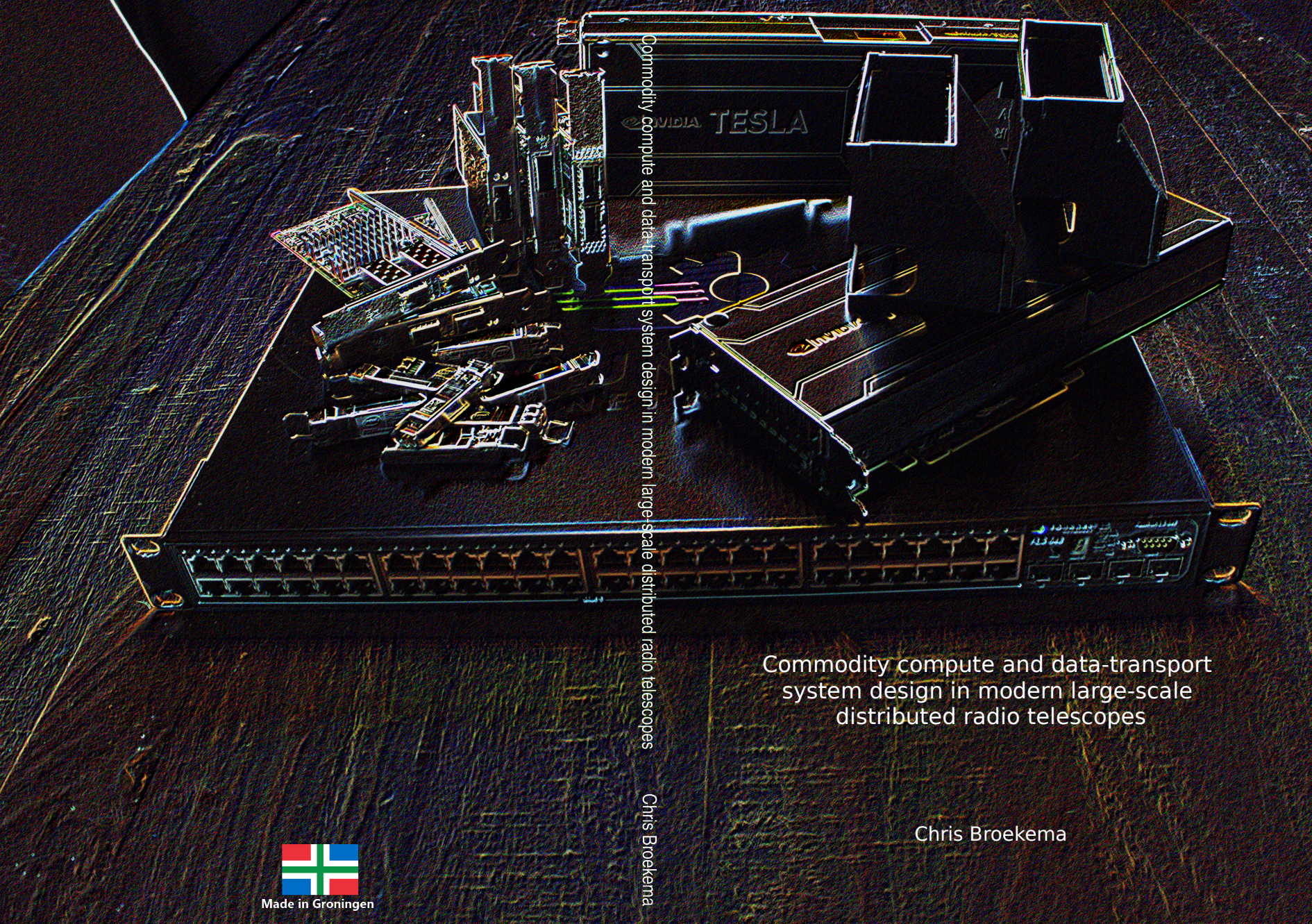


Commodity compute and data-transport system design in modern large-scale distributed radio telescopes

Chris Broekema

Made in Groningen

Commodity compute and data-transport system design in modern large-scale distributed radio telescopes

Chris Broekema

VRIJE UNIVERSITEIT

Commodity compute- and data-transport system design in modern large-scale distributed radio telescopes

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. V. Subramaniam,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Bètawetenschappen
op vrijdag 20 maart 2020 om 13.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Pieter Christiaan Broekema

geboren te Winschoten

promotoren: prof.dr.ir. H.E. Bal
prof.dr. R.V. van Nieuwpoort

Members of the thesis committee:

prof.dr. H. Bos	Vrije Universiteit Amsterdam, chair
prof.dr.ir. H.E. Bal	Vrije Universiteit Amsterdam, promotor
prof.dr. R.V. van Nieuwpoort	University of Amsterdam, NLeSC, promotor
prof.dr. P. Alexander	University of Cambridge
prof.dr.ir. M.J. Bentum	Eindhoven University of Technology, ASTRON
prof.dr.ir. C.T.A.M. de Laat	University of Amsterdam
dr. J.A. Templon	Nikhef

Cover photo: Commodity compute and data-transport components used is this thesis. Photo and manipulation by Chris Broekema (`chris@fuzzms.nl`).

Contents

Summary

Abstract

In this thesis we study the architecture and design process of commodity compute and data-transport systems in large-scale distributed radio telescopes. The goal of such systems is to facilitate the maximum viable amount of scientific discovery for the investment made. To aid our discussions, we first introduce a model to more formally express value and cost of compute systems. This allows more granular evaluation of various systems, based not just on their cost, but on their value potential as well.

We argue that, since modern radio telescopes are generally capable of producing overwhelming volumes of data, the data-transport system in such an instrument should be architected and designed together with the compute infrastructure. Examples both in the current LOFAR telescope, as well as in the Square Kilometre Array still under development, show that this co-design of data-transport and compute systems has significant value benefits.

When these systems are considered together, interesting optimisations on the boundary of the two may be considered. We show two such optimisations that focus on controlling data-flows and reducing energy consumption of such data-flows respectively.

Finally we consider the future. Over the last couple of decades compute capacity has continued to increase both predictably and dramatically. Current manufacturing technologies are approaching physical limits, which make this trend unsustainable, at least for conventional technologies. We inventory possible alternative technologies and how these may be applied in support of a radio telescope.

High level overview

The primary goal of the work presented in this thesis is to facilitate radio astronomy by designing efficient compute systems and associated data-transport infrastructures. This is of course much easier said than done, if only due to the difficulty in defining what it exactly means to design an efficient compute system. Which summarises our research question: *What is the optimal way to design a compute and data-transport system for modern radio telescopes, and how do we define optimal?* In order to place our research into context, and at least partially guide the design process, this thesis introduces four propositions as a central theme in the first chapter. These can be summarised as follows:

1. Before designing a system, bound the problem in terms of requirements and available resources
2. Compute- and data-transport systems for radio astronomy must not be developed in isolation
3. An architecture should not optimise for cost alone, but aim for the optimal combination of cost and value
4. Interesting optimisations are possible on the boundary between data-transport and compute systems if these are developed jointly

These propositions concisely express the recommendations and experiences that are further documented and explored in this thesis, and can be considered its main high-level contributions. Each chapter has some relation to one or more of these propositions, as will be explained at the end of each chapter.

Research highlights

While the propositions introduced in Chapter 1 concisely summarise the contributions in this thesis, we take this opportunity to highlight some of the research results here.

Introduction and concepts

To set the stage we start by describing the anatomy of a radio telescope. This is done at high level, and from the point of view of a computer scientist focusing on the application of commodity compute and data-transport systems, to establish the conceptual requirements on such systems. In further chapters, in particular Chapters 3, 4 and 5, such requirements are explored in more detail for their specific applications.

Furthermore, we establish a conceptual model that allows us to express the quality of designs relative to others. We argue that cost, currently often the most important or only design consideration given, should be offset by some expression of value, whatever that value may be. The most suitable design is the one that offers the most value, in the case of radio telescopes this is likely science, per invested Euro. Even though we acknowledge the difficulty in measuring science output, in particular in the design phase of an instrument, the focus on not just cost, but also value in the selection of a design or architecture is a valuable contribution.

Architecture and design work

The concepts introduced in Chapters 1 and 2 follow from extensive relevant design experience, some of which is shown in Chapters 3, 4 and 5.

Chapters 3 and 4 describe the architecture and design of the Square Kilometre Array Science Data Processor. These chapters are based on two papers that were published three years apart, and the evolution of the architecture, and the requirements that drive it, is clearly visible. Considering that even the most recent of these chapters was published

at least ten years before the full capacity of the Science Data Processor is needed, the design is by necessity limited to high-level concepts. As construction draws closer, more detailed designs will have to be made, requiring detailed knowledge of the available hardware and how these fit in a radio telescope. In Chapters 8 and 9 we recommend some future work to facilitate this.

The Cobalt project described in Chapter 5 illustrate on a relatively small scale how the concepts are applied in a real-life design process. Here we describe in exhaustive detail how initial high level requirements are turned into an prototype and eventually a production system. The problems encountered, both in finding suitable equipment and in adapting systems to make them suitable for our application, are detailed. This project, and in particular the detailed analysis of the shortcomings of than available hardware systems, gained a lot of attention. The design was used for the Wilkes general purpose computing cluster at the University of Cambridge and discussions with industry representatives showed their interest in such detailed analysis. While we have no proof that the Cobalt project has directly influenced industry, we have noticed an increased availability of suitable hardware in more recent years.

Optimisations and their impact

One of the primary recommendations in this thesis is to design compute and data-transport systems together. When applied properly, this allows interesting optimisations to be considered, primarily on the boundary between data-transport and compute systems. In Chapters 6 and 7 we show two of such optimisations, as well as their potential impact on the efficiency of the system.

While both of these are examples of optimisations that have the potential to significantly improve efficiency of a radio telescope system, they target radically different parts of that efficiency spectrum. Chapter 7 aims to decrease the energy required to receive large volumes of data by avoiding expensive kernel operations. We show that energy can be saved by applying RDMA (Remote Direct Memory Access) technologies, thus reducing cost without any impact on the science.

In Chapter 6 on the other hand we show that a Software-Defined Network (SDN) allows more robustness, flexibility and potentially even functionality to be gained, without significant additional cost increase. This optimisation aims to increase the amount of science that can be done, keeping cost similar.

Future developments and conclusions

We have shown work done to design the central compute and data-transport infrastructure for the Square Kilometre Array. The design phase of this instrument has just concluded, and construction is expected to commence in 2021. However, compute capacity will only be required in significant from around 2026 onward. Furthermore, cost considerations make it likely that continuous procurements will have to be made during the entire fifty year expected life-span of the telescope. It is therefore useful to look at future technologies that may become available in the future, and how these may be used in the a modern radio telescope. In Chapter 8 we summarise our current under-

standing of both conventional and less conventional future compute systems and their applicability.

The conclusions in this chapter, and the future work identified, form the basis of Chapter 9. Here, we summarise the research in this thesis, and how the various chapters contribute to the propositions introduced in Chapter 1. Some future work is suggested based on the risks identified in the previous chapter.

Final words

In summary, in this thesis we present on overview of extensive design experience, combined with the insights this has produced. The concepts and lessons learned are We show the evolution of the system design of the Square Kilometre Array Science Data Processor as it draws closer to construction. A more detailed analysis of a central component of the LOFAR telescope is presented as an example of a smaller scale but end-to-end design that follows the recommendations introduced in this thesis. Some optimisations are discussed on the boundary between network and compute systems that improve the relative value of the system by focusing on different parts of the equation, cost and value. Finally, we look at how future technologies may be used effectively in radio astronomy. All this combined gives the reader an all-round overview of the considerations when designing commodity compute and data-transport systems for modern distributed radio telescopes.

Samenvatting

Resumé

Dit proefschrift beschrijft het ontwerpproces van reken- en datatransport-systemen voor grootschalige gedistribueerde radiotelescopen, van concept tot implementatie. Het doel van zulke systemen is het faciliteren van zoveel mogelijk wetenschap voor de gemaakte investering. Om ons onderzoek vorm te geven, introduceren we eest een model om de kosten en vooral ook de waarde van een geïntegreert datatransport- en reken-systeem uit te drukken. Dit geeft ons de mogelijkheid om gedetailleerder systemen te evalueren, niet alleen open basis van kosten maar ook op basis van hun waardepotentieel.

We stellen dat, aangezien moderne radiotelescopen over het algemeen in staat zijn overweldigende hoeveelheden data te transporteren, het datatransport systeem in een dergelijk instrument gelijktijdig en samen met het rekeninfrastructuur moet worden ontworpen. Voorbeelden zowel in de operationele LOFAR radiotelescoop als in de nog te bouwen Square Kilometre Array laten zien dat zulk gezamelijk ontwerp van datatransport en rekensystemen aanzienlijke een meetbare voordelen biedt.

Wanneer deze twee componenten als één geheel worden bekeken, worden interessante optimalisaties op de grens van de twee mogelijk. In dit proefschrift laten we twee van dergelijke optimatlisaties zien, die zich respectievelijk richten op het beheersen en sturen van datastromen het het verminderen van het energieverbruik die het ontvangen van dergelijke datastromen met zich meebrengt.

Tot slot kijken we naar de toekomst. In de afgelopen decennia is rekenkracht dramatisch en voorspelbaar blijven toenemen. De huidige productietechnologiën voor rekencomponenten nadert echter fysieke grenzen en het is onwaarschijnlijk dat die trend nog lang stand houdt, althans voor conventionele technologiën. We inventariseren mogelijke alternatieve technologiën en hoe deze kunnen worden toegepast als rekensysteem in een radiotelescoop.

Overzicht op hoog niveau

Het belangrijkste doel van het werk dat in dit proefschrift wordt gepresenteerd is het ondersteunen en faciliteren van wetenschappelijke ontdekkingen in de radioastronomie door het ontwerpen van efficiënte computersystemen en bijbehorende datatransportsystemen. Natuurlijk is dat veel makkelijker gezegd dan gedaan, al was het alleen maar

omdat het moeilijk is precies te definiëren wat het betekent dat een computersysteem efficiënt is. Dit alles leidt ons tot onze onderzoeksvraag: *Wat is de optimale manier om een data-transport en rekensysteem te ontwerpen voor moderne radiotelescopen, en hoe is optimaal gedefinieerd?* Om ons onderzoek in de juiste context te plaatsen, en ten minste gedeeltelijk het ontwerpproces te begeleiden, introduceert deze dissertatie in het eerste hoofdstuk vier stellingen die als rode draad door het proefschrift lopen. Deze kunnen als volgt worden samengevat:

1. Voor aanvang van een ontwerp, bakenen we eerste de grenzen wat betreft eisen aan het systeem en beschikbare middelen om het te realiseren af
2. Datatransport- en reken-systemen voor radioastronomie moeten niet afzonderlijk van elkaar ontworpen worden
3. Een ontwerp moet zich niet alleen richten op het minimaliseren van kosten, maar streven naar een optimale combinatie van kosten en waarde
4. Interessante optimalisaties zijn mogelijk op de grens tussen datatransport- en reken-systemen wanneer deze gezamenlijk worden ontworpen

Deze stellingen formuleren kort en bondig de aanbevelingen en ervaringen die in dit proefschrift verder worden uitgewerkt en gedocumenteerd en kunnen worden gezien als de belangrijkste bijdragen op hoog niveau van dit proefschrift. Elk inhoudelijke hoofdstuk verwijst op enige wijze terug naar of draagt bij aan één of meer van deze stellingen, zoals zal worden toegelicht aan het einde van elk hoofdstuk.

Hoogtepunten en blikvangers

Hoewel de in hoofdstuk 1 geïntroduceerde stellingen de belangrijkste bijdragen in dit proefschrift beknopt samenvatten, maken we van de gelegenheid gebruik om hier een aantal van de belangrijkste onderzoeksresultaten te belichten.

Inleiding en concepten

Om ons onderzoek in de juiste context te plaatsen, beschrijven we eerst de anatomie van een moderne gedistribueerde radiotelescoop op hoog niveau. Hiermee laten we zien waar in een moderne radiotelescoop computer- en datatransport-systemen worden toegepast, en wat de conceptuele eisen hieraan gesteld worden. In latere hoofdstukken, met name Hoofstukken 3, 4 en 5 die dieper ingaan op componenten van specifieke instrumenten, worden deze eisen verder uitgewerkt.

In Hoofdstuk 2 introduceren we een conceptueel model waarmee we de kwaliteit van de ontwerpen kunnen uitdrukken ten opzichte van andere. We stellen hierin dat de kosten, op dit moment vaak de belangrijkste of enige ontwerpoverweging die wordt gegeven, gecompenseerd moeten worden door een of andere uitdrukking van waarde, wat die waarde ook moge zijn. Het meest geschikte ontwerp is een die de meeste waarde biedt, in het geval van radiotelescopen is dit waarschijnlijk wetenschappelijke ontdekkingen, per geïnvesteerde euro. Hoewel we erkennen dat het moeilijk is om de

waarde wetenschappelijke ontdekkingen te meten, met name in de ontwerpfase van een instrument, is de focus op niet alleen de kosten maar ook de waarde bij de keuze van een ontwerp of architectuur een waardevolle bijdrage.

Architectuur en ontwerp

De concepten die we in Hoofdstukken 1 en 2 worden geïntroduceerd komen voort uit aanzienlijke en relevante ervaring met het ontwerpen van rekensystemen voor radiotelescopen. Een deel hiervan tonen we in Hoofdstukken 3, 4 en 5. In Hoofdstukken 3 en 4 beschrijven we het ontwerpproces van de Square Kilometre Array (SKA) Science Data Processor (SDP). Deze hoofdstukken zijn gebaseerd op twee wetenschappelijke artikelen die drie jaar na elkaar zijn gepubliceerd. Ze tonen dan ook de evolutie van de architectuur en eisen hieraan tijdens gedurende het ontwerpproces. Aangezien zelfs de meest recente van deze hoofdstukken minstens tien jaar voor de bouw van de volledige Science Data Processor is gepubliceerd, is het ontwerp noodzakelijkerwijs beperkt tot concepten op hoog niveau. Naarmate de bouw vordert, zullen er meer gedetailleerde ontwerpen moeten worden gemaakt, waarvoor gedetailleerde kennis nodig is van de beschikbare rekensystemen en hoe deze in een radiotelescoop passen. In de Hoofdstukken 8 en 9 suggereren we dat een continue proces van testen en uitproberen van nieuwe componenten een essentieel onderdeel vormt in het vergaren van deze informatie. Het ontwerp dat in deze hoofdstukken is beschreven heeft is in Januari 2019 succesvol gepresenteerd tijdens de SDP Critical Design Review en zal de basis vormen van het uiteindelijk te bouwen SDP ontwerp.

In Hoofdstuk 5 illustreren we hoe de eerder geïntroduceerde concepten op relatief kleine schaal zijn toegepast in het Cobalt project. Hier beschrijven we uitgebreid hoe de eerste eisen op hoog niveau worden omgezet in een passend prototype en uiteindelijk naar een werkend productiesysteem. De ondervonden problemen, met name de gedetailleerde analyse van de tekortkomingen van de op dat moment beschikbare hardware-systemen, bleek uitermate relevant. Het ontwerp werd, met kleine aanpassingen, overgenomen voor het Wilkes rekencluster van de Universiteit van Cambridge en uit gesprekken met industrie vertegenwoordigers bleek hun belangstelling voor een dergelijke gedetailleerde analyse. Hoewel we geen direct bewijs hebben dat het Cobalt project de industrie direct heeft beïnvloed, hebben we de laatste jaren een verhoogde beschikbaarheid van geschikte hardware opgemerkt.

Optimalisaties en hun effecten

Een van de belangrijkste aanbevelingen in deze dissertatie is dat reken- en datatransportsystemen gezamenlijk ontworpen dienen te worden. Wanneer dit wordt gedaan, opent dit mogelijkheden voor interessante optimalisaties op de grens tussen datatransport- en rekensystemen. In Hoofdstukken 6 en 7 demonstreren we twee dergelijke optimalisaties, alsmede hun potentiële impact op de efficiëntie van het systeem als geheel. Hoewel het werk in deze hoofdstukken beide tot doel heeft de efficiëntie van de datatransport- en computer-systemen te verbeteren, richten ze zich op totaal verschillende onderdelen van het spectrum.

Het werk in Hoofdstuk 7 heeft als doel de energie die nodig is om grote hoeveelheden gegevens te ontvangen te verminderen door dure kerneloperaties te vermijden. We demonstreren dat energie kan worden bespaard door het toepassen van RDMA (Remote Direct Memory Access) technologie, waardoor de kosten worden verlaagd zonder dat dit gevolgen heeft voor de wetenschappelijke waarde van het systeem. In Hoofdstuk 6 laten we zien dat een Software-Defined Network (SDN) kan leiden tot een flexibeler, robuuster systeem dat zelf mogelijk meer functionaliteit biedt, zonder dat dit een significante kostenpost met zich meerbrengt. Deze optimalisatie heeft als doel de hoeveelheid wetenschap die kan worden gedaan te vergroten door middel van het verbeteren van de betrouwbaarheid en het introduceren van functionaliteit, terwijl kosten min of meer gelijk blijven.

Toekomstige ontwikkelingen en conclusies

In de voorgaande hoofdstukken hebben we het ontwerpproces van de Square Kilometre Array Science Data Processor laten zien. De ontwerpfase van dit instrument is net afgerond en de bouw zal naar verwachting in 2021 beginnen. De volledige rekencapaciteit zal echter pas nodig zijn wanneer het gros van de ontvangers zijn gebouwd, rond 2026. Bovendien is het waarschijnlijk dat om kosten te besparen gedurende de gehele verwachte levensduur van de telescoop, vijftig jaar, continu rekencapaciteit zal worden ingekocht. Het is daarom essentieel dat er in een vroeg stadium wordt gekeken naar de toepasbaarheid van nieuwe en toekomstige technologiën. In Hoofdstuk 8 geven we een samenvatting hoe zowel conventionele als minder conventionele computersystemen zich zullen ontwikkelen in de afzienbare toekomst, en wat hun rol zou kunnen zijn in een radiotelescoop als de SKA.

De conclusies van dit hoofdstuk, en het werk we hierin suggereren om toekomstige technologiën in de gaten te houden, vormen de basis van Hoofdstuk 9. Hierin vatten we het onderzoek in dit proefschrift samen en laten we per stelling zien hoe de verschillende hoofdstukken hieraan hebben bijgedragen. Op basis van risico's die in het vorige hoofdstuk zijn gesignaleerd suggereren we aanvullend werk voor de toekomst.

Afsluitend

Samenvattend presenteren we in dit proefschrift een overzicht van uitgebreide ontwerp ervaring van reken- en datatransport-systemen voor radiotelescopen, gecombineerd met de inzichten die dit heeft opgeleverd. De concepten en geleerde lessen zijn samengevat in een viertal korte stellingen die als een rode draad door het proefschrift lopen. We tonen de evolutie van het systeemontwerp van de Square Kilometre Array Science Data Processor gedurende het ontwerpproces. Een meer gedetailleerde analyse van een centraal onderdeel van de LOFAR telescoop, de correlator en beamformer, wordt gepresenteerd als een voorbeeld van een kleinschaliger maar succesvol end-to-end ontwerpproces dat de aanbevelingen in deze dissertatie volgt. Enkele optimalisaties worden besproken op de grens tussen datatransport- en rekensystemen die de relatieve waarde van het systeem verbeteren door zich te richten op verschillende delen van de kosten en de waarde vergelijking die we eerder hebben geïntroduceerd. Tot slot kijken we

hoe toekomstige technologieën effectief kunnen worden gebruikt in de radioastronomie. Dit alles bij elkaar geeft de lezer een gebalanceerd overzicht van de overwegingen die bij het ontwerpen van reken- en datatransportsystemen voor moderne gedistribueerde radiotelescopen aandacht verdienen.

CHAPTER 1

Introduction

Modern radio astronomy and computer science are both relatively young sciences that have fairly similar histories. Both were perhaps not born out of the ruins of the second world war, but have at least benefited greatly from the technologies developed during that time.

The second world war saw the first programmable computers built to aid the cryptographic effort of legendary computer scientists like Alan Turing. At the same time, tumultuous developments in radar technology, and the skilled technicians trained in their design and use, were of great help to verify the existence and kick-start the study of the newly discovered radio universe.

The modern study of physics, in particular astrophysics, relies critically on powerful compute systems and software to leverage these to produce cutting edge science. Arguably, the continued development of ever more powerful compute resources has driven the development of ever more capable scientific instruments. Indeed, for several radio telescopes past and present we can identify timing considerations that synchronise compute developments and construction of the instrument. Generally this is done to maximise the science impact of the instrument per invested Euro. In other words, instruments are often designed when the supporting computing equipment is infeasibly expensive, relying on continued developments in processor, memory and storage technologies to make the supporting equipment and software affordable when construction actually starts.

This reliance on cutting edge and affordable computing systems, make the development of more specialised and tailored solutions attractive. Whereas most conventional high-performance computing facilities need to support a wide variety of applications, infrastructure supporting an observing instrument can be optimised for a single, or small selection, of applications.

1.1 Compute systems and radio astronomy

The histories of radio astronomy and computer science are both short and tumultuous. Indeed radio astronomy, and in particular aperture synthesis, parallels that of computer science due to its heavy reliance on compute resources. It was the (re)invention of the Fast Fourier Transform [1], and development of mini-computers fast enough to run these at scale, that drove the development of the first aperture synthesis radio telescopes in the late 1960's and early 1970's, as Martin Ryle discussed in his Nobel lecture in 1974 [133].

1.1.1 Anatomy of a modern aperture synthesis radio telescope

Modern radio telescopes generally consist of multiple smaller receivers, combined into a single large virtual receiver using aperture synthesis. This concept, pioneered in the 1960s and 1970s with telescopes such as the one-Mile telescope in the UK and the Westerbork Synthesis Radio Telescope in the Netherlands, is a cost-effective way to dramatically increase the sensitivity and resolution of radio telescopes, without having to build massive dishes. The theory is based on the van Cittert – Zernike theorem, which states that two geographically separated receivers will sample a coherent signal from an incoherent source, provided the source is much farther from the receivers than the receivers are from each other. The coherence function of two such receivers has a Fourier relationship with the brightness distribution of the incoherent source sampled ([151], chapters 2 and 15[2]). This is called interferometry.

In other words, when two receivers sample a wavefront from a distance source, the complex visibility function of these samples represents a point in Fourier space of the brightness distribution of that source. Taking many such points, from a collection of receiver pairs, allows us to reconstruct a sparse representation of the distant source by means of an inverse Fourier transform ([151], chapter 3). Observing for a longer time allows us to take advantage of earth rotation to fill in more of the image. In essence we construct a large virtual telescope from multiple smaller ones, and we therefore call this aperture synthesis. We will not go in to the details of aperture synthesis theory, instead we refer the interested reader to the excellent standard textbook on radio interferometry by Thompson et al. [151].

Other modes of operation, apart from the aperture synthesis imaging mode described above, that tries to reconstruct a visual representation of the radio brightness of the source from the coherence between multiple receivers, are generally available as well. The study of radio pulsars and transient radio sources, such as Fast Radio Bursts (FRBs), generally do not require aperture synthesis, but instead focus on time-domain data. Such observation modes co-exist in modern radio telescopes, making the instrument more flexible and able to serve a larger scientific community, but the additional, possibly conflicting, requirements these additional observation modes place on the instrument signal processing components make their design more complex.

[1] While the invention of the FFT is generally credited to Cooley [48], analysis of work done by Gauss in the early 19th century indicates at least some aspects had been conceived before [75].

[2] Chapter 14 in the second edition of Thompson et al.

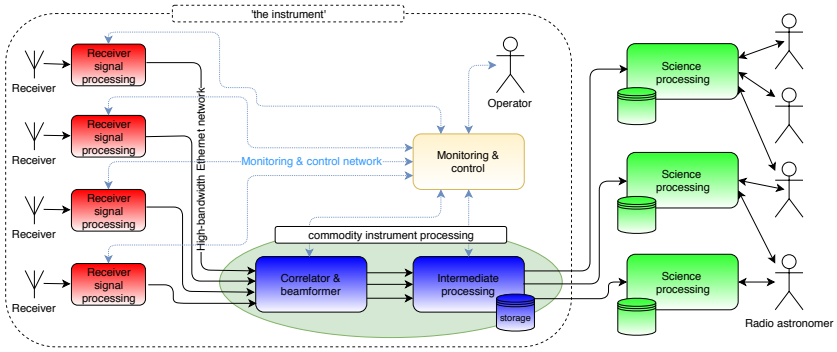


Figure 1.1: Top level overview of a generic distributed aperture synthesis radio telescope

In practical terms, a modern aperture synthesis radio telescope consists of the following components, as shown in Figure 1.1:

1. antennas, receivers and digitisers
2. receiver signal processing
3. correlator and beamformer
4. intermediate processing into science-ready intermediate products
5. science processing into science products

In Figure 1.1 we have identified the area of interest for this thesis. This is based on the potential to use commodity computing, instead of custom designed hardware. While the boundary between custom- and commodity hardware differs between instruments, we can generally state that this decision is based on reduced data rates as data moves through the system, and increased compute requirements per bit of data. On this boundary, we generally have a number of real-time processes on commodity hardware to receive data large volumes of data and process at high spectral and temporal resolution. When this data volume is reduced, the real-time requirement is reduced as well and data can be temporarily stored for further, more iterative, processing. This mix of compute profiles makes commodity computing in modern distributed radio telescopes such an interesting challenge.

Aperture synthesis imaging modes generally drive the compute- and data-transport requirement in modern radio telescopes. Therefore, the analysis of these observation modes is often prioritised, with time domain science cases following closely behind to verify they fit within the scope defined by the imaging modes.

Receivers

The receivers, the antennas and associated electronics needed to capture and condition the radio signal we want to observe, are the most visible and recognisable component of a radio telescope. Large, fully movable dishes, such as the ones shown in Figure 1.2, have been the poster child for radio astronomy for some time.

However, more recently and at lower frequencies, such large and expensive dish systems have been supplemented by large numbers of simple, omni-directional receivers that are combined in software. An example of an operational instrument based on many such low-cost antennas is LOFAR [158], the central *Superterp* of which is shown in Figure 1.3.

Receivers may contain analogue signal processing hardware for amplification and filtering. In some cases, in particular the LOFAR high-band antennas, several such receivers are combined using an analogue beamformer, essentially a weighted or unweighted addition, into a single virtual receiver. This results in a smaller and more sensitive beam pattern compared to that of a single receiver and reduces the data rate by a factor equal to the number of elements in the antenna array.

While still in the analogue domain, initial filtering and amplification is applied. Next, the analogue signal is digitised, often after being down converted. This is generally done early and close to the receiver to take advantage of more robust digital data transmission technologies over longer distances, although advances in Radio Frequency over Fibre have made short range transfers of analogue data possible [165]. A generic receiver and digitisation pipeline is shown in Figure 1.4. No digital computing is done in the receiver, and we will not discuss receivers in detail in this thesis.

Receiver signal processing

Receiver signal processing usually consists of a digital filter, often implemented as a polyphase filterbank consisting of a number of Finite Impulse Response (FIR) filters, feeding into a single Fast Fourier transform (FFT). This results in a spectrally separated channelised signal, the number of channels depending on the size of the Fast Fourier Transform and the number of FIR filters. A selection of channels can be discarded to reduce data transmission load and save on required compute capacity in the components further upstream. When omni-directional receivers are used, an optional spatial beamforming step, a coherent addition of weighted omni-directional signals, may be applied. This beamformer requires spectrally similar signals from different receiver, while the polyphase filterbank results in a collection of spectral channels per receiver. Therefore, before beamforming, the data needs to be re-ordered as shown in the generic receiver signal processing pipeline in Figure 1.5.

Receiver signal processing is characterised by very high data rates and very low computational intensity [3]. Furthermore, data flows through the pipeline continuously, and the processing is static in that the algorithms are well known and not expected to change during the lifetime of the instrument. This means that this component is well suited to be implemented in dedicated hardware, often based on FPGA (Field Programmable Gate

[3] Computational intensity is defined as the number of floating point operations per byte of data moved

Figure 1.2: The Westerbork Synthesis Radio Telescope. ©ASTRON.

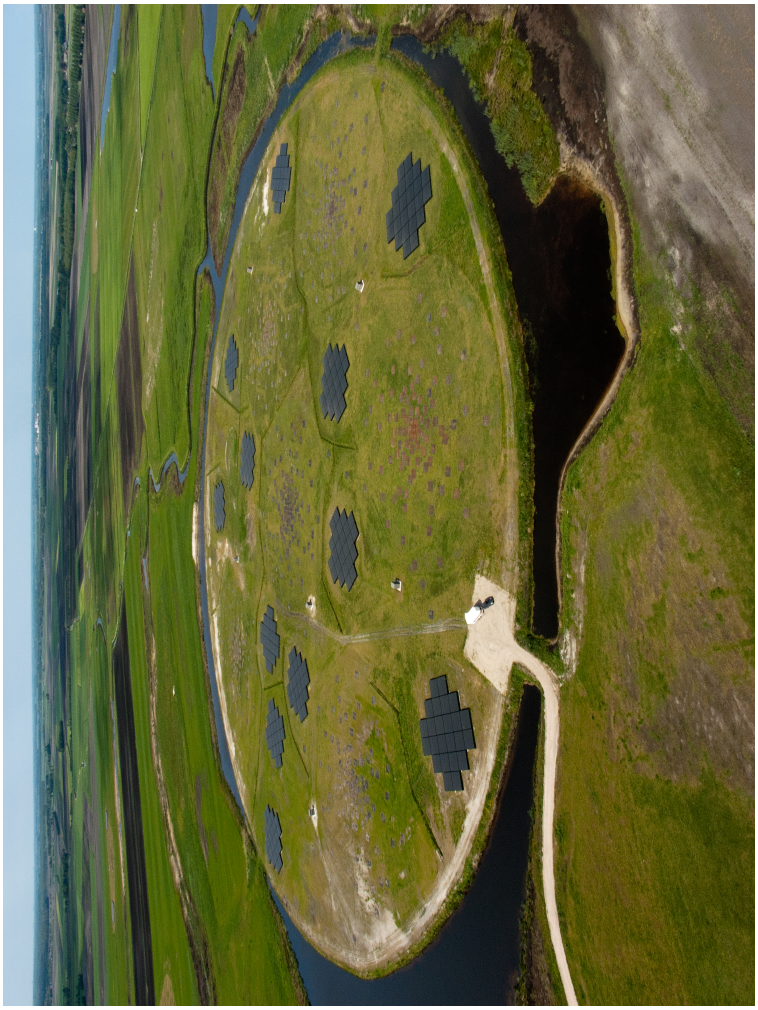


Figure 1.3: The LOFAR *Superterp*. ©Top-Foto, Assen.

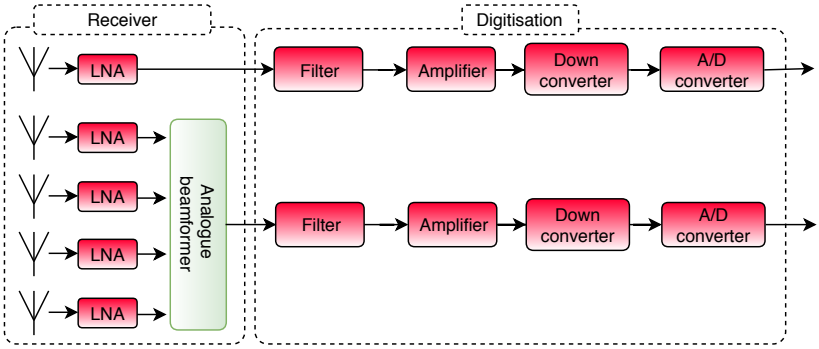


Figure 1.4: A generic receiver pipeline, including Low Noise Amplifiers (LNAs), analogue filters and digitisation components, with and without analogue beamforming.

Array) technology (for example Uniboard^2 [136]). For these reasons, receiver signal processing will not be discussed in this thesis.

Correlator and beamformer

The correlator and beamformer component receives data from the receiver signal processing system and applies a comprehensive re-ordering. Receiver signal processing results in a data stream that, for a single receiver, contains all selected frequency channels. The correlator and beamformer requires data from all receivers per frequency channel. For instruments with widely separated receivers or receiver stations, the observed wavefront may arrive at receivers delayed by multiple samples. This is corrected by delay compensation, by shifting samples appropriately. Next, a second filter is applied to that data, increasing the spectral granularity of the data. This increased spectral detail offers opportunity to mitigate clock drift effects and the ripple introduced by the first bandpass filter bank, as well as the sub-sample delays caused by geographical separation of receivers.

The correlator produces the product from each receiver pair, while the beamformer, similar to the receiver processing beamformer, applies weighted addition of all receivers into a spatial beam. Incoherent (non-weighted) addition is often offered when sensitivity over a large field of view is desired. After the correlator and / or beamformer is applied, data can integrated in time and / or frequency to reduce the volume of data to manageable levels. A generic correlator and beamformer is shown in Figure 1.6.

The correlator and beamformer are configurable components, able to run a combination of polyphase filterbanks, complex correlators and coherent (i.e. weighted) or incoherent (unweighted) beamformers. Although configurable per observation mode, this sub-system is fairly static, unlike the intermediate processing sub-system described in the next section.

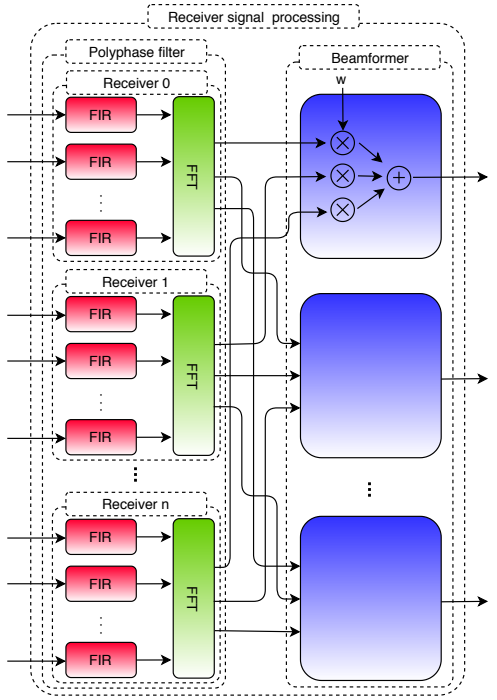


Figure 1.5: A generic receiver signal processing pipeline.

Whereas receiver processing is generally local to the receiver, the beamformer and correlator brings together data from all receivers in the instrument. Therefore, in general, there is only a single, centrally located and highly optimised, correlator and beamformer system per instrument. A geographically distributed correlator and beamformer is possible in theory, but in particular in modern large scale radio telescopes with many receivers data volumes explode during processing, making a single central correlator and beamformer a much more attractive solution.

Compared to receiver signal processing, the correlator and beamformer are generally characterised by marginally lower data rates and, depending on the number of receivers or receiver stations, have a higher computational intensity. The correlator and beamformer are real-time processing components, meaning that data streaming from receiver processing and data rates are challenging. These characteristics mean that the correlator and beamformer may be efficiently implemented in dedicated hardware, as is currently envisioned for the SKA, or in commodity hardware, as in LOFAR. Both have their advantages, which we will not discuss in detail in this thesis, and depending on

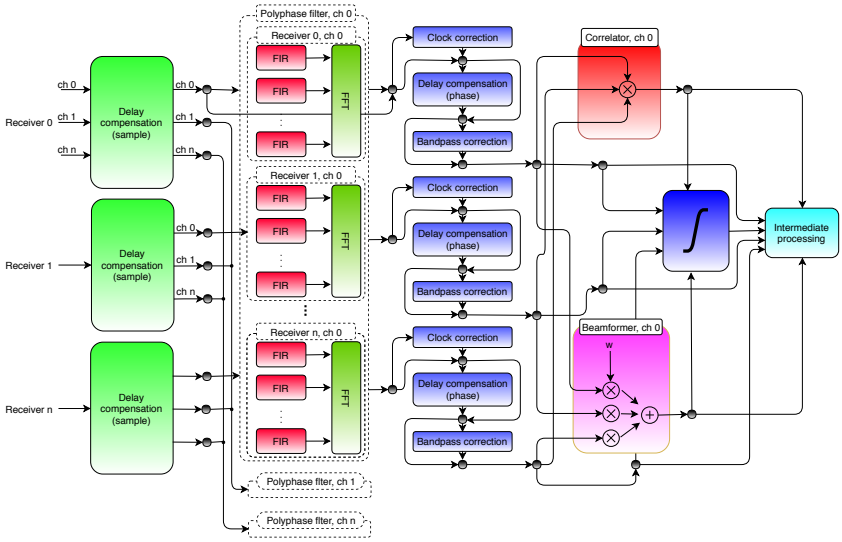


Figure 1.6: A generic correlator and beamformer with associated corrections shown.

the frequency range the instrument is designed to observe at and number of receivers or receiver stations in the instrument either may be better suited for the task. However, we will show the design and implementation of a highly optimised correlator and beamformer for the LOFAR telescope, based on commodity hardware.

Intermediate processing

Intermediate processing makes science-ready data products from the beams or visibilities that have been produced by the correlator and beamformer. Whereas the previous components have well defined functionality (even though not all are implemented in every telescope and some of these are configurable), intermediate processing is much more diverse. The functionality can roughly be summarised as taking the instrument data as delivered by the previous components and turning it into science-ready data products for analysis by the astronomer. This generally involves:

- removing interference caused by local sources
- removing strong sources outside of the field of view that have leaked into the image
- correcting for instrumental effects
- generating sky images, source catalogues, pulsar candidate lists and various other science-ready data products

- producing calibration data for the other components in the systems
- Searching for unknown pulsars
- Timing spin-rate of known pulsars
- Detection and analysis of transient events, such as fast radio bursts (FRBs)
- Store intermediate data products and distribute these to science processing

In Figure 1.7 we show the intermediate processing component in the Square Kilometre Array. Here we illustrate the relative complexity of this component by first showing the top level context, with inputs and outputs, and then drilling down into the processing components required. More detail is shown for the calibration and imaging component using a functional and data flow breakdown of both functions.

Contrary to the processing done in the instrument so far, intermediate processing is iterative. While still data-intensive, processing has much higher computational intensity. This means that intermediate processing is no longer suited to be run on custom hardware, and requires general purpose computing hardware instead. Intermediate processing is generally the last step done before the data is released to the astronomer for further, possibly interactive, analysis in the science processing component.

At the end of the intermediate processing stage data rates dropped sufficiently to be able to preserve all data products generated. Therefore, the intermediate processing component may also be required to store, index and make available the data produced by the instrument, either indirectly to the user via the science processing component described below, or directly. Since intermediate processing is still part of the instrument, user interaction is limited or impossible. While data may be preserved in by the intermediate processing component, this is for backup purposes only. Data is generally exported to science processing where it is made available to the astronomer.

Science processing

Science processing is where intermediate products, delivered by the instrument, are analysed and further processed into science products. While we can argue about the exact boundary of the instrument, science processing is generally not considered to be part of the instrument, and the science processing facility is likely local to the astronomer, not necessarily the telescope. Traditionally science processing is done on the astronomer's workstation or laptop, but modern data rates and volumes in modern instruments are such that this is no longer feasible.

Contrary to the processing done so far, science processing is often in interactive and iterative process, wherein the scientist manipulates the data and verifies the result. Data volumes for modern instruments are significant, and processing is highly diverse.

Although the work in this thesis is applicable to the science processing facility architecture and design, we will not explore this part of the system in detail.

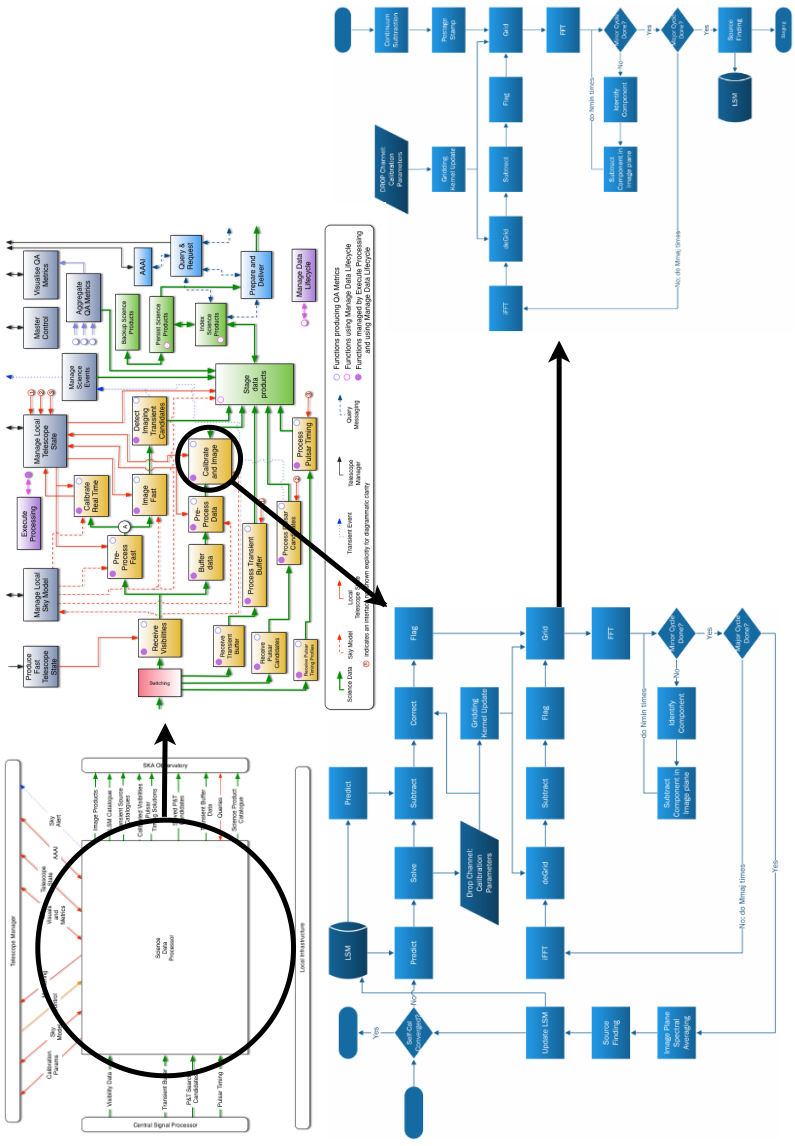


Figure 1.7: An overview that shows the relative complexity of the SKA SDP intermediate processing system (collected from its Critical and Preliminary Design Review documentation).

1.1.2 Compute- and data-transport systems in modern large-scale distributed radio telescopes

In Figure 1.1 we have identified the area of interest for this thesis: instrument processing. This is generally implemented on commodity general purpose hardware. While the components differ per instrument, generally we can state that commodity compute systems that are part of a modern distributed radio telescope are characterised by:

1. high-bandwidth, long-range data transport into the compute system
2. a boundary between custom designed and commodity hardware, generally communicating via high-performance Ethernet networks
3. applications include some that are soft real-time, some that are data-driven, but all are highly data-intensive
4. a mix of performance profiles, from data-dominated low computational intensity, to highly compute intensive iterative processing
5. automated, non-interactive processing of data streams, with none or limited user codes running on the systems

In designing compute- and data-transport systems for modern radio telescopes, we take inspiration from high-performance computing and big data systems. However, the performance profiles seen in radio astronomy, in particular those with a low computational intensity, are relatively unique. Furthermore, we are often heavily cost constrained. This drives a desire to design a cost-effective system, which is the main subject of this thesis. Previous experience with LOFAR systems has shown that separate development of data-transport system and compute system, in particular in combination with the non-standard custom hardware often employed, can lead to problems [37].

1.2 Research driven by architecture and design experience

The research presented in this thesis is based in part on extensive architecture and design activities. Lessons learned in architecture and design of compute- and data-transport systems for radio astronomy have played a large part defining the direction of the research presented in this manuscript. Apart from the usual peer-reviewed publications, the Curriculum Vitae on page 183 of this work highlights a selection of the large number of documents and design studies that have recently been produced for a number of radio astronomical sub-systems and architectures authored and co-authored by the author of this thesis. This not only shows the complementary nature of the research done in this thesis, it is also an indication of the value of architecture and research work done side-by-side. Experiences designing and architecting complex and specialised systems for radio astronomy drive interesting research questions, while the results of that research can significantly benefit future architecture work.

1.3 Research question for this thesis

Computer science, in particular as practised at an institute like ASTRON (the Netherlands Institute for Radio Astronomy), can be considered a facilitating science. The hardware and software systems that are architected, designed and built in this thesis are all meant to facilitate and enable other sciences, in this case in radio astronomy and astrophysics. In general science budgets are limited, and only a fraction of these can be used for compute hardware. Consequently there is an obvious drive to maximise the effectiveness of any investment in computational infrastructure, be it hardware or software. This drive to architect and design compute systems that maximise the scientific impact of such systems per invested Euro is the central premise of this thesis. Considering the reason of being of compute systems in radio astronomy and other physical sciences, our main research question can therefore be summarised as follows:

RQ. *What is the optimal way to design a commodity compute and data transport architecture for modern distributed radio telescopes, and how do we define optimal in this context?*

In this thesis we will endeavour to answer this question and show, using a number of propositions introduced in the next section, how our proposed solution addresses this. We also show that our proposed approach offers optimisation opportunities that would otherwise be difficult or impossible to achieve. Although these do not directly address our main research question, the offered advantages are significant and they add a significant in-depth component to our research.

1.4 Propositions summarising the research in this thesis

To address the high level research question introduced above, we propose a number of design priorities and recommendations. These are heavily inspired by the architecture and design experience mentioned above and should be taken into account during all phases of the architecture and design process for an IT infrastructure supporting a data-intensive science instrument. We have summarised these recommendations in four propositions:

Proposition 1. *Before embarking on an architecture or design, bound the problem in terms of requirements, such as capacity and functionality, and available resources, such as funds, facilities, manpower and interfaces.*

When designing a compute or data-transport system, it is useful to first bound the problem, both from a requirements and available resources perspective. This not only places the project in context, it also requires all parties involved to carefully articulate the requirements of the system under design. Furthermore, by defining requirements at the start of the project, a clear definition when the project is *finished* is explicitly documented. If either is expected to change significantly at some point, scalability is also a significant consideration. This is the *bounding* proposition.
Addressed in Chapters 2 (partially), 3, 4 and 5 of this thesis.

Proposition 2. *The compute- and data-transport systems supporting modern radio telescopes must not be developed in isolation.*

In traditional compute infrastructures and high-performance computing centres, the compute- and data-transport systems are generally developed separately. This often extends to the administration of these systems, which is usually done by different, siloed departments. In a data-intensive and often streaming instrument, like a radio telescope, the synergy between compute- and data-transport makes a coherent design of these two components essential. Otherwise, there is a considerable risk that a bottleneck in the one will significantly impact the other. In this thesis we will refer to this as the *co-design* proposition.
Addressed in Chapters 4, 5 and 8 of this thesis.

Proposition 3. *A system's architecture and design should not only optimise for cost, but instead closely consider the optimal ratio between value and cost.*

Scientific instruments generally have fairly limited budgets, and only a fraction of these can be reserved for computing and data-transport. There is therefore a strong and obvious desire to minimise the cost of these components. This is amplified by procurement regulations, that often drive the choice towards the cheapest solution available that meets the required specifications. However, the minimal viable solution is not necessarily the optimal one. We argue that by considering both value and cost, more science can be done using the same investment. In Chapter 2 we introduce a new conceptual model that allows more effective and structured reasoning about value in eScience solutions. We will refer to this as the *value* proposition.
Addressed in Chapters 2, 3(partially), 4, 5 and 8 of this thesis.

Proposition 4. *When both the compute- and data-transport system are considered jointly, optimisations can be conceived on the boundary between these two that greatly benefit the whole.*

Considering the two recommendations above, we can take advantage of the now integrated architecture and design of the data-transport and compute systems. The boundary between the two, in particular the interface between the high-bandwidth custom hardware that generates instrument data, and the commodity compute system that produces scientific data, offers many interesting and challenging optimisation opportunities. We investigate both functional and operational (i.e. energy-saving) improvements. For the purposes of this thesis, we will refer to this as the *optimisation* proposition.
Addressed in Chapters 6 and 7 of this thesis.

1.5 Support of our propositions per chapter

Since this thesis consists of a number of publications, it is useful to identify how each of these contribute to the propositions that are central to this manuscript. For each of the chapters we discuss how these support the various propositions. This is also shown in a more visual manner in Table 1.1.

	bounding	*co-design*	*value*	*optimisation*
Chapter 2, *'Cost and Value'*	✓		✓	
Chapter 3, *'Exascale computing in the SKA'*	✓		✓	
Chapter 4, *'SKA compute platform design'*	✓	✓	✓	
Chapter 5, *'Cobalt'*	✓	✓	✓	
Chapter 6, *'Software-defined networking'*			✓	✓
Chapter 7, *'UDP RDMA'*			✓	✓
Chapter 8, *'Future developments'*		✓	✓	✓

Table 1.1: Mapping of propositions to the chapters in this thesis. Light grey check marks signify partial applicability.

1.5.1 Chapter 2

In Chapter 2 we take a step back from the actual design of compute- and data-transport systems, and introduce a more formal and structured way to reason about the value and cost of solutions. Here, we introduce some of the tools that are used to identify which of any number of possible system designs is optimal for the applications in question. This chapter supports the *value* proposition, and to some degree the *bounding* proposition.

1.5.2 Chapters 3

Chapters 3 and 4 offer an insight into the design process of a component of a large radio telescope, the Square Kilometre Array (SKA). Chapter 3 was written in 2012, just after the Conceptual Design Review (CoDR) for the software and computing component of SKA. This paper summarised the computing challenges, which at that stage were considered quite challenging, and summarises how the SKA processing challenge is different from more conventional high-performance computing applications. An extrapolation of existing HPC systems at the time showed that sufficient compute capacity should become available by 2018-2019, but notes that this is only valid for the LINPACK benchmark used in the Top500 list. The argument is made to avoid data transport as much as possible to reduce energy consumption, since it was assumed, based on available information, that data transport over larger distances would be prohibitively expensive in terms of energy consumption. Even though this chapter significantly predates Chapter 2, some of its recommendations are already implicitly taken into account. This chapter illustrates both the *bounding* and *value* propositions to some degree.

1.5.3 Chapter 4

Chapter 4 was written a couple of years later (2015), and clearly shows that the analysis had progressed dramatically. The scale of the SKA was reduced slightly, considerably reducing the required compute- and data-transport capacity, making the resulting system far more affordable. Furthermore, a more detailed picture of the required processing could be sketched, and as a result a highly scalable but feasible architecture was introduced. A number of value measures, scalability, affordability, maintainability and sup-

port for current state-of-the-art algorithms, were defined that could be used to evaluate possible implementations. This chapter strongly supports the *bounding*, *co-design* and *value* propositions. We refer to some possible optimisations that are under investigation, supporting the *optimisation* proposition.

1.5.4 Chapter 5

In Chapter 5 we show a practical example of a relatively small system that is intensively optimised for a specific task. Both cost and value, as defined in Chapter 2 are considered in this project. Furthermore, a design methodology focused on data flow, rather than just compute capacity, was introduced, as recommended in this thesis. This chapter strongly supports the *bounding*, *co-design* and *value* propositions.

1.5.5 Chapters 6

A primary recommendation in this thesis is that compute- and data-transport infrastructure should be designed in close collaboration. Since the latter of these is more often overlooked, we have concentrated on this particular component in some of our more detailed experimental work. Chapter 6 explores a potentially revolutionary development in networking: an affordable and highly flexible programmable network. Using a use-case based on hard-won experience with the LOFAR radio telescope, the OpenFlow features available in a number of commercial of-the-shelf network switches are analysed This chapter primarily supports the *optimisation* proposition and to some degree the *value* proposition.

1.5.6 Chapter 7

In Chapter 7 we take a closer look at the energy consumed by receiving large volumes of data. Since radio telescopes are generally characterised high bandwidths of data streamed into a centrally located facility, the energy consumption of the receive component may be significant, and any reduction of this may well be highly desirable. This chapter primarily supports the *optimisation* proposition and to a limited degree the *value* proposition.

1.5.7 Chapter 8

Finally, in Chapter 8 we take a careful look at developments in compute and date transport technology in the near to mid future. While we have been fortunate in the past to be able to rely on continued Moore's law scaling, either by ever increasing clock frequencies, or, more recently, even increasing concurrency, this will shortly cease to be the case. Instead, a whole slew of revolutionary new technologies are being developed to satisfy the insatiable demand for more IT resources. In Chapter 8 we make an initial assessment, using the *value* proposition as our guideline, if and how such technologies can be used in radio astronomy. The assessments made rely heavily on the *co-design* proposition. Furthermore, we can argue that many of the proposed technologies are in fact

special purpose accelerators. This opens optimisation opportunities as recommended by the *optimisation* proposition.

1.5.8 Chapter 9

Finally, in Chapter 9, we will end this thesis with a brief retrospective summary, conclusions and some future work. Here, we look back at the manner in which ech of the propositions we have defined in this chapter are used throughout the thesis. Furthermore, we identify some of the major more detailed contributions each of the chapters have made. A small selection of future work is identified that build upon the work described in this thesis. Conclusions and some discussion end this thesis.

CHAPTER 2

On optimising cost and value in compute systems for radio astronomy

P. Chris Broekema [1], *Verity L. Allan* [2], *Rob V. van Nieuwpoort* [3] *and Henri E. Bal* [4]

Context and Contributions

This chapter was accepted for publication in the January 2020 issue of Astronomy and Computing[5] (available online since November 2019). An earlier version of his work was presented at the 14th IEEE eScience conference in 2018 [6]. Broekema conceived the work, did all research unless otherwise stated and wrote the vast majority of the paper. Allan did the historical research into TITAN and wrote most of the TITAN and SDP case studies in Section 2.7.

This chapter is key to the *value* proposition, providing both the context and the theoretical basis.

[1]ASTRON, the Netherlands Institute for Radio Astronomy
[2]University of Cambridge
[3]University of Amsterdam and Netherlands eScience centre
[4]Vrije Universiteit Amsterdam
[5]https://doi.org/10.1016/j.ascom.2019.100337
[6]https://doi.org/10.1109/eScience.2018.00118

Abstract

Large-scale science instruments, such as the distributed radio telescope LOFAR, show that we are in an era of data-intensive scientific discovery. Such instruments rely critically on significant computing resources, both hardware and software, to do science. Considering limited science budgets, and the small fraction of these that can be dedicated to compute hardware and software, there is a strong and obvious desire for low-cost computing. However, optimising for cost is only part of the equation; the value potential over the lifetime of the solution should also be taken into account. Using a tangible example, compute hardware, we introduce a conceptual model to approximate the lifetime relative science value of such a system. While the introduced model is not intended to result in a numeric value for merit, it does enumerate some components that define this metric. The intent of this chapter is to show how compute system related design and procurement decisions in data-intensive science projects should be weighed and valued. By using both total cost and science value as a driver, the science output per invested Euro is maximised. With a number of case studies, focused on computing applications in radio astronomy past, present and future, we show that the hardware-based analysis can be, and has been, applied more broadly.

2.1 Introduction

Modern large-scale science instruments critically rely on specialised data-intensive computer technologies, to turn instrument data into useful science results. Considering limited science budgets, of which only a small fraction can be dedicated to computing, there is a strong desire to use these expensive systems in an optimal way. The design of such an optimised system is heavily influenced by experience from previous installations. For instance, the design priorities of the GPU-based correlator and beamformer system for the LOFAR radio telescope, in particular its focus on an I/O optimised design, borrowed heavily from previous experiences with Blue Gene based systems [40].

In this chapter we discuss both the cost and value of computing technologies, and how to optimise the combination of these two for maximum science impact. Since these are difficult to measure for the complex combination of hardware, middleware and software that are generally required, we focus our detailed analysis on hardware. We enumerate some of the factors that impact the total cost of a system. However, we propose that total cost over the lifetime of a system is only part of the equation: the computational and scientific performance of different solutions may radically differ for the applications in question, depending on system and application characteristics. A more valuable metric would look at the useful output of a system per invested Euro. For example, the Distributed ASCI Supercomputer (DAS) [18] consortium tracks the effectiveness of its distributed cluster infrastructure via the number of awarded PhDs per cluster generation, as shown in Table 2.1[7]. Considering the nearly constant budget for these systems, between 1.2 and 1.5 M€, discounting inflation, the cost per supported

[7] source: `https://www.cs.vu.nl/das4/phd.shtm`, `https://www.cs.vu.nl/das5/phd.shtml` and historical data

PhD has dropped considerably over the lifetime of the DAS consortium. Alternatively, we can argue that the relative science value per invested Euro has dramatically increased.

	Year	PhDs	€/ PhD	Research agenda
DAS-1	1997	7	€ 214.285	Wide-area computing
DAS-2	2002	22	€ 68.181	Grid computing
DAS-3	2006	36	€ 41.666	Optical grids
DAS-4	2010	33	€ 45.454	Clouds, diversity, green IT
DAS-5	2015	40	€ 37.500	Harnessing diversity & complexity

Table 2.1: Awarded PhDs per Distributed ASCI Supercomputer generation

In this chapter we study a number of cases in radio astronomy, a computationally- and data-intensive science that has been using high-performance computing technologies since the very early days of computing to achieve scientific results. We show how the methodology proposed in this chapter has informally been used in the past. The main contributions in this chapter are:

- the introduction of the concepts *relative science value* and *total value of ownership*, including two potential ways to estimate total value of ownership over the lifetime of a system,
- the introduction of a way to reason about compute system technology beyond just cost,
- a number of case studies that show practical trade-offs between cost and value in radio astronomy.

Although we present a number of equations in this chapter, it is not our intention that these are used to generate a numeric merit value for a particular system or technology. Rather, they are intended to illustrate which components contribute to the cost and merit of a system and as a starting point for a more detailed discussion on the relative value of various compute systems. With these components, and some examples of cost and value past and present in this chapter in mind, system designers and architects have the tools needed to better balance their designs, and evaluate their design choices within this framework.

The intent of this chapter is to show how compute system related design and procurement decisions in data-intensive science projects should be weighed and valued. By using both total cost and science value as a driver, the science output per invested Euro is maximised. While the general concepts discussed in this chapter are known in systems engineering, we hope to introduce them to a broader audience of scientific decision makers, principal investigators, and system architects and designers.

2.2 Compute systems for large-scale science

The study of Physics, in particular Astrophysics, has relied on state-of-the-art computer science and high-performance computing. Modern aperture synthesis radio astronomy in particular was made possible by the development of the Fast Fourier Transform (FFT) [48][8] and computers fast and cheap enough to use them at scale. For example, the One-Mile Telescope, built at the Mullard Radio Astronomy Observatory, Cambridge, in 1964, relied on the computing advances of the EDSAC II and TITAN computers, as is illustrated in our Case Studies in Section 2.7.1. This telescope, and others, like the Half-Mile Telescope at Cambridge and the Westerbork Synthesis Radio Telescope in the Netherlands, depended on the abundant and increasingly cheap computation available to develop the new scientific technique of aperture synthesis, which unlocked new science and ultimately won a Nobel Prize.

More recently, the range of applications that benefit from large-scale computing has increased dramatically with the rise of Data Science, and the ease with which high-performance (if not world-leading) compute infrastructures have become available via Cloud Computing. This chapter is thus presented at a timely moment, to provide decision makers, principal investigators and designers of new compute systems and applications with a framework to help evaluate and guide their design choices.

2.3 On relative science value

In the previous section we have argued that modern data-intensive science relies heavily on computing. Given the high cost of such resources, there is an obvious desire to maximise their usefulness, or minimise their cost. We introduce a system's *Relative science value*, defined as its value per invested Euro over its lifetime, as a measure for the merit of a system over its lifetime. The definition of *value* will be discussed in Section 2.4.

The computational systems supporting modern data-intensive science are often a complex collection of hardware, middleware and software. Quantifying the cost and relative value of such a complex integrated system is nearly impossible. To start our exploration we will focus on one of the more tangible components: hardware.

By first exploring ways to quantify hardware cost and value, we reduce the complexity of the system under investigation without impacting the value of the analysis. In section 2.7 we show that the methodical hardware-based analysis can be applied more broadly, as similar considerations can be used to evaluate other system costs, such as software development, maintenance and power consumption.

The relative usefulness of a hardware system, its relative science value (M_S), depends on its total aggregate value accrued over time (total value of ownership, TVO) and aggregate cost over the lifetime of the system (total cost of ownership, TCO):

$$M_S = \frac{TVO}{TCO} \tag{2.1}$$

[8] Ryle in his Nobel lecture credits Dr. David Wheeler with the invention of the FFT in 1959 [133]

Total Cost of Ownership is a well known concept, both as a tool to inform purchasing decisions in general [62], and in computer science. In this chapter we give our own definition of the Total Cost of Ownership of a system. We introduce the generic concept of Total Value of Ownership in this chapter.

From Equation 2.1 it is obvious that there are two ways to maximise the relative science merit of a compute system: reduce Total Cost of Ownership, or increase Total Value of Ownership of a system. In practice, a carefully considered combination of the two is likely to produce the optimal result. Obviously, total cumulative value TVO is not easy to quantify, and we note that the time over which value is accumulated may extend well beyond the lifetime of the system. In the next sections we will explore the components that make up TVO and TCO.

2.4 Total Value of Ownership

Whereas the concept of Total Cost of Ownership is well known and established, the same can not be said for its value counterpart; we shall therefore introduce this first. In economic terms, we are interested in the return on investment, which we'll refer to as Total Value of Ownership (TVO) in this chapter, to contrast to Total Cost of Ownership. While this is an essential question to ask during the definition phase of a project, the answer is seldom easy to quantify. The success of science projects is generally measured in the importance of its scientific results, often expressed in the number of published peer-reviewed papers produced. However, from a system design perspective, it is attractive to use a more easily measured metric, such as compute power, throughput or storage capacity, to describe the value of a system. While such metrics are convenient and may be useful in their own right, we argue that these do not necessarily provide an accurate reflection of how the system will be used. Furthermore, these do not necessarily take computational efficiency, scientific impact, or average required capacity per accepted paper into account. In this section we propose two measures for a system's TVO that are designed to more accurately reflect the actual scientific usefulness of a system: total lifetime computational value (V_c), and total lifetime scientific value (V_s). While we provide equations, these are not designed to be used to model TVO; but rather to capture the relationship between some of the various elements that define system value.

Total performance, computational or otherwise, of a system can be a useful measure for the value of a (hardware) system. However, even this can be difficult to quantify beforehand. Whereas peak computational performance is relatively easy to determine, often only a small fraction of this can be achieved in practice. The same can be said for other metrics like peak network and storage performance. The fraction of the computational resources that can effectively be used by an application is determined by its computational efficiency. A discussion on the factors that impact computational efficiency is beyond the scope of this chapter, but we note that these factors should be foremost in the mind of a hardware system architect. To illustrate this point, we look at the yearly Top500 list of the fastest supercomputer in the world for the HPL benchmark [9]. Computational efficiencies of these systems, shown in Figure 2.1, range from 15.6% to

[9] www.top500.org

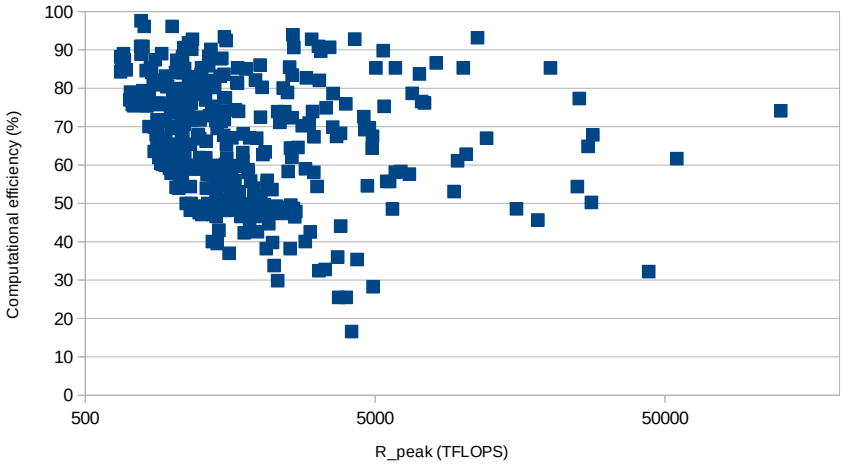


Figure 2.1: HPL computational efficiency in the Top500 (November 2017)

97.6%, which shows that the impact of unexpectedly low computational efficiency may be catastrophic.

By taking into account the target applications for a specific system, we introduce an estimate for its total lifetime computational value (V_c, in FLOP), as shown in Equation 2.2.

$$V_c = T_l \; A_o \sum_{p=0}^{P} \Big(f_p \; R_{max,p} \Big), \text{ with} \sum_{p=0}^{P} f_p \leq 1 \tag{2.2}$$

Here, we take the total lifetime of the system, T_l, and its availability as a fraction of total lifetime, operational availability (A_o), to get the effective time the system is usefully available over its lifetime. For each application p, its maximum achieved performance on the target system ($R_{max,p}$), and the fraction of operational time it is expected to be run (f_p) are taken to get a value for the average maximum achieved performance over all applications to be run on the system. Combined, these two components make up the system's total lifetime computational value.

Equation 2.2 thus takes computational efficiency into account over all target applications, and considers both system lifetime and operational availability. Similar analyses could be done for other performance metrics, such as network bandwidth and storage system performance. We do not measure average performance of applications, rather we determine the total effective performance over the lifetime of the system. However, the eventual goal of a compute system is not the delivery of capacity per se, but rather to facilitate science. A discussion on appropriate metrics for scientific output is well out of scope for this chapter, for the purpose of the discussion in this chapter we use *scientific publication* as a placeholder. This is most easily measured in peer-reviewed journal or

conference publications; however, one may also consider monographs and PhD theses, or even awards (see section 2.7.1).

To illustrate these points, we introduce a system's *Total lifetime scientific value*, which in our example is based on its previously introduced total lifetime computational value. Since not all science requires the same amount of resources, processing power or other, per scientific publication, we add the *average computational resource required per scientific publication*. An appropriate *impact factor*[10], which is not necessarily the same as a journal impact factor, may be added to differentiate potential Nobel prize winning research from more generic projects. Notably, this impact factor may be highly time sensitive, in the sense that ground-breaking projects generally have very high impact factors (see section 2.7.2 for an example). We note that these two factors may be subjective, highly sensitive, and may have significant political implications.

$$V_s = T_l \; A_o \sum_{p=0}^{P} \left(f_p \; \frac{R_{max,p}}{C_{cpp,p}} \; I_p \right), \text{with} \sum_{p=0}^{P} f_p \leq 1 \tag{2.3}$$

Total lifetime scientific value V_s is defined in Equation 2.3 by the maximum achieved performance of the application associated with science case p on the system under investigation $R_{max,p}$, divided by the average amount of resources required per scientific publication for that science case $C_{cpp,p}$. This results in the number of scientific publications per unit of time for that science case and system. Multiplied by some impact factor per science case, I_p, and summed over all science cases targeted by the system P and normalised using the fraction of time each application is expected to consume (f_p), gives us a measure for scientific impact per measure of time for that system. Multiplying that by the total lifetime of the system T_l and the fraction of that time the system is actually available (operational availability A_o) gives us the total scientific value of a system, in a unitless *scientific impact*. For convenience we use computational resources, in floating point operations (FLOP), as a measure for resources required per scientific publication in this model, but other metrics (such as bandwidth, storage capacity, etc), or a combination of such metrics, may be used instead.

The two value measures introduced in this section are by no means the only ones that can be defined. They are intended to start the discussion and offer an initial indication of the processes and thinking involved. Characterising the performance of a compute system in a single number is notoriously difficult, which has been studied in some detail. Previous work suggested the use of harmonic means of runtime of a number representative benchmarks to express the useful performance of a computer [145], which expresses performance in terms of the total runtime of a set of benchmarks. While benchmarks certainly have their place, and runtime is an appropriate measure for performance of a system, this is not necessarily equivalent to value. However, we note that the V_c is equivalent to the weighted harmonic mean suggested by Smith et al. multiplied by $T_l \; A_o$. Essentially, instead of computing average performance, we focus on aggregate performance over the effective lifetime of the system, taking system lifetime and operational availability into account.

[10] We are aware that constructing a useful impact factor has many pitfalls. See [87] for a discussion about journal impact factors as an example.

2.5 Total Cost of Ownership

Having looked at various ways to define the value potential of a system, we now turn to more familiar ground: cost. The aggregate cost of a system over its lifetime is usually referred to as its Total Cost of Ownership. While the definition of TCO is relatively easy to give, calculating it a priori may not be as simple, in particular in large-scale science installations. The lifetime of a particular system may be unpredictable, and the often non-conventional use of such systems may lead to unexpectedly large operational costs. Furthermore, complex and highly integrated systems make for difficult deployment and integration, which is hard to plan and budget for. Having said that, TCO can be defined as a combination of capital investment (C_{cap}), engineering cost (C_{eng}, often called non-recurring expense, or NRE), installation, deployment and integration cost (C_{int}), development cost (C_{dev}), recurring operational cost (C_{ops}) over the lifetime of the system (T_l) and miscellaneous costs not covered elsewhere C_{misc}, as shown in Equation 2.4.

$$TCO = C_{cap} + C_{eng} + C_{int} + C_{dev} + \sum_{t=0}^{T_l} C_{ops} + C_{misc} \tag{2.4}$$

The one time investment to acquire a system is referred to as its capital cost, C_{cap}. This includes all readily available hardware required to install and commission the system. Capital cost is usually either capped, or relatively easy to estimate. We note, however, that even capital cost becomes highly uncertain when predicted several years in advance, due to fast moving markets and uncertain performance characteristics and pricing of newly developed components. Models often resort to extrapolation from existing systems using some form of Moore's law scaling to estimate future cost and performance (see for instance the SDP costing for the SKA telescope [12]). While this has historically been somewhat accurate, the demise of Dennard scaling [57] around 2005 has made modelling much more complicated. This uncertainty is exacerbated by an erratic market that is increasingly dominated by single players without significant competition.

When a system requires engineering investment in order to be usefully employed, this is engineering cost, C_{eng}. This may involve custom cooling solutions, or other non-standard equipment specific to the system (see for an example the LOFAR GPU-based correlator and beamformer [40]). Costs associated with certification of a custom solution may also be considered engineering cost. General purpose systems generally have no or very little engineering cost, but in more specialised systems this may be a significant cost driver.

Any investment needed to integrate and commission the system into an existing infrastructure is captured in integration and commissioning cost, C_{int}. Note that in software systems, especially if the source code of this software is available, integration, commissioning and development may be closely related.

It is unlikely that the application software of a science instrument or experiment remains static over the lifetime of the instrument. Part of the software evolution will be to add additional functionality or implement advances in algorithmic or scientific understanding of the problem. Another part of this development will be to adapt existing

code to run (efficiently) on a newly installed platform. The cost of this particular development effort is the development costs of that system (C_{dev}). Such costs may be small (e.g., porting code to a newer system with the same or a similar architecture), or very large, for example, porting functionality from a CPU cluster to a GPU-based system, as was done for LOFAR correlator [40]. These costs may be difficult to predict during the design phase of a long-lived instrument, which, in the LOFAR case, was a decade earlier. It is likely that not all development effort is expended before the system is deployed, and C_{dev} may extend significantly into the lifetime of the system. Furthermore, added development effort may have a significant impact on computational efficiency, with a corresponding effect on Total Value of Ownership. There is a direct coupling between the development costs, and thus Total Cost of Ownership, and Total Value of Ownership. Conversely, if a system performs well enough, there is no need to expend more development effort to improve performance, unless this opens opportunities for, for instance, additional science cases.

Whereas all previously mentioned costs, with the exception of Development costs, are expended before the system becomes operational, Operational cost (C_{ops}) is a recurring line-item during the lifetime of the system. This includes costs associated with energy consumed, infrastructure cost (i.e. rack space, network connectivity, both physical links and bandwidth, heat dissipation, etc), maintenance and system administration. We have simplified our model by using a single operational cost component; reality is often more complex, especially in a hosted environment where the components mentioned above are provided by different entities or organisations. While we have opted to keep operational cost in our model flat over the lifetime of the system, this is again a simplification. Operational cost in the initial phase of the system may be higher both due to early failure of hardware and staff unfamiliarity and training. Near the end of the operational lifespan of the system, often after four or five years in general purpose computing, an increase in hardware failures may be observed, which may increase operational cost, depending on the chosen service model. Furthermore, operational cost may depend on inherently volatile pricing of, for instance, electricity. Energy costs are often estimated using the previously mentioned extrapolation using Moore's law scaling, while staffing levels and costs may be based on industry standard fractions of FTE per rack or PetaByte [69].

Finally, staff costs not included in the components above, such as those required to secure funding, acquire the system (e.g. writing tender documentation and evaluating responses) and to decommission the system after its useful lifetime, as well project management and support other than system administration, are included in miscellaneous cost (C_{misc}).

The remainder of this chapter takes the concepts introduced, and shows, using artificial and real-world case studies taken from radio astronomy past and present, the value of this structured approach to compute system design.

2.6 A synthetic instructive example

In the previous sections we identified a metric that we can optimise for: total relative science value as defined in Equation 2.1, M_s, but its definition is (deliberately) ambigu-

ous. While it is not our intention to advocate numeric values for the total relative science values for eScience technologies, we can use the equations introduced above to identify ways to optimise their usefulness.

In this section we illustrate the value of the proposed methodology using a thought experiment. We have constructed an example that is obviously manipulated to show the desired results. However, using this example we show that, depending on the value measure selected, any of the offered solutions can be judged superior to the others.

Table 2.2 describes hypothetical responses to a hypothetical request for tender for the replacement of key computer hardware. A set of ten key applications was identified that cover the lifetime of this system, and performance of each system for these applications was measured, as shown in Table 2.3.

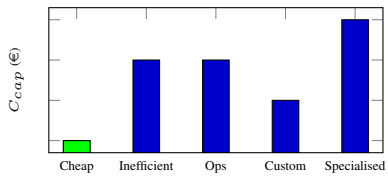
	Cheap	Inefficient	Ops	Custom	Specialized
C_{cap} (€)	250.000	350.000	350.000	300.000	400.000
C_{eng} (€)	-	-	25.000	-	25.000
C_{int} (€)	25.000	-	25.000	-	25.000
C_{dev} (€)	750.000	600.000	1.250.000	1.250.000	1.000.000
C_{ops}/yr (€)	50.000	25.000	75.000	25.000	25.000
C_{misc} (€)	25.000	25.000	25.000	25.000	25.000
T_l (yr)	5	5	5	5	5
A_o	0,9	0,95	0,85	0,95	0,95
TCO (€)	1.300.000	1.100.000	2.050.000	1.700.000	1.600.000

Table 2.2: The offered solutions, with detailed cost, lifetime and availability information

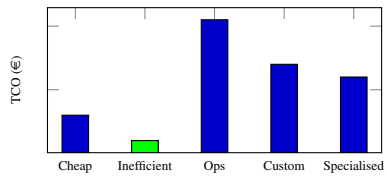
Each of these offers were evaluated using the model introduced in this chapter, the results of which are shown in Figure 2.2. For each value measure, the superior solu-

	f_p	C_{cpp}	I_p	Cheap	Inefficient	Ops	Custom	Specialised
A	0,04	$1 \cdot 10^4$	5	$2 \cdot 10^8$	$1 \cdot 10^8$	$5 \cdot 10^8$	$4 \cdot 10^8$	$2,5 \cdot 10^8$
B	0,08	$1 \cdot 10^4$	5	$2 \cdot 10^8$	$1 \cdot 10^8$	$5 \cdot 10^8$	$4 \cdot 10^8$	$2,5 \cdot 10^8$
C	0,02	$1 \cdot 10^4$	5	$2 \cdot 10^8$	$1 \cdot 10^8$	$5 \cdot 10^8$	$4 \cdot 10^8$	$2,5 \cdot 10^8$
D	0,02	$1 \cdot 10^4$	5	$2 \cdot 10^8$	$1 \cdot 10^8$	$5 \cdot 10^8$	$4 \cdot 10^8$	$2,5 \cdot 10^8$
E	0,40	$1 \cdot 10^4$	5	$2 \cdot 10^8$	$1 \cdot 10^8$	$5 \cdot 10^8$	$4 \cdot 10^8$	$2,5 \cdot 10^8$
F	0,11	$1 \cdot 10^4$	5	$2 \cdot 10^8$	$1 \cdot 10^8$	$5 \cdot 10^8$	$4 \cdot 10^8$	$2,5 \cdot 10^8$
G	0,07	$1 \cdot 10^4$	5	$2 \cdot 10^8$	$1 \cdot 10^8$	$5 \cdot 10^8$	$4 \cdot 10^8$	$2,5 \cdot 10^8$
H	0,08	$1 \cdot 10^4$	5	$2 \cdot 10^8$	$1 \cdot 10^8$	$5 \cdot 10^8$	$4 \cdot 10^8$	$2,5 \cdot 10^8$
I	0,02	$1 \cdot 10^4$	100	$2 \cdot 10^8$	$1 \cdot 10^8$	$5 \cdot 10^8$	$4 \cdot 10^8$	$10 \cdot 10^8$
J	0,16	$1 \cdot 10^4$	5	$2 \cdot 10^8$	$1 \cdot 10^8$	$5 \cdot 10^8$	$4 \cdot 10^8$	$2,5 \cdot 10^8$

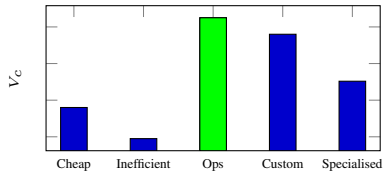
Table 2.3: Application characteristics and performance per offered solution



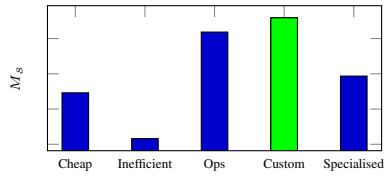
(a) Capital investment (C_{cap}, lower is better)



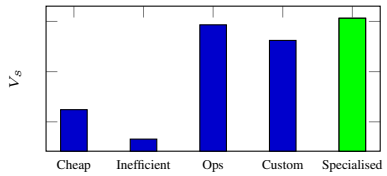
(b) Total cost of ownership (TCO, lower is better)



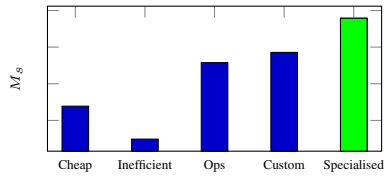
(c) Total lifetime computational value (V_c, higher is better)



(d) Relative science value ($M_s = \frac{V_c}{TCO}$, higher is better)



(e) Total lifetime scientific value (V_s, higher is better)



(f) Relative science value ($M_s = \frac{V_s}{TCO}$, higher is better)

Figure 2.2: The offers evaluated against six cost and value measures. The superior offers for each measure are shown in green.

tion is shown in green[11]. While the offers are fictional and the use-case is obviously constructed, it is clear that, depending on the chosen selection criterion, a different solution wins, highlighting both the power and importance of the concept introduced in this chapter. More importantly, this example shows the dangers of selecting the wrong value measure for convenience or not carefully considering all possible components that make up the selected value measure.

In section 2.1 we postulate that the useful (scientific) output of the system per invested Euro is the most useful value metric of a system. Not using such a metric, and instead focusing solely on total cost of ownership, would, in this example, lead to the selection of the far inferior *Inefficient* solution.

[11]All underlying data and analysis used in this chapter are available here: `https://doi.org/10.5281/zenodo.2270842`.

2.7 Case studies

To further illustrate the value of the conceptual model introduced in this chapter, three radio astronomy use cases will be discussed: the use of the TITAN computer for one of the first operational radio interferometers, the LOFAR array and the SKA Science Data Processor. We also highlight the variability of value over the lifetime of a compute system using the performance impact of a recent hardware vulnerability as an example.

2.7.1 The TITAN Computer and the Mullard Radio Astronomy Observatory

The One-Mile Telescope, sited at the Mullard Radio Astronomy Observatory (MRAO) near Cambridge, was an early aperture synthesis telescope, and was the first designed to use the Earth rotation aperture synthesis technique. It was conceived when the EDSAC II computer at Cambridge University was in operation, and was completed in 1964, as the TITAN computer came online. TITAN was then used by the One-Mile, the Half-Mile and Interplanetary Scintillation Array (IPSA) telescopes, until TITAN was decommissioned in 1973. The One-Mile was explicitly designed to used the improved computing resources provided by TITAN, first to provide the control tapes for the telescope and then using the Fast Fourier Transform (FFT) to power the data analysis [63], [131]. As Wilkes recalls:

> One day, Ryle came to me to say that he was planning the erection of a much larger telescope and to ask whether the Mathematical Laboratory could undertake to provide the computing support required.[166], p.193

TITAN was a ground-breaking computer itself, with hardware procured from the Ferranti company, and software developed by staff at the University of Cambridge, mostly from the Mathematical Laboratory. The Mathematical Laboratory was, unusually for the time, already running as an effective computing service, where users applied for time with their projects (as is common with HPC resources today)[101], [166]. This differed from companies such as Ferranti and IBM, which were producing computers for the commercial market, and other universities, which were producing computers primarily as a way of investigating computers themselves (the purpose of instruments such as the Manchester Baby and CSIRAC), without explicit support for scientific research [50].

Although the University wished to buy a new computer to replace EDSAC II, it did not have a large capital budget available. Thus they bought a heavily-discounted Ferranti Atlas (usual cost £ 2 million; price actually paid: £ 350,000 (approximately £ 6-7 million today [102]) with an additional £ 75,000 for a large disk store [8]). However, the University now had to spend a lot of money on salaries to develop the software, but this cost was not explicitly tracked by the University, and their decisions were made purely on the C_{cap}.

The performance of TITAN, combined with David Wheeler's FFT algorithm [133], allowed TITAN to do the calculations necessary for the first Earth-rotation aperture synthesis observations with the One-Mile telescope, and then to produce the first maps

of the radio sky [131]. It was also used to support IPSA, which was used by Dame Jocelyn Bell Burnell to discover the first pulsar.

These scientific breakthroughs, backed by TITAN, won Tony Hewish and Sir Martin Ryle their joint Nobel prize for their innovative telescope design work [132]. Furthermore, at least 30 PhD theses using the One-Mile, and the subsequent Half-Mile and IPSA, used TITAN-processed data, or used TITAN for theoretical modelling.[12] It is unfortunately not possible to track all the papers that were produced with TITAN, as it was not comprehensively tracked at the time and not all authors note their use of TITAN. Therefore there are aspects of TITAN's value that are not captured.

Nevertheless, TITAN delivered exceptional TVO extending well beyond its lifetime. It was not only used in radio astronomy, although radio astronomy made unique use of its capabilities, but also in computer science (applications included one-way functions for storing passwords, timesharing systems, computer language research, early version control systems [8]), crystallography (another field that used the FFT), statistics, Computer Aided Design, agronomy, and quantitative economic methods (for which one TITAN user, Sir Richard Stone, won the Nobel Prize for Economics) [146][13]) [103].

Many of the people who designed, programmed for, and used, TITAN were or became leaders of their fields, bringing rewards (both financial and reputational) to their institutions in the subsequent decades; thus TITAN provided a TVO that far outweighed cost of purchasing, developing, and running the system. There is a significant "long tail" to TITAN's value, exemplified by Dame Jocelyn Bell Burnell's receipt of the Royal Society Royal Medal in 2015, and the Special Breakthrough Prize in Physics (2018), both of which specifically cite her work on pulsars. To illustrate the disparity between the lifetime of the TITAN system and its value, we have plotted major prizes won by TITAN users in radio astronomy, as compared to TITAN's lifespan, in Figure 2.3.

Awards were not confined to the radio astronomy community. Eleven TITAN users have been elected Members of the Royal Society.[14] TITAN users have won many other major awards in their fields, including: the Royal Statistical Society Guy Medal in Silver [15], the Gold Medal of the Royal Astronomical Society[16], the London Mathematical Society De Morgan Medal [17], the Faraday Medal [18], the IEEE John von Neumann medal [19] and the Karl G. Jansky Lectureship, which "is an honor established by the trustees of Associated Universities, Inc., to recognize outstanding contributions to the advance-

[12] One of us (VA) checked the archived PhD theses of the Astrophysics Group, Cavendish Laboratory, University of Cambridge. TITAN may have been used for PhD theses in other groups; however, it is difficult to locate all of these 40 years later.

[13] A copy of this work is held in the Library of the Department of Computer Science and Technology, University of Cambridge, classmark V75-14.

[14] Frank Yates, Donald Lynden-Bell, David George Kendall, Maurice Wilkes, Sir Martin Ryle, Peter Swinnerton-Dyer, Malcolm Longair, Brian Pippard, Roger Needham, John Baldwin and Dame Jocelyn Bell Burnell [129]. "The Royal Society is a Fellowship of many of the world's most eminent scientists and is the oldest scientific academy in continuous existence", and members must have made "a substantial contribution to the improvement of natural knowledge, including mathematics, engineering science and medical science" [127, 128].

[15] Won by Georgle Kendall and MJR Healy

[16] Won by Donald Lynden-Bell, Sir Martin Ryle, and Jeremiah P. Ostriker

[17] Won by D. G. Kendall

[18] Won by Ryle, Maurice Wilkes, and Roger Needham

[19] Won by Wilkes

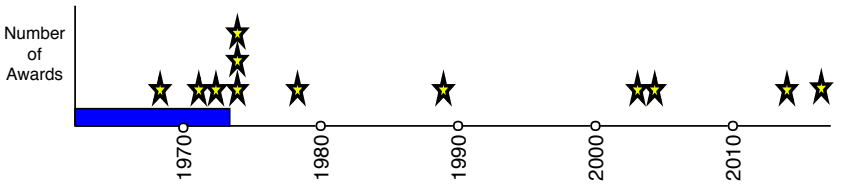


Figure 2.3: Awards given to TITAN radio astronomy users over time. The blue bar indicates when TITAN was active.

ment of radio astronomy" [20]. The precise role of TITAN in these awards is difficult to quantify; however, having TITAN available clearly provided important support and enablement for people at all stages of their careers — precisely the purpose of a scientific computing resource. Moreover, this lists only the very highest achievers amongst TITAN users; there are shallower network effects from the existence of TITAN, which are even harder to account for, but which indicate that resources such as TITAN are vital for the scientific community. Although a significant investment, both capital, as well as engineering and development, was required, TITAN's ten-year lifespan and high-impact and long-lasting contributions make its relative science value exceptional. Even if the full list price of the hardware had been paid, the capital outlay would still have been justified by its scientific success, which far outshone other contemporaneous systems.

2.7.2 LOFAR

LOFAR, the LOw Frequency ARray [158], is a modern low-frequency large-scale distributed radio telescope in the Netherlands, with international stations in various European countries. The concept and design of the LOFAR telescope, which started in the late 1990s, is a study in trading off value and cost. A number of early papers discussing the telescope concept and initial design [29, 30], as well as some retrospective analysis of the design considerations [32], make this a particularly interesting instrument to study.

As discussed above, modern radio interferometry was made possible by the availability of abundant and affordable compute resources. In LOFAR, this concept is taken even further by replacing a small number of large parabolic reflectors with many simple, cheap and omni-directional dipole antennas and software-based digital beamforming. Essentially, many simple antennas are combined, by coherent addition, into a single virtual receiver. Early design concepts for this low-frequency array that could act both as a technology demonstrator for the future Square Kilometre Array, as well as scientifically open a relatively unexplored frequency range, identify a "processing window of opportunity". This early concept predicted that, while computational cost for the processing required for this low-frequency array was at the time infeasibly large, it would become affordable, assuming Moore's law continued to apply, after 2003.

[20] Awarded to Bernie Fanaroff and Dame Jocelyn Bell Burnell, both of whom used TITAN during their PhDs.

In further work, instrument sensitivity was defined as the key value parameter (and thus a measure for the TVO of the instrument) for the design trade-offs in this instrument [31], although other value measures such as survey speed and resolution were also taken into account. In order to achieve optimal performance over cost, all main constituents of the complete LOFAR system were designed to have a similar marginal performance over cost ratio.

This analysis shows that both TVO and TCO for the LOFAR telescope in general, and the digital processing systems in particular, were carefully considered early on in the conceptual design phase of the instrument. A clear choice was made to use sensitivity over other technical or scientific metrics, such as survey speed or resolution, as a measure for the total value of the instrument. We note that this implicitly assumes this technical measure translates to scientific value. Regardless of this technical measure, suitability for a small number of key science projects was also a key design consideration in the development of LOFAR. Furthermore, the cost of the digital processing system was analysed, and, more importantly, judged to become affordable at some point in the mid future. This realisation allowed development of the instrument, and its associated software infrastructure, to start before the required compute capacity became financially feasible.

Since its opening in June 2010, one measure of LOFAR's science value, the number of peer-reviewed scientific publications using LOFAR produced data, has been monitored [21]. A different way to express the value, or in this case more accurately the return on investment of a science instrument, is to evaluate how much of the invested money is reinvested in the local (national) economy. A Dutch research institute that specialises in research on the impact of science, Rathenau, recently studied the LOFAR telescope and the Dutch contribution to three other major science instruments: CERN [22], ESRF [23] and ITER [24]. They defined a return coefficient (R) as the capital reinvested in the national economy, divided by the Dutch contribution in that instrument. The results, published in Dutch [148], are summarised in table 2.4. While it is difficult to compare four completely different instruments, this work shows that the financial return of the LOFAR telescope for the Dutch economy has been excellent. This is due to the fact that most, if not all, of the IP was developed in the Netherlands, and therefore production of those components, even for international stations, is likely to occur there as well.

The hierarchical and modular nature of the LOFAR system has allowed several dedicated systems to be added to the telescope to increase its scientific value at modest cost. While some, like Dragnet (described in section 2.7.2), were just plug-in systems that required little to no additional engineering to add to LOFAR, others, like AARTFAAC (see section 2.7.2), require raw antenna data not available in standard LOFAR observation modes. We will explore the cost and value considerations of some of these components in the following sections.

[21] https://old.astron.nl/radio-observatory/lofar-science/lofar-papers/lofar-papers

[22] https://home.cern/

[23] https://www.esrf.eu/

[24] https://www.iter.org/

	CERN	ESRF	ITER		LOFAR	
	average per year	average per year	2008-2017 construction incl. grants	2008-2015 construction grants only	2004-2013 construction	2014-2017 operations per year
Total investment	1,104M CHF	90M €	6,120M €	4,581M €	92M €	4.3M €
Total Dutch investment	50,9M CHF	2.7M €	161M €	120M €	81.2M €	3.4M €
Dutch contribution	4.61%	2.97%	2.63%	2.63%	88.3%	77.8%
Total expenditure	343M CHF	57M €	4,330M €	101M €	92 €	4.3M €
To the Netherlands	8.7M CHF	0.58M €	7.9M €	4.3M €	89.2 €	4.1M €
Dutch ROI	2.54%	1.01%	0.07%	4.18%	97%	96.5%
Return coefficient	0.55	0.34	0.07	1.59	1.10	1.24

Table 2.4: Return coefficients for the Dutch economy for four large scale science infrastructure projects (source, Rathenau institute [148])

LOFAR correlator and beamformer systems

A key signal processing component of the instrument, the LOFAR correlator and beamformer, and specifically its hardware evolution, is relatively well described. This part of the instrument is algorithmically simple and the required functionality is fairly constant. Therefore, for this specific example, cost (with all its different components), operational availability, and lifespan mostly determine the relative science value of the correlator and beamformer. Early concepts for the LOFAR central processor show a 1600 node hybrid cluster compute system that uses conventional processors and dataflow co-processors to process the data [53, 156]. While feasible, the considerable size of this compute concept meant that a bespoke supercomputer was a viable and, more importantly, cost-effective alternative. In 2003, an IBM Blue Gene/L, briefly the fastest supercomputer in Europe, was installed as the central correlator and beamformer for LOFAR [122]. This was upgraded to a much smaller, IBM Blue Gene/P in 2008 [123], that was not only more powerful, but also considerably more energy efficient. Whereas the total lifetime computational and scientific value of this new system was similar, its reduced operational costs, as well as improved software environment made its relative science value considerably higher than the previous Blue Gene/L. However, supercomputers are inherently expensive, so research into more cost-effective solutions continued [157, 160]. This eventually resulted in the procurement and commissioning of a much smaller and more affordable GPU-based correlator and beamformer platform, Cobalt [40]. A more capable second generation of this system, Cobalt 2.0, started operations in 2019 [25].

The timeline of the LOFAR correlator and the construction of the instrument as a whole is interesting to study. As mentioned above, the telescope was opened in 2010 and at that time the initial Blue Gene/L correlator and beamformer has already been replaced. While it is not accurate to say Blue Gene/L was never used in production as the LOFAR correlator and beamformer, it is clear that it was procured and installed early. Arguably, its cost was considerable (although the actual investment was never made public), and its value limited. However, the strategic alliance and collaboration agree-

[25] https://old.astron.nl/cobalt20-sets-stage-fully-multitasking-lofar

ment between ASTRON and IBM was an important consideration in securing sufficient construction funding for LOFAR. Furthermore, spare computational capacity was made available to other scientific users. Therefore, while the total lifetime scientific value of the Blue Gene/L correlator for the LOFAR telescope was low, its general value for the LOFAR telescope was extremely high and its total lifetime scientific value for the wider community was comparable to other high-performance computing systems. Nevertheless, the Blue Gene/L system was never used to its full potential in the LOFAR telescope, and even the Blue Gene/P system was significantly under-utilised for most of its lifetime. These systems did however provide extremely valuable experience that was essential to the success of Cobalt and was used to great effect in the hardware design of the SKA Science Data Processor. Whether this was worth the significant initial capital investment is beyond the scope of this chapter.

When the LOFAR Blue Gene/P was nearing the end of its service life, a feasibility study into possible upgrades was undertaken [76]. Four drop-in replacement options (Blue Gene/Q, an FPGA-based Uniboard system, a CPU-based cluster with GPU accelerators, and a CPU-based cluster) were evaluated for risk, development effort, cost, power consumption and scalability. It is clear from these selected criteria that various cost components were carefully considered, while value was expected to be equal among the contenders considering any new system was expected to replicated the functionality of the existing Blue Gene/P based correlator and beamformer. A cluster with GPU accelerators was judged to be the most cost-effective solution, based on low cost and power consumption, good scalability, and relatively little development effort required. By extension this was therefore also the option with the highest relative science value, and selected for implementation as the Cobalt correlator and beamformer [40].

AARTFAAC

While the LOFAR correlator and beamformer described above are integral parts of the original LOFAR design, AARTFAAC is an add-on system that was added to increase functionality and enable additional science cases. The Amsterdam–ASTRON Radio Transients Facility and Analysis Center (AARTFAAC) system [114] is a real-time all-sky transient detection system. Data from a subset of LOFAR antennas is duplicated during normal LOFAR operations and processed independently into all-sky images of the low-frequency radio sky that can subsequently be monitored for bright transient events. This is a significant advance over the capabilities of the original LOFAR system, which was only possible due to investments made early in the LOFAR project to over-provision both the bandwidth of the LOFAR station ring network and the LOFAR Wide Area Network. For this specific addition, a custom shim was added the station data transport ring: Uniboard-RSP Interface boards. These duplicate raw antenna data, normally beamformed in RSP boards, to AARTFAAC Uniboards. Furthermore, the FPGA firmware on the LOFAR core stations that take part in the AARTFAAC system had to be modified to generate the additional AARTFAAC packets. Data from the AARTFAAC system is transported to dedicated processing nodes located in the same central processing facility as the LOFAR correlator and beamformer, sharing spare network capacity.

While AARTFAAC adds undoubtable value to the LOFAR telescope, its addition required significant additional engineering and manufacturing. In particular the additional

firmware programming requires the use of scarce resources that are generally overcommitted. We will not discuss the relative merits of this addition over others, or whether the investment was valuable or not. However, we do note that additional investment in the development of data spigots at the LOFAR station during construction would have made the development of AARTFAAC much cheaper and easier. This was considered during design, but technology had not progressed sufficiently; the additional cost would have been significant and the idea was shelved.

DRAGNET

Whereas AARTFAAC is a real-time transient monitor that operates in UV-space, the DRAGNET cluster [19] is a non-real-time pulsar and transient search system that operates in the time domain. It takes beamformed data from the Cobalt correlator and beamformer and uses blind coherent de-despersion to identify fast transients and millisecond pulsars. This system has demonstrated its value by the discovery of the second fastest-spinning pulsar to date, and one of the first at such low observing frequencies [20].

The DRAGNET system consists of 23 nodes, each of which has 4 NVIDIA Titan X GPUs that provide the bulk of the processing capacity. Its source data is produced by the LOFAR Correlator and Beamformer, Cobalt. Data is stored locally and processed non-real-time, resulting in a pulsar and/or transient candidate list for further analysis.

Since DRAGNET uses a standard LOFAR data product as input, only limited modifications were necessary to integrate the system into the LOFAR telescope. The only major investment, apart from the cluster and dedicated software for DRAGNET itself, was the integration of the system into the LOFAR monitoring and control system. DRAGNET has added significant capability to the LOFAR telescope: the ability to search for extremely fast-spinning pulsars, and a way to detect fast transient events that would be missed by the original LOFAR telescope. This adds significant additional value to the instrument, since it allows new science cases to be explored. The majority of the additional investment was in the actual cluster and the software development needed to process the data, with limited investment needed to modify the existing system.

International LOFAR stations

International LOFAR stations are not just valuable parts of the International LOFAR Telescope (ILT), these can also operate in *local* or *standalone* mode. In this mode, station data is not sent to the central LOFAR correlator and beamformer, but instead redirected to a local system and can thus act as a fully functional telescope in its own right. The comparatively small size of these stations, and the low observation frequency, make them relatively unsuited for imaging observations, so most effort has gone into local transient and pulsar search. The ARTEMIS backend [16] was developed as a real-time GPU accelerated suite of software to search for these events in data from modern radio telescopes. Four international stations are equipped with such systems [141].

Changing an international LOFAR station to stand-alone mode is, from a high level, as easy as changing destination IP number and MAC address of the receiving nodes. The ability to use these international stations in this mode can be partly attributed to the extensive use of standardised protocols and interfaces, as well as the modular nature

of the LOFAR telescope. This means that LOFAR is potentially a large collection of independent instruments.

One international station, the French station near Nançay, differs significantly from any other antenna field in the LOFAR instrument. Apart from the low- and high-band antennas as in every international station, an unused third analog data path in the LOFAR station hardware is used to add a cluster of 96 mini-arrays, each of which consists of 19 antennas sensitive from 10 to 87 MHz [170]. The resulting giant extension of LOFAR, NenuFAR, while not as large as the LOFAR telescope, adds a similar number of low-band antennas to the instrument as all other stations combined (1938 vs $\sim$2700). In stand-alone mode, NenuFAR, currently under construction and accepting early science proposals[26], intends to support a wide range of data products, very similar to those produced by the LOFAR telescope. This shows that NenuFAR is a powerful instrument itself, especially for pulsar and radio transient science. A dedicated correlator and beamformer, based on the newly commissioned Cobalt 2.0 correlator and beamformer, is currently being installed.

This extension to the French international station was made possible by the availability of an unused analog data path in the LOFAR station hardware. This data path was intended for a third receiver type, eliminated early in the design process for cost reasons. In Dutch LOFAR stations this data path is used to connect half of the low-band antennas.

Finally, a LOFAR station was constructed in Lapland, Finland, near Kilpisjärvi, well above the arctic circle [95]. This station, KAIRA, is not part of the International LOFAR Telescope (ILT) and not connected to the rest of the LOFAR network. Instead it is used exclusively in stand-alone mode, primarily for atmospheric imaging using reflected transmissions from a number of remote radar sites. Experiments have shown fringes on recorded data between it and the international LOFAR station at Effelsberg in Germany, proving that for exceptional experiments it is possible to add the station to the LOFAR array, albeit not in real-time.

Retrospective

The LOFAR concept design identified a period in time where the relatively high impact of ground-breaking radio astronomical research in a relatively unexplored frequency range, combined with dropping costs for computing, would result in an instrument with optimal relative science value. During its design and operational lifetime, the LOFAR correlator and beamformer in particular has benefited from continued development of cost-optimised solutions to improve the relative science value of an already successful and cost-effective instrument. The modular nature of the LOFAR telescope enabled the addition of additional systems to the instrument, further increasing its science value.

We note that the cost and value analysis of these additional systems was not as rigorous as that done for the original LOFAR system. While the engineering challenges of such add-on system were generally considered, the operational impact was often underestimated and (un)availability of critical development resources lead to significant slippage in project schedules. Within ASTRON this has led to a more formal and structured

[26] https://nenufar.obs-nancay.fr/en/astronomer/

application process for funding and the adoption of more rigorous systems engineering practices. For the second phase of LOFAR we are considering the establishment of a LOFAR architecture team to centralise and formalise the responsibility for the considerations on cost and value for the instrument. Modern distributed radio telescopes are, due to their inherent modular nature, exceptionally adaptable and extendable. Taking possible extensions into account during the design of a new instrument will make the addition of such extensions easier and thus cheaper.

When looking at radio telescope systems as a whole, instead of just the compute systems they rely on, scientific value is the better understood factor while the sum of all costs is often not fully appreciated. This is, at least to some degree, a result of the funding model for scientific instruments. Funding proposals are evaluated on scientific merit first, and cost second. Furthermore, costing an addition to a complex distributed sensor system, like the LOFAR telescope, is exceedingly complex and prone to overseeing non-trivial component costs.

2.7.3 Spectre and Meltdown: how value of an existing system may change unexpectedly

We argue in this chapter that we can try to estimate the total lifetime computational value of a hardware system beforehand. However, value is not constant over time and may be impacted by external factors beyond the control of the user. In January 2018 a number of critical and widespread flaws in the hardware design of current generation processors were published [89, 84]. These unparalleled hardware vulnerabilities hit virtually every installed compute system currently in operation. While many software bugs may cause temporary performance issues, or cause delays in achieving top performance, the mitigations implemented to address these unprecedented flaws in processor design caused a completely unexpected and major reduction in performance of current systems, including otherwise well-performing systems. Due to the nature of these flaws, critical separation failures in performance-critical speculative execution, mitigation efforts in processor microcode and operating system kernel, have resulted in significant performance impacts, thus reducing the value of existing compute systems. In particular I/O heavy workloads, such as those encountered in the LOFAR correlator and beamformer, that cause large numbers of context switches are expected to see performance reduced by very significant amounts [33]. For the Linux kernel, the dominant operating system in both high-performance computing, as well as distributed computing applications, these are known as Kernel Page Table Isolation (KPTI). These are kernel level fixes, that can be activated or deactivated at boot-time with a kernel boot parameter.

We illustrate the performance impact of these mitigating efforts in Figure 2.4. We test three Linux kernels, one released just before the announcements mentioned above (4.13.16), one that includes the initial mitigating patches (4.14.14) and one more recent kernel (4.19.1) in which the mitigations have been in place for some months. Since a key task in the correlator and beamformer systems in LOFAR involves receiving large numbers of UDP/IP streams, we measure performance impact, and therefore the hit on value, by trying to receive as many UDP/IP packets as possible on a CPU-bound system with a 40 GbE device. Results are normalised to the performance of the oldest kernel,

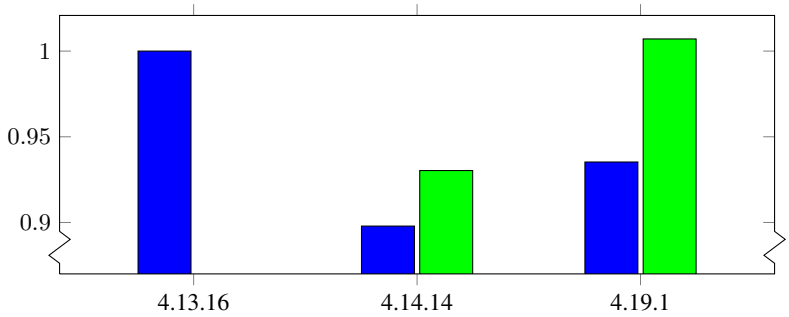


Figure 2.4: Maximum UDP/IP packet receive performance for three kernels, normalised to the oldest kernel. Blue shows the default configuration, green when Spectre and Meltdown v2 mitigations are turned off.

which, for reference, achieved around 1,65 million packets per second.

This measurement shows that the value of a system has the potential to change over time (here between 5% and 10%), and may be affected by factors and risks outside its operators' and designers' control. In this particular case, most of the performance impact may be avoided by turning off page table isolation (`nopti`) and retpoline (`nospectre_v2`) at boot time, at the cost of accepting that the system is trivially exploitable (which may be acceptable for a dedicated cluster behind a firewall).

2.7.4 SKA SDP

The Square Kilometre Array (SKA) is a next-generation radio telescope, currently in the design phase. It will consist of two telescopes, a low-frequency telescope (50-350MHz) consisting of 130,000 antennas in over 500 stations in Western Australia, and 133 mid-frequency dishes (4350MHz-14GHz) in South Africa, which latter will be joined to the existing MeerKAT telescope. These telescopes will be constructed and deployed in phases across a 5-year period. The SKA is designed to achieve exceptional scientific value, and to enable potential Nobel Prize-winning research [144].

A key component of this instrument is the Science Data Processor (SDP), where instrument data, produced by specialised correlator hardware, is turned into science-ready data products, such as radio astronomy images, using high performance general-purpose compute systems [10]. There will be data centres in Perth and Cape Town, which will host an SDP for each country, where the data will be received, processed, and transmitted for use by astronomers. The data rates from SKA will be extremely large: each telescope will output up to 3.1Tb/s from the correlator. The main function of the SDP is thus to perform a data reduction, outputting data products that are able to be used by scientists, but which are also somewhat easier (and cheaper) to store and transmit. The (in)ability of the SDP to perform this function may impact the science

that can be performed by the SKA as a whole: if it takes too long to reduce the data, or the SDP cannot reduce the data by a sufficient factor, less data- or compute-intensive observations will have to be scheduled [10]. Thus the design of the SDP is critical to the scientific value of the telescope.

In order to maximise its relative science value, the SDP will use a mix of custom-designed software components and off-the-shelf software. In order to reduce TCO, the SDP will make extensive use of existing technologies. A platform management system is envisioned to provision and organise its compute resources. Such a system allows for the automation of compute deployment, at the cost of a mild computational overhead. This saves on operator time, and allows for reliable and reproducible deployment of operating systems and other support services. The reduced operator time needed and increased reliability drive down operational costs (C_{ops}); reproducibility renders it easier and quicker (hence cheaper) to detect bugs. OpenStack, an open source platform management product in use by HPCs and data centres, is a candidate solution for this, in part because SKA is already working with CERN on improving OpenStack technologies. The SKA will save cost of development (C_{dev}) and ongoing maintenance costs by using this off-the-shelf open source software, rather than writing their own suite of complicated software for the same purpose. The viability of this approach has already been prototyped [150].

In addition, in order to improve TVO, a new suite of astronomy data processing software will be developed, focusing on a highly reusable modularised architecture. The principal idea is to create low-level software modules that can be reused by many data processing pipelines [9]. However, rather than using existing code from existing telescopes, these modules will be newly implemented for two reasons: scalability in parallel environments and maintainability over the expected 50-year telescope lifetime.

Providing modules that can easily be connected for use in large clusters is key for the SDP, as, without taking advantage of the inherent parallelism available in a lot of astronomy data processing, it will be difficult to achieve the data throughput necessary. This modularisation not only allows designing for an embarrassingly parallel processing environment, it also permits programmers to quickly and easily provide new modules for optimised use with new hardware, and implement new algorithms for new science without rewriting other parts of the software infrastructure. This is explicitly to reduce C_{dev}, by anticipating the need to port code to new, potentially very different, architectures, in contrast to the issues LOFAR experienced, as noted in 2.7.2. This also allows the SDP to run on generic COTS hardware, while also allowing for future software-hardware co-design for key algorithmic components. Similarly, modularisation of code handling hardware interfaces allows for pivoting to new technology — an inevitability in a long-lived project.

Maintainability is also a key driver for writing new code: technical debt accrues in software projects over time, as programmers can end up prioritising writing code quickly, rather than writing it well, or with an eye to help reduce maintenance costs. Some of the commonly-used radio astronomy code, such as CASA, has parts that are nearly 40 years old, and which were not designed to be used in highly parallel compute systems. Thus the SKA has the opportunity to reduce its total lifetime costs by investing in new code that is designed to be more easily maintainable, especially around providing new algorithms and pipelines for its highly parallel environment. Proofs of concept

of this approach have similarly been prototyped, to verify ease-of-use and explore the scalability required for SKA [13, 49].

This requires a significant up-front investment for rewriting code – SKA SDP software accounts for approximately 8.2% of the SKA construction budget, compared to 7.1% for the VLT, 5.7% for ALMA and 4.3% for ASKAP [83, 74]. We note that for SKA SDP this is processing software only, excluding telescope manager – functionality that is included in the figures for VLT, ALMA and ASKAP. However, this will improve TVO, by making it easier to make efficient use of the data processing hardware, by making it easy to implement new algorithms, and by isolating where code changes to support those algorithms are needed. This should thus reduce some of the maintenance cost of the SDP, and improve its ability to unlock new science across the lifespan of the telescope, albeit at an increase in upfront development cost (C_{dev}). The SDP will also undertake a phased hardware deployment, to provide compute when it is needed to support the increasing number of antennas and dishes on the ground, which will both keep capital costs lower, and reduce overall operational costs (C_{ops}). Furthermore, the deployment of hardware later on in the project allows hardware to be tailored to the software and vice versa, similar to the Cobalt correlator and beamformer in LOFAR, improving total relative science value of the resulting system. The SDP is for the SKA thus deliberately considering and trading off in different areas, the TCO and TVO of the system, with some decisions made to manage cost, and others to maximise total lifetime scientific value.

2.8 Related work

This work is a form of hardware-software co-design, as practised in the design of compute systems for large-scale science instruments. However, up to now, hardware-software co-design has focused mostly on more easily measured metrics, such as cost, power consumption and peak performance. Furthermore, while the literature often speaks of the importance of application co-design, the metrics used are agnostic and described mostly in terms of cost functions and constraints in energy and capital. In this chapter we explore what these systems are really built for, and what a suitable measure for their performance would be.

This work can be considered a specialisation of general cost-benefit analysis in economics. Whereas cost-benefit analysis normally evaluates the social or financial benefit of a certain investment, this chapter focuses on the scientific benefit in particular. There is research that introduces the concept of total value of ownership [167] in accounting, but this is introduced as potential future research as an extension to TCO based decision making and not expanded upon. In that paper it is claimed that TVO builds on the concept of value as described in marketing literature.

Total value of ownership, also referred to as total value of opportunity, is also a metrics-based methodology for measuring and analysing the business value of enterprise IT investments [15]. This is an extension of TCO analysis, where both cost and any benefits of the proposed investment, tangible or intangible, are considered.

Value Engineering, Value Management and Value Analysis in Systems Engineering describe processes to achieve an optimal solution [164]. This optimal solution is

based on stakeholder value metrics; the processes are agnostic to these. In this chapter we take the stakeholder view, describing and enumerating the value metric, while not considering the detailed processes required to optimise these.

Some work was done to analyse the societal impact of the High-Luminosity Large Hadron Collider (HL-LHC) upgrade of the LHC [66, 21], predicting a larger than 90% chance of positive net economic benefit to society based on Monte Carlo simulations. These simulations estimate the economic returns from diverse benefits such as value of training for students, technological and industrial spillover, cultural effects for the public and academic publications. A comparison of the impact of the upgrade to the LHC with the non-upgraded instrument was also presented. An impressive effort is made to estimate the total cost of the current LHC, a difficult task even though all CERN expenses are well documented, due to the many in-kind contributions by member and non-member states. Societal impact analysis are very useful for funding agencies to gauge the value of an instrument to society using *scientific* and *objective* criteria. However, analysis of the methodology found many ambiguities and the scientific benefits of the LHC is given as less than 2% of the total impact of the instrument using this method: a drastic underestimation [137]. Furthermore, it was found that the societal impact of CERN's mission, "promote science and bring nations together", was impossible to measure, since no way has been developed to measure in economic terms the success of the second objective. In comparison , the concepts introduced in this chapter look at more immediate impact, computational or scientific, and attempt to be more directly useful when making design choices.

Recent work on design optimisation of low-frequency telescopes using cost constraints [26] takes a slightly different and more domain specific approach. Here, an attempt is made to model both cost and scientific performance of a radio telescope using Lagrange multipliers. Scientific performance, defined by two instrumental figures of merit – sensitivity and survey speed, is optimised using both models and an assumed fixed capital budget. The LOFAR architecture as built, and the SKA phase 1 baseline design are analysed using the introduced model and variants optimised for survey speed and sensitivity are proposed. This methodology focuses on receiver and front-end optimisation and mostly ignores the cost required for compute capacity or how this scales with the number of stations and length of baselines. While the cost model does include a central correlator and beamformer, its model is exceedingly simple. Calibration, imaging and other post-processing costs, as well as long-term storage of data products, monitoring and control and operational costs are not modelled. Furthermore, we note that the chosen degrees of freedom in this paper, number of stations and number of antennas, have an enormous impact on required compute capacity for calibration and imaging. In this chapter we take a more generic approach that is not limited to radio astronomy and that focuses on the cost and value of the compute systems that are not considered by Boonstra et al.

2.9 Summary and conclusions

In this chapter we introduced a more formal way to reason about cost and value of compute resources, both hardware and software. We suggested that a focus on minimising

cost alone is not sufficient to design an optimal solution. The introduction of several new concepts, total value of ownership, total lifetime computational value, total lifetime scientific value and relative science value, gives us the *vocabulary* to effectively discuss routes towards more optimal solutions. Although both total lifetime computational value and especially total lifetime scientific value are difficult to quantify, and we do not expect anyone to do so using the formulas given in this chapter, we do show a number of components that allow us to reason effectively about this metric.

We provided a number of case studies in which we demonstrate the concepts introduced in this chapter. We can see the utility of explicitly considering a metric of total lifetime scientific value, as the TITAN computer sought only to minimise capital cost (which happily led it to deliver truly exceptional value), whereas the SKA designers are explicitly allowing for relatively high costs in some areas to maximise total scientific value. In the LOFAR use case we noted the explicit trade-off made between high-impact science and dropping cost for computing, which led to an identified "processing window of opportunity" some years in the future where relative science value was perceived to be optimal. Some of the later additions to LOFAR were discussed, each adding their own value to the complex machinery that is the LOFAR telescope. Finally we showed, using a recent highly publicised processor flaw and its mitigating patches, that the total computational value of a system may potentially change over a system's lifetime. Together, these concepts and case studies provide a framework for decision makers, principal investigators, designers, and engineers of computing solutions to reason about the optimal solutions, in hardware or software, for their applications.

2.10 Our propositions in this chapter

2.10.1 The bounding proposition

In this chapter we mainly focused on ways to measure cost and value. Even though these are important bounds to consider, this chapter does not materially contribute to the *bounding* proposition. However, the case-studies in section 2.7, in particular the LOFAR case-study, give a clear example of this proposition in active use. Initially the computational requirements of LOFAR were judged too expensive, but to become affordable in the mid-future. This shows a number of bounds considered: compute requirements, capital investment and time.

2.10.2 The value proposition

This chapter lays the theoretical foundation for the value proposition. We define the concepts of Total Value of Ownership to express the value a compute and data transport system accrues over its lifetime, and total relative science value to define the value per invested Euro of such systems. While quantifying these measures is still difficult, we give a number of possible ways to estimate relative value. A synthetic example shows that choosing an appropriate measure for value is exceptionally important, and that choosing poorly may lead to sub-optimal implementation choices. Several use-

cases are analysed, suggesting that the concepts suggested are in use, but informally and would benefit from more structured analysis, as suggested by in this thesis.

Acknowledgements

The authors would like to thank Yan Grange, Ágnes Mika, Bram Veenboer and Cees Bassa at ASTRON for their valuable feedback on early versions of this manuscript. We would like to thank Dr Elizabeth Waldram and Professor Malcolm Longair from the Cavendish Laboratory, Cambridge, for their help with understanding the impact of the TITAN Computer, and the Librarian of the Department of Computer Science Library at the University of Cambridge for access to the TITAN archival material.

CHAPTER 3

ExaScale High Performance Computing in the Square Kilometre Array

P. Chris Broekema [1]*, Rob V. van Nieuwpoort* [2] *and Henri E. Bal* [3]

Context and contributions

This chapter is a slightly modified version of a paper that was presented at the first Workshop on High-Performance Computing for Astronomy in 2012. Broekema was the first author, and wrote most of the paper, with some illustrations reproduced with permission from the owners as mentioned in the acknowledgements.

This paper was one of the very few accurate and up to date descriptions of the tasks faced by the SKA science data processor, until it was replaced by the Journal of Instrumentation paper that is shown in the next chapter. A section detailing a soon abandoned analysis tool was removed from this chapter. While some parts described in this chapter are superseded by chapter 4, we kept most of the original text intact to show the gradual refinement of the *bounding* proposition as a project progresses.

This chapter reflects the early design stages of a large project and there is some evidence of the *bounding* and *value* propositions in use here.

[1]ASTRON, the Netherlands Institute for Radio Astronomy
[2]Netherlands eScience Centre
[3]Vrije Universiteit Amsterdam

Abstract

Next generation radio telescopes will require tremendous amounts of compute power. With the current state of the art, the Square Kilometre Array (SKA), currently entering its pre-construction phase, will require in excess of one ExaFlop/s in order to process and reduce the massive amount of data generated by the sensors. The nature of the processing involved means that conventional high performance computing (HPC) platforms are not ideally suited. Consequently, the SKA project requires active and intensive involvement from both the high performance computing research community, as well as industry, in order to make sure a suitable system is available when the telescope is built. In this chapter, we present a first analysis of the processing required, and a tool that will facilitate future analysis and external involvement.

3.1 Introduction

The Square Kilometre Array (SKA) is a next-generation radio telescope currently entering its pre-construction phase. The central processor for the SKA will require computational resources well in excess of what even current top-of-the-line supercomputers can offer. Additionally, the processing done is quite different from conventional high performance computing applications, in computational intensity[1], in its streaming nature, and in its relative robustness against hardware failures.

In this chapter, we present an analysis of SKA central processing on future supercomputer hardware, for as far as possible considering the limited information available. We also present an analysis tool that will allow the accurate and detailed analysis of SKA processing on a system level. This is intended to show not only architectural shortcomings of current or near-future high performance computing platforms, but also to identify design points that would make future systems particularly suited for radio astronomy.

This chapter is structured as follows. First, we will briefly describe the Square Kilometre Array. In Section 3.3, the computational requirements of the first phase of the SKA are identified, followed by an analysis of radio-astronomical algorithms, as compared to conventional HPC applications. We then present an analysis of the HPC roadmaps and identify where these fall short of our requirements. In Section 3.6, we present a SKA analysis tool, followed by our conclusions. We end by briefly discussing future work.

3.2 The Square Kilometre Array[2]

The Square Kilometre Array is a next-generation radio telescope, with a total collecting area of approximately one square Kilometre. It will operate over a very wide frequency

[1] Computational intensity is defined as the number of floating point operations per byte of I/O. Normally I/O is considered to be a memory access, but for the SKA this is often a network read.

[2] While this section is outdated, it is kept in tact to illustrate project progression. See Section 3.8 for more recent information.

range, from 70 MHz to 10 GHz, which will require several different receiver types. It will be built in the southern hemisphere, either in South Africa or Western Australia. The SKA will be capable of extremely high sensitivity and angular resolution.

The extremely high sensitivity and angular resolution requires an array with a very large number of receivers, covering a very large area. SKA sites are projected to extend up to 3000 km from the central core and will contain millions of receivers.

The SKA will cover a frequency range from 70 MHz to 10 GHz. This can't be covered with a single antenna type, so arrays consisting of two, possibly three, different receptors will be built.

- The low end of the frequency range, from 70 to 450 MHz, will be covered by low-frequency sparse aperture arrays of simple dipole antennas (AA-low). Clusters of around 11,200 of these dipole antennas will be grouped into stations of about 180 meters in diameter. It is expected that 250 of these stations will be built. Each AA-low station will eventually produce between 10 and 30 Tb/s of data, to be transported to the central processor, depending on the number of simultaneous beams required. In phase 1 this will be limited to $\sim$1 Tb/s per station. In total these stations will generate up to 7.5 Pb/s.
- The mid-frequency range may be covered by a dense aperture array design, using tiles grouped in stations of about 60 meters in diameter (AA-mid). Up to 250 of these stations may be built. AA-mid stations produce the same amount of data as AA-low stations, totalling $\sim$2.5 Pb/s for 250 stations. The AA-mid concept is part of the Advanced Instrumentation Program (AIP). Technologies in this program will be assessed in terms of science impact, cost, and technical readiness, and deployed in SKA phase 2 if shown to be feasible and cost-effective.
- The high end of the frequency range, upward from 500 MHz, will be covered by relatively small dishes, with a Single Pixel Feed (SPF). These will be around 15 meters in diameter. A subset of the dishes may be equipped with a Phased Array Feed (PAF), or a Wide Band Single Pixel Feed (WBSPF). These advanced feeds are also part of the Advanced Instrumentation Program. Around 3000 dishes will be built. Each dish produces $\sim$120 Gb/s, assuming single pixel feeds. PAFs will produce up 10 times more data, WBSPFs around twice the amount of a SPF. All dishes together produce $\sim$360 Tb/s, if fitted with single pixel feeds.

The SKA stations will be divided into four regions. Figure 3.1 shows the four regions of the SKA and the distribution of the collecting area.

- The core will have a diameter of around 1 km, for each of the receiver types. The dish core and the AA core will be spatially separated from each other.
- The inner region has a diameter of around 5 km. The core and inner regions together will contain more that half of the total collecting area of the SKA.
- A mid region, extending out to 100 km for phase 1, and up to 180 km for phase 2, will contain dishes and pairs of AA-low and, possibly, AA-mid stations. In each case, they will be randomly placed within the area, with the density of dishes and stations falling off towards the outer part of the region.

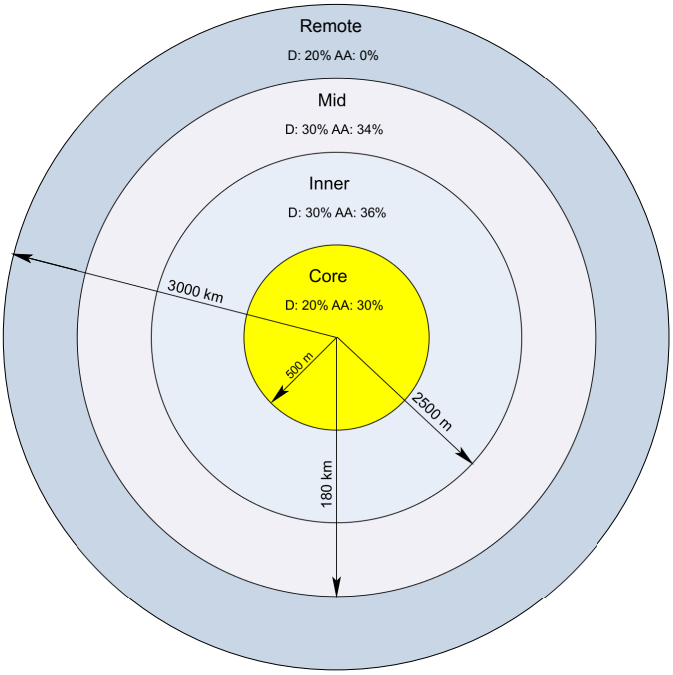


Figure 3.1: The distribution of collecting area for SKA phase 2 (D = Dishes, AA = Aperture Array stations)

- The remote region extends from 180 km to 3000 km from the central core. This will comprise five spiral arms, along which dishes, grouped into stations of around 20, will be placed. The separation of the stations increases towards the outer ends of the spiral arms.

The data from the receivers are transported, using long haul optical links, to a central processor facility. The aggregate data rate into the central processor will be in the order of 10 Pb/s, fittingly described as a data deluge. There is no known way around this central processing; the correlator requires data from all stations in an observation. Figure 3.2 shows a high level overview of the SKA system.

The central processor can conceptually be divided into the central signal processor, handling correlation and beamforming, visibility processors, responsible for gridding, and image formation, generating images and calibration solutions. The central processor, located at the Science Computing Facility shown in Figure 3.2, takes data from the SKA stations as input, and delivers calibrated science data as output. The scientific data are stored in the science data archive at the Regional Science Centre(s), intermediate

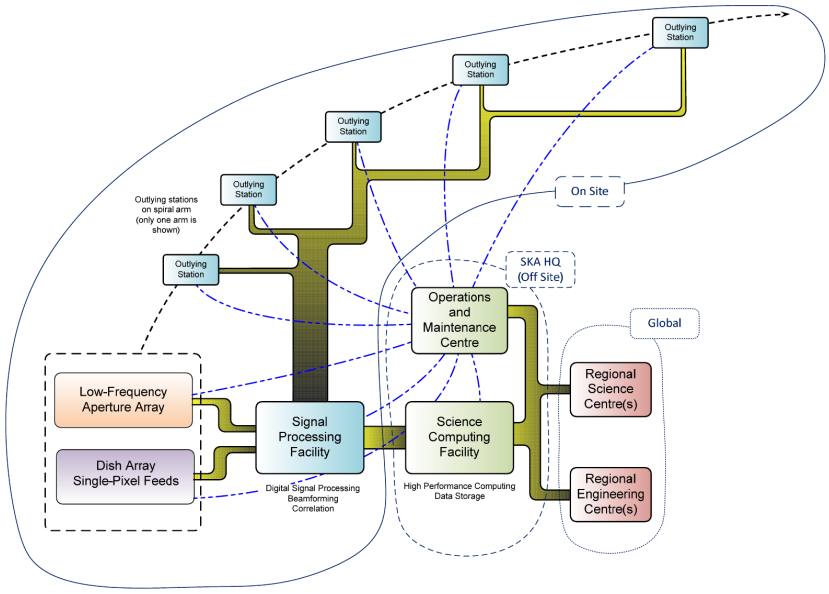


Figure 3.2: A high level overview of the Square Kilometre Array

data are, in principle, not stored. The data rate out of the SKA central processor is several orders of magnitude smaller than that coming in.

The feature that sets the SKA central processor apart from conventional high performance computing applications, is its very high input data rate. Since it is unlikely that enough high performance storage will be available, or affordable, to allow batch processing, and the system needs to keep up with the input data stream or risk dropping data, the SKA central processor can be described as a pseudo real-time streaming processor. Note that there is no hard real-time deadline, apart from the requirement to keep up with the input data stream, making the real-time requirements less strict than in a classic real-time application.

An important consequence of the streaming nature of the SKA central processor, is the need to dimension the system to be able to comfortably handle the most demanding application. Since there is no intermediate storage available, insufficient resources will inevitably and immediately lead to data loss if the central processor is under dimensioned.

3.3 SKA phase one requirements[3]

As a risk- and cost-reduction measure, the SKA will be built in two phases. Phase 1 (often identified as SKA_1), scheduled to start construction in 2016, will consist of two receiver types, low-frequency sparse aperture arrays and high-frequency dishes, and have a maximum baseline of around 200 kilometers. The preliminary specification of phase 1 defines 50 sparse aperture array stations and 250 dishes, although the science analysis may require more, but smaller, aperture array stations be built. It is likely that instead up to 250 smaller aperture array stations will be built. The cumulative data rate of these smaller stations will be similar to the original 50 bigger stations, but post-processing requirements will increase considerably.

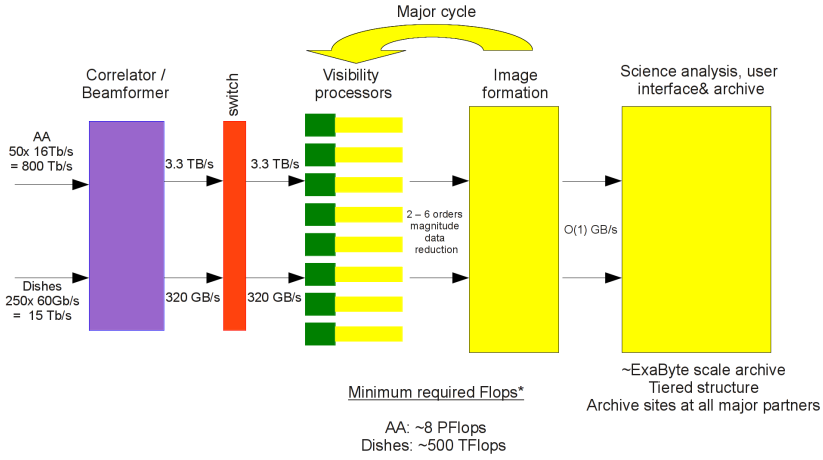


Figure 3.3: A schematic overview of the SKA phase 1 central processor

Although the exact computational requirements of the SKA phase 1 are not well defined yet, it is clear from the preliminary requirements[59] that these are well beyond the capabilities of current HPC systems. A full analysis of the science requirements will be done in the pre-construction phase, starting in 2012. An early analysis of the current science requirements, based on the SKA phase 1 Design Reference Mission (DRM)[72], showed considerable differences with the preliminary requirements mentioned earlier[11].

It is of course unrealistic to expect the complete requirements for the software and computing component for a complex system like the SKA to be known at such an early stage. In fact, both software engineering experience[93, 130], and experience from the only operational radio telescope comparable to the phase 1 SKA, LOFAR[43], show

[3]This section is again outdated, but kept to illustrate project progression.

that requirements are likely to be subject to change. Having said this, the preliminary requirements currently available, give us a good first order estimate of the compute power needed for phase 1 of the SKA.

Figure 3.3 shows a schematic overview of the SKA phase 1 central processor, based on these preliminary requirements. Note that the compute requirements shown in this figure are an absolute minimum. They assume a computational efficiency of 100%. Furthermore, recent experience has shown that the assumption that it takes 10^4 operations to completely process an input sample, may be a serious underestimate. If we assume a computational efficiency of 10%, quite reasonable in HPC, and 10^5 operations per input sample, we would need to scale the central processor to a peak performance of approximately 800 PFlop/s. This only covers the visibility processor and image generation parts of the central processor, it does not include the correlator and beamformer.

Output from the correlator for SKA phase 1 will be in the order of several terabytes per second. This extremely high data rate makes the SKA central processor quite unique among high performance computers. Most supercomputers in the Top500 are designed to handle complex simulations, which typically have a very high computational intensity. If we consider a software based correlator and beamformer, the input data rate increases to around 100 terabytes per second. Although the concept of a highly integrated software based central processor, including the correlator and beamformer, is intriguing and we will argue in section 3.7 that this may offer significant advantages, in the rest of this chapter we will mostly ignore the beamformer and correlator, and concentrate on what Figure 3.3 calls the visibility processors and image formation.

3.4 Radio-astronomical HPC

When analysing the computational feasibility of an instrument, like the SKA, the natural performance figure to look at centres around the floating point arithmetic performance, usually expressed as a LINPACK[88] performance figure. These figures form the basis for the Top500 list of the fastest supercomputers in the world[152].

The LINPACK benchmark is characterised by a large number of parameters, like matrix sizes, that can be freely chosen to allow the user to tailor or optimise the benchmark to the hardware being tested. On the one hand this allows an extensive exploration of the efficiency space, but it also makes LINPACK a highly unrealistic benchmark.

The computational performance of the LINPACK benchmark depends strongly on the double precision floating point performance of matrix-matrix operations, and interconnect latency. Compared to LINPACK, algorithms in radio astronomy are characterised by a very low computational intensity, expected to be in the order of 1 floating point operation per byte of I/O, often even less[77]. Data are mostly independent in frequency, which means that low latency in an interconnect is far less important than high bandwidth.

Figure 3.4 shows a flow diagram of the calibration and imaging steps in the SKA. This shows the current state of the art, which may obviously change. It is beyond the scope of this chapter to go into the details, but it is important to note that, unlike current radio telescopes, data must be calibrated and imaged online, the data rates involved don't allow temporarily storing intermediate data products.

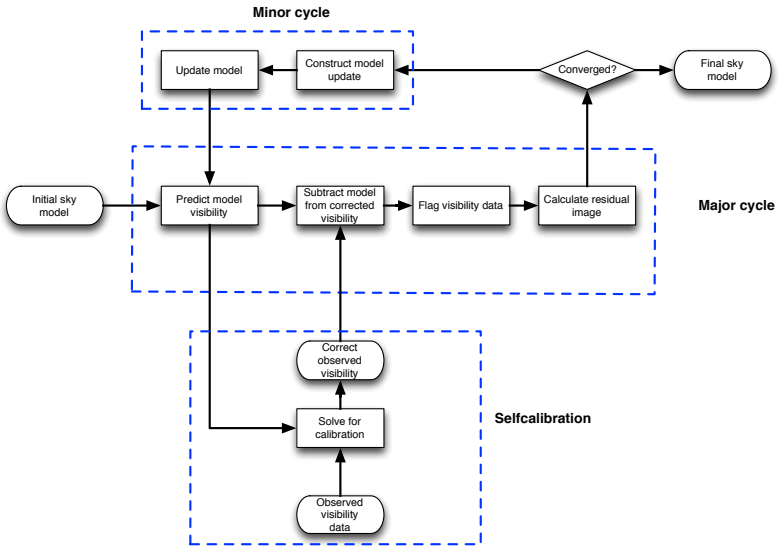


Figure 3.4: Flow diagram of SKA calibration and imaging

This exposes an interesting challenge. The very high input data rate demands a streaming processing model, without intermediate storage of data products. Unfortunately there is no known single-pass calibration method. Therefore, a very high performance buffer is required to store intermediate data, while the multi-pass calibration algorithm converges.

Nearly all computations are done on complex numbers. Fourier transforms, vector and matrix operations, and complex multiply-adds play an important role. In contrast to most conventional HPC applications, we can probably get away with single-precision floating point operations for a lot of our processing. This may significantly reduce data rates and, assuming appropriate hardware support, the size of the central processor. The exact ratio of single- and double-precision processing, as well as an accurate decomposition of the exact operations required and the exact computational intensity of the various processing steps, will be part of a detailed investigation in the upcoming preconstruction phase.

While the LINPACK benchmark gives us a reasonable idea of the performance characteristics of current state of the art supercomputers, it is of limited use for the evaluation of a system for the SKA. For a more detailed discussion on this subject, we refer back to Chapter 2, specifically Section 2.4.

3.5 HPC roadmap analysis

The most powerful supercomputers in the world, according to their performance in the LINPACK benchmark, are ranked in the Top500 list. This list is updated twice per year and goes back to 1993. Plotting the aggregate performance of the Top500 machines shows a steady increase in available compute power over the last two decades, mostly consistent with Moore's law.

A straightforward extrapolation of the past lists shows that a machine capable of handling the phase 1 central processor requirements, should be available by 2018 - 2019. This is illustrated in Figure 3.5. This only shows the development in compute power, as measured using the LINPACK benchmark.

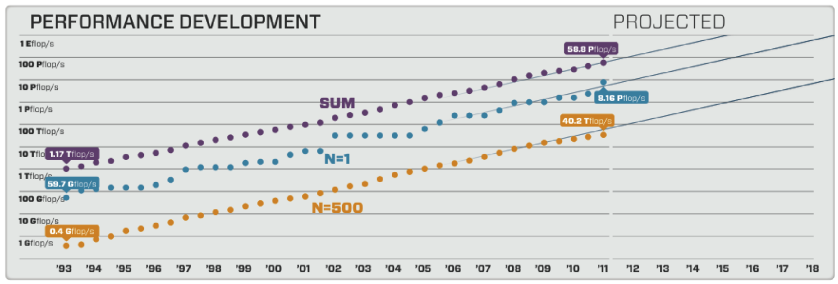


Figure 3.5: Top 500 extrapolation shows ExaScale systems available by 2018 - 2019. From top to bottom the lines represent the the performance of last entry in the Top 500, number 1 entry in the Top 500, and the sum of the performance of all supercomputers in the Top 500 combined.

In 2008 the ExaScale panel published a report on the expected developments in high performance computing in the coming decade[111]. By looking at current, and expected developments in the different components of a high performance computer, a projection was made of a feasible ExaScale system. Although one should carefully weigh the value of long term predictions like the ones in this study, they did highlight some disturbing trends.

One of the most obvious developments in HPC is the increase in overall concurrency. Moore's law, as interpreted as the doubling of the number of components per unit of area on a chip every 18-24 months, is expected to continue to hold for the next decade or so, which means that feature sizes in future processor will continue to decrease for the foreseeable future. Due to increased leakage power at small feature sizes, processor clock frequency has levelled off. Future systems are expected to continue to run at a clock frequency in the order of one to several Gigahertz.

The power budget available for a single processor socket has also levelled off. The practical limit for commodity cooling solutions is around 150W per socket. Water-cooling may raise this limit slightly. In the future we will see aggressive and fine grained power gating shutting down unused parts of a CPU, allowing the remaining components

to dynamically scale in performance to fill the available thermal budget. It is likely that the available thermal budget per socket will be insufficient to allow all components in a processor to run at full power simultaneously.

The trend going forward is that individual cores tend to not increase in performance very much, certainly not sufficiently to follow Moore's law. Shrinking feature sizes, however, allow us to add ever-increasing numbers of cores on a CPU. Additionally, many-core architectures, like GPUs and special purpose accelerators, can now be integrated onto the CPU. These developments lead to a massive increase in required application concurrency to efficiently use the available resources. Nevertheless, these developments are, by themselves, not enough to reach the performance levels shown in Figure 3.5. In order to bridge that gap, an increase in the total number of processors is also required, possibly with additional accelerator hardware. Both relative memory size and bandwidth are unlikely to keep up.

So ExaScale systems will be characterised by massive parallelism on many levels. Huge numbers of nodes, possibly of various types, will be connected to a cohesive but highly complex system. Within a node, and even within a processor, we will see various levels of parallelism. It is probable that processors will be heterogeneous, consisting of both a smaller number of general purpose, complex, super-scalar, out-of-order cores, and many, much simpler, cores optimised for floating point operations. It is possible that these will be augmented by a number of special purpose accelerators. The heterogeneous nature of such processors makes them relatively hard to program, but the potential performance and efficiency of such a system is tremendous.

Conventional HPC applications are often relatively compute intensive; the number of Flops per bit of I/O is very large. SKA processing, in contrast, contains a significant portion of operations with very low computational intensity. The streaming nature of the SKA central processor emphasises this. Although most HPC applications will notice the significantly reduced memory bandwidth per Flop available in future systems, the I/O bound and streaming nature of SKA processing makes this a particularly significant problem for us.

In Table 3.1 two recent top-of-the-line supercomputers are compared to the projected ExaScale machine as predicted by the ExaScale panel. A number of interesting features have been selected for comparison. Particular pain points are highlighted in bold face, the decrease in available memory bandwidth per Flop is especially worrying. In the next few sections we will investigate the impact of the various trends shown in this study.

Energy consumption

The ExaScale study used a total power consumption of 20MW as a design target for an ExaScale machine. They argued that this allowed some growth beyond that of today's largest systems, but still not be so high as to preclude it from deployment in anything other than specialised strategic national defence applications. The same study also concluded that even the most optimistic projections with respect to improvements in energy consumption per Flop, still fall short of this target by several factors.

[4] Mean Time To Interrupt

	2009 Jaguar	2011 'K' computer	2018 (projected)	2009 vs 2018	2018 Summit
System R_{peak}	2 PF	10 PF	1 EF	*O(1000)*	201 PF
Node R_{peak}	125 GF	128 GF	1 - 15 TF	*O(10 - 100)*	42 TF
Energy	6 MW	10 MW	20 MW	*O(10)*	13 MW
Energy/Flop	3 nJ/F	1 nJ/F	20 pJ/F	*-O(100)*	65 pJ/F
System memory	0.3 PB	1 PB	32-64 PB	*O(100)*	2.8 PB
Memory/Flop	**0.6 B/F**	**0.1 B/F**	**0.03 B/F**	***-O(10)***	**0.01 B/F**
Memory bw/node	25 GB/s	64 GB/s	2 - 4 TB/s	*O(100)*	900 GB/s
Memory bw/Flop	**0.2 B/s/F**	**0.5 B/s/F**	**0.002 B/s/F**	***-O(100)***	**0.0000045 B/s/F**
Total concurrency	**225,000**	**548,352**	**O(10^9)**	***O(10^5)***	**167 10^6**
MTTI[4]	days	days	hours	*-O(10)*	N/A

Table 3.1: A projected 2018 supercomputer compared to two current ones. This table was updated with information from the fastest supercomputer available in June 2019, Summit, to show actual progression compared to the estimate. Summit was installed and commissioned in 2018, so that year is used for our comparison.

The scale of the problem becomes clear when we look at the current state of the art. IBM's Blue Gene/Q prototypes are currently the most energy efficient machines in the Top500, by a significant margin[70]. These machines offer an efficiency of 2 GFlop/s/W. The design goal of an ExaScale machine within a 20 MW power envelope requires an efficiency of at least 50 GFlop/s/W.

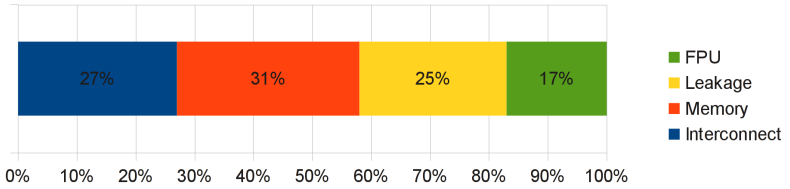


Figure 3.6: Energy distribution in an ExaScale system

Figure 3.6 shows the energy distribution in the most optimistic projection of a 2018 ExaScale supercomputer. Even in this very optimistic projection, more than half of the energy consumed is needed for I/O. It is worrying to note, though, that external I/O, i.e. data coming into or going out of the machine, is not taken into account at all. The ExaScale panel mainly considered conventional HPC applications, which are characterised by very little external I/O. The SKA central processor, in contrast, is characterised by a massive data stream into the system and a much smaller, but in comparison to conventional HPC applications still significant, stream of data out to the science data archive.

Input/output

The problem of input/output is closely related to that of energy consumption. As was shown in the previous section, a world-class supercomputer in the SKA time frame will consume the bulk of its energy moving bits around. Unfortunately, most HPC roadmaps limit I/O predictions to memory and interconnect bandwidth, and they often ignore external data transport.

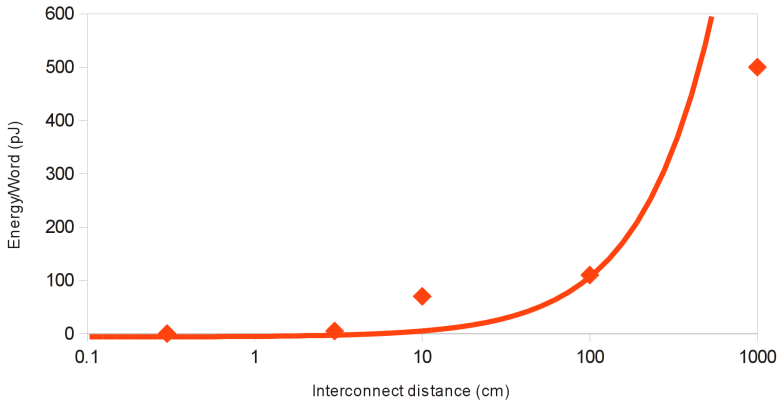


Figure 3.7: Energy required for I/O (Source: [86])

Figure 3.7 shows the energy required to transport a word of data over a given distance [86]. As distance to the CPU increases, the energy required increases super-linearly. The streaming nature and the massive input data rate of the SKA central processor mean that the energy consumed for I/O in the SKA central processor will account for much more than the 58% shown in figure 3.6.

Experience with the LOFAR radio telescope has shown that, although a real-time streaming central processor is a feasible and very flexible proposition[98, 123], significant work is often needed to modify the system software designed and optimised for conventional HPC applications[79, 119, 169].

Programmability

In the next decade or so we'll see disruptive changes in the way compute hardware is designed and built. Massive parallelism, combined with heterogeneous and bandwidth starved architectures, will have to be handled by all HPC developers, but for the SKA the problem is amplified.

The streaming nature of the SKA central processor, as well as the tremendous input data rate, make it unique in high performance computing. This may mean that the programming models that will be developed to handle the features of future supercomputers are not suitable for our application. To make sure we can efficiently use future hardware, we may need to develop these programming models ourselves.

Current radio telescopes use a significant number of legacy codes unsuitable for deployment on an ExaScale system. Some of these handle relatively simple things, like coordinate transformations and so on. These, casacore[44] and wcslib[65] are some examples, will have to be rewritten for future systems. This is time consuming, but the algorithms are well known and this should not be a significant risk. Others deal with processing the massive data stream from the correlator and beamformer. For these codes, merely porting existing codes used by the pathfinder and precursor instruments is not enough. Significant algorithm development is needed as well, to make sure it can handle the data volumes, achieve the science requirements, and efficiently use future ExaScale systems. This must be considered a significant risk.

Reliability

In the next few years we'll also see a tremendous increase in the number of components in a supercomputer, as illustrated by the total concurrency row in table 3.1. Since reliability per component is not expected to increase significantly, it is inevitable that total reliability of the system will decrease. In fact, one can say that the system must be considered somewhat broken all of the time.

In contrast to conventional HPC applications, SKA processing is relatively robust against failures. Within reason, the data are embarrassingly parallel, and loss of a small portion of data is often quite acceptable. As long as the system software is capable of detecting and reporting failed nodes, this reduced system reliability should not present a serious problem.

3.6 HPC involvement

The SKA community has been remarkably successful in attracting interest in the project, both from industry, and from the HPC research community. Since the SKA community itself lacks much of the experience and, frankly, critical mass required to handle the disruptive technology changes expected to occur in the next couple of years, it is essential to leverage this generated interest and get an increased involvement of industry and academia in the project.

One way of doing this is demonstrated by Lawrence Livermore National Lab. For their new supercomputer, Sequoia, they've published a set of sixteen representative benchmark codes[140]. These codes give a useful insight into the scalability and efficiency problems these applications face, but the obvious goal is of course to enable industry to optimise their architectures, software, and programming models to perform optimally for these applications.

3.7 Conclusions

It is clear that the central processor of the SKA is quite different from conventional HPC applications. Even though the LINPACK benchmark is already of limited value for normal HPC applications, its emphasis on low-latency interconnects and high computational intensity matrix-matrix operations means LINPACK figures are often misleading for our application.

This highlights the underlying problem: due to the markedly different properties of normal HPC applications versus radio-astronomical reductions, conventional future HPC installations are ill suited for SKA processing. How this can be improved should be part of a detailed study in the oncoming SKA pre-construction phase.

The very high data rates involved in the central processor are a major concern, especially considering the fact that moving bits around is going to account for the bulk of the energy consumed in a future supercomputer. Any design or technique that limits the amount of data movement should be seriously considered. This also means that optimisation of code should no longer focus on maximum utilisation of the computational resources, but instead try to minimise energy consumption. In practice this will often mean minimising I/O.

A highly integrated central processor, combining correlator, beamformer and science data processing in a single integrated solution that avoids highly inefficient switches in favour of integrated backplane communication, may well be more energy efficient than separate components. It is also important to realise that, although the efficiency in Joules/Op of custom hardware, FPGAs for instance, is still, and will probably continue to be, unbeatable for simple operations, like the correlator or beamformer, this is offset by power hungry data transport between the dedicated hardware solution and the general purpose science data processor. In other words, Joules/Op is really not a suitable metric to evaluate the efficiency of a system, since this ignores data transport. Instead a measure for the energy consumed by the entire central processor for a unit of scientific data, Joules per generated image pixel for instance, should be adopted.

Streaming processing support, both in hardware, and in software, is, and will continue to be, limited. Unfortunately, the SKA is quite unique in HPC in its requirement for very high input data rate and streaming processing of data. This means that we cannot expect any off the shelf solution to be immediately suited for SKA central processing. LOFAR experience has shown that significant work, especially streaming I/O related, is often needed to optimise an otherwise excellent HPC platform for radio astronomy.

The unique nature of our problem should be leveraged as an interesting case study for industry and research alike. Getting a challenging streaming application to work efficiently on a platform requires all components involved, hardware, operating system, communication middleware, and software, to work together in the most optimal way possible. In other words, if a streaming pseudo real-time application, like the SKA central processor, works efficiently on a platform, most other more conventional applications will also benefit from the optimisations required to get to that point.

3.8 Retrospective

This chapter was written before the detailed design process for the SKA kicked off. It is therefore instructive to retrospectively see how the high level design has changed since, and how this chapter has generally fared in terms of long-term accuracy. Chapter 4 is a more recent description of the SKA and its Science Data Processor.

This chapter was written before the site of the SKA was selected. In Section 3.2 we state that the SKA will be built in South Africa or Western Australia. In January 2013, the decision was made to build the mid-frequency array in South Africa, and the low-frequency array and survey instrument in Western Australia.

Furthermore, the high level design of the Square Kilometre Array introduced in this chapter underwent a major rebaselining in March 2015. This was intended to control cost and limit the excessive compute requirements for the original high-level design. This exercise, as well as various other delays both technical and organisational, have significantly shifted the schedule introduced in this chapter. At the time of writing (April 2019), pre-construction has mostly finished. System Critical Design Review preparations are well underway, and the SKA treaty, forming the basis for the construction and exploitation of the SKA instrument, was signed recently. While pre-construction did start in 2012, as mentioned in Section 3.3, construction is now expected to start in 2020.

As a final note we highlight that the projected and extrapolated increase in available computational and other resources resources has not materialised. The latest release of the top500 list at the time of writing shows a machine with 200 PFlop/s peak performance (Summit, a Power9 + GPU machine) as the fastest supercomputer in the world, whereas we expected to have reached ExaFlop/s by this time. This marked difference makes that the original timeline discussed in this chapter would have made the SKA SDP much more expensive, at least partially justifying the suffered delays. Table 3.1 was updated to include the characteristics of Summit, which shows that the identified trends have continued. Memory and memory bandwidth are increasingly scarce, and the advent of GPU processing means a massive increase in concurrency.

3.9 Our propositions in this chapter

3.9.1 The bounding proposition

The paper that this chapter is based on, was written very early in the design process of the SKA Science Data Processor. In fact, this chapter precedes the name *Science Data Processor*. As such, even though this is more of a feasibility study than an architecture or design, the *bounding* proposition is in clear evidence here. First and foremost, the SDP is defined in context of the instrument, and an initial estimate is presented of the required computational capacity, with the provision that this is a very rough and initial estimate. Furthermore, we analysed the development of hardware over time, inspired mainly by the exascale developments in high-performance computing at the time, to see if and when the required capacity would become feasible. The initial estimates of the required capacity, both computationally and in the form of I/O and storage, and feasibility of these requirements over time are bounds explored in this chapter.

3.9.2 The value proposition

Although this chapter represents the very start of the design process for the SKA Science Data Processor, we can already identify the *value* proposition in limited use. We note that, in the design and architecture of the system we draw inspiration from high-performance computing and in particular the ExaScale developments therein. However, we also immediately identify that the computational profile of most high-performance computing applications are quite different from those in a radio telescope, and consequently that systems optimised for high-performance computing are not necessarily suitable for the SKA Science Data Processor. Data flow is identified as a key distinguishing factor, which is confirmed in later work, the lack of which would impact the value potential of conventional HPC systems. Operational cost of I/O is also highlighted as an issue, a possible solution of which is addressed in Chapter 7 of this thesis.

Acknowledgements

Tim Cornwell (CSIRO/CASS) provided the flow diagram showing imaging and calibration (figure 3.4). Figures 3.2 and 3.1 were kindly provided by Peter Dewdney of the SKA Project Office (SPO). This work is supported by the SKA-NN grant from the EFRO/Koers
Noord programme from Samenwerkingsverband Noord-Nederland, and the ASTRON / IBM Dome project, funded by the province Drenthe and the Dutch Ministry of EL&I.

CHAPTER 4

The Square Kilometre Array Science Data Processor – Preliminary Compute Platform Design

P. Chris Broekema [1]*, Rob V. van Nieuwpoort* [2] *and Henri E. Bal* [3]

Context and contributions

This chapter is a slightly modified version of a paper that was published in the Journal of Instrumentation in 2015. Broekema was the first author and wrote almost all of the paper. The content of the paper was based heavily on Preliminary Design Review (PDR) documentation for the SKA Science Data processor, in which Broekema played a large role. While the paper focuses on the hardware design of the Science Data Processor, as conceived by Broekema, some of the more general concepts, such as the architectural priorities mentioned in Section 4.3, must be credited to the entire Science Data Processor consortium as is mentioned in the acknowledgements section of this chapter.

When published, this paper replaced the AstroHPC paper that was described in the previous chapter as the most recent and accurate published description of the SKA Science Data processor.

[1]ASTRON, the Netherlands Institute for Radio Astronomy
[2]Netherlands eScience Centre
[3]Vrije Universiteit Amsterdam

There is a clear progression visible from this chapter and the previous one. The bounds of the problem are now much more defined, clear evidence of the *bounding* proposition in active use. Furthermore, the design of the compute platform includes both compute- and data-transport components, following the *co-design* recommendation. A key component of this work is introduced in Section 4.3. Here, the SDP design priorities and principles are explained, of which cost is only one. This is a clear example where the *value* proposition is used, even though it was not defined in so many words yet.

Abstract

The Square Kilometre Array is a next-generation radio-telescope, to be built in South Africa and Western Australia. It is currently in its detailed design phase, with procurement and construction scheduled to start in 2017. The SKA Science Data Processor is the high-performance computing element of the instrument, responsible for producing science-ready data. This is a major IT project, with the Science Data Processor expected to challenge the computing state-of-the art even in 2020. In this chapter we introduce the preliminary Science Data Processor design and the principles that guide the design process, as well as the constraints to the design. We introduce a highly scalable and flexible system architecture capable of handling the SDP workload.

4.1 Introduction

Handling the data flow from the future Square Kilometre Array (SKA) radio telescope is one of the iconic IT challenges of the next decade. Phase one of this instrument will challenge the state of the art in high-performance computing (HPC) even in 2020, while the far more ambitious second phase is likely to be at the forefront of computing in the decades to come. The Science Data Processor (SDP) for the SKA is generally described as a large HPC system, but the requirements on the SDP are quite different from those on a general-purpose supercomputer. While some of these requirements are more stringent and require careful attention, the very targeted nature of the SDP system allows us to be much less generic in our design, potentially saving money and reducing energy consumption.

This chapter starts with an analysis of the requirements and constraints that bound the SDP design space. Based on these constraints, we define four SDP-wide priorities that guide our design work, and discuss some of the underlying principles for our detailed design. In this chapter we introduce a flexible but workload-driven system design philosophy that allows us to tune the SDP hardware to its specific set of tasks. The concept of SDP Compute Islands, independent and self-sufficient units that represent the basic building blocks of the Science Data Processor, is introduced next. Finally, we introduce a software-defined network to improve the flexibility and robustness of the data flow system. To explain the workload-optimised system design strategy, we first introduce and analyse the required workload.

4.1.1 The Square Kilometre Array

The SKA is a next-generation radio telescope, phase one of which is to be built in South Africa and Western Australia starting in 2017. This global project is currently in its detailed design phase. When completed, the instrument will consist of two distinct telescopes, optimised for low-frequency and mid-frequency observations. For a detailed description of the SKA1 system, we refer to the SKA1 baseline design [60] and the corrections thereof [94]. Here we suffice with a summary of the characteristics of the SKA phase one telescopes as shown in Table 4.1. In March 2015, the SKA underwent a major rebaselining [4], the consequences of which are still being evaluated. We provide an initial estimate of the computational requirements for this redesigned SKA, but these numbers are still being refined by the SDP consortium [1] .

	SKA1 mid	SKA1 low
Location	South Africa	Western Australia
Number of receivers	197 (133 SKA + 64 MeerKAT)	131,072 (512 stations x 256 elements)
Receiver diameter	15 m (13.5 m MeerKAT)	35 m (station)
Maximum baseline	150 km	80 km
Frequency channels	65,536	65,536
SDP input bandwidth	5.2 Tbps	4.7 Tbps
Req'd Compute capacity [2]	24 PFLOPS	5.7 PFLOPS

Table 4.1: SKA1 system characteristics. Input bandwidth includes protocol overhead and meta data. Required computational capacity is a work in progress estimate and does not take computational efficiency into account.

The SKA, in addition to being one of the premier science instruments of this century, is considered a major IT challenge. Table 4.1 shows the input bandwidth expected into the SDP facilities and the compute capacity required as indicated by our initial parametric modelling efforts [107, 2]. Computational efficiency is not taken into account, therefore in reality the installed system (peak) capacity needs to be several times larger. We expand on this in section 4.2.1.

Figure 4.1 gives a high-level overview of the SKA1 system, showing the two (distributed) telescope receivers, the Central Signal Processor (CSP, see Section 4.1.2) systems and the Science Data Processors. This chapter will concentrate on the Science Data Processor.

4.1.2 The SKA Science Data Processor

The Square Kilometre Array is an astronomical radio interferometer. Data from the antennas are transported to the Central Signal Processor, where the correlator produces cross-products for each antenna or station pair. These so-called visibilities are transported to the Science Data Processor, where they are calibrated and turned into sky

[1] See section 4.9 for a short overview of the work done since.

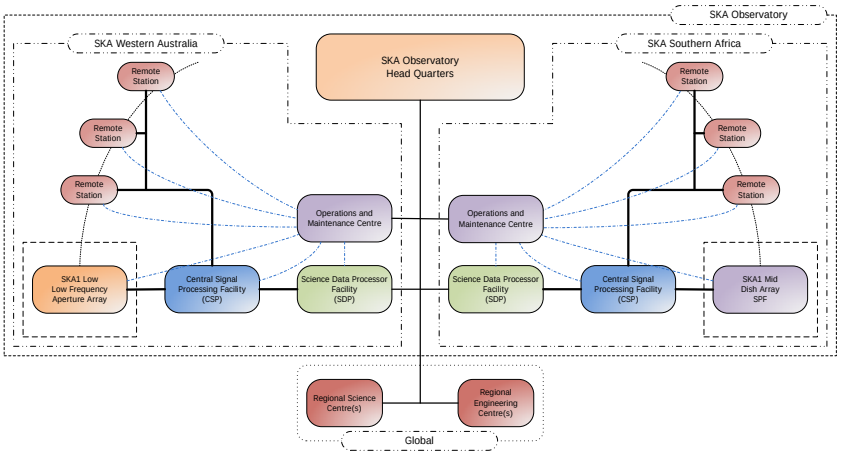


Figure 4.1: The Square Kilometre Array top-level system overview for phase one of the project. This figure is based on similar images by the SKA Office.

images. In the SKA Science Data Processor, bulk data is ingested from the Central Signal Processor, located at the telescope sites in the South African and Western Australian deserts several hundred kilometres away. Meta data is provided by the Telescope Manager, and merged into the bulk data stream at this stage, making the SDP internal data products self-describing.

Each visibility represents a point in the Fourier plane of the observed sky. Making sky images, and calibrating these, involves Fourier transforming these back into the image plane using a two-dimensional FFT, making sure the visibilities line up on the FFT grid (gridding) and applying corrections to these (convolution). A detailed discussion on the required processing is well outside the scope of this chapter. The interested reader is referred to the wealth of information available on the subject, in particular [107, 134].

The SKA SDP is responsible for receiving SKA data products from the CSP and for processing these into science-ready data products. Furthermore, the SDP is responsible for the safekeeping of these data products for the lifetime of the telescope and delivery of these to external entities. Finally SDP needs to compute calibration parameters and feed these back into the system.

One SDP instance will be built for each SKA telescope system, one in Perth, Western Australia, and one in Cape Town, South Africa. While the compute requirements and input bandwidths are similar for both telescopes, as shown in Table 4.1, compute characteristics such as required memory footprint and bandwidth may be different. However, to simplify design and operations, we aim to provide a single unified SDP design, which is shared between the SKA telescopes.

4.2 Requirements and constraints

The design of the Science Data Processor is bound by three main constraints: science, power and capital. First and foremost, the Science Data Processor is required to provide the systems and tools needed to meet the science requirements. The high-priority SKA science cases have been identified [163], but these are only described in limited detail [162]. Although much of the detailed information is missing, especially with respect to the implications of these science priorities, we can begin to sketch a requirements outline. The primary requirement on the Science Data Processor is that the system must be capable of efficiently running the processing pipelines required to reduce astronomical data. Models of the processing required to produce science-ready data have been developed, using current state-of-the-art algorithms, to estimate the required compute power [107, 2].

The locations of the SDPs are likely to impose a hard limit on the power that can be consumed without incurring very significant additional cost. Although no exact numbers are available yet, the SKA Office has indicated that these limits are not likely to exceed 5 MW per site. Furthermore, the operational budget for the SKA is bound to impose a limit on the amount of money that can be spent on electricity consumed by the SDP, which may translate into a lower soft limit on power consumption, which may be averaged over time.

Finally, as with any major science instrument, capital constraints are a major issue. The SKA board has approved a cost cap for the construction of the SKA1 of 650 million Euros. It is expected that the Science Data Processor will be allocated approximately 20% of this budget. This includes both SDP facilities, one for each telescope, and all software procurement and development needed, but excludes the building, cooling and power delivery. Software from existing precursor and pathfinder telescopes is not expected to scale to SKA requirements, which means that the SKA software will have to be rewritten almost entirely from scratch. This software development is likely to dominate the SDP budget, which means that it is expected that less than half the SDP budget will be available for hardware. To ensure optimal use of the hardware, and considering the software will need to be developed in parallel with the evolving hardware design, we will spend significant effort designing a system that provides maximum useful computational performance for minimal cost. The Science Data processor design needs to fit within at least these three constraints.

4.2.1 Defining the required SDP capacity

The required aggregate compute capacity of the SDP (R_{SDP}), assumed to be in double precision[2] floating point operations per second (FLOPS), is defined by:

$$R_{\mathrm{SDP}} = \frac{I_{\mathrm{bw}}\, q}{E}, \tag{4.1}$$

[2]This assumption, and the possibility of using mixed precision during some of the processing steps, is subject to further investigation.

where I_{bw} is the input bandwidth which is given in the baseline design [60]. q is the computational intensity[3] of the processing required in FLOP/byte, an estimate of which for each pipeline component is given in our parametric models [107]. Finally, E is the computational efficiency of those same algorithms in fraction of available peak performance (R_{peak}). Of these, computational efficiency is arguably the most difficult to estimate since it depends on many factors, such as chosen implementation, programmer talent, target platform and data access pattern. There is an element of hardware dependency in computational efficiency. This makes it almost impossible to give an accurate estimate for the SDP efficiency, the hardware of which will only be procured after 2020. There are currently no roadmaps, public or under NDA, that look that far into the future. Consequently we cannot say with any degree of certainty what hardware will be used for the SDP. We can investigate computational efficiency of the most costly algorithmic components, an estimate based on our current understanding of the required processing, on current day best-of-breed hardware. This shows very poor efficiency of **at most** 20% of R_{peak} [118, 3][4].

4.2.2 Preliminary timeline

While the SKA phase one project starts its construction phase immediately after finalising the detailed design, an analysis of the required compute capacity over time shows that building the SDPs for both telescopes can be postponed. Considering the blistering pace of developments in computing hardware, buying as late as possible has obvious advantages. In addition, this avoids having massive amounts of expensive operational hardware being idle. To support commissioning of the receivers and early science, we introduce the concepts of milli- and centi-SDPs. These are quite literally $\frac{1}{1000}$ and $\frac{1}{100}$ the size of a full SDP and will be designed and built not for efficiency but for convenience. It is important to note that the size of these initial SDP installations does not allow testing of our software at scale. Figure 4.2 shows the preliminary timeline for the SDP roll-out for the three systems.[5]

4.3 SDP design priorities and principles

The scale of the SDP surpasses that of all existing major science instruments. We take a pragmatic approach to ensure the feasibility of the SDP. In order, the Science Data Processor as a whole prioritises the following characteristics:

1. Scalability
2. Affordability
3. Maintainability

[3] Computational intensity is defined as the number of floating point operations per byte of data moved.

[4] This percentage has been the subject of much debate. We will discuss some of the more recent results in section 4.9

[5] Since publication this timeline has slipped considerably. Whereas construction was expected to start in 2017, this is now scheduled for 2021.

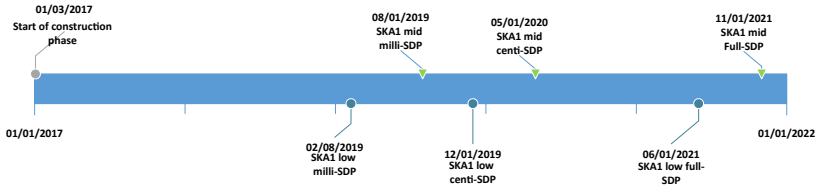


Figure 4.2: Preliminary roll-out schedule for both SDP systems, based on the preliminary roll-out schedule of the antennas.

4. Support current state-of-the art algorithms

To ensure the feasibility of the SDP, we will first and foremost focus on designing a scalable system. We will prioritise this even over an affordable system, since there is no use in having an affordable system if it cannot scale to the required size. Maintainability is a key challenge in this system, since it will be orders of magnitude larger than anything done before in radio astronomy. There are examples of similar sized systems, in terms of numbers of nodes, in HPC and cloud environments, but these have radically different requirements to SDP, which we will explore in more detail in the next section. Finally, we need to support, and more importantly size our system based on, current state-of-the-art algorithms. In other words, we cannot count on future developments in algorithm design to solve our problems, even though efforts in that area will obviously continue. Note that these priorities are not limited to the hardware design, but span the entirety of the SDP design.

4.3.1 SDP top-level design considerations

Taking into account the design principles introduced above, we make some key observations. The SKA SDP will be an integral part of an operational instrument, not a general-purpose HPC system, handling massive amounts of signal processing tasks. Some of these tasks will work on streaming high bandwidth data, some on buffered data. There is a near real-time component, handling the streaming data, and in general the instrumental nature of the system brings with it different reliability requirements compared to either HPC or cloud environments.

This fact can also be leveraged. Since we don't need to support all workloads, the SDP can be designed to exactly match the limited set of applications that it is required to run most effectively. Furthermore, experience with pathfinder and precursor instruments, LOFAR in particular [159], has taught us that the vast bulk of SDP-like processing is embarrassingly parallel in frequency and communication between tasks can be limited by parallelising in that dimension. In our system design we exploit this characteristic by designing a workload-optimised system.

We also observe that the scale of the SDP will greatly exceed that of existing large science instruments, such as the Large Hadron Collider [24]. Since the SDP is an integral part of an operational system, hardware failures, and the associated loss of scientific

data, may have an impact on science quality. A flexible data flow system that allows data to be easily redirected from failed SDP components is therefore essential to avoid having these disrupt operations.

4.4 Data flow model

The defining characteristic of the SKA Science Data Processor is the data flowing through the system. The streaming nature of data into the system from the CSP correlator, and indeed the bandwidth involved, is unprecedented. While the computational challenges faced by the Science Data Processor are significant, the data flow and relatively low computational intensity of the processing involved, make the problem particularly hard to solve. Since the data flow defines the SDP, it is logical to use the data flow, and in particular minimising this, as a key design priority. Data transport systems, in contrast to compute capacity, have the tendency to scale super-linearly in cost and energy consumption, which supports this decision.

Moving large volumes of data is expensive, in time, energy and required hardware. We therefore make use of the embarrassingly parallel nature of the SDP data flow and design the SDP system to minimise the (inherently large) flow of data. Data flow is directed such that all subsequent processing requires little or no additional (long-range) communication. The SDP is divided into numerous independent components, the Compute Islands described in section 4.6, that are sized to support end-to-end processing of the data directed to them. Figure 4.3 shows a high-level overview of the SKA SDP data flow.

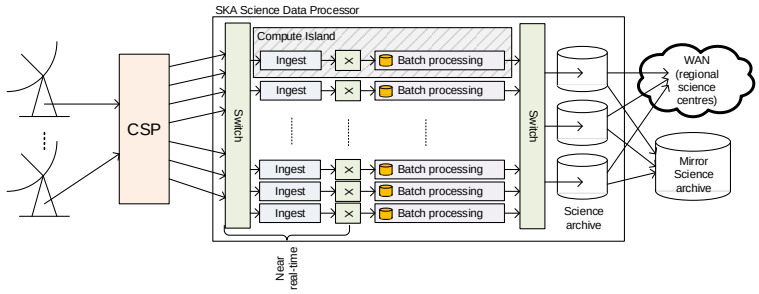


Figure 4.3: The SDP top-level data flow. Data flows into the SDP switches from CSP, where it is directed to the subscribed SDP component(s). After ingest and optional reordering of data through the Compute Island switch, identified by an X, data are buffered for iterative batch processing. Science ready outputs are stored in the science archive and exported to the world outside of SKA SDP.

The scale of the SDP means hardware failures will be a regular occurence. A flexible data flow system is essential to redistribute and redirect data flows around failed components in the Science Data Processor. On a high level, SDP components can be seen

as subscribing to data flows from CSP correlator entities. Every CSP *entity* produces a number of data streams, each representing a fixed chunk of visibility space. Each SDP component is responsible for a subset of visibility space by subscribing to these CSP streams, directed by the SDP local monitoring and control system.

4.5 Top-level network design

The top-level SDP network architecture is shown in Figure 4.4. Three distinct networks are shown:

- the bulk data network, handling data ingress from CSP
- low-latency network, handling potential data reordering and intra-island communication
- science archive network, handling data egress to the world outside of SKA SDP

While these are shown as distinct entities, they may share hardware resources. However, this must not impact performance of in particular the bulk data network, since the data stream from CSP is an unreliable UDP/IP based stream that does not support retransmission of lost packets. On the other hand, the ingress and egress networks are both used almost exclusively in one direction each, making sharing of hardware resources an obvious and attractive cost-saving option. A small-scale prototype will determine if this is indeed a feasible design option.

4.5.1 Software-defined networking in the SKA SDP

Experience with Ethernet-based precursor instruments, such as LOFAR, has shown that such infrastructures are static and fairly difficult to maintain [37]. The classic split between network and compute systems, in design, procurement, and maintenance, does not fit well in our data flow driven design philosophy. Since the data flow is the defining characteristic of the SKA Science Data Processor, network and compute systems must both be considered integral parts of one and the same system.

In addition to this, a classic Ethernet-based network imposes a very strong coupling between sending and receiving peers, in this case the CSP-based correlator, and SDP ingest. Any change in the data flow needs to be carefully negotiated between sender and receiver, which may be hundreds of kilometres apart. This contrasts with our desire for a flexible data flow environment to effectively handle failures in the SDP.

We therefore propose to build a software-defined network (SDN) infrastructure, which will become an integral part of the SDP dataflow, and will fall under the direct control of the SDP monitoring and control system. This means that the network is no longer a static piece of infrastructure, but may dynamically change configurations to suit the work flow requirements. Such a software-defined network also allows an effective decoupling of sending and receiving nodes. In this model, the sending peers, the CSP correlator nodes, effectively send to a virtual receiving node, which may or may not physically exist. Receiving nodes subscribe to data flows from the CSP, as directed

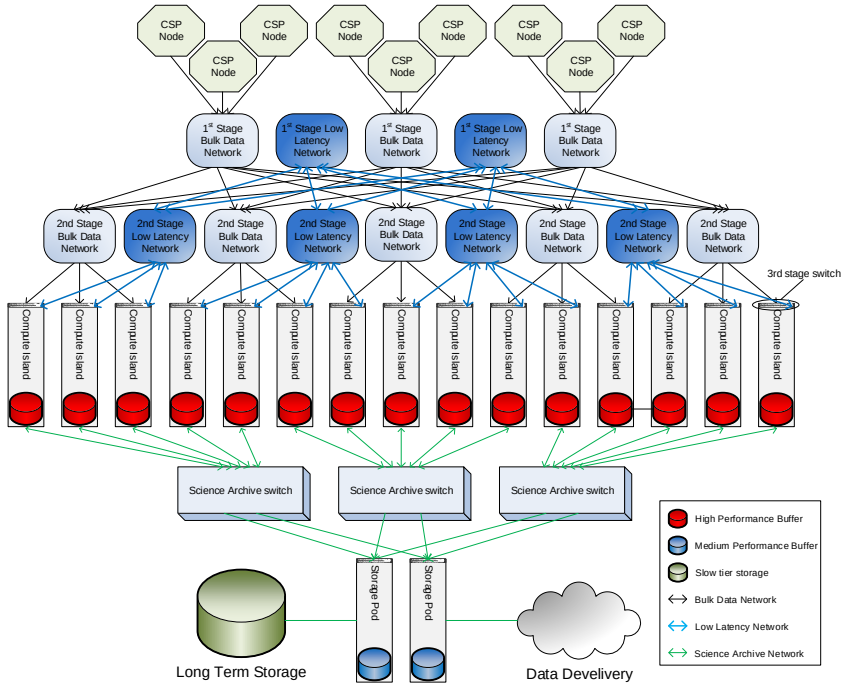


Figure 4.4: The SDP top-level network design

by the data flow manager. A software network controller directs physical data flows by having switches modify Ethernet headers in transit to match receiving peers: a classic publish-subscribe model, implemented in a network. Support for these technologies is currently available in many newer Ethernet switches from a variety of vendors. However, this is a novel approach to building a sensor network, that needs to be prototyped. A more in-depth discussion on the relative merits of this approach is given in a recent SDP memo [37]. In Chapter 6 we discuss this in detail.

4.6 Compute Islands

In this chapter we introduce the concept of a Compute Island, a self-contained, independent collection of compute nodes, as the basic replicable unit in the SKA SDP. Compute Islands are sized such that they need mostly to process data that is contained in the island itself and intercommunication between islands is limited. Some applications, such as multi-frequency synthesis, require a number of gathers to be performed before end-

products can be combined. However, at this stage data volumes are greatly reduced and a limited intra-island interconnect is sufficient to support this.

Figure 4.5 shows an overview of the Compute Island concept. Note that although a Compute Island is represented by a single rack of hardware in this figure, this is only illustrative. The actual size of the Compute Island may span multiple racks, or be limited to a fraction of a rack, depending on various parameters discussed in more detail in section 4.7.

A Compute Island consists of a number of interconnected Compute Nodes and associated infrastructure and facilities, such as master and management nodes, networks and filesystems. This makes each Compute Island self-sufficient and largely independent of the rest of the system. The characteristics of the Compute Nodes, in terms of compute resources, memory and storage resources, are defined by the application pipelines expected to run on them. In Figure 4.5 we show a current state-of-the-art host and accelerator system as a candidate Compute Node design, in which the CPUs handle the near real-time ingest processing and the accelerators the non real-time batch processing. Note that all components in the Compute Island are currently expected to be commercial of-the-shelf (COTS), both to reduce cost and to avoid lock-in. Most of the infrastructure will be similar between the two SDPs, but it is conceivable that the size of an island (e.g. the number of compute nodes within an island) or the compute node design itself differs between SDPs.

Within a Compute Island, a fully non-blocking interconnect, with a per node bandwidth far in excess of the per node ingest rate, is provided. This is primarily used for reordering data between processing steps, ideally within a single island. The same interconnect facilitates communication between islands for inter-island reordering or global processing, but in this case bandwidth will be much more limited and end-to-end transfers may require several hops.

4.7 SDP scaling

While the total useful capacity of the Science Data Processor depends on many components, we identify three defining characteristics that we will use to scale the system:

- Total computational capacity
- Computational capacity per Compute Island
- Characteristics per compute node.

The total computational capacity of a SDP, the aggregate peak performance (R_{peak}) expressed in PFLOPS, is defined by the number of Compute Islands that make up the Science Data Processor, a parameter that is freely scalable due to the Compute Islands' independent nature, and the capacity per Compute Island. While this number is a useful way to express the size of the system, its usefulness is limited since it does not take computational efficiency into account. Ideally, the total capacity of the system would be defined by the science or system requirements, but considering the constraints discussed above, it is more likely that total capacity will be defined by the available budgets (energy, capital or operational).

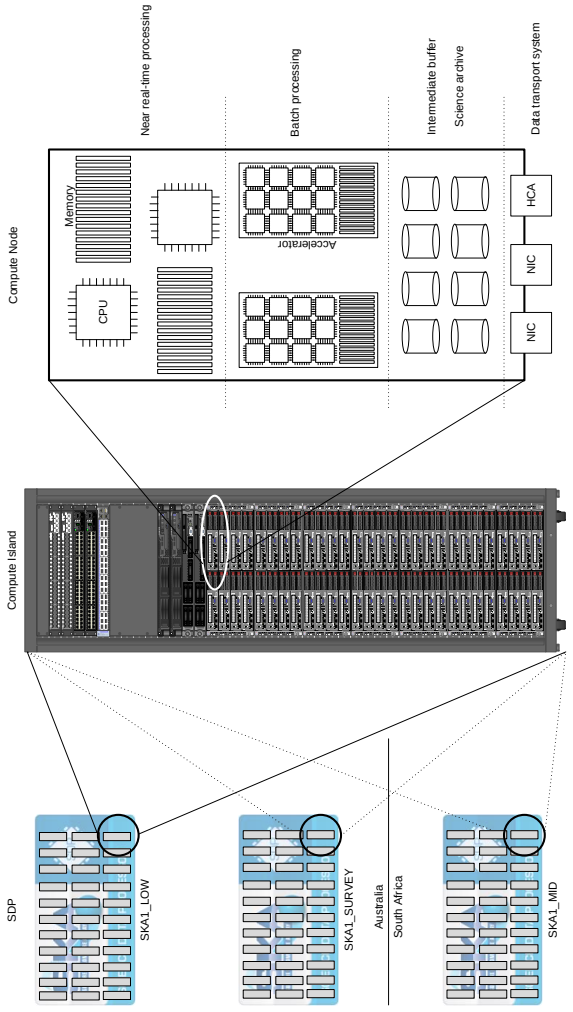


Figure 4.5: SDP scaling using Compute Islands. Each SDP instance contains many self-contained Compute Islands, each containing multiple compute nodes.

Capacity per Compute Island is defined by the number of compute nodes per Island, and the performance characteristics of these nodes. This capacity is expressed in terms of peak computational capacity, i.e. TFLOPS, but it is likely that computational capacity will not drive the sizing of the Compute Islands. Island capacity is defined by the most demanding application, in terms of required memory (capacity or bandwidth), network bandwidth, or compute capacity that requires a high-capacity interconnect.

The basic building block of a Compute Island is the Compute Node. The characteristics of these nodes are defined by the design equations in [107] but within these bounds a vast number of valid node designs can be identified. Considering the timeframe of the SDP roll-out, which extends well beyond the available industry roadmaps, the node definition is perhaps the least well-understood component of the SDP design. The SDP parametric model defines a number of ratio rules that describe suitable node designs. Within the bounds of these rules, cost, energy efficiency and maintainability are considerations that may be used to select optimal node implementations.

There is one key requirement that a compute node needs to satisfy: if used to ingest data, only a very small percentage of that data may be lost. In other words, these nodes need to be scaled such that they comfortably satisfy the ingest real-time requirements and a sufficient number of these nodes need to be available to receive all data from the CSP.

One interesting consideration is whether or not both SDPs will be standardised on a single node design. Answering this question requires a trade-off between the standardisation of components on the one hand, and workload optimisation of those same components on the other hand. Operational costs, in particular energy versus deployment and maintenance cost, will also play a key role in this decision. It is clear that this decision cannot be made until more information is available on the likely technology options available for nodes.

4.8 Conclusion and discussion

In this chapter we present an overview of the design considerations and constraints for the SKA Science Data Processor. This chapter analyses the design constraints put on the SDP hardware and identifies a number of key design priorities that guide the design process. We present an initial, highly scalable, preliminary design for the SDP which should both be suitable and scalable while minimising procurement and operational costs.

The preliminary design, presented in this chapter, satisfies all of these constraints. The independent and self-sufficient nature of the Compute Islands make the design extremely scalable. This modular approach also aids maintainability, since it allows for easy replacement of failed components. Our flexible data flow model, thanks to the software-defined network, is also tailored specifically to account for failures. The focus on hardware/software co-design and COTS components make for a system that is as affordable as possible.

Although we are confident in the suitability of our design, the detailed design is still in flux. Our timeline for construction of the full systems in 2021 is well beyond any industry roadmap at the time of writing, which makes technology selection difficult.

This also makes the scale of the SDP very difficult to estimate, since computational efficiency is very hardware dependent. However, the preliminary design presented in this chapter is scalable to such a degree that we feel confident that it can act as a good basis for the detailed design during the next couple of years.

4.9 Current status

Since this chapter was published, the SKA Science Data Processsor consortium has progressed considerably. In January 2019 it passed its Critical Design Review based on a large body of work addressing some of the issues mentioned in this chapter.

As part of this review process an updated system sizing estimate was produced. This revealed an updated requirement for 13.6 PFLOPS for the SKA1-Low instrument and 11.5 PFLOPS for SKA1-Mid [17], based on continued and extensive modeling of the pipelines involved [2]. While the computational requirements were significantly changed compared to the initial numbers shown in this chapter, the scalable architecture introduced was sufficiently flexible to accomodate these changes [35].

Computational efficiency that can be achieved in the Science Data Processor was a controversial and heavily debated subject. This is in part due to the complex work flows that will be run on the system, but also due to uncertainty about the hardware platform that will be procured during contruction. A summary of the prototyping efforts concludes that the lower bound of the achievable computational efficiency is around 10% of R_{peak}, with the upper bound, for some algorithms, being around 40% [41]. This conclusion was reached based both on roofline analysis and hardware benchmarking of various state-of-the-art algorithms on different hardware platforms.

4.10 Our propositions in this chapter

Having looked at developments since the publication of the paper this chapter is based on, we can now direct our attention to our propositions. In the next sections we show how the three applicable propositions are supported by this chapter.

4.10.1 Bounding proposition

This chapter shows a clear progression compared to the previous one. The bounds to the system are much more clearly defined, and these are clearly articulated in this chapter. Section 4.2 identifies required compute capacity, the likely hard limit on the available energy on the site, the budget on the first phase SKA and an initial timeline. Furthermore, Section 4.3 defines a number of design principles and priorities that can also be considered bounds on the architecture. These priorities and principles, summarised as *scalability, affordability, maintainability and support of current state-of-the-art algorithms* drive the further initial design of the SKA SDP, as described in this chapter.

4.10.2 Co-design proposition

This chapter also shows the initial SKA Science Data Processor design. In this chapter we go to some lengths to not only quantify the required compute capacity, but also define how data flows through the system. Section 4.4 defines both the external interfaces of the Science Data Processor, and ways data flows internally. This analysis results in a top level design, shown in Figure 4.4, that focuses on data flow, with the required compute capacity being delivered by basic building blocks known as Compute Islands, as shown in Figure 4.5. Clearly the *co-design* proposition is used in the entire design process.

4.10.3 Value proposition

The initial SKA Science Data Processor architecture that is discussed in this chapter is designed to maximise the value potential in several ways. First and foremost we should point out that the design principles and priorities adopted from the very start are a balanced mix of cost and value, as recommended in this thesis. The priorities mentioned, in particular scalability and maintainability, were driven by value considerations. Furthermore, while the design is mostly limited to high-level concepts due to the limited amount of information, both with respect to requirements and available components at the time of procurement, the data flow structure is well established. This is indended to minimise the risk of data flow bottlenecks, maximising the value potential of the eventual system, and can be used as a solid basis for a scalable compute system design. Finally, we point out that, while the architecture is fairly generic, the node characteristics have to be defined by the performance profiles of the applications. These result in design equations, which define ratios between compute, memory capacity and bandwidth and external I/O, among other things. Based on these, a value potential of a node can be determined.

Acknowledgements

This work is based heavily on the SKA SDP Preliminary Design Review documentation, specifically the compute platform component [36]. In addition, many of the concepts introduced in this chapter, in particular the four design priorities in Section 4.3, evolved during discussions with the rest of the SDP consortium. This work is supported by the ASTRON/IBM Dome project [38], funded by the province Drenthe and the Dutch Ministry of EL&I.

CHAPTER 5

COBALT: a GPU-based correlator and beamformer for LOFAR

P. Chris Broekema [1], *J. Jan David Mol* [1], *R. Nijboer* [1], *A.S. van Amesfoort* [1], *M.A. Brentjens* [1], *G. Marcel Loose* [1], *W.F.A. Klijn* [1], *J.W. Romein* [1]

Context and contribution

This chapter is slightly modified from a paper that was published in Astronomy and Computing in 2018. While it documents work done by all authors, the focus of the paper is on the hardware design aspects of the project, which was led by Broekema. Broekema is the first author, defined the focus and structure of the paper and has written the majority of the paper. Sections 5.7.1 to 5.9 were were written by Mol, with significant input from Broekema and van Amesfoort (Section 5.7.1). Mol was also instrumental in gathering and verifying the information used in section 5.12 This work documents the COBALT project, in which Broekema acted as the lead hardware engineer.

The design process documented in this chapter clearly follows the recommendations introduced in this thesis. Bounds of the system are defined and articulated as recommended by the *bounding* proposition, and an explicit choice is made to very carefully consider data-flow as well as required computational resources in the design (*co-design* proposition). Furthermore, the architecture of the COBALT system and the selection of its components clearly focuses of a number of value metrics, in particular optimal data flow, other than just cost, as recommended by the *value* proposition.

[1] ASTRON, the Netherlands Institute for Radio Astronomy

Abstract

For low-frequency radio astronomy, software correlation and beamforming on general purpose hardware is a viable alternative to custom designed hardware. LOFAR, a new-generation radio telescope centred in the Netherlands with international stations in Germany, France, Ireland, Poland, Sweden and the UK, has successfully used software real-time processors based on IBM Blue Gene technology since 2004. Since then, developments in technology have allowed us to build a system based on commercial off-the-shelf components that combines the same capabilities with lower operational cost. In this chapter we describe the design and implementation of a GPU-based correlator and beamformer with the same capabilities as the Blue Gene based systems. We focus on the design approach taken, and show the challenges faced in selecting an appropriate system. The design, implementation and verification of the software system shows the value of a modern test-driven development approach. Operational experience, based on three years of operations, demonstrates that a general purpose system is a good alternative to the previous supercomputer-based system or custom-designed hardware.

5.1 Introduction

The LOw Frequency ARray (LOFAR) [159] radio telescope is often described as one of the first of a new generation of software telescopes. LOFAR has pioneered the use of a combined software correlator and beamformer in an operational radio telescope since 2004 [122, 123, 98]. One key characteristic of a software telescope is the ability to ride the technology wave to increase functionality and/or reduce operational cost by leveraging new developments. In this chapter we discuss the hardware design of the third generation Graphics Processing Unit (GPU) based LOFAR software correlator and beamformer: Cobalt (**CO**rrelator and **B**eamformer **A**pplication for the **L**OFAR **T**elescope), as well as the design and development of the associated software.

Since the tasks of this real-time central processor are well known and clearly defined, this application is an excellent candidate for a focused hardware/software co-design approach.

In this chapter we describe the following concepts that in combination led to the success of Cobalt:

- A data flow-driven design philosophy;
- Hardware/software co-design;
- Data flow analysis and task mapping to identify potential weaknesses in available HPC solutions for our streaming application;
- Close public-private collaboration in the hardware design, which showed the clear advantages of such a partnership in this kind of project;
- A simplified system engineering approach in the design and implementation of the project;

- An agile software engineering methodology to ensure timely delivery within budget;
- A Test-driven software development process to improve the robustness of our system.

5.2 Related work

The Cobalt project built on previous experience with combined software correlator and beamformer systems in the LOFAR telescope [122, 123, 98]. We discuss some aspects of these in more detail in Section 5.3.1. Cobalt shared common ancestry with the AART-FAAC correlator [113], although the radically different I/O ratio led to different design decisions.

There are several other software correlators in use in radio astronomy. Here we briefly discuss some of these in relation to the Cobalt system. We limit ourselves to FX-correlators, that combine a filter- and Fourier transform (F) stage with a cross-correlation (X) stage

The correlators used by the Murchison Widefield Array (MWA) [109], the Large Aperture Experiment to Detect the Dark Ages (LEDA) [85], and PAPER [110] all share the same general architecture. Whereas Cobalt implements both the filter (F-stage) and the correlator (X-stage) in GPUs, the above mentioned instruments employ a hybrid FPGA-GPU approach. The F-stage is implemented in FPGA, the X-stage is implemented in GPUs using the xGPU library [46]. A high bandwidth switch connects the F- and X-stages.

The Giant Metrewave Radio Telescope (GMRT) real-time software backend [126] uses a structure similar to the MWA correlator, with nodes dedicated to three specific tasks. In this case the software backend relies on conventional CPUs only, with heavy use of off-the-shelf performance optimised libraries.

For Very Long Baseline Interferometry (VLBI) a number of software correlators have been developed. Examples of these are SFXC, developed by JIVE [82], and DiFX [55, 54]. These perform tasks similar to Cobalt, although DiFX doesn't include a beamformer. However, data rates are usually modest compared to those generated by the LOFAR stations.

A real-time software correlator has been developed and deployed for the Canadian Hydrogen Intensity Mapping Experiment (CHIME) pathfinder [56]. The correlator stage of this system is very similar in concept and size to Cobalt, but implements the F-stage in FPGAs, requiring additional communication from the F to the X stage. In Cobalt the F and X stage use the same hardware.

5.3 LOFAR: the Low Frequency Array

LOFAR, the LOw Frequency ARay, is a new-generation radio telescope, built in the northern part of the Netherlands, with international stations distributed across Europe. As the name suggests, it is designed to observe in the relatively low and unexplored frequency range of 10 to 250 MHz. The array consists of 40 stations: 24 core and 16

remote, in the Netherlands, with an additional 13 international stations, 6 in Germany, 3 in Poland and one each in France, Ireland, Sweden, and the UK.

Each station consists of two receiver types, low band dipole antennas and high band antenna tiles, covering either side of the commercial FM band. A LOFAR station consists of 96 Low Band Antennas (LBAs), operating from 10 to 90 MHz. In addition, Dutch core stations have 48 High Band Antenna (HBA) tiles in two clusters that cover the frequency range from 110 to 250 MHz. Remote stations in the Netherlands have the same number of HBA tiles, in a single cluster. International stations provide a single cluster of 96 HBA tiles.

At each LOFAR station dedicated processing equipment samples, digitises and digitally filters data using a polyphase filter bank. This filterbank produces 512 frequency bands with a spectral bandwidth of 195 kHz. By coherently adding the same frequency bands of individual antennas or tiles, station beams are created. Such a beamformed frequency block is referred to as a *subband*, 488 of which may be selected per observation, giving a total spectral bandwidth of 95 MHz (in the most common 8-bit mode). Spectral bandwidth may be exchanged for beams, essentially allowing up to 488 independent (narrow-band) pointings to be made. Core stations may be split, allowing the two HBA clusters to be treated as smaller, but fully independent, stations. To distinguish them from physical stations, these are called *antenna fields*, 77 of which currently make up the LOFAR array. Subbands produced by these antenna fields are transported to the central processing facility, hosted by the University of Groningen, about 50 kilometres from the LOFAR core area, using UDP/IP over many 10 Gigabit Ethernet links.

The central processor can be divided into three distinct components: the real-time processor, the post-processing cluster, and the archive. The real-time processor (Cobalt), which implements a correlator and beamformer, is a soft real-time system that collects data from the antenna fields, conditions this data, applies a second polyphase filter, and subsequently combines all antenna fields (beamformer mode) or all antenna field pairs (correlator mode) to produce intermediate results. Although there is no hard deadline in the sub-second range as in a classic real-time system, it is required that the central processor keeps up with the antenna field data streams, otherwise data is irretrievably lost. In Section 5.3.2 we discuss the processing steps that make up the real-time system.

Output from the real-time processor is stored on the post-processing cluster. This is a conventional Linux cluster, with significant disk capacity to store intermediate products and facilitate further processing. Here, instrument calibration is performed, possible interference is identified and removed, and final products (images, pulse profiles, source lists) are created.

Final products are exported to the LOFAR long-term archive, which is currently distributed over three sites: Amsterdam hosted by SURFsara in the Netherlands, Jülich hosted by Forschungszentrum Jülich in Germany, and Poznań hosted by the Poznań Supercomputing and Networking Center in Poland. Astronomers retrieve their data from one of these archive sites, no end-user interaction with the LOFAR system is required. Figure 5.1 shows a top-level overview of the LOFAR system.

The remainder of this chapter will focus on the LOFAR real-time processor.

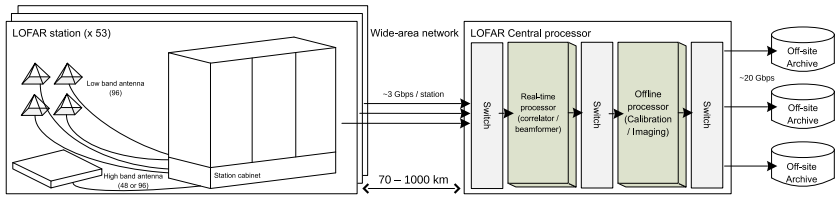


Figure 5.1: Top-level overview of the LOFAR system.

5.3.1 The LOFAR real-time processor

The LOFAR project decided early on to employ general purpose computing for the real-time processor, both to exploit the fact continued developments in general purpose processor technology had made this feasible, and to save precious FPGA development resources for the station processing boards. Initially, the requirements for the LOFAR real-time processor were quite challenging for a general purpose compute system. The only feasible option available was to use a supercomputer. In 2004, an IBM Blue Gene/L was installed at the LOFAR central processor. At the time, the LOFAR real-time processor was the fastest supercomputer in the Netherlands, and the second fastest in Europe [153]. Although compute performance of the Blue Gene/L was sufficient, significant research and development was required to achieve the required I/O performance [122, 79].

In 2008 the six rack Blue Gene/L system was upgraded to a slightly more powerful, but much smaller and more energy-efficient three rack Blue Gene/P system. While a significant improvement over its predecessor in terms of programming environment and general hardware features, considerable research was again required to achieve the I/O performance required [119, 168, 169].

While the Blue Gene real-time processors were operational, research continued into various other software alternatives [123, 106]. The advent of many-core architectures for high performance computing, in particular Graphics Processing Units (GPUs), allowed us to move away from supercomputers, and instead build the third-generation correlator and beamformer based on general-purpose server hardware and accelerators. In this chapter we discuss the design approach taken, we show some of the problems encountered and how these were tackled, and we conclude with the successful commissioning into operational service of a new, GPU-based, LOFAR correlator and beamformer. Several years of operational statistics are presented in Section 5.9.

5.3.2 Processing steps

The LOFAR real-time processor receives data from LOFAR antenna fields as continuous UDP/IP data streams. Missing and out-of-order packets are identified and, where possible, corrected. Data that has not arrived after a short deadline is considered lost. Each incoming data stream contains all frequency data from a single antenna field, while

each processing node for a given frequency range requires data from all antenna fields. Therefore, a transpose is required on the incoming data, before further processing.

The processing component is complex and involves a number of optional sub-components, as shown in Figure 5.2. Data is converted from fixed to floating point to better match the hardware available in a general purpose computer. The current LOFAR real-time processor uses single precision complex floating point throughout, with one exception: calculating delays for delay compensation. Two main pipelines are implemented that can run in parallel. The correlator pipeline implements an FX-style correlator, the components of which were described in more detail in earlier work [123]. The beamformer pipeline consist of coherent and incoherent components, as well as a complex voltage pipeline, details of which were previously published as well [98]. In Section 5.7.5 we describe how these processing steps are implemented in Cobalt.

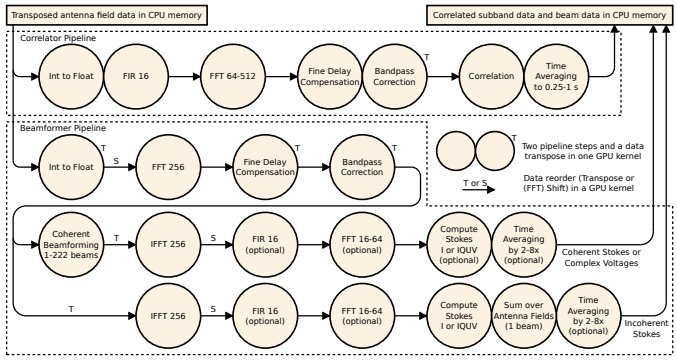


Figure 5.2: Signal processing steps in the correlator and beamformer pipelines.

5.4 Development process

The relatively modest scale of the project allowed us to use a slightly simplified system engineering approach in the design of this system. First, the system requirements, both functional and non-functional, were identified (see Section 5.5). From these, a high-level architecture was derived. This, combined with a detailed analysis of the various aspects of the application performance profile, such as network data flow (Section 5.6.2), memory footprint (Section 5.6.3) and computational load (Section 5.6.1 and Section 5.6.4) led to a detailed design of the system hardware (Section 5.6.8). During the hardware implementation phase, a single prototype node was used to verify that the selected hardware implementation met the performance requirements (Section 5.6.5). Finally, the fully deployed system was verified against the system requirements (Section 5.8). This process closely mirrors a traditional systems engineering approach, but the relatively small project size meant we could simplify the process by eliminating most of the formal documentation.

Before the start of the Cobalt project the feasibility of a GPU-based solution was researched and optimised GPU kernels had been developed. Moreover, a highly optimised and proven Blue Gene implementation of all required functionality was available. However, a different hardware architecture and steep performance, maintenance and reliability requirements, necessitated redesign of our on-line processing software, except for wrapper and support libraries. The beamformer pipeline was redesigned, so several GPU kernels had to be adapted or rewritten.

The development process was paramount to obtain correct output and adequate performance within a limited time frame. We used the Agile/Scrum development process [138, 139] to focus a small software team on a common goal, divide and plan remaining work, and periodically tried to improve our practices.

5.5 System requirements

The following hard and soft requirements were put on the Cobalt system. In terms of functional requirements, the Cobalt system *must*:

- be able to correlate 64 antenna fields in 16 bit mode and in 8 bit mode at full bandwidth (for a single beam), i.e. with 244 resp. 488 subbands, down to 1 sec time resolution and at maximum 256 channels per subband frequency resolution. In this mode up to 8 independent beams can be made, in which case the number of beams, the total bandwidth and the number of bits have to be traded against each other.
- be able to create 127 time domain data streams using all 48 core antenna fields in 16-bit mode at full bandwidth. These can be recorded in one of three modes: 1) coherent addition (Stokes I only or Stokes IQUV, referred to as a coherent tied-array beam), 2) incoherent addition (Stokes I only or Stokes IQUV, referred to as an incoherent tied-array beam), or 3) coherent complex voltage (XX, XY, YX, YY). Time resolution can be traded against frequency resolution, within the resolution of a subband.

In addition, there were the following non-functional requirements. The system must be delivered in time and within budget. It must have hardware, software, and data input/output connections installed, tested, and debugged in a staged approach. The system must have a design that allows to scale up and be prepared for future planned modes and functionality. Furthermore, the system must have an operational availability greater than 95% (excluding planned service), while having a system maintenance staff effort of less than 0.25 FTE, delivered during business hours only. The total operating costs per year must be lower than 50% of the (one-time) capital investment costs.

The non-functional requirements on Scalability, Operational Availability (i.e. robustness) and Maintenance Effort (i.e. maintainability) translated into software quality, programming environment, software support, test environment, non-monolithic design, etc.

In addition to the hard requirements there were the following soft requirements, i.e. nice-to-haves. The Cobalt system *should* be able to handle more LOFAR data, such

as e.g. doing parallel LBA and HBA observing (doubling the number of available subbands, and doubles the required Cobalt capacity for a given observation), correlating more antenna fields (up to 80), correlating longer baselines (up to 3500 km), creating more beams (up to 200), or operating in 4-bit mode (which would double the number of available subbands, at the cost of some dynamic range, again doubling required Cobalt capacity for a given observation).

It should have the capacity to handle additional online tasks, e.g. Fly's-eye mode in which we store antenna field data without central beamforming, handling of more than 8 independent beams in correlator mode, online flagging, and beamforming the six central stations ("superterp") before correlation. Cobalt should also have the capacity to handle additional offline processing tasks including automatic flagging, (self-) calibration and averaging, coherent de-dispersion of pulsar data and production of dynamic spectra and additional parts of the pulsar pipeline: online folding and online searching.

Finally, Cobalt should prepare for future extensions, such as commensal observing, parallel observing with sub-arrays, responsiveness to triggers and interrupts, and additional observing modes.

5.6 Hardware design and implementation

In the design process we focused on data flow rather than compute requirements. One of the key characteristics of the LOFAR real-time processor is a relatively high data rate. While making sufficient compute capacity available was a key requirement, efficient use of this capacity critically depends on efficient data flow through the system. Furthermore, previous many-core correlator research meant that the computational requirements and challenges were relatively well understood. We therefore made the conscious, if somewhat counter-intuitive, decision to focus our hardware design for the Cobalt system on data flow, with computational capacity a crucial but secondary design goal.

5.6.1 Hardware requirements and design priorities

Cobalt was intended to be a drop-in replacement for the existing Blue Gene based correlator and beamformer for LOFAR. As such, the primary requirement for Cobalt was to provide performance equal to the previous system. In addition, the desire to increase the number of antenna fields that can be correlated from 64 to 80 was expressed. Table 5.1 summarises the top-level hardware requirements for Cobalt.

Component	Requirement	Design target
Input bandwidth	$\sim$192 Gbps (64 antenna fields)	$\sim$240 Gbps (80 antenna fields)
Output bandwidth	80 Gbps	$\geq$80 Gbps
Interconnect	input + output bandwidth	$>$2 * (input + output bandwidth)
Correlator	64 antenna fields, 244 (16-bit) - 488 (8-bit) subbands	80 antenna fields, 244 (16-bit) - 488 (8-bit) subbands
Beamformer	127 beams, 48 antenna fields, 244 (16-bit) subbands	200 beams, 48 antenna fields, 244 (16-bit) subbands

Table 5.1: Top-level hardware requirements for Cobalt

In order to translate these requirements into a system design, we estimated the required compute capacity. The main contributors were expected to be:

- correlator
- polyphase filter bank (essentially many Finite Impulse Response (FIR) filters, feeding into a Fast Fourier Transform (FFT))
- bandpass and clock corrections

Figure 5.3(a) shows how we expected the compute load to scale with the number of LOFAR antenna fields. This figure takes the theoretical compute load of each of the contributors, and takes into account a rough estimate of the achievable computational efficiency[1]. Extensive prototyping, as well as experience with previous software correlators, showed that the correlator itself can be highly optimised [122, 123, 106]. Computational efficiencies well above 90% of theoretical peak performance have been observed. The FFT on the other hand is notoriously inefficient. So while the computational complexity of the correlator is $\mathcal{O}(N^2)$, compared to $\mathcal{O}(N \log N)$ for the FFT[2], the effective contribution to the computational load of both is much closer than these theoretical computational complexities suggest. For the purposes of this estimate, we assumed a computational efficiency of 90% for the correlator and 15% for the FFT, based on the published performance of the CuFFT library. The other components were not expected to contribute much to the required compute resources, therefore a computational efficiency of 50% for all other contributors was used.

The efficiencies quoted here are much higher than the ones mentioned in the previous chapters, due to different application profiles run. The Cobalt system is a correlator and beamformer, as described in 1.1, whereas the SKA Science Data Processor described in chapters 3 and 4 are examples of intermediate processing systems.

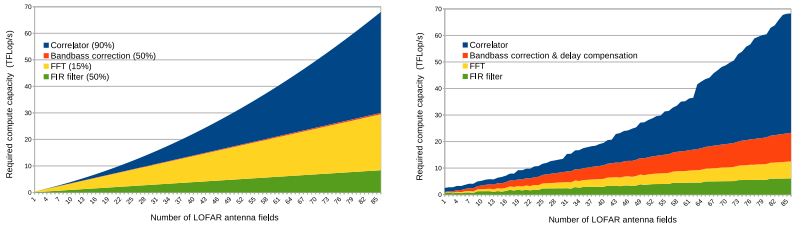
In Figure 5.3(b) we show measured performance scaling of the operational Cobalt system. While the total consumed resources are very close to the estimate in Figure 5.3(a), there are some marked differences. The cost of the FFT was significantly overestimated, due a the conservatively chosen complexity. Both correlation and FIR filters are close to the estimate, but bandpass correction consumes much more compute resources than estimated. This is due to the addition of delay compensation, compensating earth rotation, but more importantly due to a performance regression discussed in more detail in Section 5.8. However, Figure 5.3(b) shows that the current implementation fits within the available system resources, and therefore further optimisation is not necessary. We also note the value of conservative scaling estimates, to account for unexpected regressions during the implementation phase.

5.6.2 Design and data flow

Having identified the top-level requirements of the system, we derived detailed requirements and identified possible suitable implementations. While several options were

[1] Computational efficiency is defined as the percentage of the theoretically peak performance that is obtained in practice.

[2] To complicate matters further, note that in these complexity measures N may not necessarily refer to the same parameter.



(a) Predicted compute scaling (assumed computational efficiency in parentheses).

(b) Measured compute performance (November 2017).

Figure 5.3: Predicted and measured scaling of required compute capacity against number of processed LOFAR antenna fields.

considered, analysis showed that a highly integrated system where a single node type will handle all tasks, was the most attractive solution. Each node will therefore need to

- receive data from LOFAR antenna fields
- transpose this data
- run a GPU correlator and/or beamformer
- send the resulting data to the post-processing cluster

A much simplified representation of the data flowing through the Cobalt system is presented in Figure 5.4. Station data streams into the system at up to 240 Gbps, using UDP/IP over Ethernet. Output data is sent to the storage cluster, also using Ethernet, but here we are free to choose a reliable protocol such as TCP/IP instead.

Within the Cobalt system we need to fully reorder the data. The data from antenna fields contains all frequency bands for a single antenna field, while the correlator requires data from all antenna fields for a single frequency band.

The LOFAR core network is based on 10 Gigabit Ethernet (GbE) technology. Data from up to three LOFAR antenna fields is sent through each 10 GbE link. Combined with the required number of supported antenna fields, this gives a lower bound on the number of 10 GbE ports we require in Cobalt (a minimum of 22, we designed for at least 27). At this stage of the design process we considered 40 Gigabit Ethernet a viable and more dense alternative to four 10 GbE ports.

The reordering of large volumes of data was considered a risk. The efficiency of such a transpose, and the achievable bandwidth of the required low-latency interconnect, are difficult to estimate. To mitigate this risk, we considerably over-dimensioned the network intended for this operation. The transpose bandwidth is the same as the input bandwidth. Our design target was to provide double the input Ethernet bandwidth specifically for the transpose. Each Fourteen Data Rate (FDR) Infiniband Host Channel Adapter (HCA) provides a theoretical maximum achievable bandwidth of 54.54 Gbps.

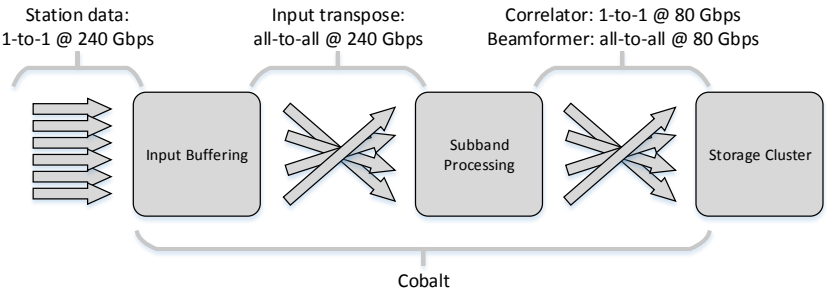


Figure 5.4: Data flow from and to external systems, as well as within Cobalt.

We therefore designed our system to provide one FDR Infiniband port for every two 10 GbE ports, noting that this ratio needs to apply for every node.

5.6.3 Memory bandwidth

The Cobalt system is characterised by a sustained and high rate of data streaming into the system. This data stream needs to be received, conditioned and processed without loss. Modern general-purpose operating systems are inherently inefficient at receiving data, due to the need to copy data several times before an application can access it[3]. This puts a considerable load on the memory subsystem, in particular on the available memory bandwidth. Figure 5.5 shows the way the various tasks described in Section 5.6.2 were expected to be mapped on hardware. We noted that the main memory bus was a potential bottleneck.

An analysis of the memory bandwidth requirements was undertaken to estimate the system requirements in this respect, based on the input bandwidth and the number of times data is expected to be copied. Handling of input was expected to drive this requirement, all other tasks combined were estimated to take less memory bandwidth. The impact of hitting a memory bandwidth bottleneck was estimated to be high, we therefore took a conservative approach and limited maximum memory bandwidth use to 50%. Caching effects may positively affect used memory bandwidth, but are exceptionally unpredictable and were therefore not considered. This, combined with the available memory bandwidth in the most recent Intel Xeon generation available at the time, gave us a lower bound on the minimum number of processors, and thus nodes, needed in the system. Cobalt would require a minimum of six dual socket nodes in order to provide the required memory bandwidth, and our design target required eight dual socket nodes.

[3]While this is an essential security feature, avoiding this potential bottleneck, for instance by the use of Remote Direct Memory Access (RDMA), is an active area of research.

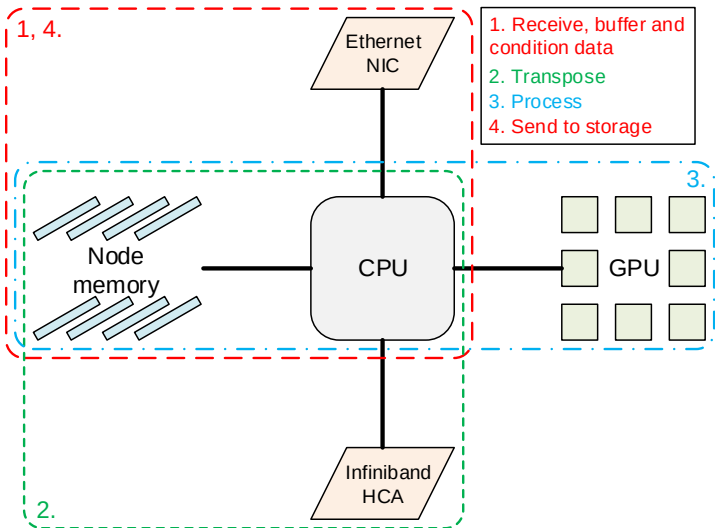


Figure 5.5: Mapping of the various Cobalt tasks onto node hardware. This shows that node main memory, and in particular the memory bus, is used for each task, highlighting a possible bottleneck.

5.6.4 Selecting the accelerator

In selecting a suitable accelerator, we evaluated device specifications, performance of prototype code (per Watt and per Euro), software quality and programming environment. Three vendors were evaluated: NVIDIA, AMD and Intel, with a total of four devices investigated in more detail.

Although Intel's Xeon Phi was commercially available at the time, prototype code on this accelerator performed poorly due to the early state of the software stack. It was therefore not considered further. Both NVIDIA and AMD had two devices available that would suit our applications. The AMD FirePro S9000 and S10000, as well as the NVIDIA Tesla K10 and K20X were evaluated in more detail, shown in Table 5.2.

	NVIDIA Tesla K10	NVIDIA Tesla K20X	AMD FirePro S9000	AMD FirePro S10000
Architecture	Kepler	Kepler	Tahiti PRO	Tahiti PRO
GPU	2x GK104	1x GK110	Tahiti PRO GL	2x Zaphod
Single Precision (GFLOPS)	4577	3935	3225.6	5913.6
Double Precision (GFLOPS)	190.7	1312	806.4	1478.4
Memory (MB)	2x 4096	6144	6144	2x 3072
PCIe	PCIe 3.0 x16	PCIe 2.0 x16	PCIe 3.0 x16	PCIe 3.0 x16
Programming	Cuda	Cuda	OpenCL	OpenCL

Table 5.2: Considered GPU options.

	Minimum	Design target	Proposed Cobalt
Nodes	6	8	8
10 GbE ports	22	27	32
FDR HCAs	11	14	16
NVIDIA Tesla K10s	10	14	16

Table 5.3: Detailed lower bounds for the Cobalt system.

Experience with prototype code showed that AMD devices generally performed better, but software and drivers stability for these devices was a potential problem. This was considered unacceptable for a system that is an integral part of an operational instrument. The NVIDIA devices, although providing less computational performance, were superior in terms of stability, software quality and programming environment. The Cobalt system does not require extensive double precision floating point support. The data-driven nature of the processing made support for PCIe v3 a secondary requirement, which K20X does not support. Coupled with the superior single precision performance and lower energy consumption, NVIDIA's Tesla K10 was selected as the accelerator of choice. Cuda was selected over OpenCL as a programming model to take advantage of the superior debugging and profiling tools available, at the cost of having to rewrite the OpenCL based prototype code. This selection, combined with the analysis in Section 5.6.1, gives a lower bound on the number of accelerators required for Cobalt. A minimum of 10 K10s (42.8 TFLOPS / 4.577 TFLOPS = 9.6) were needed. Our design target required at least 14 K10s (61.3 TFLOPS / 4.577 TFLOPS = 13.4).

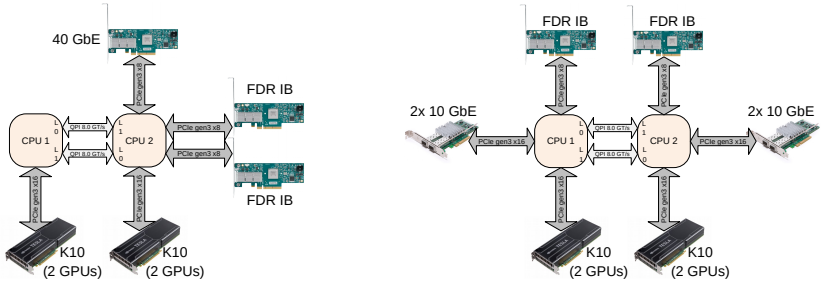
5.6.5 Prototyping

In Table 5.3 we show a summary of the detailed lower bounds on the Cobalt system. Based on the lower bounds discussed in the previous Sections, and a first order approximation of what may be a suitable node design, a list of components for Cobalt was proposed.

Based on the lower bounds identified above, we proposed a baseline Cobalt system that consisted of at least 8 nodes. Each of these nodes would have four 10 GbE ports (or equivalent), two FDR Infiniband ports and two accelerators. We noted that dual-port FDR Infiniband HCAs are inherently bottlenecked by their limited PCI-express bandwidth, so two single-port HCAs were required. Our task next task was to find a suitable commercially available node, and evaluate a representative sample for performance. The entire product line of all major vendors was evaluated, based on suitability, availability and maintainability. Having surveyed a large number of nodes from a variety of vendors, we selected our initial prototype based on a Dell R720 chassis. This node had a single 40 GbE port instead of the four 10 GbE ports, but matches all other requirements.

5.6.6 PCI-express balancing

The primary data transport interfaces in Cobalt nodes is PCI-express (PCIe). Our system consists of many inter-communicating components, so a well balanced PCI-express infrastructure is vital to an efficiently operating correlator and beamformer. We investigated the configuration of a prototype Cobalt node, the standard Dell HPC node at the time, a Dell R720 (shown in Figure 5.6(a)). In this figure, a clear inbalance can be seen, as the vast majority of PCIe connectivity is provided by a single CPU. All data for the other CPU, or the accelerator attached to that CPU, had to cross the Quick Path Interface (QPI) boundary between CPUs at least twice. Based on experimental data, it was considered highly likely that this would be a significant bottleneck.



(a) Dell R720 (b) Dell T620

Figure 5.6: PCIe configurations encountered in the two prototype systems.

Finding a system that exposes next to all the available PCIe lanes in a more balanced manner, turned out to be quite difficult. Figure 5.6(b) shows the configuration of a Dell T620 workstation node. Even though these nodes were not specifically designed for HPC use, the balanced PCIe configuration shown led to this being selected as our base node type. These nodes also allowed for the installation of two dual-port 10 GbE Network Interface Controllers (NICs), in place of the single port 40 GbE NIC in the R720 that was found to be unsuitable in the existing 10 GbE network. In Section 5.7.1 we leverage the symmetrical architecture of these nodes by essentially using them as two mostly independent nodes, one for each CPU socket, both for clarity and performance.

5.6.7 Cooling the GPUs

The Dell T620 chassis was designed as a workstation, rather than a HPC node. The NVIDIA Tesla K10 was only available as a passively cooled unit, which relies on the chassis to provide sufficient cooling. These two facts combined meant that we ran into serious cooling issues for the selected GPUs. Early tests showed that the K10s ran at approximately 70°C while idle, with an optional fan-bar installed. No load tests could be performed, since the GPUs would overheat and switch off before any meaningful test

results could be obtained. Improvised cardboard and duct tape airflow baffles showed that sufficient cooling could be provided to the GPUs. Better fitting baffles were designed and 3D printed in-house at ASTRON. By directing the airflow generated by the fan bar through the NVIDIA Tesla K10 GPUs, we successfully reduce the operating temperature of the GPUs to acceptable levels. Using these custom baffles, shown in Figure 5.7, the Dell T620 and NVIDIA Tesla K10 combination ran about 10°C cooler than a comparable Dell R720 system, probably due to the additional space in the (large) Dell T620 chassis. We outsourced the production of twenty of these baffles by injection moulding rather than 3D printing, sufficient for ten Cobalt nodes.

5.6.8 The Cobalt system

Apart from the issues described above, no other performance limitations were identified with the Dell T620 nodes. The fully deployed Cobalt system consists of ten of these nodes, eight production and two hot spare and development nodes, fitted with two NVIDIA Tesla K10 GPUs each. Each node contains two dual-port Intel X520 10 GbE NICs and two single-port Mellanox ConnectX-3 FDR HCAs.

5.7 Software design

The software part of Cobalt consists of two applications that manage the data flow through networks and GPUs, and store correlated and/or beamformed data products on persistent storage as shown in Figure 5.4. Cobalt also interfaces with several other subsystems for control, monitoring, logging, and metadata. No data is fed back from post-processing into Cobalt.

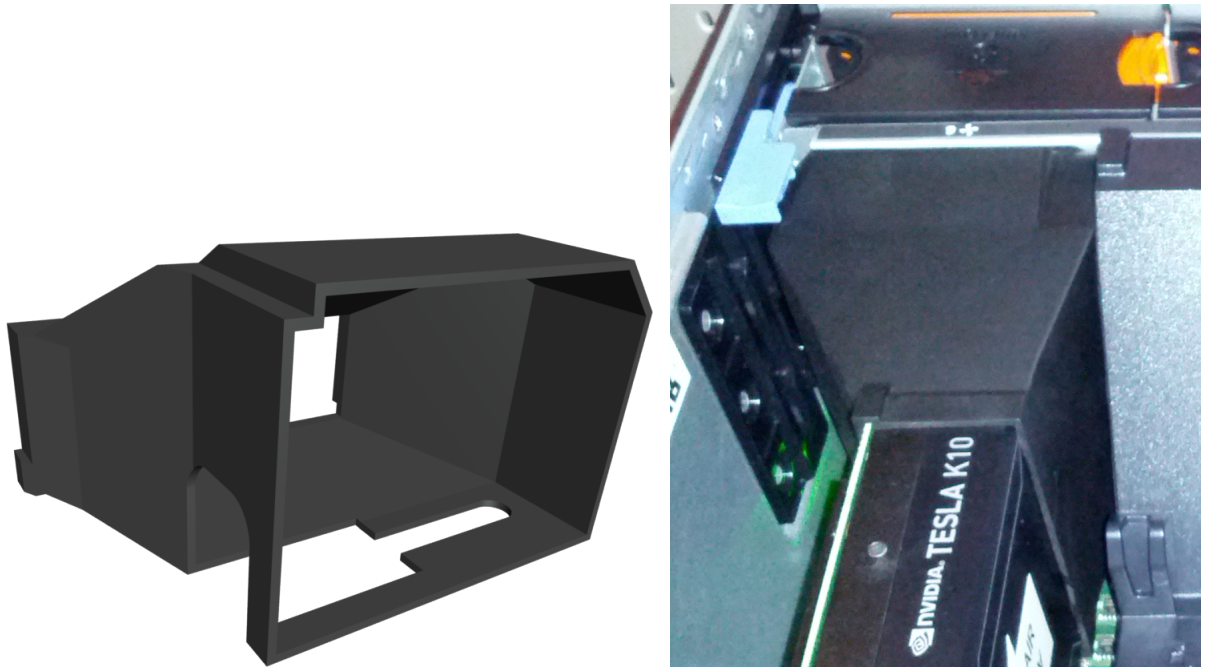
The following Sections describe the Cobalt software architecture and considerations for parallelism at different layers.

5.7.1 Software architecture

The component diagram in Figure 5.8 shows high-level LOFAR Cobalt components (here named in `typewriter` font), dependencies and data flow.

Observation control starts `Output` processes on all allocated nodes in the storage cluster. It then uses MPI (Message Passing Interface) to start two processing applications (MPI ranks) per GPU cluster node, one per CPU socket. Each data processing instance connects to `Output` processes it needs to send data to, and opens sockets for its two 10 GbE interfaces to receive antenna field data. Just after the observation start time, data flows through Cobalt producing data products on the storage cluster. On late establishment or failure of network connections, Cobalt retries until the observation stop time. Then, observation meta data such as LOFAR system health statistics are gathered from databases and written into the data products. Before shutting down, Cobalt gives its vote for the observation end status to LOFAR control.

All software components along the data flow path forward data blocks of about 1 second using MPI, thread-safe bounded FIFO queues or TCP/IP. We allocate block space once during initialisation and keep pools of free blocks. The block size is a trade-off:



(a) 3D render of the production Cobalt airflow baffle

(b) Cobalt node with custom baffle installed

Figure 5.7: Custom cooling solution in a Cobalt node. Note the fan at the top of the image, providing forced air cooling to the NVIDIA Tesla K10 via the installed custom duct.

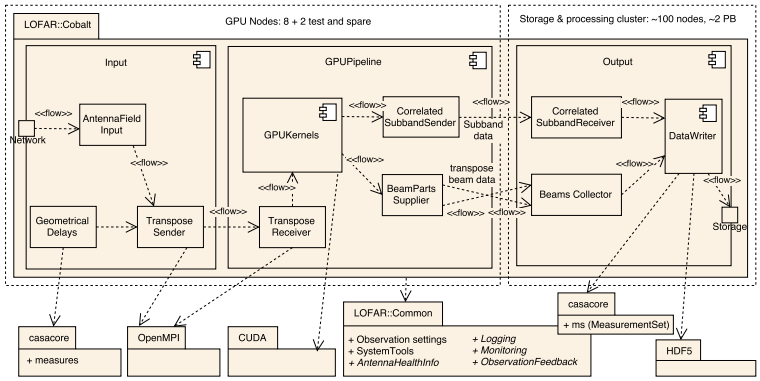


Figure 5.8: High-level component diagram of the LOFAR Cobalt software with data flow and dependencies.

efficient network transfers and processing favour larger blocks, but the size is limited by GPU memory (4 GiB), and affects how many beams the beamformer can form, as well as main memory footprint and overall latency. The exact block size is a multiple of all work unit sizes in each signal processing step to limit the number of edge cases to implement, test and debug.

Each `AntennaFieldInput` receives UDP datagrams on two 10 GbE network interfaces and forwards valid data to the `TransposeSender`. `Transpose` uses a circular buffer to perform coarse delay compensation by shifting the sample streams by an integer number of samples ($\sim$5.12 µs). These delays are computed in a separate CPU thread by `GeometricalDelays`, and are used to compensate for different signal arrival times at different antenna fields and to form beams. The remaining (sub-sample) delay is compensated for later using phase rotation on the GPU.

`TransposeSender` also deals with (rare) out-of-order UDP datagrams and drops data that arrives after a deadline. `TransposeReceiver` in the `GPUPipeline` component transposes data per antenna field to data per subband using MPI over InfiniBand. The `GPUPipeline` component pushes subband data through the signal processing pipeline on GPUs, producing correlated and/or beam data, as explained later. Each correlated subband is sent to an `Output` component on a single host using TCP/IP, but beam data needs to be transposed over the network to collect all subbands for each beam, produced at different GPUs, to be combined in a single storage host. The `Output` component stores correlated data in the MeasurementSet format using casacore[4] and beam data in the LOFAR HDF5 format[5].

[4] `https://github.com/casacore/casacore`
[5] `https://www.hdfgroup.org/HDF5/`

5.7.2 Dealing with jitter and hardware failure

Cobalt is part of a large, operational system and as such uses the LOFAR `Common` library to communicate with several monitoring and control systems, and to reuse other common functionality. Antenna fields send data at a fixed rate, but contention on computing and especially on network and storage resources may vary. As a complex system with different sites, jitter, hardware failures and misconfigurations do sometimes occur. We therefore designed Cobalt to conceivably drop data rather than fail or wait in several key places. The network or operating system may drop incoming UDP data, the `TransposeSender`'s circular buffer may drop data if not read out in time. Both `CorrelatedSubbandSender` and `BeamPartsSupplier` have bounded queues that drop when full. Any overload or failure in the pipeline will fill the previous component's queue, propagating until such a dropping point is reached. Cobalt encodes lost or dropped data in metadata that is aggregated and written into the data product for post-Cobalt processing to interpret.

We routinely correlate 488 subbands (about 96 MHz wide) from up to 78 antenna fields (230 Gbps input) or produce 222 beams (37 Gbps) from 12 antenna fields (or a compromise of both) using 80 storage and 8 GPU nodes. Correlation is GPU compute-bound, but for beamforming output bandwidth to the storage cluster is the limiting factor, which is not bound by Cobalt. Most beamforming science needs high time resolution and as many beams as we can form, up to the available capacity. Measurements show that up we can form up to 146 beams for 288 16-bit subbands, which is well in excess of our original requirement, even if we cannot store the resulting beams at the desired time resolution.

5.7.3 Workload distribution

The Cobalt hardware is fundamentally different from its predecessor, the IBM Blue Gene/P supercomputer. In Blue Gene/P we needed several cores to process a single subband, but in Cobalt a single GPU is powerful enough to process several subbands. In Blue Gene we designed a complex round-robin work-distribution scheme to avoid contention on the internal torus network [123]. In Cobalt a static assignment of subbands to GPUs is sufficient. Table 5.4 shows the levels of hardware parallelism in Cobalt.

Table 5.5 indicates application data dimensions that we must map to hardware parallelism. The independence within dimensions (e.g. process two antenna fields independently) is not available throughout the complete processing pipeline: at several points data has to be combined or forwarded jointly (i.e. synchronised). In terms of scaling direction, most dimensions scale *out*. When adding more antenna fields, beamforming and correlation output also scale *up*, the latter quadratically.

Apart from data parallelism, processing and I/O task parallelism are also possible: receive input data, geometrical delay computation, input data transposition, control of GPU data transfers and kernels, data transfer to storage, and data product write-back all run in parallel on the same hardware. We leverage all levels of parallelism mentioned in Tables 5.4 and 5.5 to ensure we can keep up with the most demanding observation setups.

Layer	Type (Qty.)	API
Cluster	Multi-node (8), multi-CPU (16)	MPI
Half-node	Multi-core (16 SMT), multi-GPU (2)	OpenMP, pthreads
GPU	SMs (16), cores (1536)	CUDA

Table 5.4: Hardware available for parallel execution.

Data dimension	Size (typical)
UDP datagrams	48828 per antenna field per second
Antenna fields	38–78 (correlator), 12–48 (beamformer)
Freq. subbands	200–488
Freq. channels	64–256 (correlator), 1–16 (beamformer)
Samples (time)	768–196608 per freq. channel per second
Beams	1–222

Table 5.5: Dimensions to map to (data parallel) hardware.

We partition the antenna field streams over all 10 GbE interfaces and the subbands over all MPI ranks and their GPUs. Work partitioning and mapping to GPU resources is compute kernel specific. To utilise all compute resources, a GPU needs to be supplied with many blocks each with many (semi-)independent work units. I/O and memory access need to be carefully considered too, as many of our compute kernels are bound by GPU memory bandwidth. Exact partitioning and mapping differs between observation setups, especially for dimensions that are traded off against each other. For example, fewer frequency channels implies more samples in time, providing a different dimension for parallelism. We compile our CUDA kernels at runtime to turn observation-specific constants into compile-time constants. Run-time compilation increases performance by removing branches and by lowering the register pressure, and allows more freedom with respect to workload distribution within the GPU. To control and process on GPUs we use CUDA [105] and the CUFFT library. In contrast, the Blue Gene/P PowerPC CPUs required handcrafted assembly to fully exploit their processing power.

5.7.4 Parallelisation libraries

Cobalt uses *OpenMP*, *OpenMPI*, *CUDA*, *fork/wait* (for runtime kernel compilation), *pthreads*, and *signals* (to initiate shutdown) in the same processing application. Some of these were not designed to work together and require careful programming.

To exploit task parallelism we need to determine task granularity and mapping, such that tasks both run and forward data blocks in time. The *OpenMP* pragma `omp parallel for` is an easy way to iterate over the subbands in parallel. Around that we placed `omp parallel sections` to divide pipeline work into parallel tasks. Tasks forward blocks through *thread-safe bounded queues* that use *pthreads* condition variables, not available in *OpenMP*, to avoid busy waiting. Although *OpenMP* and *pthreads*

are not intended to be used together, this results in excellent readability of the multi-threaded code, while allowing the use of powerful primitives like *thread-safe bounded queues*. Multi-threading remains in a local scope and both data flow and control flow remain very clear. Another upside is that our *OpenMP* pipeline allows us to trivially adjust task granularity and count, without requiring a direct mapping to CPU cores. Downsides include the non-portability of our combined use of *OpenMP* and *pthreads* and that this use favours to have as many threads as tasks, as otherwise some tasks have no dedicated thread and thus may not empty their input queue, causing deadlock. As a result, some observation setups end up with an order of magnitude more threads than (logical) CPU cores. While there is room for CPU task management, the current *OpenMP* code is well readable and further optimisations will not improve *system* capability, since CPU power has never turned out to be a bottleneck in our system.

The InfiniBand and GPU cards need the same CPU memory used for DMA (Direct Memory Access) to be pinned and registered with their driver. Pinning and registering come with an overhead, which we have mitigated by allocating all of these buffer during initialisation. Both the MPI and GPU library offer interfaces to explicitly allocate memory for DMA, but only CUDA can mark an existing allocation as such, so we allocate shared buffers via MPI and then register them with CUDA.

Cobalt deals with a lot of mostly independent data streams that are handled in parallel without inter-dependencies. To optimally utilise the available hardware, every level of available parallelism needs to be exploited. However, none of the MPI libraries we looked into offered good multi-threading support, they were either not thread-safe, used a global lock, or failed to compile or run with fine grained thread synchronisation. We therefore wrap our MPI calls with a global lock, which turns out to be efficient enough in combination with non-blocking sends and receives using `MPI_Isend` and `MPI_Irecv`. We do need a separate polling thread to frequently check for completion of pending transfers using `MPI_Testsome`, otherwise MPI throughput suffers.

On the storage cluster, we distribute all subbands and beams over the nodes. Some beamforming observations need full resolution, both spectral as well as temporal, which limits the number of beams that can be sent to storage due to limited network bandwidth. In such setups, we have to store each beam across multiple storage nodes. This split is less convenient for post-Cobalt processing, to be executed on the same cluster.

5.7.5 Signal processing with GPU kernels

This section focuses on the digital signal processing GPU kernels shown in Figure 5.2 as executed within the `GPUKernels` component.

The correlator pipeline first channelises subbands in a polyphase filter using FIR filters and FFT kernels. We carry FIR filter history samples across to the next block. The pipeline then applies fine delay compensation and bandpass correction. This marks the end of processing per antenna field. To efficiently operate *across* antenna fields, the delay and bandpass kernel transposes data on write-back to GPU device memory. The last kernel computes the correlations of all pairs of antenna fields and averages in time to approximately 1 s.

The beamformer pipeline forms many *coherent* and/or *incoherent* beam(s). Both beam types have the first four kernels in common. Cobalt performs delay compensation,

bandpass correction and beamforming at 256 channels per subband as a good compromise between time and frequency resolution, and then transforms to the requested output resolution, often 1 or 16 channels per subband. After bandpass correction, the coherent and/or incoherent specific steps of the beamforming pipeline execute. Coherent beamforming first adjusts the beam direction with a phase shift and sums over antenna fields, then optionally computes Stokes parameters, while incoherent beamforming first computes Stokes parameters and then sums over antenna fields. Coherently formed beams are more sensitive but cover a much smaller sky area. During an observation many adjacent beams can be formed to mosaic a somewhat larger sky area, although some projects also add an incoherent beam to quickly search for bright signals [47]. If we do not convert to coherent Stokes I (intensity only) or IQUV (full polarisation), we retain complex voltage data with phase information allowing coherent dedispersion (after Cobalt). However, complex voltages cannot be time averaged.

Due to differences in required frequency/time resolution and averaging, the beamformer and correlator pipelines diverge quickly in how they transform the incoming signal. This means that our beamformer cannot share initial steps with the correlator and needs to reorder data often as shown in Figure 5.2.

All kernels operate on single-precision complex floating-point data, except for delay compensation, which uses mixed precision. Fine delay compensation (i.e. subsample) uses the residual delay from coarse delay compensation by the `TransposeSender`. From the residual delays at the start and end of a block, we compute the channel-dependent phase angles in double precision. Within a 1 s block these angles can be interpolated linearly to obtain the angle for each sample. Then back in single precision, we determine the phase shift factor (sin/cos) and rotate back the phase of each sample (complex multiplication). The beamforming kernel operates in a similar way to form beams with an offset from the centre. The Tesla K10 GPU has low double precision throughput, but as long as the kernel is memory bound, the limited use of double precision has little impact.

Most kernel parameters are fixed throughout an observation. We avoid using registers for these parameters and obtain more efficient kernel binaries by using runtime compilation supplying fixed parameters as C-style defines. The resulting code is also more readable. We reduce GPU memory usage by using a small number of buffers that the CUDA kernels alternate between as their in- and output.

The number of observation parameters supported by Cobalt is large. This affects kernel complexity, kernel execution configuration (CUDA block and grid dimensions, as well as input/output data dimensions and some transpose alternatives. This complexity cannot lead to observation failures. To deal with execution configuration, GPU buffer sizes and performance counters, we use a wrapper class for each kernel. This also wraps the type unsafe argument passing when launching a CUDA kernel. Each kernel unit test covers the wrapped kernel. Furthermore, we centrally document which buffers are (re)used by which kernels and what the array dimension order and sizes are.

Although the development of highly optimised GPU kernels is a critical Cobalt ingredient, the details are outside the scope of this article. For more insight into radio astronomy signal processing for Cobalt and beyond on various accelerator platforms, we refer the interested reader elsewhere [120].

5.8 Verification, validation and optimisation

Before Cobalt could be taken into operational use it needed to be extensively tested and tuned. Regression testing and integration happened continuously during (software) development. We determined science readiness during *commissioning*, a phase in the last part of development where domain experts and instrument engineers work closely together towards system-wide integration, validation, tuning and performance characterisation. Some of these tests are still performed on one Cobalt node and LOFAR station before deploying a new software release at full scale.

During Cobalt development we added about 400 tests in 100 test programs. Some are unit tests, others test a feature, uncommon observation settings, across an interface, or a complete Cobalt pipeline on a tiny amount of data. About another 100 unit tests were already in place for the `LOFAR Common` package.

Incrementally developing tests was a substantial amount of work. Extending the test set and updating documentation are part of delivering a new feature. What added to the effort was dealing with tests that generally pass, but occasionally fail due to race conditions or non-real-time testing of real-time code. We used the Jenkins[6] *continuous integration* service to manage regression test builds. The extensive use of testing was critical for Cobalt to minimise regressions, both on component and on system level. Furthermore, tests kept the code maintainable, by providing confidence and freedom to improve or even refactor the Cobalt code.

Cobalt needs various non-default system settings to perform well. System firmware (BIOS/EFI) and Linux kernel settings needed to be tuned for performance and predictability, such as (minimum) network buffer sizes, CPU frequency scaling, and mapping GPU and NIC interrupts to the CPU they are linked to. We do not need to bind threads to cores within a socket, as long as we raise the CPU and I/O priority of threads receiving UDP input and writing to storage. We also do not need to run a PREEMPT_RT (real-time) patched Linux kernel. Our multi-homed network and VLANs to international stations required changes to ARP and routing settings to function correctly.

To get good performance for the input transpose via MPI, we needed to tune OpenMPI RDMA settings, for which we used the point-to-point tests from the SKaMPI benchmark [117]. We also send transfers between CPU sockets over InfiniBand instead of directly between the CPUs via the on-board QuickPath Interconnect.

Due to a performance regression that couldn't be resolved by reverting code commits, we had to rework the MPI transfer scheme. Instead of supporting all surrounding tasks independently by scheduling their many point-to-point transfers, we applied message combining to send fewer but larger messages. While the new implementation solved the performance regression, this came at the cost of increased use of memory/cache bandwidth, and it introduced more dependencies between producers and consumers of MPI data. This is an example where we sacrificed an over-dimensioned resource (CPU cache/memory bandwidth) for a scarce resource (development effort).

We have more examples of unexpected regressions during development and operations, but in general, debugging performance issues silently introduced with system software updates, changed system & network settings, or replaced hardware was time

[6] `https://jenkins.io/`

consuming and difficult. To alleviate this risk, we used performance and configuration verification scripts. This *operational readiness check* was especially useful when the line between responsibilities for high performance software and system and network administration blurred. When major hardware/software functionality had passed verification, the project scientist (i.e. an Observatory astronomer) was responsible for the validation effort to deliver a science capable instrument.

Radio telescopes essentially sample electromagnetic noise, including radio interference, and then perform stochastic signal processing. Thus there was no reference output to bit-wise compare our output to. Moreover, the existing BlueGene-based system used double precision and a different beamformer DSP filter chain. We therefore chose to analyse Cobalt output to comply with signal and noise properties required for the most demanding science cases. This proved the validity of the Cobalt output without having to be bit-wise equal to its Blue Gene predecessor.

In total, we planned and performed about 30 experiments and worked with astronomers and software developers to get issues resolved and the system tuned and characterised. This effort took several months. Several experiments required custom tools or software hooks and resolving issues can be time consuming. This was a substantial project risk that had to be mitigated with a solid development process and extensive and early testing.

During commissioning we observed no perceptible increase in system noise between the Blue Gene/P based correlator and beamformer and the new Cobalt implementation. Considering the difference in numerical precision used – double precision in Blue Gene, single precision in Cobalt – this warrants some discussion. We note that these choices were driven primarily by the selected hardware, not by necessity. Blue Gene was designed for double precision processing. There was no advantage in using lower precision arithmetic. In contrast, the selected NVIDIA K10 GPU was optimised for single precision processing. As shown in Table 5.2, this GPU has abysmal double precision performance. Only delay compensation was considered vulnerable to this loss of precision. Comparative analysis showed that single precision delay compensation led to an insignificant increase of the total noise [34]. Calculating the delays themselves does require double precision, this is the only part of the cobalt pipeline to do so.

5.9 Operational experience

Cobalt has been LOFAR's secondary correlator since January 2014 and its primary since March 2014. In May 2014, Cobalt also took over for beamformer observations.

We have collected statistics from three years of operations with the Cobalt system. Figure 5.9 shows the relative number of failed observations, with a break down into four failure modes ($N \approx 23000$). On average, 97.3% of submitted observations were successful, clearly exceeding the required operational availability of $> 95\%$ described in Section 5.5.

Observations may start at any moment (24/7), but normally, issue investigation starts the next working day. Observations scheduled between the occurrence of an issue and the start of the next working day may be adversely affected. The availabilities of other LOFAR sub-systems are not shown, but were generally lower than that of Cobalt. How-

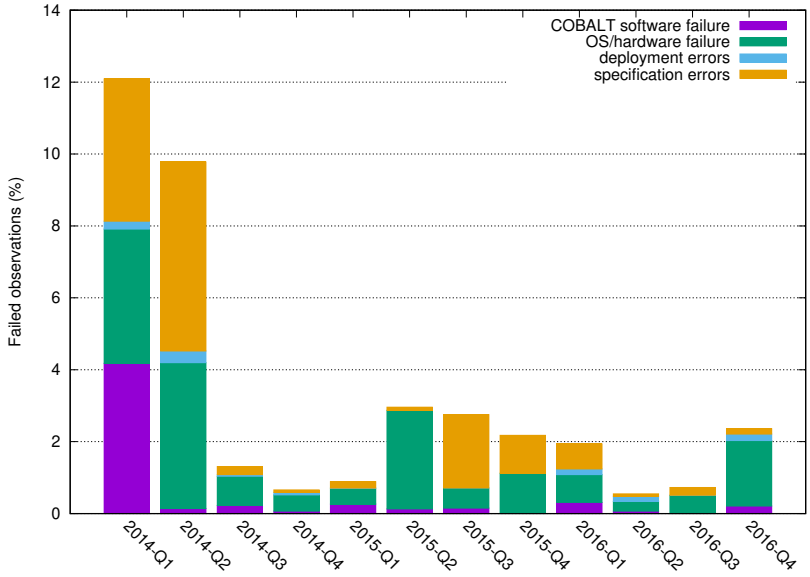


Figure 5.9: Failure modes of the Cobalt system over three years, and their occurrence in percentages.

ever, these generally work on non-volatile data where failures do not automatically result in irretrievable data loss.

There was a noteworthy increase in availability after six months in operations that can be attributed both to burn-in, as well as bug fixes in the Cobalt software and in the scheduling system. Current operational Cobalt failures mostly originate in network configuration or services, or in non-standard observation settings. Hardening and monitoring the network and settings have reduced their impact (until such monitoring services fail). We have run into several Linux kernel bugs, unexpectedly exposed with new software releases or changes in work load. This includes a failure mode that caused occasional Linux kernel panics in our system, due to a memory allocation bug that was fixed with a newer kernel release. While this shows the value of keeping low-level software updated and patched, we note that the regression mentioned in Section 5.8 may in part have been caused by similar updates.

5.10 Summary and discussion

In this chapter we presented the Cobalt GPU-based correlator and beamformer system for the LOFAR radio telescope. This system has successfully replaced the earlier

Blue Gene based systems and has been in operations for almost four years now. We introduced the hardware design, as well as the data flow-driven simplified system engineering process that led to the final implementation. The challenges that were faced during prototyping were described, as well as some of the engineering efforts that were necessary to keep the GPUs at an acceptable operating temperature. Finally, we showed some of the details of the software design, the verification process, and we discussed the operational experience with the Cobalt system.

All nodes in Cobalt are identical and perform all necessary processing, there are no dedicated nodes for a task. This requires careful programming, as was shown in Sections 5.7 and 5.8, but also makes for a highly efficient system with few idle components.

In contrast to similar chapters describing software correlators, we focused heavily on the development process of the system design. We showed how hardware/software co-design, in close collaboration with a commercial partner, can lead to an efficient and affordable system. None of the systems aimed at the HPC market were, for various reasons, suitable for our application. Close interaction between hardware vendor, hardware system designer and software architect in the design and prototyping phases was instrumental in finding a suitable node design.

An Agile test-driven development process was introduced to ensure timely delivery of a system that is fit for purpose and meets the requirements described in Section 5.5. We also noted in Section 5.8 that the test-driven aspect had great advantages in a system that cannot be completely deterministic. As another example of co-design, the experiences with previous LOFAR beamformer systems showed that a redesign of this component would better match the requirements of the majority of the science users. While this delayed the delivery of the Cobalt beamformer slightly, we took this opportunity to improve LOFAR non-imaging capability.

5.11 Impact

This project has generated a significant amount of interest. Discussions with the University of Cambridge HPC team, showed that they faced very similar issues, although their applications are very different. The University of Cambridge used our Cobalt hardware design as the basis for their Wilkes general purpose cluster[7], which reached #2 on the November 2013 edition of the Green500 [70] list. The size of this cluster made this decision particularly note-worthy. It was a 128 node cluster, with just 8 nodes per rack, taking up 16 racks in total. At 4U per node, this was not a particularly dense solution, but the, at the time, unique and abundant PCIe structure in these nodes was judged sufficiently desirable to justify the additional expense in terms of rack space.

Informal presentations of the Cobalt design to several industry partners, including senior Dell management, have resulted in an increased awareness of radio astronomy as an eScience. It was difficult to find a chassis from any one of the major vendors that could meet the requirements of the Cobalt project. We hope that our discussions with industry, using this project as an example, will improve the suitability of future HPC system designs for the next generation of radio telescopes.

[7] `http://www.hpc.cam.ac.uk/services/wilkes.html`

The initial design approach taken in this project, where the hardware is closely matched to the software requirements, has since been successfully employed in the SKA Science Data Processor preliminary design, described in chapter 4.

5.12 Retrospective

During the preparation of this thesis we noted that the original paper did not specify whether the Cobalt system as commissioned meets its requirements. At time of writing, Cobalt is reaching the end of its lifespan, with its successor being commissioned. We can now retrospectively conclude that most requirements were met by the Cobalt system.

In more detail, both functional requirements were met, although the second requirement approaches the limits of our available resources. Cobalt is able to produce 127 streams of time domain data with a frequency resolution of 256 subbands, slightly surpassing the 244 subband requirement.[8] The hardware can handle up to 88 stations in correlator mode, with up to 77 physical stations available and used in production at the time of writing.

The system was delivered on budget, although full functionality was delivered about 3 months later than scheduled, missing our *on time* requirement. Delivery was in stages, as mentioned in Section 5.9. In this same section we also show, using operational data shown in Figure 5.9, that its operational availability far exceeds the required 95%, although not until Q3 2014. System maintenance amounted to an average of 0.2 FTE from the hosting institute over the lifetime of the system, excluding some incidental additional effort by ASTRON domain specialists, totalling less than the required 0.25 FTE. Total operational cost, including above mentioned system maintenance, energy consumption and rack-space rent, amounted to less than 15% of the one-time capital investment (CAPEX), well below the 50% requirement.

All further requirements were soft, i.e. *nice to haves*. Nevertheless, many of these were achieved over the lifetime of the system, demonstrating that Cobalt was a highly flexible and capable system. Table 5.6 shows the various requirements and whether they were satisfied by the Cobalt system. Online flagging was implemented to only detect data that was lost during transmission. Due to the limited amount of memory available on the GPUs, filter window sizes would have to be relatively small, limiting the performance of any interference flagging.

5.13 Our propositions in this chapter

To close this chapter, we show how the various propositions, as introduced in chapter 1, are supported. Three of the four propositions apply to this chapter, the bounding, co-design and value propositions. We shall briefly discuss these below.

[8] In coherent complex voltage mode, required output bandwidth exceeds the available network capacity. Although the Cobalt system meets the requirement, this requirement was in hindsight unrealistic.

Requirement	Achieved	Remarks
Correlate 64 stations	✓	hardware can handle up to 88
Create 127 Beams, 244 subbands	✓	hardware limited to 256 subbands
On time	✓	partial functionality on time
Within budget	✓	for capital investment
Staged delivery	✓	see Section 5.9
Availability $\geq 95\%$	✓	see Figure 5.9
Maintenance ≤ 0.25 FTE	✓	average 0.2 FTE/yr
Ops cost $\leq 50\%$ CAPEX	✓	$< 15\%$ of CAPEX
Fly's eye mode	✓	
8 Beams in correlator mode	✓	achieved 488 beams
Online flagging	✓	only lost data, not interference
Superterp beamforming	-	hardware capable, no requirement
Additional offline processing	-	hardware capable, not implemented
Commensal observations	✓	
Parallel sub-arrays	✓	
Responsiveness to triggers	✓	
Additional observation modes	-	hardware capable, not implemented

Table 5.6: Cobalt requirements and whether they were achieved.

5.13.1 The bounding proposition

The Cobalt project, as documented in this Chapter, is a further excellent example of the *bounding* proposition in active use. In the introductory sections of this chapter, in particular section 5.5, the bounds of the system are clearly articulated. We note that, due to the fact that the Cobalt system was intended to be a 'drop-in' replacement of an existing system, this problem was fairly well bound. The experience gained with systems before Cobalt, the Blue Gene/L and Blue Gene/P based LOFAR correlator and beamformers, was instrumental here. This chapter defined the bounds of the Cobalt design in terms of functional and operational requirements on the system, as well as project bounds such as capital investment and development effort available. Compared to the previous two chapters, which describe the conceptual evolution of a system, rather than the design of a physical system meant as a drop-in replacement of an existing solution, the bounds are much more pronounced and detailed.

5.13.2 The co-design proposition

In this Chapter we arguably take the *co-design* proposition even further than in previous Chapters. Here we note, in Section 5.5, that for the Cobalt system we took the slightly unusual approach of focusing on data flow first, and compute requirements a close second. We argue that, in this case, the effective and efficient use of the available compute capacity relies critically on the efficient flow of data through the system. We go on to

design a tailored compute solution that tries to avoid data-transport bottlenecks on all levels of granularity, the various networks, the PCIe infrastructure in the node, down to the available memory bandwidth per processor. The careful attention given to the variety of data transport systems in Cobalt during the design process was a key contributing factor to its success.

5.13.3 The value proposition

In the Cobalt project the value potential of the system, as articulated by the *value proposition*, was a key factor in the design process. The system only has a very limited set of applications to run, the correlator and beamformer pipelines, and we could therefore afford to heavily optimise for these. While cost was an important consideration in the design process, value, defined by the functional and operational requirements, was more important. By focusing heavily on data flow in the design of the system, value potential of the system was maximised. Our retrospective shows that the operational cost of Cobalt, designed to be as close to an off the shelf system as possible to exploit available mass-market experience, was much lower than required. This results in a better relative science value, as defined in chapter 2, than expected.

Acknowledgements

The authors would like to thank the Center for Information Technology (CIT) at the University of Groningen and in particular Wietze Albers for his indispensable efforts in the design and prototyping phase and Hopko Meijering and Arjen Koers for their system and network administration during the development of Cobalt and its operational use. We thank Dell Netherlands, and Patrick Wolffs in particular, for their close involvement in the prototyping phase of this project and their help in getting our custom cooling solution supported and certified. We would like to express our appreciation to the anonymous reviewers for their detailed comments on an earlier version of this manuscript.

ASTRON is an institute of NWO, the Netherlands Organisation for Scientific Research.

CHAPTER 6

Software-defined Networks in large-scale radio telescopes

P. Chris Broekema [1]*, Damiaan R. Twelker* [2]*, Daniel C. Romão* [2]*, Paola Grosso* [2]*, Rob V. van Nieuwpoort* [2,3] *and Henri E. Bal* [4]

context and contributions

This chapter is based on a paper presented at the Computing Frontiers Conference in 2017. It has been extended to include additional context and reasoning to properly place the work into context.

This work was conceived by Broekema, who also acted as first author for the paper. Initial work was done by Twelker, based on a plan by Broekema and under supervision of Broekema and Romão. This resulted in a very highly graded Bachelor's thesis by Twelker [154]. Broekema did all of the research presented in this Chapter, using some of Twelker's initial work and code, and wrote the vast majority of the content, with some input from the co-authors.

In this chapter we show an example of the *optimisation* proposition in use.

Abstract

Traditional networks are relatively static and rely on a complex stack of inter-operating protocols for proper operation. Modern large-scale science instruments,

[1]ASTRON, the Netherlands Institute for Radio Astronomy
[2]University of Amsterdam
[3]Netherlands eScience Centre
[4]Vrije Universteit Amsterdam

such as radio telescopes, consist of an interconnected collection of sensors generating large quantities of data, using high-bandwidth IP over Ethernet networks. Recently the concept of a software-defined network (SDN) has gained popularity, moving control over the data flow to a programmable software component, the network controller. In this chapter we explore the viability of such a software-defined network in sensor networks typical of large scale radio telescopes. Based on experience with the LOw Frequency ARay (LOFAR), a recent radio telescope, we show that the addition of such software control adds to the reliability and flexibility of the instrument. We identify some essential technical SDN requirements for this application, and investigate the level of functional support on three current switches and a virtual software switch. A proof-of-concept application validates the viability of this concept. While we identify limitations in the SDN implementations and performance of two of our hardware switches, excellent performance is shown on a third.

6.1 Introduction

Packet switching ASICs at the heart of networking equipment have significantly grown in capability and flexibility over the years. To expose the increased functionality of these networks, the concept of a software-defined network has appeared. In a traditional IP over Ethernet network control over the data flow is mostly implicit. A software-defined network moves the control plane to a programmable network controller, allowing positive and explicit control over the data flow. This concept has gained remarkable popularity in recent years, with all switch manufacturers including support in their mainstream products. In this chapter, we will study the usefulness of SDNs for an important application: large-scale radio telescopes.

Modern large-scale science instruments, and in particular radio telescopes, are often distributed sensor networks. They consist of large numbers of sensors, that sample the object of interest. Specialised custom hardware is used to convert the collected data into the digital domain and perform dedicated signal processing. A centralised general-purpose computing facility reduces data volumes, calibrates the instrument and allows the user to extract science.

These components are interconnected with high-volume networks, often based on off-the-shelf IP over Ethernet equipment. This combination of specialised custom-designed components, interconnected with general-purpose compute systems, may lead to unexpected challenges. As an example (studied in this chapter), IP over Ethernet relies on a complex stack of protocols to ensure traffic reaches its intended target as determined by the destination IP address. This is regardless of the physical identity or location of the host. Custom hardware may omit parts of that protocol stack to simplify implementation, and the data flow in a radio astronomical sensor network is strictly unidirectional. The network may not be able to populate their MAC address tables through MAC learning, leading to packets being forwarded on all output ports.

We argue that a software-defined network adds valuable additional functionality, while mitigating some of the challenges an IP over Ethernet network may cause. This is illustrated by comparing several use cases based on experiences with a state-of-the-art operational instrument in the Netherlands, LOFAR. By implementing a proof-of-

concept application, we explore the required SDN functionality for this application, and the current level of support for these SDN features in three available hardware switches and a software-based virtual switch. In addition, we present some limited performance measurements. While these give a useful indication of the expected latency, data loss and potential bottlenecks, we note that the investigated switches are of modest scale and performance.

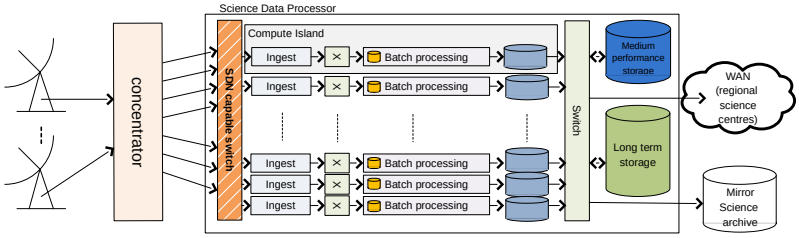


Figure 6.1: Top-level system overview of a radio telescope.

While two of our switches exhibit significant limitations in their OpenFlow implementation, software-based flexibility allows us to work around these. One of our SDN-capable switches shows excellent performance for our application, exhibiting very low latency, no significant data loss and a richly implemented OpenFlow feature set.

We show that the way in which packets with unrecognised destination addresses are handled and the ability to explicitly program the port on which packets are forwarded improve the robustness of a sensor network. In addition, we show that the added flexibility allows us to build a conceptual publish/subscribe system in the network, which in turn makes it easy to serve multiple processing pipelines without added overhead.

This chapter is structured as follows. We start with some background information, discussing modern radio telescopes and SDNs. In Section 6.3 we show how a SDN fits in a radio telescope, and the foreseen impact of a integrated SDN. Next we describe our proof-of-concept application, the investigated switches and our experimental setup. We explore the functionality of the various switches with respect to our application and do some limited performance measurements in Sections 6.6 and 6.7. We close this chapter with discussion, future work and conclusions. Finally, we look back at the propositions in this thesis, and how this chapter contributes to these.

6.2 Background

We explore how a software-defined network may be integrated into a large-scale distributed radio telescope. Modern radio telescopes, like LOFAR and the Square Kilometre Array (SKA) are examples of such instruments.

6.2.1 Modern radio astronomy

As we mentioned in Chapter 1, modern radio astronomy is an extremely data- and compute-intensive science. Since the signal of interest is exceptionally weak, enormous amounts of data have to be collected, filtered and processed to achieve a useful result. This is generally done by digitally combining multiple receivers into a large virtual instrument. Due to the high data volumes involved, the sensors and initial processing are often highly specialised custom devices.

While we usually describe radio telescopes in terms of the processing required, it is the data flow that drives the design. Therefore we can consider a radio telescope as a high-performance sensor network. Recently, the concept of a distributed radio telescope has emerged, and LOFAR and SKA, introduced in Chapters 3, 4 and 5, are perfect illustrative examples. Figure 6.1 shows a high-level overview of a modern distributed radio telescope. In this chapter we propose a redesigned receiving end of the general purpose computing facility with an SDN capable switch, as shown in Figure 6.1.

6.2.2 Software-defined networks

From the description above and in previous chapters, it is clear that for all intents and purposes modern radio telescopes, and more generally any sensor network, can be considered a special-purpose computer network. They are a collection of nodes and links connected together, often using off-the-shelf hardware based on IP over Ethernet technology. General purpose computer networks are generally static, with tight coupling between sender and receiver. In such a computer network there is no easy way to modify the behaviour of the network without changing the behaviour of the sender and receiver nodes.

A new paradigm has recently appeared in networking, that effectively separates the control and data planes of the network [108]. Such a software-defined network consists of SDN-capable switches: the data plane, a network controller: the control plane, and a connection between these two. Special-purpose virtual services are built on top, allowing dynamic modification of the network to suit applications, transparent to the sending and receiving peers. Support for OpenFlow, a standardised method to communicate control data from the control plane to the data plane, is becoming standard in more recent switches.

A software-defined network handles traffic based on a defined flow table. Incoming packets are matched to a list of flows installed on the switch by the network controller, and handled according to rules associated with those flows. Rules may modify source and/or destination headers and forward packets to specific ports, giving the network controller complete and explicit control over the data plane and effectively decoupling the sending and receiving nodes. Non-matching packets may be either dropped or have their header forwarded to the network controller for inspection, depending on configuration.

OpenFlow

The concept of a software-controlled network data plane is not new, but OpenFlow [96] is the first standard software interface between control and data plane with a consider-

able install base. Introduced in 2009, the initial stable release of the OpenFlow specification, 1.0.0, describes a fairly sparse protocol. It allows installation, modification and removal of the switch flow table. Each flow contains a header field for packet matching, counter and action fields. Per packet processing involves:

1. Find the highest priority matching flow entry
2. Apply instructions
   a) Modify packet & update match fields
   b) Update action set
   c) Update metadata
3. Send match data and action set to next table

If a packet does not match any of the flow entries in the flow table, the switch may either drop the packet, send it to another table or forward the packet header to the controller. This behaviour differs from that of a standard Ethernet switch, which would forward a packet with an unknown destination to all ports.

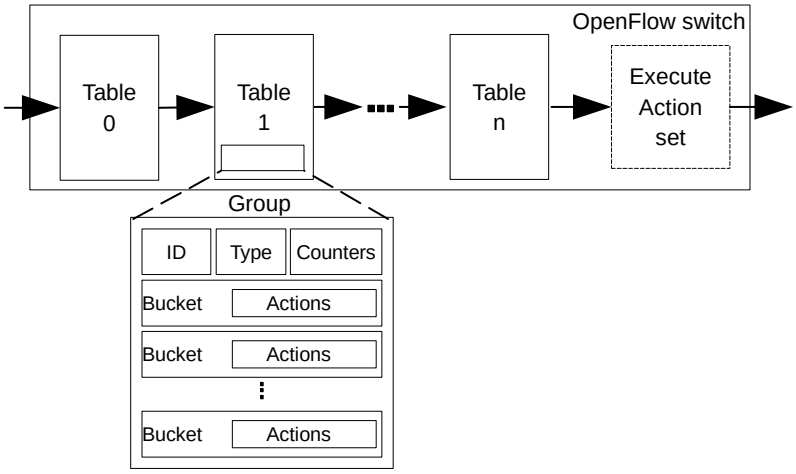


Figure 6.2: Packet processing pipeline of an OpenFlow switch, including groups.

The next major release that gained significant support, 1.3.0, introduced the concept of groups[1]. An OpenFlow group combines similar fields as in above, with a list of action buckets. Each of these provide similar functionality as the single list of action fields described above, giving the ability to have each packet be processed by multiple

[1]Groups were introduced in version 1.1.0, but this and version 1.2 are not widely supported

action lists. In addition, this release adds a far richer set of matching and action fields, although the implementation of many of these is optional. Figure 6.2 shows the per packet processing.

In this chapter, we investigate the viability of a software-defined network component in a modern radio telescope by implementing a proof of concept application using OpenFlow. Using the most prevalent SDN implementation available today, OpenFlow, and using the most recent protocol version available in all our switches (1.3.0), we investigate the functionality supported by four SDN-capable switches. These are described in Table 6.1.

6.3 Software-driven data flow

A previous SKA memo [37] explored the use of Ethernet in a modern radio telescope system, LOFAR. This showed that the unique and often custom-built hardware in such systems is difficult to combine with Ethernet networks. These networks rely on a complex multitude of supporting protocols and services to correctly direct traffic, many of which are not implemented in the highly specialised sensor nodes. Most of the challenges identified in that chapter occur on the boundary where data from specialised custom-built hardware is sent to general-purpose compute systems.

6.3.1 Impact

By replacing the traditional Ethernet-based infrastructure with a software-defined network, many of the issues previously described are alleviated. We foresee two distinct consequences:

1. The different way of processing packets in the packet processor will improve robustness of the system:
    - more direct control over packet forwarding
    - better handling of multi-homed hosts
2. The increased flexibility offered by the software-controlled data plane will open up the possibility for additional instrument functionality:
    - ability to dynamically change the data flow
    - integration of data flow and processing model

We discuss some examples below.

Robustness – Packet forwarding

A network switch typically receives data and forwards it to its destination. A MAC address table, matching output ports with host MAC addresses, is maintained in the switch, populated by monitoring source MAC address and incoming port of forwarded packets. When the appropriate output port cannot be determined, data is forwarded on

all output ports, with the exception of the ingress port, reducing the switch to a hub. This behaviour guarantees that packets have the highest possible chance of arriving at their destination, but in normal operation this should rarely be needed. Further traffic, such as unicast acknowledgements, usually allows successive packets to be properly directed.

In a specialised sensor network with custom hardware this may cause problems. This custom hardware may omit parts of the standard network stack, and it emits a continuous stream of uni-directional data generated by the sensors. A misconfiguration or failure of such a sensor may cause the network to be flooded by packets forwarded on all ports. In effect this failure mode may be considered a self-inflicted Distributed Denial of Service (DDoS) attack.

In a software-defined network, the way unrecognised packets are handled is configurable. Non-matching packets may be dropped or have their headers forwarded to the network controller for further processing. Neither of these actions will cause the network to be flooded, although the latter may cause significant load on the network controller.

Robustness – ARP flux

In high bandwidth systems, it may be necessary to connect multiple Ethernet interfaces in a node to the same network. While there are ways to *bond* these devices into a single virtual Ethernet interface, no guarantees can be given on the effective use of available bandwidth. Therefore we prefer to explicitly address each of the interfaces. Linux nodes tend to answer ARP requests for addresses they host on any interface, which may cause switches to associate the wrong port with that MAC address. This gives rise to ARP flux, in which data addressed to a node does arrive, but on the wrong interface, causing the operating system to drop this data. Figure 6.3 shows ARP flux in a system with two network interfaces.

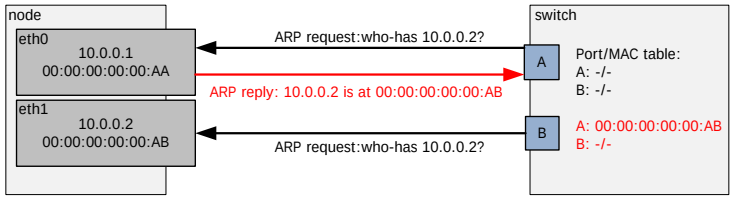


Figure 6.3: ARP flux with two network interfaces.

The positive control a software-defined network allows over the data flow mitigates this problem. Whereas a conventional network relies on MAC learning to determine the physical output port to forward packets to, in a software-defined network the output port is explicitly defined.

Flexibility – Publish-Subscribe model

Unicast traffic, traffic from one destination to another, is very difficult to redirect or duplicate at the network level. Switches may sometimes be configured to output traffic to a designated port. This is resource intensive, and data forwarded on such a monitoring port is not easily consumed by hosts. Such data will be addressed to the original destination host, both on Layer 2 (MAC address) and Layer 3 (IP address). Generally, a host operating system will transparently drop data not specifically addressed to it, unless it is running in promiscuous mode.

A software-defined network gives us the flexibility to not only forward data to a specific physical output port, but also modify the packet headers to match the receiving host. In essence, this breaks the tight coupling between sender and receiver, since the sending peer no longer necessarily has knowledge of the receiving peer's address. By moving this control to the software layer of the receiving system we can build what can conceptually be regarded as a publish/subscribe system. We note that this is similar to IP multicast in concept, but not implementation. Where IP multicast relies on the network infrastructure to properly direct data, in a software-defined network this is done by the network controller. This not only improves flexibility, but also allows us to integrate this functionality more deeply into our software, as described below.

In a radio telescope, we may use this functionality to process data more than once, without the overhead of re-transmitting data. This is useful when running multiple processing pipelines, serving multiple observation modes, or for instance to commission new processing hardware or algorithms. In the latter use-case, results from newly installed hardware or software implementations would be compared to those from established systems to verify functionality.

Flexibility – Data flow as part of the processing model

The addition of a software component into the data plane of a radio telescope allows us to integrate this data plane into our software systems. This more direct and positive control over the data flow allows us to manipulate the data flow without having to reconfigure the sending peer. Data movement may now be directly coupled to the processing schedule, creating for instance a system of data-driven round-robin processing, where data is directed to processing nodes as needed. Going even further we could envision implementing an in-network transpose, similar in concept to an `MPI_AllToAll()`. The directing of data would be based on a combination of L2 and L3 addresses that identify the content sufficiently.

6.4 SDN implementation

We explore the viability of a software-defined network as part of a modern science instrument in the remainder of this chapter. We focus on the most prevalent and most widely implemented software-defined networking protocol currently available: OpenFlow. First we investigate which features provided by OpenFlow are required to implement a suitable software-defined network for our application. We summarise the support

for these features in four available SDN-capable switches, both hardware and software, in Section 6.6.

Our proof of concept is simple, but allows us to gauge the level of support in current platforms. Two cases are implemented, both manipulating a data stream emitted from a LOFAR signal generator. This emulates a LOFAR station-processing board and emits a continuous UDP/IP data stream to a single host identified by its IP and MAC address.

Two main scenarios are tested that are representative of how a software-defined network may be used in a radio telescope.

1. redirect data destined for A to B
2. duplicate data destined for A to also go to B

Since we are most interested in the feasibility of the concept rather than performance of these platforms, we limit ourselves to a single data stream of around 700 Mbps, which is $\frac{1}{4}$th of the data rate of a single LOFAR station.

Redirecting data

First, we redirect the data from its original destination to a second host. This is achieved by modifying the Layer 2 destination MAC address and the Layer 3 destination IP address, after which the packet is forwarded to the appropriate port. We use the Set Field functionality defined in the OpenFlow standard to manipulate destination MAC and IP address fields. Note that the implementation of none of the Set Field features is required in the OpenFlow specification.

Duplicating data

Second, we modify the data flow such that it is emitted not just to its original destination, but also to a second host. We forward packets unmodified to the physical port hosting the original destination, while a second copy of the data stream undergoes modification much like described above. Since this requires multiple independent actions on the same data, a different approach is required. OpenFlow groups allow the creation of action buckets, which are processed and applied independently. We configure the Group such that all buckets are applied to a packet, a functionality that is required according to the OpenFlow standard. However, we note that the applied actions are bound by the same limitations described above. The Set Field functionality we rely on to readdress packets is still optional.

Measurements

These use cases show the functional requirements we put on the available hardware. Next we do a limited performance analysis by showing how these platforms handle data streams representative of the ones seen in the LOFAR radio telescope. Our interest is mainly in uninterrupted streaming data flows, so we concentrate our investigation on:

1. the latency between a given command and the data arriving at the appropriate node

2. any data lost when switching data streams

In normal operation, where data is forwarded to fixed output ports, we expect no data to be lost regardless of the switching paradigm used. When actively modifying packet headers we potentially introduce overhead that may lead to lost data. We test if current-day hardware is capable of modifying header information at wire speed, without losing data. In recent switching equipment this should be done in hardware, which should be fully capable of achieving a lossless line rate.

We are also interested in the latency between a given command, and the first packets being received on the appropriate host. While our application is not particularly latency sensitive, from an operational point of view a predictable latency is much preferred. Although the data stream uses an unreliable protocol and lost data cannot be retrieved, we can tolerate limited loss of data, if detected.

6.5 Experimental setup

We investigate four SDN-capable Ethernet switches, summarised in Table 6.1. These interface with a Ryu [1] network controller that exposes a RESTful interface for our test application to communicate with.

Switch type	Switch ASIC	Software version	OpenFlow version
Mininet / Open vSwitch	VirtualBox virtual machine	Open vSwitch 2.5.0	1.0 - 1.6
Pica8 P3290	Broadcom BCM56534 (Firebolt3)	Open vSwitch 2.0.90	1.0 - 1.3
Cetect V350-48T4X	Centec CTC5163	3.1.16.8.alpha Open vSwitch 1.9.90	1.0 - 1.3
Brocade ICX7250	Broadcom BCM56344 (Helix4)	08.0.40bT213	1.0 and 1.3

Table 6.1: Investigated switch platforms.

None of these platforms are particularly high performance, and per-port speed is limited to a Gigabit per second at most. Nevertheless, these experiments give a useful indication of the viability of software-defined networks for our particular application.

NTP (Network Time Protocol) synchronised nodes measure the time difference between the moment the command is given, until the data arrives at the new destination. While we do some measurements of latency, we note that our primary interest is in the functionality. Although our proof of concept only uses limited bandwidth and a single stream, due to the near embarrassingly parallel nature of the data flows in radio astronomy, it is a representative test. This application also allows for an excellent exploration of the functionality required for this application.

6.6 Functionality investigation

In the following sections we investigate the functional support for these OpenFlow features in the four platforms described in Table 6.1. Many of these OpenFlow features are

optional, which means that implementation of these is not mandatory and often omitted. Furthermore, we found that some implemented features may lack hardware support, leading to significant bottlenecks.

Implementation of the use cases described in the previous section relies on a number of functional elements in the OpenFlow standard:

- Set Field action destination MAC address
- Set Field action destination IP address
- Group Type=ALL: output to multiple ports
- Group Type=ALL: apply action instructions

We note that to be truly useful, a software-defined network must be able to support the above described functionality on Layer 3. While we can make a workable proof of concept at Layer 2 by manipulating just destination MAC address and not destination IP address, this requires either that all receiving hosts have the same IP address, or that all host Ethernet interfaces are run in promiscuous mode, which is not feasible in practice.

We investigate the available functionality in a virtualised software switch, Open vSwitch in Mininet, and three hardware switches: a Pica8 P3290, a Centec V350 and a Brocade ICX7250.

6.6.1 Mininet / Open vSwitch

Mininet is a virtual machine-based virtual network environment. We used Mininet as our primary development platform, using Open vSwitch as the virtual switch. Since the same software is also used in two of our hardware platforms, it is interesting to compare the feature set supported in a pure software environment with available hardware implementations. We used the most recent stable version of Open vSwitch at the time, which fully implements OpenFlow up to and including 1.3, including all optional features.

6.6.2 Pica8 P3290

The first physical switch we investigate is a Pica8 P3290, located in the OpenLab testbed of the University of Amsterdam, an experimentation environment for multi-domain networking research [92]. Based on a Broadcom switch ASIC and a Freescale CPU, it uses the same Open vSwitch software as described above. This switch supports all features required to redirect data based on Layer 3 packet header fields. However, modifying the destination IP address causes major packet loss, which suggests that this is handled in software, not by the switch ASIC. As a work-around, we assigned both receiving hosts the same IP address and redirected data based on just Layer 2 fields (i.e. destination MAC address). This is not necessarily a viable solution in production, since we rely on hosts sharing IP addresses, which breaks normal IP networking. To duplicate data we rely on OpenFlow groups, but in the Pica8 switch these did not function as expected. Only a single, seemingly random, apply-action was executed before packets were forwarded to their respective ports. Thus correctly addressed packets would only appear at

one receiving host, while all others would receive packets addressed to the wrong destination MAC address. For this particular platform we abandoned the use of OpenFlow groups and instead installed a single flow that sets the destination MAC address to the broadcast address and forwards the packets to all relevant output ports. We note that this solution rules out the use of hybrid networking, since the use of the broadcast address in streaming data will cause non-SDN switches to forward this data on all ports, quickly overwhelming the network.

6.6.3 Centec V350

A software-defined network was included in the latest incarnation of the distributed ASCI supercomputer, DAS-5 [18]. The installation at the Vrije Universiteit Amsterdam has two Centec V350 switches based on custom silicon. This is targeted specifically at software-defined/OpenFlow applications and uses the familiar Open vSwitch software environment. Of the investigated hardware platforms, this was the only one that performed flawlessly. All features we use to redirect and duplicate data using either Layer 2 or Layer 3 header information are supported without observable bottlenecks. While the reported Open vSwitch version on this switch is 1.9.90, it supports OpenFlow versions 1.0, 1.2 and 1.3. Since this version of Open vSwitch only supports OpenFlow 1.0, we speculate this is a fork with significant additional development.

6.6.4 Brocade ICX7250

Apart from the two specific OpenFlow-targeted switches investigated above, we looked at a more mainstream product with OpenFlow support. The Brocade ICX7250 is a top-of-rack switch based on a Broadcom ASIC, and like the Pica8 switch, located in the OpenLab testbed of the University of Amsterdam. While it supports OpenFlow versions 1.0 and 1.3, we found that there are many caveats that limit the usefulness of this hardware for our application. Although the switch is able to match flows based on Layer 3 fields, it can only modify a very limited set of header fields. IP destination address is not one of these, although destination MAC address can be modified. This means that directing or duplicating data can only be done based on Layer 2 header fields, not Layer 3 header fields. Furthermore, the switch is not able to output to multiple ports while modifying packet headers, effectively rendering duplication of data impossible.

While testing this platform, we found that we regularly had to reboot the switch to solve unexplained packet loss. Identical flows installed after reboot yielded far better results. We cannot explain this behaviour at this time, but suspect a bug in the firmware.

6.6.5 Summary

In Table 6.2 we summarise our experiences with the four described switches for our application. While we note some missing features and we were not able to duplicate data on the Brocade switch at all, we successfully implemented our proof-of-concept application on all switches. Nevertheless, it is clear that support for the desired OpenFlow functionality varies widely. Furthermore we found an undocumented deficiency in the implementation of OpenFlow groups in the Pica8 switch.

OpenFlow feature	Open vSwitch	Pica8	Centec	Brocade
Set Field action: destination MAC address	supported	supported	supported	supported
Set Field action: destination IP address	supported	supported in software	supported	not supported
Group Type=ALL: apply-action instructions	supported	not fully supported	supported	supported
Group Type=ALL: output to multiple ports	supported	supported	supported	not fully supported

Table 6.2: Level of support for the important OpenFlow features for our investigated switches.

6.7 Latency and loss

In Section 6.6 we concluded that not all platforms are capable of Layer 3 packet header manipulation at line rate. We investigate the loss and latency of all four platforms while redirecting and duplicating data using Layer 2 and, when available, Layer 3 packet header information. While the virtual network environment is functionally very well developed, it is limited in performance. Where other platforms will be expected to handle the output of a single LOFAR hardware stream, at approximately 700 Mbps, we reduce this to about 45 Mbps for Mininet to avoid packet loss. The virtual machine environment, and the software-based Ethernet switch, make the latency measurements for the Open vSwitch system less than reliable. For completeness, and as a comparison against the other hardware-based switches, including these is still valuable. In all cases ten measurements are done, and averages are shown in the Tables 6.3 and 6.4. Data loss is measured over a period of three seconds, with 12000 packets generated per second.

6.7.1 Latency and Loss – Redirecting

In Table 6.3 we summarise the latency and loss of data when redirecting data. As discussed before, Layer 3 functionality is not available on the Pica8 and Brocade switches, therefore no data is available. We note that redirecting data using an installed flow that modifies the packet header and forwards packets to the appropriate port will lead to limited loss of data on the Centec and Brocade switches and the software Open vSwitch environment. When the flow is installed, 16 or 32 packets are lost that arrive neither at the original nor the new destination. This number is unrelated to the data rate, since we ran a low bandwidth experiment and observed similar lost data. It is likely that a packet buffer is flushed when the flow is installed.

To mitigate this behaviour, we could implement the same functionality using OpenFlow groups, in much the same way as we did with the duplication of data. Once data is flowing to the new destination, the original destination may be removed from the group. Unfortunately, our chosen network controller lacks functionality to remove group entries in the RESTful interface. Our analysis above indicates that neither the Pica8 nor the Brocade platform will support this more advanced redirection of data.

The Pica8 switch exhibits significant loss of data when the appropriate flow is installed. While this amounts to approximately 1.3% of the total data flow, it is not incidental, suggesting a bottleneck in the implementation. While not nearly as significant as observed when we modified L3 headers on this switch, it is still worrisome. We note that both the Pica8 and the Brocade switch exhibit high latency and extreme variance,

Layer 2	Open vSwitch	Pica8	Centec	Brocade
average latency (ms)	202	74	5.9	265
standard deviation	316	212	0.76	74
average data loss (packets)	16	457	29	32
standard deviation	0	381	7	0
Layer 3	Open vSwitch	Pica8	Centec	Brocade
average latency (ms)	158	n/a	6.9	n/a
standard deviation	313	n/a	1.2	n/a
average data loss (packets)	16	n/a	29	n/a
standard deviation	0	n/a	7	n/a

Table 6.3: Latency and loss while redirecting data.

especially compared to the Centec switch. The stability of the Centec performance is impressive.

6.7.2 Latency and loss – Duplicating

Table 6.4 shows our results for the second case study. We install an OpenFlow group that forwards packets unmodified to the intended destination. A second copy of all packets is modified such that the destination addresses (Layer 2 and Layer 3) match the second receiving host and are forwarded on the appropriate port. The Pica8 switch does not support modification of the destination IP address at this rate, therefore for this switch we only modify the destination MAC address. In our Open vSwitch software environment we measure both L2 and L3 duplication of data for reference. Considering the excellent performance of the Centec switch, and the possible disturbance to the cluster network due to the necessary modification on the hosts needed to make this work, we only show L3 performance for that platform.

The Centec switch once again performs very well. We again note significant data loss in the Pica8 switch, while none of the other workable switches exhibit any observable loss.

6.7.3 Summary

Both the Pica8 and the Brocade switches exhibit relatively high latencies, especially compared to the Centec switch. The loss of data on the Pica8 switch is worrisome, since it is indicative of a bottleneck that will become more pronounced when we scale up this proof of concept. In all measurements, the Centec switch showed excellent performance. Latency is low and stable, and apart from the initial packets lost when a flow is installed, no further loss was observed.

Layer 2	Open vSwitch	Pica8	Centec	Brocade
average latency (ms)	29	34	-	n/a
standard deviation	38	70	-	n/a
average data loss (packets)	0	314	-	n/a
standard deviation	0	308	-	n/a
Layer 3	Open vSwitch	Pica8	Centec	Brocade
average latency (ms)	59	n/a	5.1	n/a
standard deviation	55	n/a	0.43	n/a
average data loss (packets)	0	n/a	0	n/a
standard deviation	0	n/a	0	n/a

Table 6.4: Latency and loss while duplicating data.

6.8 Discussion and future work

In this chapter we show that a software-defined network in a modern radio astronomical sensor network is feasible. Support for the required OpenFlow functionality varies widely and is often poorly documented. More detailed investigation, targeting more modern and higher performance platforms is needed. However, the concept is sound and the additional flexibility and robustness are highly desirable. Adding positive, explicit and centralised control over the data plane, adds an additional layer of control to the data flow in a radio telescope. This opens up the possibility of integrating the data plane in the processing schedule so we can steer data much more dynamically to the compute resources needed.

We note that the two hardware switches that show poor performance are both based on Broadcom switch ASICs targeted at conventional top-of-rack switches. This may indicate that chips designed for more conventional switches with added SDN functionality may be limited in performance compared to chips specifically designed for software-defined networks. Further investigation on this subject is needed.

A recent development is P4 [27], a high-level language that allows a network engineer, or an instrument engineer, to dynamically program packet processors that make up a network. The OpenFlow protocol described in Section 6.2.2 has grown in complexity over the years, adding possible header field entries and multiple stages of rule tables. This complexity is constrained by a rigidly specified and repeatedly extended protocol. P4 is designed to be a much more flexible and higher abstraction level language. A more flexible and more extensible high-level language such as P4 may make the implementation of advanced packet processor features easier. However, although we noted highly varied levels of support for the required functionality, these were independent of the OpenFlow standard. The implementation, not the standard, limits the usefulness of some of the switches we investigated.

The networks investigated in this chapter bear some resemblance to multimedia broadcast infrastructures, where the use of IP multicast is well established. The South

African MeerKAT radio telescope [80], currently under construction in the Karoo desert, uses IP multicast in its data flow design. Their work has shown that while the network can successfully distribute instrument data over the relevant subscribed processing nodes, significant engineering effort was required [7]. The programmable nature of a software-defined network makes it inherently more flexible than an IP multicast based infrastructure.

6.9 Conclusions

In this chapter we have investigated the viability of a software-defined network in a modern radio telescope. Based on an investigation on the functionality available in four switches and an implementation of a simple proof-of-concept program, we conclude that the concept is viable and valuable. A software-defined network by its very definition will mitigate some of the robustness issues we have discussed in Section 6.3. In addition, we have shown that the additional flexibility we envision is feasible on at least some of the investigated platforms.

We note that the supported functionality in the investigated platforms varies greatly. While some offered support for all required features, others do not implement critical features or implement some in software only. The OpenFlow standard is characterised by a large number of optional elements, that may or may not be implemented by the manufacturer, some of which we depend upon. Although all switches claim OpenFlow v1.3 support, support for optional functionality is often not well documented. It is noteworthy that the way Pica8 implements apply-action instructions in Groups violates the OpenFlow standard.

6.10 Our propositions in this chapter

Finally let's return to the propositions introduced in this thesis.

6.10.1 Value proposition

There is a convincing argument to be made that the optimisations discussed in this and the next chapter are conceived primarily to add additional value to the instrument without significantly adding cost. As such, while the primary contribution of these chapters are to the *optimisation* proposition, they also contribute in a limited way to the *value* proposition.

6.10.2 Optimisation proposition

This chapter epitomises the *optimisation* proposition. Experience with the LOFAR telescope had identified the boundary between custom and commodity hardware as a potential problem area. In particular the way packets are sent over the network, without the benefit of a full IP stack on the custom hardware boards, was cause for concern, considering Ethernet switches require such a stack for correct operation. Using

a software-defined switch environment, based on OpenFlow, we showed that we can mostly solve the problems identified with conventional Ethernet switches, while simultaneously adding functionality to the instrument. Unfortunately we also had to conclude that, while in a fully implemented software environment our experiments worked flawlessly, none of the available hardware switches worked nearly as well. This was mostly due to optional features that were not implemented, and severe bugs in the vendor firmware. However, the concept was proven, and the additional functionality is interesting. Therefore, investigation of more modern switches and alternative software implementations is future work, the direction of which is intended to avoid the reliance on vendor implemented firmware features.

Acknowledgements

We thank the SNE OpenLab for the equipment they made available for this research. Kees Verstoep with the Vrije Universiteit Amsterdam was instrumental in making the Centec experiments a success, for which we are thankful. We gratefully acknowledge the copy-editing and reviewing by Ágnes Mika. This work is supported by the SKA-NN grant from the EFRO/Koers Noord programme from Samenwerkingsverband Noord-Nederland.

CHAPTER 7

Energy-Efficient Data Transfers in Radio Astronomy with Software UDP RDMA

Przemyslaw Lenkiewicz [1] *and P. Chris Broekema* [2] *and Bernard Metzler* [3]

Context and contributions

This chapter is a slightly modified version of a paper that was published in the Future Generation Compute Systems journal, which itself was an extension of a paper that was presented at the Innovating the Network for Data-Intensive Science (INDIS) workshop 2016.

While most of the implementation and measurements in the paper were done by Lenkiewicz, Broekema was the originator of the research. His novel idea, looking at reducing energy consumed by significant processor overhead in receiving large amounts of streaming data, was the key insight in this research. Furthermore, it was Broekema's insight that this work was highly applicable to radio astronomy in general, and the Square Kilometre Array in particular. Broekema and Metzler initiated this research stream as part of the DOME project, and Broekema led the work done at ASTRON in collaboration with the DOME team at IBM Zürich. This work is an example of the *optimisation* proposition.

[1] IBM Research - Netherlands
[2] ASTRON, the Netherlands Institute for Radio Astronomy
[3] IBM Research - Zürich

Abstract

Modern radio astronomy relies on very large amounts of data that need to be transferred between various parts of astronomical instruments, over distances that are often in the range of tens or hundreds of kilometres. The Square Kilometre Array (SKA) will be the world's largest radio telescope, data rates between its components will exceed Terabits per second. This will impose a huge challenge on its data transport system, especially with regard to power consumption. High-speed data transfers using modern off-the-shelf hardware may impose a significant load on the receiving system with respect to CPU and DRAM usage. The SKA has a strict energy budget which demands a new, custom-designed data transport solution. In this chapter we present SoftiWARP UDP, an unreliable datagram-based Remote Direct Memory Access (RDMA) protocol, which can significantly increase the energy-efficiency of high-speed data transfers for radio astronomy. We have implemented a fully functional software prototype of such a protocol, supporting RDMA Read and Write operations and zero-copy capabilities. We present measurements of power consumption and achieved bandwidth and investigate the behaviour of all examined protocols when subjected to packet loss.

7.1 Introduction

Modern radio telescopes, such as the LOw Frequency ARray (LOFAR) [159] and the upcoming Square Kilometre Array (SKA) [61] are in essence large-scale, distributed sensor networks, characterised by large numbers of receivers producing vast amounts of data. This data is often generated by custom hardware in remote areas, while processing this data into usable science data is done in data centres in nearby cities. Receiving and processing these data streams is a computationally intensive task that may consume considerable amounts of energy. A current state-of-the-art example is LOFAR: 51 antenna stations produce around 250 Gbps of sensor data in total, to be transported over 65 km to the central processor. The Square Kilometre Array (SKA) will produce much more data, to be transported over much longer distances. This data stream, about 3 Tbps per telescope, is to be transported from the Western Australian desert to Perth, and from the Karoo desert to Cape Town, both several hundred kilometres away.

Whereas the available compute capacity in current telescopes is often limited by available capital, in the SKA it will likely be limited by available energy. Experience with the LOFAR radio telescope has shown that receiving large volumes of sensor data may consume significant compute resources [123, 124]. These consumed resources cannot contribute directly to the science result. It is therefore useful to investigate ways to minimise the resources, and energy, required to receive streaming radio astronomical data. Reducing the protocol overhead allow more of the precious resources to be dedicated to scientific processing. More energy-efficient handling of incoming data can be directly translated into additional science output within the limited available energy budget.

The Linux network and IP stack was designed with robustness and security in mind. Strict separation between user and system resources is maintained. Received data is copied several times and will trigger several interrupts and context switches before the

user application gains access to it. In the IBM Blue Gene/P supercomputer a different bottleneck, namely software handling of Translation Lookaside Buffer (TLB) misses, was mitigated by bypassing conventional kernel processing [169]. This was used to significantly decrease compute resources consumed while receiving LOFAR data. In this chapter we propose a similar approach aimed at the Square Kilometre Array. The Linux IP stack requires significant resources, in particular while receiving large volumes of sensor data. We propose to bypass the host operating system and place data directly into user memory. While this also bypasses several of the security features that are essential in typical network stack, in a tightly controlled and private network, such as found in a scientific instrument, these are less crucial. We expect a reduction in resource consumption and therefore a reduction in the amount of consumed energy. Since the SKA Science Data Processor is expected to be bound by very tight budgets, in particular in available energy, reducing the computational cost of receiving data would allow for more science, improving the scientific efficiency of the instrument.

Remote Direct Memory Access (RDMA) technology has been essential in high-performance networking to resolve similar issues, namely to allow higher bandwidth, lower latencies and lower CPU utilisation. RDMA-capable network interface controllers (RNICs) provide this by moving data directly from the user space memory of one machine to that of another, without involving either of the host operating systems. The application layer is involved only on the side where the request is issued and it can access the contents of memory buffers on a different host thanks to memory pre-registration. The RDMA technology is a very good example of how the data movement process can be optimised for a specific scenario, helping to utilise the full capabilities of the hardware. However, the currently-available RDMA solutions lack some of the features that are necessary for a scenario such as the SKA. In particular, the target scenario both requires a more efficient handling of the expected very high bandwidth-delay product of the data transfer channel, and imposes application specific requirements on time sensitive, partial data transfer reliability [39].

In this chapter we address the data transport challenges for modern radio astronomy instruments. We introduce a possible solution that measurably reduces the consumption of CPU resources and energy associated with that data transport. In particular, we design and implement an efficient communication protocol for transferring high rates of astronomical data over long distances with the goal of being more energy-efficient at the receiving end.

The main contributions of this chapter are:

1. we design and prototype in software a partially reliable, RDMA-based transport protocol suitable for modern radio astronomy applications;

2. we present experiments with results showing that the energy-efficiency of the prototyped transport stack is improved compared to standard UDP data transfer;

3. we argue that further, more dramatic improvements in efficiency are possible when support for this protocol is implemented in hardware.

7.2 The Square Kilometre Array

The Square Kilometre Array has been described is sufficient detail in Chapters 3 and 4. For this Chapter, we note that the SKA Science Data Processor (SDP) is expected to be bound by strict energy and capital budgets that will severely limit the scale of the system. In Section 7.1 we cite previous work that showed that receiving large volumes of data itself also requires significant resources. The resources consumed just receiving data do not directly contribute to the scientific output of the SDP. In this work we aim to reduce these compute resources required to receive the incoming data stream by avoiding a well known system bottleneck: the Linux IP stack and the associated kernel overhead.

Considering the large distances and volumes of data, it is not feasible to use a reliable data transport protocol for the data transport between CSP and SDP. This would require constant buffering of the transmitted packets at the sending side until confirmations from the receiving side arrive. In a highly optimised real-time environment, such as the CSP correlator system, this would incur very significant cost and performance overheads. The chosen transmission protocol for this data stream is therefore unreliable, based on UDP/IP over Ethernet, with far lower sender-side overhead. At this point the transported data is highly redundant. Loss of a fraction of this data will result in reduced signal-to-noise ratio in the end-product, but this is, within reason, acceptable. Goal of this work is to maximise the scientific output of the Science Data Processor per invested Euro and/or Joule by minimising cycles spent on data transport that don't directly contribute to the science output.

The specific set of requirements for this particular SKA data transport component can be summarised as follows:

- very high data rates, several Terabits per second
- almost entirely uni-directional traffic
- UDP/IP over Ethernet
- prioritising bandwidth over latency
- desire for very high energy-efficiency
- full reliability is not crucial, some data loss is tolerable[1]

In the remainder of this chapter we investigate how an existing industry standard RDMA implementation can be modified in such a way that it can be used to transport SKA specific data streams. By avoiding a known bottleneck we expect to save a measurable amount of computational resources and energy. This can immediately be translated into increased scientific performance for the same investment.

[1] A SKA SDP requirement states that at most 1% of data loss, excluding data flagged due to interference, is tolerable. For comparison. in LOFAR at most 5% of loss overall is tolerable, including flagging and lost data in pipelines. However, lost data will always lead to reduced signal to noise in the science product. Minimising data loss is therefore important.

7.3 RDMA, iWARP and SoftiWARP

Receiving multiple high-bandwidth UDP/IP data streams requires significant CPU resources. Since CPU cycles can be translated into consumed energy, it can be assumed that a more efficient way to receive large data streams will consume less energy. In addition, compute resources spent on receiving data cannot be utilised for data reduction or processing.

Implemented as an operating system service, the Linux network I/O stack was designed with the main focus on robustness and security while maintaining good performance. Applications access network services via the socket API. To achieve separation and protection, all communication data are copied between application buffers (user space memory) and operating system (kernel) memory within the socket layer. On the transmission path, after copying data into the kernel, network protocol output processing packetises the data, stores it for potential retransmission and informs the network adapter to fetch the packets for wire transmission. In the network packet input path data are first moved from the network card into kernel memory and an interrupt is issued, which handles network protocol processing within the kernel. As a result of protocol processing, kernel data buffers containing the received data are queued to the socket receive queue for application retrieval. Within a system call, the application eventually copies those data from kernel memory to application receive buffers, which typically involves waking up the application thread waiting for data reception. Both in the sending and receiving path, traversing the Linux networking stack incurs non-negligible overhead (interrupt handling, context switches, network protocol processing, data copy operations), which degrades application-available CPU processing power, while limiting achievable communication bandwidth and adding to end-to-end communication latency. Moving the data directly between the network device and application buffer would avoid such overhead, but if not done properly, would violate the data protection and separation principles of the operating system. However, in a tightly controlled and private environment, such as in a scientific instrument, these limitations might be acceptable.

In the past decade the Remote Direct Memory Access (RDMA) technology has been gaining more and more relevance in the field of high-speed communication. Its development was driven by the need for high throughput and low latency networking, especially in High Performance Computing. RDMA provides this by moving data directly from the user space memory of one machine to that of another, without involving host operating system and minimising host CPU usage. The application layer registers memory buffers with the local RDMA-capable network interface controller (RNIC) for remote write or read access. Under the control of local and remote RNIC, RDMA write operations transfer data from a local buffer to a tagged remote buffer that was advertised by the peer, whereas the RDMA read operation transfers data from a tagged remote buffer to a tagged local buffer. The application layer is involved only on the side where the request is issued. Any application buffer used as a source or target for an RDMA operation must be pre-registered with the local RNIC device, and is typically pinned into physical host memory. This allows the RDMA device to access the buffer in physical memory without further OS intervention. To allow overlapping communication and computation, RDMA offers an asynchronous communication interface. RDMA operations are posted as Work Requests (WRs) to a communication endpoint and are asynchronously processed by the

RDMA device. Work completions are signalled and retrieved asynchronously as well.

RDMA is provided through several network technologies, including Myrinet [25], Infiniband [112], RDMA over Converged Ethernet (RoCE) [147, 22] and iWARP [71, 116]. The functionality and performance of these standards has been evaluated and compared in various studies [90, 115]. Well-known programming interfaces, like the Message Passing Interface, may be used in order to access the RDMA functionality on different hardware [91].

Both RoCE and iWARP are deployed over Ethernet, which makes them very interesting candidates for the SKA data transport service. RoCE defines the transmission of InfiniBand packets directly over Ethernet, which limits its scope to the Ethernet broadcast domain and thus leaves it non-routable. To solve that issue, a recent protocol extension (RoCEv2) puts it on top of UDP/IP. On the other hand, iWARP defines RDMA operations on top of TCP/IP networks, giving it the advantage of being compatible with the existing Internet infrastructure. Unfortunately, both RoCE and iWARP rely on the implementation of a rather complex protocol state machine (TCP or InfiniBand) meant to provide a level of data transmission reliability which is not needed and even obstructive for the intended use: data to be transmitted have a limited relevance in time – in case of partial data loss the protocol should favour the transmission of new data over the retransmission of lost fragments. Lost data fragments shall result in just dropping the entire affected application level message at RDMA protocol level, while keeping the end-to-end connection intact.

In our work towards an energy-efficient protocol for modern radio astronomy we have chosen the iWARP standard as the baseline, but extended it with an unreliable service. This was achieved by replacing the TCP protocol with UDP and modifying the semantics of the RDMA application interface.

7.3.1 Implementation of iWARP in software

Although the full range of advantages of RDMA is only available through hardware support for iWARP (in order to offload I/O and protocol processing from the CPU), a software implementation can also be well motivated. iWARP is still a relatively young technology and therefore it is useful to be able to rely on a software solution for testing and development purposes. Furthermore, the software version can be introduced in the less stressed parts of the infrastructure, whereas the more utilised parts would be equipped with iWARP-capable NICs – provided that the software implementation can operate in such a mixed scenario. Thanks to the RDMA semantics and the asynchronous API, even a software implementation can provide benefits such as a zero-copy data transmit path and less application interaction/scheduling, which can lead to increased performance and lowered CPU load and power consumption. Software iWARP can also be used for migrating existing applications to the RDMA interface without the need for RDMA hardware. Finally, it can ease the development of new, experimental extensions to the RDMA stack without hardware prototyping. The SKA scenario is a good example of such a case, as we want to experiment with an implementation of iWARP that is tailored specifically for our needs.

The idea to implement the iWARP protocol fully in software has been already approached and there are solutions available, such as the Software iWARP implementation

by the Ohio Supercomputing centre [51],[52] or the SoftiWARP [104] implementation by IBM Research.

It is however important to note that a software implementation of the iWARP protocol will most likely not guarantee a power efficiency to meet the energy budget requirements of the SKA. This choice gives a good ground for experiments on the points relevant for the scientific instrument, namely CPU utilisation, power consumption and behaviour under packet loss. The final solution, one that can be incorporated in the design of SKA, should rely on hardware support. This work could of course be seen as a step towards such solution, as all the created code will be made available in a public repository.

7.3.2 SoftiWARP

The work presented in this chapter is based on the SoftiWARP (SIW) open source software implementation of the iWARP protocol suite, developed at the IBM Zürich Research Lab and available from GitHub[2]. SoftiWARP comprises two main building blocks: a kernel module, which implements the iWARP protocols on top of TCP kernel sockets, and a user level library. SoftiWARP integrates with the industry standard OpenFabrics[3] RDMA host stack and thus exports the OpenFabrics RDMA API to both user space and kernel space applications. Due to close integration with the Linux kernel socket layer, SoftiWARP allows for efficient data transfer operations. On the sending side, it supports zero copy data transfers out of application buffers. On the receiving side, the implementation makes use of target buffer address information available with the RDMA protocol headers: the packet payload is directly copied from their in-kernel representation (`sk_buff`) to the final application buffer without scheduling the receiving application. Since the implementation conforms to the iWARP protocol specification, it is wire compatible with any peer network adapter (RNIC) implementing iWARP in hardware.

7.3.3 Implementing an unreliable connected SoftiWARP service

In order to fulfil the requirements of the SKA we have defined and implemented a new unreliable, connection oriented RDMA transport protocol based on SoftiWARP. Here, communication between hosts is implemented over UDP kernel sockets instead of the reliable, connection-oriented TCP. The issues with TCP and UDP protocols for long-distance data transfers has been addressed in multiple studies. In [149] the authors have pointed out a poor utilisation of network capacity when using TCP in long-distance transfers and proposed a new congestion control algorithms to partially address this problem. In [42] we can find an evaluation of new advanced TCP stacks, which can give good performances on high speed long-distance network paths and limit the need of using multiple parallel streams. In contrast, other work on high bandwidth long distance connections conclude that TCP is able to achieve good utilisation, but hardware and software architectures of the computers and their IP stacks may limit performance [97,

[2] https://github.com/zrlio/softiwarp
[3] https://www.openfabrics.org

14]. The presented studies show clearly that a reliable connection brings shortcomings that we have also mentioned before, namely growing complexity of traffic management, delays, buffering and therefore, higher CPU utilisation and power consumption. An unreliable transport protocol can mitigate or avoid many of the identified shortcomings.

In SoftiWARP UDP the unreliable connection is used both for the connection management operations, as well as the data transfer. After connection setup, the application data transfer does not enforce reliability, but is implemented in an unreliable, message-oriented manner: the sender segments the RDMA message into a set of UDP datagrams, which are reassembled on the receiver side into the original message and, if completely received, delivered to the application. Messages which remain incomplete due to UDP packet loss are silently dropped at the receiver.

To retain the efficiency of the original implementation, any inbound, in-sequence data are directly placed into the application target buffer without intermediate queueing. At API level, error handling has been implemented as simple as possible: if a message remains incomplete due to data loss or corruption, the content of the target buffer remains undefined. If the lost message belongs to an RDMA Send/Receive operation, the current Receive operation remains incomplete and the receive buffer gets re-used for placing the next inbound RDMA Send. Corrupted RDMA Write messages just leave the application buffer in undefined state. While originally not defined for the iWARP protocol, an 'RDMA Write with Immediate Data' operation might further improve the handling of unreliable RDMA Writes at the target side: only if the RDMA Write operation completes successfully, the 'Immediate Data' are delivered to the application indicating the complete placement of a new RDMA Write. These data could carry additional application level information such as a message sequence number. Only InfiniBand and ROCE currently define this optional 'Immediate Data' semantics for RDMA Writes. With that, it is currently up to the application to detect corrupted data placed via RDMA Writes.

Unreliable RDMA Read operations are currently supported at an experimental level only. First of all, this operation is not required for the SKA use case: Data streaming is strictly uni-directional and only dictated by the sender delivering radio-astronomic data to a data processing entity. Secondly, supporting unreliable RDMA Reads requires a further extension of the protocol state machine at the RDMA Read initiator side, since it must detect permanently lost RDMA Read Request/Response pairs. A timer based detection of message loss appears to be a viable solution to the problem, but is currently not implemented.

The extended SoftiWARP implementation runs on both UDP and TCP and allows to select reliable connection (RC) or unreliable connection (UC) services on a per connection basis. For the UC service, the client side must first create a connection endpoint with an appropriate OpenFabrics service attribute, namely `IBV_QPT_UC`, which represents an Unreliable Connection Queue Pair. On the server side a listener endpoint for the same service type must exist. If the client connects its endpoint with the listener, a new server side endpoint will result, which is associated with the connecting client endpoint. After connection setup, both sides can use the new RDMA association for unreliable data transfer operations.

7.4 Experiments

In this section we present in-depth tests of SoftiWARP UDP and analyse how a software implementation of iWARP standard is able to perform in terms of achieved bandwidth and power consumption in comparison to standard TCP and UDP sockets. Our test platform comprises two server machines equipped with Intel Xeon E3-1240 v3 CPUs running at 3.40 GHz, 16 GB RAM and Chelsio T5-580 40 Gb RDMA-capable Ethernet cards. The machines are interconnected with a direct connection using a QSFP+ cable. The tests have been performed with:

- The Netperf[4] benchmark tool with additional tests implemented, which carry traffic over RDMA protocols, both over TCP and UDP,
- The LOFAR telescope traffic generator[5], which creates data packets at rates that correspond to that of a LOFAR telescope station. TCP and UDP Sockets as well as TCP and UDP iWARP is supported for data transport.

We use two measurement points in our experiments to precisely assess the energy consumption of the data transfers. Using the RAPL Technology [125] the values from Intel Processor's registers can be read and the power consumption of the CPU and DRAM can be estimated in a very accurate way. We use the Performance Application Programming Interface (PAPI) library[6] and the Likwid tool[7] to read the power meters. We have also constructed a custom-made power meter based on an Arduino board and voltage sensors attached to the PCI-Express slot [121]. Using this device we can measure the power consumption of the NIC with an accuracy of 1/100 Watt and 1 millisecond sampling rate.

7.4.1 Power consumption of Chelsio T5

The power consumption of the Chelsio T5 NIC has been measured using the power meter mentioned in the previous section, under numerous different test scenarios. The results of these tests are shown in Fig. 7.1 in a consecutive manner. The blue line presents the trace of power consumption of the Chelsio T5 NIC. The value of 9 W shows the idle state of the NIC and each peak of around 13.5 W represents one test being carried out. Peaks 1 to 6 represent Netperf tests over different transport protocols in the following order: SoftiWARP TCP, sending side; SoftiWARP UDP, receiving side; TCP sockets, sending side; TCP sockets, receiving side; Hardware iWARP, sending side; Hardware iWARP, receiving side. Tests 7 and 8 represent 50 instances of the LOFAR traffic generator, first the sending side, then the receiving side.

We can see from Fig. 7.1 that the power consumption of the NIC card is very similar in all cases and doesn't depend on the kind of transport protocol used. Further tests have been performed with varying message sizes and all available transport methods, on

[4] http://www.netperf.org
[5] https://gitlab.com/broekema/SDP_Controller
[6] http://icl.cs.utk.edu/papi/
[7] https://github.com/RRZE-HPC/likwid

sending and receiving side. All of them have shown nearly identical results of 9 W for idle state and 13.5 W for full link speed. Therefore, we can conclude that the power consumption of the RNIC is very consistent and doesn't show a dependency from the type of traffic. In the following sections we will focus only on the CPU and DRAM power consumption, as this is where all of the tested protocols show significant differences.

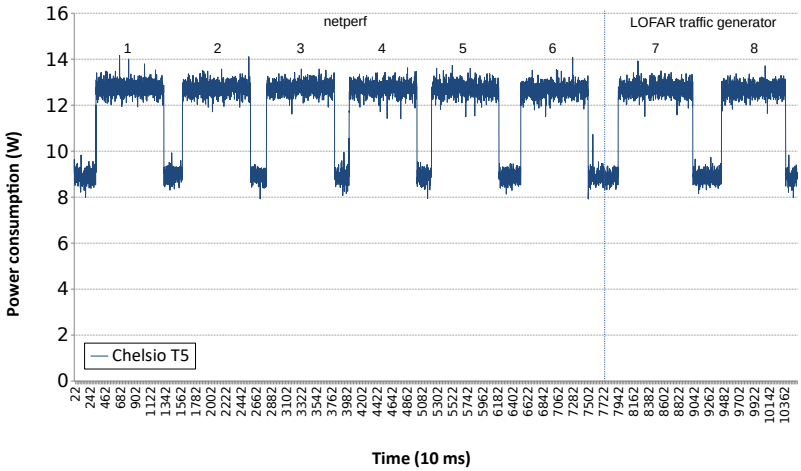


Figure 7.1: Power consumption of Chelsio T5 during eight consecutive tests using Netperf (tests 1-6) and the LOFAR traffic generator (tests 7-8).

7.4.2 Radio astronomy data flow

In this section we mimic the data flow from LOFAR, an operational radio telescope with very similar characteristics to the future SKA. A traffic generator is used to emulate the data produced by a LOFAR Remote Station Processing (RSP) board. This is a UDP/IP data stream, measuring approximately 760 Mb/s, transmitted in packets of 8 kB, which is a limit imposed by local memory on the station FPGA boards. Each LOFAR antenna field produces four of these data streams, totalling slightly more than 3 Gb/s per antenna field. LOFAR currently has 73 antenna fields, 24 core stations which may be split into two independent antenna fields, 18 remote stations and 7 international stations. Three more international stations are under construction, which brings the maximum LOFAR input data rate to almost 230 Gb/s. We generate 50 data streams in our experimental setup, which at 37.5 Gb/s corresponds to roughly $\frac{1}{6}$th of the total LOFAR data flow. Preliminary designs of the SKA system data flow make it likely that data transported between the CSP and SDP will have very similar characteristics, albeit with much higher data rates at longer distances. Considering the parallel nature of this data flow, energy consumption, and savings, scale linearly with increasing bandwidth. Apart from in-

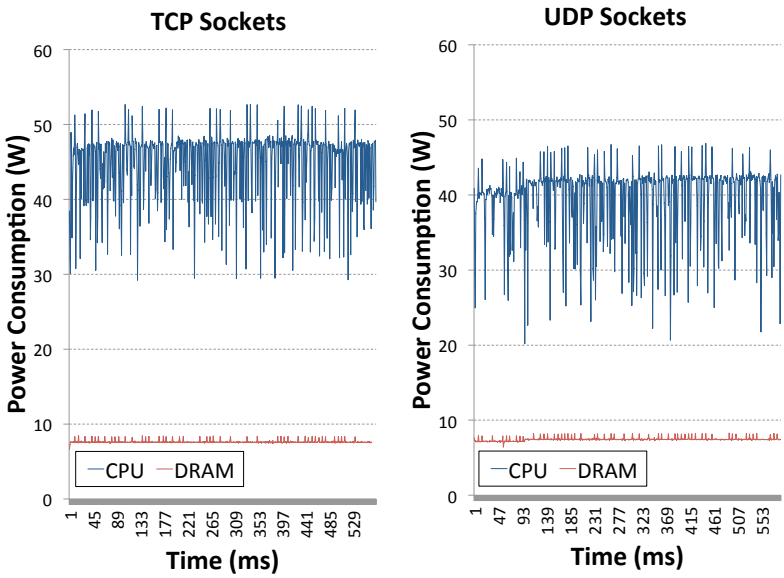


Figure 7.2: Power consumption of CPU and DRAM for receiving a transfer of LOFAR-like traffic over TCP and UDP sockets.

creased travel latency, and possibly a slightly higher chance of lost packets, distance should have little impact on the conclusions in this chapter. Our generator is capable of transmitting the said data stream using TCP and UDP sockets and also with TCP and UDP SoftiWarp.

In Fig. 7.2 we show the power consumed by receiving 50 emulated LOFAR data streams using TCP Sockets on the left image and UDP sockets on the right one. The energy consumption for receiving TCP traffic is measurably higher than when using UDP due to the additional overhead of the TCP/IP protocol stack. This is a clear indication that reducing this protocol overhead will result in a smaller energy consumption. The average power consumption for TCP in this experiment is 45.09 W and for UDP it is 40.05 W. In Fig. 7.3 we present the power consumption measurements obtained with the LOFAR traffic generator using SoftiWarp TCP on the left image and SoftiWarp UDP on the right image. The power consumption during transfers with software iWarp implementation is clearly lower than in the case of TCP and UDP sockets, presented in the previous image. The average value for the TCP experiment was 32.38 W and for the UDP experiment it was 31.01 W. The power efficiency difference between the TCP and UDP transfer in this case isn't as clear as with the sockets scenario, however the SoftiWarp UDP transfers achieved a better bandwidth, which can be seen on Fig. 7.4. We can explain this with the fact that the used message size in these transfers is relatively

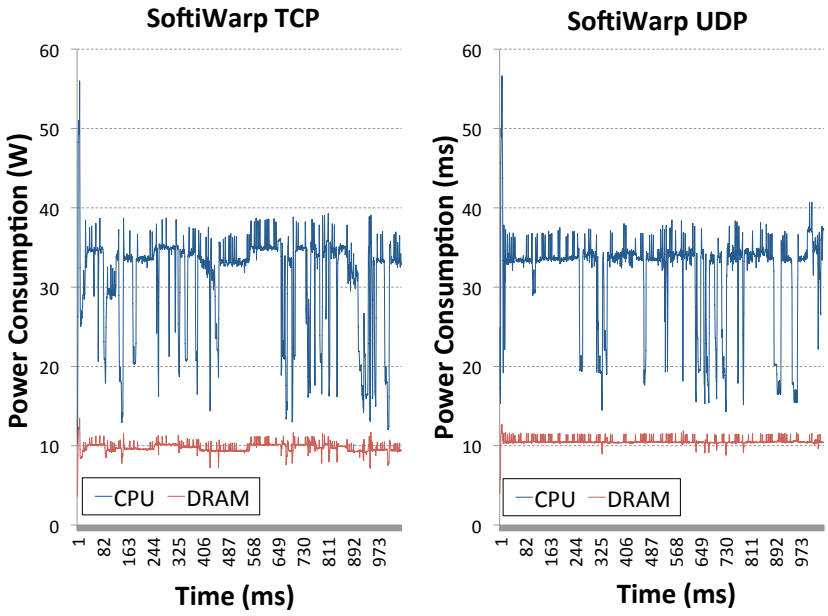


Figure 7.3: Power consumption of CPU and DRAM for receiving a transfer of LOFAR-like traffic over SoftiWarp TCP and SoftiWarp UDP.

low (8kB) and TCP-based protocol may have a problem achieving full link speed. The UDP-based protocol is more likely to achieve better speeds with smaller messages due to the lower overhead of the unreliable protocol. We will look further into the matter of achieved bandwidth and power efficiency (as Gb/s per Watt) in the following sections and present more results on this subject.

7.4.3 Power consumption of SoftiWARP TCP

In this section we carry out a set of transfers with the Netperf tool for the Sockets- and RDMA-based protocols with varying message size used. This will allow to observe the behaviour of different transport methods in different scenarios and allow to calculate the theoretical energy efficiency for all the transport methods. The tests have been performed with all of the offloading features of the NIC switched off, which was done for two reasons: firstly, we want to assess the direct effect of the transport protocol on the power consumption when no hardware support is available. Secondly, the offloading features available in modern NICs offer significantly more support for TCP protocol compared to UDP protocol, which means that with the offloading turned on the solutions

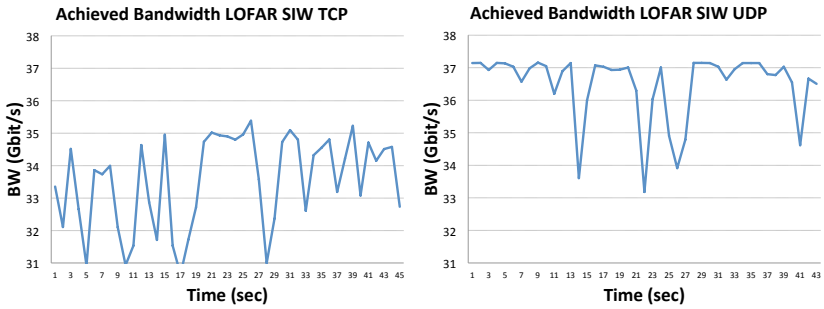


Figure 7.4: Achieved bandwidth of LOFAR-like traffic on the receiving side using SoftiWarp TCP (left image) and SoftiWarp UDP (right image).

based on the UDP protocol would be penalised. First we present the power consumption traces of different protocols and in Sec. 7.4.5 we present the complete set of numerical values and evaluate the normalised power consumption per achieved bandwidth. As

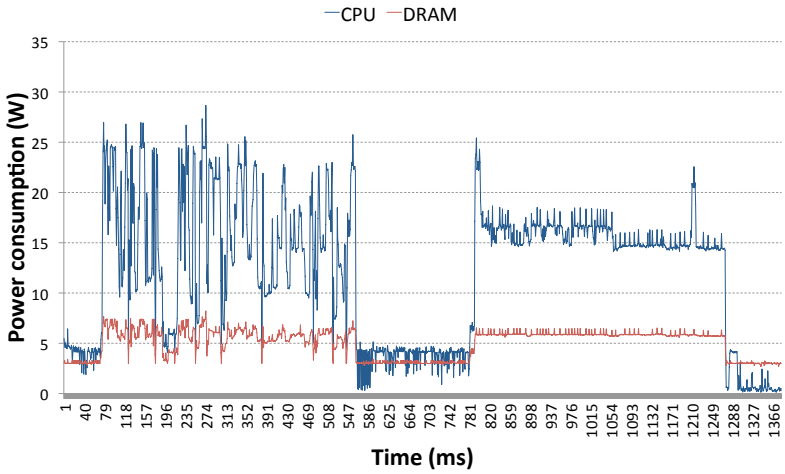


Figure 7.5: Power consumption of CPU and DRAM for data transfer over TCP sockets (first peak) and SoftiWARP TCP (second peak), receiving side.

mentioned before, we are interested in the power consumption on the receiving side of the connection, therefore we initially focus on these results. This is motivated by the

fact that the receiving sides of the data transfers in the SKA (CSP and SDP) will most likely be HPC systems, so experiments such as ours can give a good indication on the expected power consumption. Most of the sending side devices, on the other hand, will be custom-built devices. Therefore their power consumption patterns will be significantly different from a standard HPC system and the problem of their power efficiency needs to be addressed on their design level.

Fig. 7.5 shows the system power trace on the receiving side during data transfer with TCP sockets (first peak) and then SoftiWARP TCP (second peak). The blue line represents the power consumption of the CPU whereas the red line shows the DRAM power consumption. We performed six tests for both TCP sockets and SoftiWARP TCP and compared them to confirm that the power consumption follows very similar patterns in all cases. The data bandwidth achieved during the tests shown in Fig. 7.5 is 25.1 Gb/s for TCP sockets and 27.85 Gb/s for SoftiWarp TCP. As we can see, neither protocol is able to achieve the full link speed when the offloading features are switched off and we are communicating between just two instances of the testing application. However, already we can note that the bandwidth achieved when using SoftiWARP TCP is slightly larger and the power consumption is smaller. The average power consumption from six TCP socket tests is 17.4 W and for SoftiWARP the average is 15.89 W.

7.4.4 Power consumption of SoftiWARP UDP

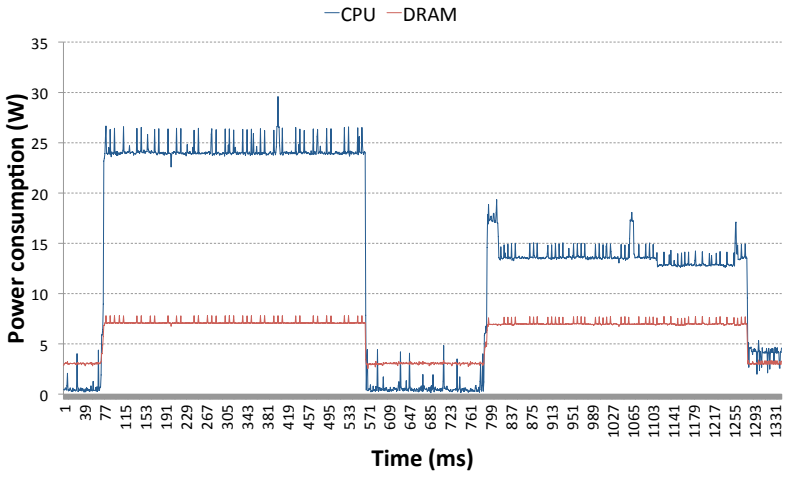


Figure 7.6: Power consumption of CPU and DRAM for data transfer over UDP sockets (first peak) and SoftiWARP UDP (second peak), receiving side.

Similarly to Sec. 7.4.3 we have performed the comparison between UDP sockets and

SoftiWARP UDP. Fig. 7.6 presents the system power trace during the execution of two Netperf tests: first peak shows the test using UDP sockets and the second one presents a SoftiWARP UDP test. It is clearly visible that in this case the power consumption difference between standard sockets and SoftiWARP is significant. The average energy consumption in the UDP socket-based tests is 24.21 W and 13.72 W for SoftiWARP-based tests. Furthermore, the near-full link speed of the connection is achieved in both cases: 39.37 Gb/s for the UDP stream test and 38.24 Gb/s for SoftiWARP.

7.4.5 Comparison of power efficiency

In order to quantify and directly compare the power efficiency of different transport protocols we performed a set of experiments in which we measured the power consumption used by the entire data transfer, including the CPU, DRAM and the NIC. Then we have calculated the normalised power efficiency, which we define as follows:

$$E = \frac{BW}{P} \tag{7.1}$$

$$[E] = \frac{\mathrm{Gb/s}}{\mathrm{W}} \tag{7.2}$$

From (7.1) it can be seen that our metric, the normalised power efficiency (E), is defined as the data bandwidth (BW) divided by the total power consumption (P), expressed in Gigabits per second per Watt (7.2). With this metric we are able to provide a good comparison on how much power is needed by specific transport protocols in a manner that is independent from the variations in bandwidth in different experiments. We note that we have opted not to use the equivalent Gb/J metric, since this does not emphasise the bandwidth aspect and could, for instance, also be used to denote the power efficiency of storage systems. We perform six experiments for each value, using message sizes in the range of 8 kB to 2 MB. The tested transport services include: TCP sockets, UDP sockets, SoftiWARP TCP and SoftiWARP UDP – both using RDMA Read and RDMA Write operations. The UDP sockets have only been tested for message sizes up to 64 kB as such size is the largest supported by this transport protocol. As before, during the first tests all of the offloading features of the NICs have been turned off. However, this time we have also performed tests with the following offloading features enabled: rx and tx checksumming offloading, generic receive offload (GRO) and generic segmentation offload (GSO). This was done to see the impact of such features on the results and compare them with the no-offload scenario.

Figures 7.7 and 7.8 show example results with hardware offloading features disabled and enabled, respectively. Both figures present results for the following message sizes: 256 kB for TCP-based protocols and 64 kB for the UDP-based protocols. At these values the given protocols have achieved their maximum bandwidth.

The tests performed without hardware offloading demonstrate that when using TCP even a relatively modern system is unable to achieve full link speed using a single core. Only the UDP-based protocols have been able to achieve the near-full link speed, however with UDP sockets this was coupled with significant power consumption on the sending and receiving sides. On the other hand, the SoftiWarp UDP tests using RDMA

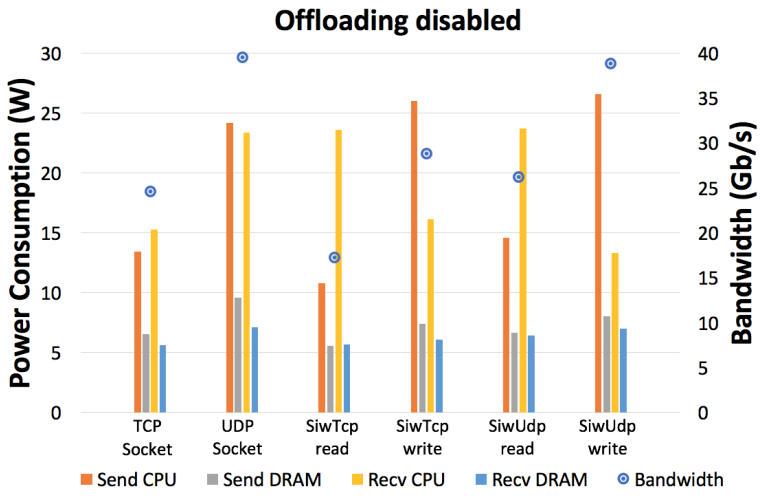


Figure 7.7: Results of power consumption tests with the offloading features of the NIC disabled.

Write have been able to achieve nearly identical results with the hardware offloading enabled and disabled, which was around 38.79 Gb/s bandwidth with only 13.64 W of average power consumption on the receiving end. The power consumption on the sending side remains among the highest in the above table, but this is not a crucial issue for radio astronomy applications as the sending side will most likely not be a standard computer but rather a custom-built FPGA unit, designed specifically to issue RDMA Write operations. Therefore, the power consumption of the sending side is a research topic on its own and cannot be evaluated using experiments similar to those presented in this chapter.

The results presented in Fig. 7.8 confirm our assumptions from Sec. 7.4.3, namely that the TCP-based protocol family receives significantly more support of hardware offloading. In the second set of tests almost all protocols achieved full link bandwidth, except for SoftiWarp TCP RDMA Write. The plain UDP Socket test didn't receive any support from the hardware offloading features, achieving the same bandwidth and power consumption. The SoftiWarp UDP RDMA Read test has achieved the full link speed due to the Receive Offload and Segmentation Offload features. Finally, it is important to notice that the SoftiWarp UDP Write test still offers the lowest power consumption on the receiving side of all the protocols, even when competing with the hardware-supported TCP sockets or SoftiWarp TCP.

Fig. 7.9 depicts how individual bandwidth values (left panel) and power consumption (right panel) correspond to varying message sizes. These charts allow for visualising the

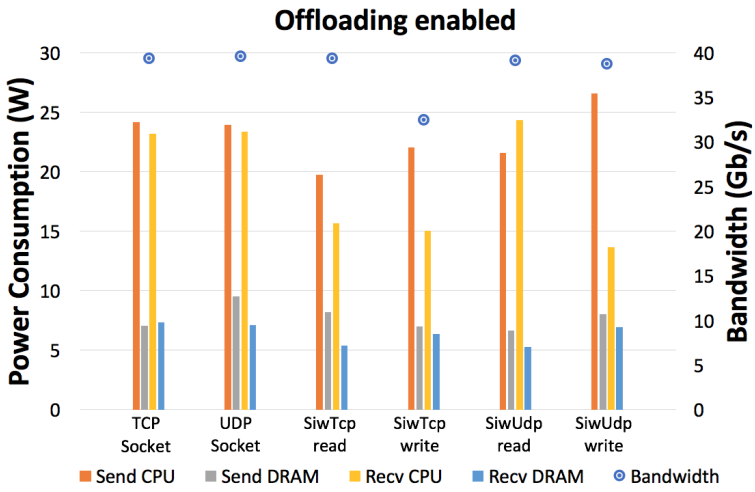


Figure 7.8: Results of power consumption tests with the offloading features of the NIC enabled.

trends and optimal values for specific protocols. The achieved bandwidth, but also the resulting power consumption, increases with increasing message size for all protocols. The optimal value is around 512 kB for the TCP protocols and around 64 kB for the UDP protocols. UDP sockets are the highest consumer of energy of all protocols, but also they are the only ones that achieve the best link speed without hardware support. SoftiWarp UDP is able to achieve very similar results with regard to bandwidth, but shows much lower power consumption, therefore its achieved power efficiency is higher.

The above results are used to calculate the normalised values of the power efficiency as expressed by (7.1) and (7.2). The calculated values are depicted in the efficiency chart shown in Fig. 7.10. Comparing the TCP and UDP groups, the former one is less efficient, which can be explained by the low bandwidth achieved by TCP protocols as shown in Fig. 7.9 left. Comparing SoftiWARP protocols to plain sockets, both over TCP and UDP, we can see that SoftiWarp is more power efficient in both cases. In all of the experiments SoftiWarp TCP performs better than TCP sockets and SoftiWarp UDP better than UDP sockets. This advantage results from the design of the SoftiWARP receive path implementation: after receiving iWARP packets into kernel memory, SoftiWARP directly copies their content into the target application buffers. Making use of the one-sided semantics of RDMA communication this final data placement does not involve the scheduling of the receiving side application process.

Although these results are still based on software prototype of SoftiWARP UDP, we can already confirm that the reduced data touching and the decreased overhead from the OS lead to very desired characteristics and promising results. The power consumption of

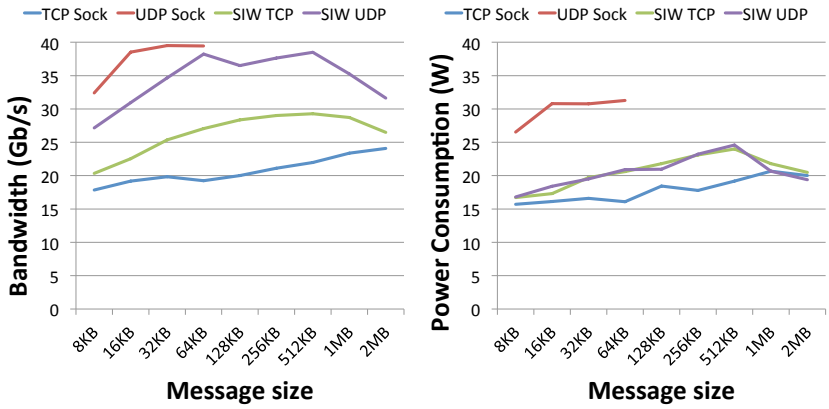


Figure 7.9: Bandwidth (left panel) and power consumption (right panel) results for tests with varying message sizes.

SoftiWARP is lower than TCP or UDP sockets in all cases and the achieved bandwidth is at least as good.

7.4.6 Behaviour in case of packet loss

Finally, we wanted to assess the behaviour of all the tested protocols in case of significant packet loss. We have done this by emulating packet loss using the Netem network emulation tool[8] in the range of 0.1% to 10%. The achieved bandwidths can be seen in Fig. 7.11. The difference between TCP-based and UDP-based protocols is significant. The former tend to sustain their original bandwidth in the initial part of the tests as all the lost packets are re-transmitted. However, with larger packet loss the network is no longer capable to keep up with re-transmission and the bandwidth gets significantly reduced. The UDP-based protocols do not rely on the retransmission-based reliable communication implemented by the TCP protocol and are able to maintain the transfers on the same level, regardless of the problems occurring along the link. The only decrease in bandwidth is the actual amount of packets that have been dropped. As we can see in the chart, this doesn't hold true for the results of SoftiWarp UDP RDMA Reads, which - as discussed earlier - are not yet fully supported in our implementation. The protocol does not yet recover from completely lost RDMA READ request/response pairs, which results in transfer breakdown as soon as the packet loss reached 3%.

The above results show that the use of a protocol that relies on two-way communication and tries to provide full reliability on the transport level, such as the TCP, can be infeasible for a scenario such as the SKA. It is true that the introduced packet loss in

[8] https://wiki.linuxfoundation.org/networking/netem

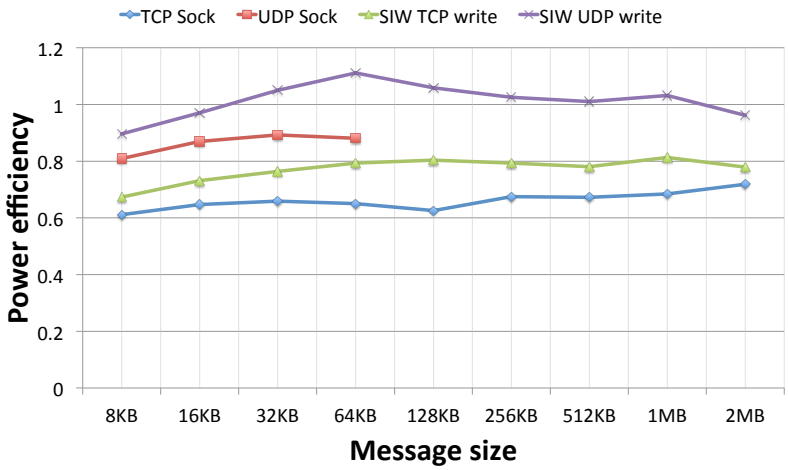


Figure 7.10: Power efficiency (in Gigabits per second per Watt) of 10 s Netperf tests, receiving side. X axis - message size, Y axis - Power efficiency.

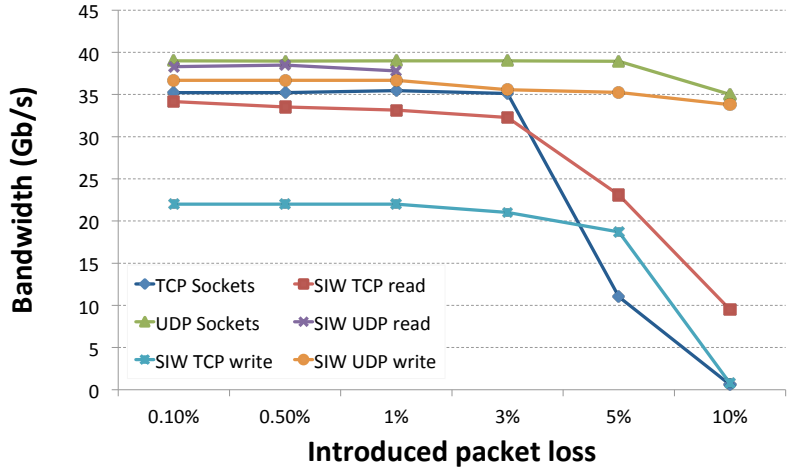


Figure 7.11: Achieved bandwidth with introduced packet loss.

our experiments was very high, but the tests were performed for a short, local connection. In the case of the SKA, where the connections spread over hundreds of kilometres in length, we would see a much more drastic influence of packet loss on the achieved bandwidth. This result confirms another reason for the choice of an unreliable transport protocol for our purposes. The power consumption in different packet loss scenarios didn't show any noteworthy behaviour. It corresponded to what we have seen in our previous experiments, namely that with growing packet loss the energy consumption was lower, because the achieved bandwidth was also lower.

7.5 Conclusions and Future Work

In this chapter we presented the data transport requirements of the world's largest radio telescope, the Square Kilometre Array (SKA). We proposed a solution to meet these requirements, namely an unreliable, datagram-based iWARP protocol implementation. We have then presented a software prototype of such a protocol, called SoftiWARP UDP, and evaluated its performance and power efficiency together with those of TCP and UDP sockets. We have confirmed that UDP is a very good choice for long distance transfer of astronomical data. The protocol overhead is lower, which leads to lower power consumption. Furthermore, the use of a reliable transport protocol is not feasible in a scenario such as the SKA, as it (1) leads to higher power consumption, and (2) the data transfer quality soon becomes unacceptable in case of non-negligible data packet loss.

Our software prototype of SoftiWARP UDP is already capable of outperforming TCP and UDP sockets by as much as 70% in terms of power efficiency (when comparing SoftiWarp UDP with standard TCP/IP sockets, see figure 7.10). This is a very desired result, however we expect a much higher improvement of the power efficiency with implementation of the SoftiWARP UDP protocol in hardware, e.g. using FPGAs and the source code of SoftiWARP UDP, which we leave for future work on this subject. A solution desired for the SKA purposes would have typical RDMA characteristics, where all four lower network layers are handled in hardware. Considering our results, we believe that with a hardware implementation the SoftiWARP UDP protocol would bring all the benefits of RDMA, namely an outstanding power efficiency, low latencies and CPU utilisation and high bandwidths, while meeting the specific requirements of the radio astronomy data transfer service.

Finally, our tests show that the DRAM power consumption was not reduced using SoftiWarp UDP, and now has a much more significant impact (approximately 23%, see figure 7.6) on the total consumption. Solutions for its reduction should be explored. We will look into using flash storage technology instead of DRAM for data ingress, which is energy efficient and offers high bandwidth and low-latency access.

7.6 Our propositions in this chapter

We now return to the propositions that from the basis of this thesis.

7.6.1 Value proposition

As mentioned in the previous chapter, we can argue that the optimisations identified and explored in this thesis are primarily intended to increase the relative science value of the system, as defined in chapter 2. While this chapter primarily contributes to the optimisations proposition, we cannot ignore the underlying reasoning for these optimisations. Therefore this chapter also contributes partially to the value proposition.

7.6.2 Optimisation proposition

In chapter 3 we identified that the large volumes of data moving in and out of the SKA Science Data Processor may require significant amounts of energy. In this chapter we not only verified this claim, we also provided a potential solution to the problem: a software RDMA based receive stack that requires significantly less energy. While the particular solution is one of many implementations available, the energy savings of all of these should be similar. It is noteworthy that the energy required to receive large volumes of data would generally be considered inevitable. Because we consider data-transport and compute together, this source of energy consumption becomes an obvious potential optimisation target.

Acknowledgements

This work is conducted in the context of the joint ASTRON and IBM DOME project and is funded by the Netherlands Organisation for Scientific Research (NWO), the Dutch Ministry of Economic Affairs (EL&I), and the Province of Drenthe.

CHAPTER 8

Future developments in compute and data-transport systems for radio telescopes

So far in this thesis we have focused on ways to architect and design efficient compute systems in support of radio astronomy and astrophysics instruments. One key element we have seen several times in these considerations is timing the procurement of compute infrastructure for maximum science impact. This relies on a continuous and predictable increase of compute capacity per invested Euro, generally referred to as Moore's law scaling [1].

8.1 Computational scaling and the demise of Moore's law

Computing as we know it today started in earnest in the second world war, but it was the introduction of the first microprocessors in the late 1970s that started the dramatic and sustained growth of compute capacity we have enjoyed for the last decades and that has propelled us into the information age. Initially the main source of computational scaling over time was derived from Dennard scaling [57]. This states, roughly, that as transistors get smaller, their power density stays constant. In others words, as transistors get smaller they require less energy to operate and thus required voltage and current are reduced. Dynamic power consumption of integrated (CMOS) circuits is to its clock

[1] Moore's law states that the number of components in an integrated circuit doubles every year [99], a number later adjusted to 24 months [100].

frequency. This meant that at the same or similar energy levels integrated circuits with smaller transistors could be run at higher clock frequencies, with corresponding higher performance.

Around 2006 transistors had shrunk to a level sufficiently small that leakage currents in the components started to dominate energy consumed by the parts. Intel proved to be unable to scale its Netburst architecture based Pentium 4 processors beyond 3.8 GHz, not even half the originally expected 10 GHz (after several fabrication process updates). This unexpected setback required a radical re-think of architectures and led to the (at that time rare) cancellation of the relatively new Netburst architecture and more focus on energy efficiency and performance per clock cycle.

This focus on performance per clock cycle has driven two main developments. First, multi-core systems have increased concurrency in systems, in theory allowing for an n-fold increase in performance for an n-core system without increasing clock frequency. However, only parts of a workflow will benefit from this, since Amdalh's law states that scaling in such a way is limited by the sequential part of the code. Second, ever more, and more complex, SIMD (Single Instruction Multiple Data) instructions on ever longer vectors were added to systems to increase the number of operations per clock cycle that can be done. Third, hardware features were put in place to increase the occupancy of the available compute resources, such as the ability to execute instruction out-of-order and the addition Simultaneous Multithreading (SMT). In particular the first two put a heavy burden on programmers and compiler designers to use these instructions properly, and, more importantly, make sure sufficient data is available in the correct ordering to effectively use these instructions.

More recently accelerators, like GPUs (Graphics Processing Units) have become commonplace, which can roughly be considered as specialised co-processor with many, not necessarily general purpose (or even Turing Complete) compute units. These offer excellent computational performance, and while programming these is not easy, they have been the mainstay of modern high-performance computing for several years now.

However, all of these developments still rely on the fact that, so far, we have been able to reliably produce a new production process with smaller transistors every two years. Shortly this will no longer be the case, and we already see significant slowdowns in new processes being rolled out. Intel has had trouble introducing their 10 nm technology, due to poor yields [5]. In January 2019, Intel announced its first that its first mass-production-ready 10 nm processors, originally expected to be available in 2016, would be released by the end of 2019 [6], some three years late. Even if 10 nm will eventually be an economically viable node, it is expected that only two more production nodes will be available before physical limitations (i.e. quantum tunnelling and features approaching the size of single atoms) prevent further scaling. It is expected that the prohibitive cost of (the facilities to produce) these nodes will delay their introduction significantly. Consequently, it is likely that the current slowdown of developments in compute capacity will continue. If no alternative technologies are developed, these developments will come to a complete halt with the introduction of a fabrication process in the order of 3-5 nm.

8.2 Post-Moore computing

The physical limitations of current technologies have been known for quite a while. Academia and industry have proposed a large number of alternative solutions to the problem. While a survey of these alternatives is not within the scope of this thesis, a recent report for the intelligence community [81] summarised them into four classes:

1. Classic Digital computing
2. Analog Computing
3. Neuro-inspired computing
4. Quantum computing

While the future of compute architectures as we known them today, and have known for decades, is sombre, we note that this represents an extremely large market with excessive amounts of available capital. Therefore, capital intensive solutions that extend the viability of current technologies for even a short period of time are not beyond the realm of possibilities. The cost of these solutions will however be such that performance per invested Euro will no longer increase at the same rate as before. This development is exacerbated by limited competition in the market of high-performance micro processors and accelerators, driving component cost up.

While the number of alternative solutions identified below is large, only a subset of these do not represent a radical change in architectures, making most, if not all, of the theory underpinning modern computer science obsolete. The cost of such radical new solutions is high. Not only does such a new paradigm likely require all-new software to be written, it is likely that the entire algorithmic and theoretical basis underpinning current state-of-the-art radio interferometry needs to be re-evaluated. This means that a clear process to determine the applicability and viability of such technologies, and its cost, both capital and other, is essential. The propositions introduced in this thesis, in particular the *value* proposition, and its theoretical background introduced in Chapter 2, are highly relevant to that process. This will also drive a desire to extend the useful lifespan of current and next-generation more conventional digital processing techniques, since these can more easily and cheaply be used.

We will give a short overview of the four different classes of compute technologies, and the ways they may be used in radio astronomy. Note that none of these have been proven, and viability statements are best-guess estimates.

8.2.1 Classic Digital Computing

Considering the massive investments that have been made in all areas of information technology, from application software to the algorithmic theory, hardware and manufacturing, it is very likely that classic digital computing will remain the dominant computing technology for some time. This will extend even beyond the practical scaling limit of CMOS, somewhere after 2020. At that stage, any improvement in capacity per Euro (or Joule for that matter) be derived from architectural improvements. One of these

that we see in production already, is 3D stacking, currently employed in high-bandwidth memory (HBM), which drastically reduces the distance for information to travel. Consequently, the bandwidth available is often exceptional, with the recently introduced AMD Radeon 7 graphics card offering an unprecedented 1 TB/s bandwidth to a stack of 16 GB of memory in a consumer grade device [2].

Another development that is likely to continue is the appearance of specialised hardware in general purpose compute resources. Recently Nvidia has added special purpose deep learning (tensor cores) and ray tracing (RT cores) hardware to their line of GPUs. While this has drastically increased the theoretical performance of these devices for some applications, the cost of programming such systems will likely increase, due to their heterogeneous nature.

Non-conventional or special purpose systems

Current compute systems are generally based on technology that has been developed over years. One of the key characteristics of these systems is that they are backward compatible, which requires significant chip real-estate to implement. More efficient systems, built from the ground up to support a small sub set of applications, may be feasible and even affordable. Such systems would be designed to excel at one, or a small set of similar, application(s), potentially with degraded performance for other applications.

8.2.2 Analog Computing

Fundamentally, an analog computer attempts to build an analogy for the system that is being studied. While an analog surrogate of a classic digital computer has been studied in some detail, the complexities of having a continuous state space over continuous or discrete time, whereas digital computers operate on a discrete state space and discrete time, means that progress has been slow. It has been shown that a Turing machine can be simulated by an analog computer, suggesting that analog computers are at least as powerful as digital computers [28].

It is important to note that analog computers are much older than digital ones. Mechanical analog computers have been used for centuries, the oldest of which, the abacus, dating back to 2500 BC. An interesting relatively recent discovery is that of an ancient Greek clockwork analog computer designed to calculate astronomical positions, the *Antikythera mechanism* [67]. Some of the most complicated mechanical analog computers were produced for sophisticated targeting systems in naval ships and heavy bombers during the second world war.

Electronic analog computers are more recent. An interesting example used the analogy between the motion of water and the flow of electricity to model the slope of rivers (by DC) and tides (by AC) [161]. An analog computer based on this analogy, Deltar, was constructed in the Netherlands in the wake of the catastrophic 1953 floods to model tidal flows in preparation of the Delta works plans [155]. Until gradually replaced by

[2] `https://www.amd.com/en/products/graphics/amd-radeon-vii`

numerical methods run on digital computers in 1983, Deltar was used to study the effect of the Delta works, built to protect the Dutch coast against further floods.

Digital computers replaced most of their analog counterparts when they became abundantly available in the 1970s. Recently, analog computers leveraging the massive advances in CMOS technology, are being reconsidered due to their potential energy efficiency. However, although some analog computers are Turing complete, this does not guarantee they are easily (re)programmable. In other words, analog systems may be single- or narrow-purpose, making them essentially unsuited for general purpose computing. Furthermore, while the precision of digital computers is only limited by word-size and arbitrary precision arithmetic provides any precision required, in analog computers this is a function of design and quality of components. Moreover, since numbers are represented by physical phenomenon, there are practical limits to the precision achievable with an analog computer. For instance, compare the size of the universe ($\approx 10^{27}m$) with the smallest measurable scale (Planck length; $\approx 10^{-35}m$), which is $\approx 10^{62}$. This implies that the maximum accuracy on distance measurement is 62 digits. Similar fundamental limits exist for other physical quantities, such as time, mass, charge, voltage, etc. We should note that hybrid analog / digital approaches to represent numbers do not necessarily have this limitation.

When modern electronic analog systems become more mainstream, accuracy of these must be considered first and foremost when assessing their use in a radio telescope. For static, high throughput processing that does not change over the lifetime of the instrument, analog computing may be a viable option. In particular parts of the receiver signal processing component, such as the filters and beamformers shown in Figure 1.5 on page 8 of this thesis, may be suitable candidates for analog implementation. In essence this would move the analogue/digital converter further upstream. Considering that these generally use relatively few bits to represent data (12 bits in LOFAR for instance), the accuracy of analog computers should be sufficient. It seems unlikely that analog computing would be suitable once data has been digitised, but further research is required to confirm this. The fact that analog computers are particularly well suited for solving differential equations can be key differentiating characteristic in this research.

8.2.3 Neuro-inspired computing

No amount of developments in classical digital computing is likely to approach what the human brain can do within a power envelope of only about 20 W. Neuro-inspired computing aims to mimic the structure of the brain, not simulate its characteristics. This is characterised by highly interconnected and approximate computing, and unsurprisingly its architecture looks similar to a classic neural network, with neurons applying weights, which encode the knowledge learned from training sets, to information flowing through the system.

Neuromorphic systems can be implemented using CMOS technology [78]. Conventional digital computers can be used to gain experience with this style of computing, but due to limited interconnect availability these are currently orders of magnitude less efficient than dedicated, possibly hybrid digital/analog [135], computers may be. General purpose cores with bespoke and dedicated routing engines have been built to mitigate this problem [68].

Neuro-inspired systems are likely to excel at stream processing of sensor data. They will be data driven, without a clock as in classic digital systems. Such systems are expected to be exceptionally energy efficient, but approximate in terms of accuracy. Apart from the obvious applications of such systems in machine learning and other artificial intelligence tasks, it may be possible to adapt even small scale neuro-inspired systems to the tasks we currently entrust to custom designed Field Programmable Gate Array (FPGA) boards at the receiver level in modern radio telescopes. These are generally digital filters, fast Fourier transforms and beamformers that are applied immediately after digitisation on very large data streams, but involve comparatively little compute resources per bit of data. Besides a possibly significant reduction on required energy for such tasks, the data driven nature of such neuro-inspired compute systems may be a significant advantage that warrants additional investigation. It potentially negates the need for synchronisation of the FPGA clock and the data stream coming into the board, considering the data is in itself triggering the processing.

Whatever the advantages may be, it is unlikely that neuro-inspired computers will displace classic digital computers any time soon. They may however be an interesting addition to a hybrid system designed in a holistic manner to leverage the specific advantages of each specific technology. The scientific impact of the approximate nature of neuro-inspired computing needs to be determined, especially when applied near the initial receivers.

8.2.4 Quantum computing

No chapter on future computing can be complete without at least acknowledging the developments in quantum computing. Conceived fairly recently by the legendary physicist Feynman [64], it has been the promised technology ever on the edge of practical applications for years. While a conventional digital computer operates on bits, with a definite value of either 0 or 1, a quantum computer operates on quantum bits, or qubits, that each are in a superposition of states. A key distinguishing feature between qubits and digital bits is that multiple qubits can exhibit quantum entanglement, which broadly means that the superposition of two qubits maintains higher correlation than is possible in classical systems. In other words, if two qubits are entangled, measuring the value of one qubit will result in the same value as measuring the other, even if they are separated after entanglement.

One of the earliest papers considering the utility of quantum computers showed that a universal quantum computer is indeed Turing complete and can perfectly simulate any Turing machine [58]. In other words, any problem that can be solved using a Turing machine, and therefore using any general purpose computer, can be solved on a universal quantum computer. This was rigorously confirmed twelve years later [23].

Since universal quantum computers are Turing complete, we should be able to map any *computable* problem to a quantum computer. However, not all problems will benefit from this mapping. In other words, quantum supremacy is not guaranteed, and indeed for many problems there is little to no benefit in the move to a quantum computer. There are a fairly small number of algorithms that are expected to show quantum supremacy, Shor's algorithm for factorising large integers being the most well known [142].

Quantum algorithms

Since the idea of a quantum computer was conceived, only a hand full of classes of quantum algorithms have been discovered. These can be summarised as:

1. using Fourier Transforms to find periodicity, Shor's algorithm [142] being the prime example.
2. Grover's search algorithm [73] and its generalisations.
3. algorithms for simulating or solving problems in quantum physics, exemplified by Feynman's initial ideas [64].
4. quantum walks, quantum analogues to classic random walks, first proposed by Childs [45].

While for a small set of problems quantum computers seem to offer tremendous advantages over conventional computers, as typified by these classes of algorithms, this set has not grown significantly in years. Some research has been done into the reasons for this [143], which concludes that either these systems are so different from normal computers that our techniques for designing algorithms are unsuitable, or there are only a handful of problems that actually benefit significantly from quantum computers. Computer scientists and physicists have been thinking about ways to use quantum computers for a while, although focus has been on problems that are difficult or impossible to solve with a conventional computer. It is certainly too soon to tell definitively, but the current state of the art limits applicability of quantum computers to the subset of problems that map well onto the four classes of algorithms mentioned above.

While not enough research has been done to identify the applicability of any of these classes of algorithms for radio astronomy or astrophysics, we can make some initial assessments. In Chapters 1 and 2 we argue that modern aperture synthesis radio astronomy was born by virtue of the Fast Fourier Transform and the development of (mini-)computer fast enough to run these at scale. At the heart of Shor's algorithm for factorisation is quantum Fourier transform [142]. However, implementing an accurate quantum Fourier transform is difficult, and Shor proves in his seminal paper that an approximation is sufficiently accurate for his factorisation algorithm. For quantum computers to be useful for aperture synthesis radio astronomy, we must first determine if a quantum Fourier transform is a suitable alternative to the FFT, and what the required accuracy would be. It is important to note that while the quantum Fourier transform at the heart of Shor's algorithm *is* exponentially faster than even a Fast Fourier Transform, it is unclear if it can be integrated into the normal radio astronomy work flow as is, and if the potential increase in performance is worth the additional cost and complexity that will inevitably accompany such a complex, and likely heterogeneous, system.

A possible further application is in the area of pulsar search. The current state of the art uses an exhaustive search of a 5-dimensional space (x, y, pulsar dispersion rate, pulsar timing and pulsar acceleration), which is computationally an extremely expensive task. Alternatively this could, in theory, be seen as a search in an unstructured set of data, for which Grover's algorithm [73] may be suited. A quadratic speedup over conventional search may be achievable, but further research is required to properly gauge the suitability of this class of algorithm for pulsar search.

We do note however that both of these applications are relatively data-intensive and that data transport will be a critical weakness of early quantum computers. Current systems measure data transport capabilities in kilobytes per second, rather than the gigabytes per second that are the norm in modern radio telescopes. Extreme cooling requirements of the current generation of early quantum chips limit the ability to transport large volumes of data quickly. We do not expect this to change significantly in the initial batch of operational quantum computers, and this may severely limit the applicability of these for radio astronomy and astrophysics.

8.3 Data transport systems

Whereas developments in compute technology are expected to be challenging and potentially revolutionary, the same is not true for the underlying technology for data-transport systems. While photonics will continue to be applied closer to, and be extended into, the node, there is no reason to assume that physical limitation will the growth of available bandwidth at more or less the same rate we're used to. That said, other developments, such as the appearance of programmable networks and open source network operating systems, may offer interesting new ways to apply the propositions introduced in this thesis. This development is arguably a reaction to the closed nature of networking vendors and the vendor lock-in that entails. As a result, many major cloud providers have started to develop their own networking equipment, with some, Microsoft and Facebook in particular, open sourcing their hardware [3] and software [4]. This combination of highly commoditised hardware and extendable open source software makes for an attractive option to build application specific code on, essentially further integrating the network into the compute system, as recommended by the *co-design* proposition in this thesis. Furthermore, the open source software environment should allow easier and more extensive optimisation options, facilitating the *optimisation* proposition, by limiting reliance on vendor software quality, although the Switch Abstraction Interface (SAI), the layer that interfaces hardware with a common software stack, may still vary in quality.

Analogous to quantum computing, quantum networking facilitates the transmission of quantum information using qubits between physically separated quantum processors. While progress in this area has been slow, even though physical links are based on standard telecom fibres, the efficient and fault-tolerant transmission of quantum information is essential to the successful application of quantum computers in any application. The high data rates involved in radio telescopes make this particularly important.

8.4 Data storage technologies

As with data-transport systems, the expectation is that developments in data-storage technologies will continue more or less at the same rate as in the past. Although the density of current magnetic storage media such as hard drives is such that continued

[3] https://www.opencompute.org/wiki/Networking/SpecsAndDesigns
[4] https://azure.github.io/SONiC/

growth of density without assisting technologies is difficult, a number of solutions have already appeared in the market. While there is no physical limit in sight that will fundamentally limit the growth in storage density, the storage throughput to and from the medium has lagged behind. This trend will likely continue, to mitigate this it is expected that a new tier of high-performance intermediate storage systems will appear. Integrating such systems in a co-designed compute- and data-transport infrastructure may require an interesting extension of the concepts introduced in this thesis.

Classic spinning media will continue to evolve for some time, with increased densities driving higher capacities in the same form factor. The gap between these spinning media and solid state storage will continue to grow in terms of bandwidth to the media, and decrease in terms of cost per unit of capacity. Eventually solid state storage, in a number of different implementations or tiers, will completely replace magnetic storage. There is no practical limit in sight yet for the development of solid state storage, both in capacity per Euro, and in bandwidth to the storage.

One worry is the cost of, in particular high-performance, storage systems. Whereas storage media itself are relatively inexpensive, it is interesting to note that enterprise storage solutions generally increase the cost by several factors. Arguably, data produced in radio astronomy is not very precious, and the measures taken to ensure data integrity in such expensive systems are unnecessary. Furthermore, complex enterprise and high-performance computing storage solutions are complex, difficult to maintain and tune, and frankly fragile.

Abundant, affordable and relatively reliable storage solutions are not very interesting to major vendors, since they offer slim margins over the media cost. There are, however, open source hardware and software solutions available that may offer superior price per capacity over commercially available solutions, at the cost of modest investment in man power.

8.5 Tackling a challenging future

Since the advent of modern micro processors, we have been spoiled by a continuous and predictable increase in available compute capacity over time. Physical limitations have slowed down this development and within this generation will grind it to a halt. This will lead to a diverging set of evolutionary and revolutionary developments that have the potential to significantly disrupt the landscape of systems available for procurement.

While we can at this point speculate about the suitability of proposed new technologies, there is currently no solid basis to base any definite statements either way on. However, the propositions we have proposed in this thesis, in particular the *value* and *co-design* propositions, apply equally to both to current and future technologies.

Indeed, it seems likely that future systems will consist of a heterogeneous collection of varying compute resources. Finding the most efficient combination of technologies is essential, and while the design space is different, the process will be very similar to the one introduced in Chapter 2 and applied in Chapters 4 and 5. However, beforehand we need to investigate the viability of the various conventional and non-conventional alternatives. Unless we have a clear understanding if and how radio astronomy and

astrophysics can benefit from potential new technologies, there is a potential risk that radio astronomy can not effectively take advantage of these when they become available.

It is our recommendation that a fundamental research project be started with the following goals:

1. Closely track developments in both conventional and less-than-conventional compute systems, collaborating widely with both industry and academia.
2. Investigate the viability and, where possible, the performance of the various other-than-conventional compute systems for key radio astronomy and astrophysics applications.
3. Fundamental algorithmic research should be undertaken with the goal of building a new theoretical basis for radio astronomy, to best utilise emerging other-than-conventional compute resources, such as neuromorphic and quantum computers, and compare this with current state-of-the-art implementations currently in use.

8.6 Our propositions in this chapter

8.6.1 Co-design proposition

While, contrary to chapters 3 and 4, we can not show practical applications of the co-design proposition in action in this chapter, it does play a major role in our evaluation of future technologies. In particular our initial estimate on the viability of quantum computing in radio telescopes is based on this proposition. Whereas computationally it may be advantageous to investigate whether quantum superiority is viable for some of the applications in radio astronomy, much depends on the ability to transport data into and out of such a system.

We also identify an interesting new trend towards openness in the development in programmable networks. Major internet-based companies that rely on abundant and cheap compute and data-transport capacity are moving towards an open ecosystem, where hardware and software are open sourced, procured from multiple sources and adapted to suit specific purposes. This may allow much further and more flexible integration of network and compute systems, extending the impact of this proposition even further.

8.6.2 Value proposition

When evaluating new technologies, as we did in this chapter, it is essential to consider the value potential of such new technologies relative to existing more conventional systems. This must the a key factor when deciding whether such future non-conventional technologies are viable and applicable as part of the signal processing resources in large-scale distributed radio telescopes. While not enough research has been done to evaluate to what degree the various discussed technologies are suitable, the value proposition does show how such evaluation should be done.

8.6.3 Optimisation proposition

This chapter identifies an interesting new trend. Whereas networking equipment previously relied on vendor specific firmware, open source alternatives have appeared more recently. These would allow development of specialised applications to be run in the network, creating opportunities for optimisations that were previously difficult or impossible to implement. In chapter 6 we identified that vendor firmwares limited our concepts. By reducing our reliance on vendor provided firmware to support our optimisations, this development potentially strengthens the value and impact of the *optimisation* proposition.

CHAPTER 9

Conclusions

In this thesis we endeavoured to answer a research question that is both very generic and highly specific. Architecture and design of a compute or eScience system is a broad subject that opens opportunities for numerous other questions. At the same time, our question focuses on a fairly small set of applications in a very defined area of research: radio astronomy.

9.1 Summary of contributions in this thesis

In Chapter 1 of this thesis we posed the main question that drives the research presented in this thesis:

- *What is the optimal way to design a commodity compute and data transport architecture for modern distributed radio telescopes, and how do we define optimal in this context?*

To address this question, and based on extensive architectural design experience, we proposed four propositions that articulate some of our high-level recommendations. Combined, these four propositions provide the basic starting point for the design of any commodity compute system in a distributed radio telescope.

In Chapter 1 we introduced four propositions, summarised as the bounding, co-design, value and optimisation propositions. These are based on extensive architecture and design experience, both in the LOFAR radio telescope and in the design of the Square Kilometre Array (SKA) Science Data Processor (SDP). The propositions drive the direction of the research in this thesis and can also be taken as proposed design priorities and recommendations to take into account when architecting or designing

the general purpose compute component in any modern distributed radio telescope, or indeed large-scale scientific instrument.

To summarise the contributions in this thesis we shall consider each proposition in turn, and discuss how these were addressed. We also briefly review some of the more detailed contributions made in the various chapters that are not directly covered by the propositions.

9.1.1 Proposition 1: the *bounding* proposition

Before embarking on an architecture or design, bound the problem in terms of requirements, such as capacity and functionality, and available resources such as funds, facilities, manpower and interfaces.

At first glance, this proposition may seem obvious. This proposition is also in line with systems' engineering best practice. However, scientific instruments are not always designed and built using proper systems' engineering methods, and this elementary first stage is often neglected. It is therefore quite relevant to explicitly articulate this proposition.

In Chapters 3, 4 and 5 we showed how this proposition was used in practice, in the design of the SKA Science Data Processor and the GPU-based LOFAR correlator and beamformer respectively. The cases studied in Chapter 2 also show the *bounding* proposition in use, even though the chapter itself does not significantly contribute to this proposition. Especially in the SKA, where construction of designed system is years away, bounding the problem has proven to be very useful. The progression shown between Chapters 3 and 4 was achieved in part due to considerable developments of the bounds used, in particular required resources, computational and other, thanks to extensive modelling of the high priority science cases. While this indicates that there is significant value in a continuous revisiting of the defined bounds as the architecture or design matures, this does not invalidate the proposition itself.

9.1.2 Proposition 2: the *co-design* proposition

The compute- and data-transport systems supporting modern radio telescopes must not be developed in isolation.

Although we refer to this as the *co-design* proposition, our intention is slightly different than the more general use of the term implies. Whereas co-design often refers to hardware and software being developed side-by-side, here we mean the co-design of compute- and data-transport systems. However, extensive and hard-learned experience with the LOFAR telescopes has shown that this is essential, considering the data-intensive nature of commodity processing in radio telescopes. This does mean a slight loss in modularity, with the data-transport and compute components essentially being a monolithic component.

Chapter 5 embodies this proposition well. This chapter describes the design of a data-intensive component of the LOFAR radio telescope. Considering the streaming nature of the processing, efficient data-flow through the system, within and between nodes, was a primary design consideration, while compute resources were an important, but secondary, consideration. The ultimate success of this project was at least partly due

to the careful and simultaneous design of the compute- and data-transport systems, both internal and external, in Cobalt.

The conceptual design of the SKA Science Data Processor (SDP), discussed in chapter 4, is less detailed, but similar to Cobalt, the data-transport system is designed in concert with the compute resources. While the conceptual design of the SKA SDP does not extend to the node level, the lessons learned in Cobalt apply, and a very similar design process can be expected to occur during the construction phase of the SKA SDP. This focus on both data-transport and compute ensures scalability and fitness for purpose of procured systems. It is natural to assume that any new and less conventional solution to these problems will require a similar focus. Considering the specialised nature of many of the non-conventional solutions discussed in chapter 8 it is likely that compute systems will shift to interconnected collections of heterogeneous resources, making the data-transport infrastructure even more important.

9.1.3 Proposition 3: the *value* proposition

A system's architecture and design should not only optimise for cost, but instead closely consider the optimal ratio between value and cost.

This proposition is one of the core contributions of this thesis. It tries to articulate a significant body of experience, exemplified by the partial CV shown on page 183 of this thesis. While the message in itself is clear, it is generally exceptionally difficult to quantify cost and, particularly, value of a data-transport and compute system, as we show in Chapter 2.

Nevertheless, in particular in Chapters 4 and 5, we show that compute and data-transport systems designed for radio astronomy benefit greatly from a balanced look at both cost and value. Furthermore, the optimisations discussed in Chapters 6 and 7 must either decrease cost, improve value in some way, or a combination of both in order to be considered successful.

9.1.4 Proposition 4: the *optimisation* proposition

When both the compute- and data-transport systems are considered jointly, optimisations can be conceived on the boundary between these two that greatly benefit the whole.

This proposition is a consequence of proposition 2. As it turns out, when data-transport and compute systems are designed together, and the data-transport infrastructure is not merely seen as a black-box interface between different components, very interesting avenues for optimisation become available. We have explored two of these on the boundary between commodity and specialised hardware, in detail in Chapters 6 and 7 of this thesis. A third possible optimisation opportunity, looking at open source network operating systems and the programmability of these, was identified as future work. At the time of writing, initial work on this has been started in collaboration with the University of Amsterdam.

9.1.5 Other contributions

The propositions summarised above represent the primary contributions in this thesis. However, during the course of this work, many other, more detailed, valuable contributions were made. In this section we will highlight the most important ones.

While we certainly did not invent the concept of value, we did introduce the concepts of Total Value of Ownership, total lifetime computational value and total lifetime scientific value in sections 2.3 and 2.4. By articulating the explicit desire to maximise the scientific value of a system over its lifetime, rather than just minimise its total cost of ownership regardless of the performance impact, we contribute a way to greatly improve how we evaluate compute- and data-transport systems for radio telescopes.

The progression of the bounds, architecture and design of the Square Kilometre Array Science Data Processor is also a key contribution of this thesis. Two chapters were dedicated to this system, and apart from their support of the propositions, these show the evolution of a highly scalable, data-driven, hardware architecture for the Science Data Processor. The hardware design of this system, including the documentation delivered as part of the preliminary and critical design reviews, should also be considered as significant contributions, even though these are not published in journals or proceedings. We note that scientific publications on this subject are planned.

We also contributed a detailed description of the design and engineering process and construction of the GPU-based LOFAR correlator and beamformer. This chapter is unique in that it describes the design, implementation and operational experiences with a system designed specifically for production use in a radio telescope. We were therefore able to retrospectively show that the system meets all requirements, not just in synthetic benchmarks, but on actual production data as well. The design and implementation of the Cobalt system itself was, of course, also a significant contribution that was essential to the continuity of the LOFAR telescope.

We contributed two prototypes that demonstrate two different optimisations in the way that data is received into a general purpose compute system. One moves the control of data flows from the data plane into the control plane, while the second significantly reduces the energy overhead associated with receiving large volumes of UDP/IP data by avoiding parts of the Linux kernel network stack.

Finally we discussed, in some detail, how future technologies may be applied in a modern radio telescope. Even though this work is not yet published in a peer-reviewed journal or conference, we aim to write an extended and more detailed version in the near future.

9.1.6 Research question

Having looked at the contributions in this thesis we now return to our original research question reproduced at the start of this chapter. We have proposed a number of recommendations, articulated as propositions, that guide a design process to an optimal solution. These recommendations can be further summarised as:

1. bound the problem
2. consider data-flow and compute together

3. consider both cost and value
4. identify and investigate possible optimisations and research opportunities that increase value or decrease cost (or a combination of the two)

This requires a combination of domain specific knowledge and general computer science experience. In order to consider data-flow and compute, a detailed picture of the performance profile of the target application must be available. Similarly, significant knowledge about the system and target applications is needed in order to accurately estimate the value potential of a system under design.

Essentially we argue that, to ensure optimal utilisation of the procured resources, these need to be carefully balanced against their cost and their contribution to the scientific value of the total system. In order to achieve this, the system designer needs to be aware of the application intended to run on the designed system, in particular the way data flows through the system. Similarly and equally important, the programmer needs to be aware of the strengths and weaknesses of the target system.

9.2 Hardware components not discussed in this thesis

In this thesis we focused on the compute- and data-transport systems in radio astronomy, and their co-design opportunities. While we have not discussed data-storage technologies in any meaningful way, these may also be handled similarly. An excellent example of this can be found in the preliminary architecture for Square Kilometre Array Science Data Processor, as described in Chapter 4. Here, three different tiers of storage infrastructure, the high-performance buffer and medium- and slow-tiers of science archive storage, are designed to integrate neatly in the compute- and data-transport systems. Indeed, the compute-, data-transport, and data-storage subsystems in this architecture are designed as a single coherent system, although the data-storage system was not judged to require a very detailed design, since this is likely to be procured as a stand-alone appliance.

It is likely that the *co-design* proposition will need to be expanded in the near- to mid future, with the appearance of near-line non-volatile storage tiers. These generally fit between volatile main memory in nodes, and slower, non-volatile storage media either in the node, or in a dedicated storage system accessible via a high-performance network, and are intended to mitigate the high latency and relatively low performance of slower tiers of storage. Since these near-line storage technologies are integrated into the node for performance reasons, the classic boundary between compute- and storage sub-systems will inevitably blur. Such storage technologies are appearing today, in the form of Intel's 3D-Xpoint for instance, and will require a more detailed and integrated architecture of the data storage systems similar to the one advocated in this thesis for the compute- and data-transport systems.

9.3 Future work

The software-defined networking concept introduced in Chapter 6 is the basis for further work in that area. In this chapter we investigated the functional availability of features required for our application, but this used Gigabit Ethernet networks and did not extend to more high-performance networks. This work will be extended to more high-performance networking equipment, 40 and 100 GbE, to more accurately match the networks probably used in future radio telescopes such as the Square Kilometre Array. Initial experiments offered no different insights when using such higher performance equipment, with significant limitations in firmware remaining an issue. Furthermore, it is likely that the concepts introduced in Chapter 6 can be realised with different technologies, such as IP Multicast, BGP or P4. An extended paper is planned to investigate these.

In Chapter 6 we argue that the quality of the software running on the network equipment currently limits the applicability of a Software-defined network in radio astronomy. The Open Compute project [1] aims to create a fully open source stack data centre components, both in hardware and software. As part of this, a stack of open source network operating system components has been developed, including a bootloader (Open Network Install Environment, ONIE), a full Linux based operating system (Open Networking Linux, ONL), and a standardised interface to vendor specific hardware (Switch Abstraction Interface, SAI). To mitigate the impact that the quality of switch software has on the applicability of software-defined networking to our application, we intend to implement our own software to implement this functionality on top of this open source software stack.

Chapter 8 has not yet been published at a peer-reviewed conference or journal. It is our intention to extend this chapter into an overview paper that offers an initial assessment of the applicability of various newly developed compute concepts for data-intensive science. Furthermore, the concepts introduced in this thesis, in particular those in Chapter 2, combined with the new technologies and architectures that are being developed to address the demise of Moore's law scaling, may be the subject of a new direction of research. The combination of evolutionary and revolutionary developments in compute and data-transport technology we expect to become available in the next decade or so offers both interesting challenges an opportunities for research.

9.4 Discussion & conclusions

The work in this thesis is based on extensive experience designing, building and maintaining IT infrastructure in support of radio telescopes and astrophysics. The propositions introduced in this thesis are supported by a number of peer-reviewed publications. Furthermore, many architecture, design and procurement documents for current and future telescopes take advantage of the same lessons learned. A number of these are referenced in the Curriculum Vitae chapter that concludes this thesis.

[1] https://www.opencompute.org/

In this thesis we propose that the architecture and design of compute- and data-transport systems should be co-developed. The same is true for the management and administration of these systems. Whereas conventional high-performance computing centres and generic data centres generally split the responsibility for network and systems in separate departments, based on the *co-design* proposition there is a convincing argument to be made that this is not an efficient solution in systems supporting a data-intensive instrument. Since both the network and compute infrastructure rely heavily on eachother and indeed are highly interdependent, cross-trained staff in a single unified support department will be a more effective in supporting the instrument. This proposition is based on anecdotal evidence, proof being difficult to obtain without significant institutional changes, which are difficult without convincing evidence, the beginnings of which were presented in this thesis.

In this thesis we have articulated and documented extensive design and architecture experience gained in the development of commodity compute and data-transport systems for both the LOFAR radio telescope, currently operational, and the Square Kilometre Array, the construction of which is still years away. The recommendations described herein have proven to be essential when designing such systems to be effectively and efficiently operated as part of a radio telescope. They also turn out to provide a solid starting point when evaluating newly developed technologies for the use in such instruments. Such revolutionary systems, or more likely the combination of commodity and future technologies, will become ever more important to sustain the increase in computational resources that we need to build more powerful telescopes. The demise of Moore's law has made it clear that more diverse and heterogeneous systems will become commonplace, dramatically increasing the design space for the compute and data-transport system architect for radio telescopes. However, the experience documented in this thesis provides an excellent basis upon which we can base the evaluation of such complex systems. In all, the future for the design of commodity compute and data-transport systems is challenging, but this offers not only avenues for interesting research, opportunities for optimisation with dramatic performance potential are also abound.

Bibliography

[1] The Ryu SDN framework. URL https://osrg.github.io/ryu/.

[2] SKA SDP parametric model. URL https://github.com/ska-telescope/sdp-par-model.

[3] CUDA 6.5 Performance Report, September 2014. URL https://developer.nvidia.com/cuFFT.

[4] The world's largest radio telescope takes a major step towards construction, March 2015. URL https://www.skatelescope.org/news/worlds-largest-radio-telescope-near-construction.

[5] Intel Reports First-Quarter Financial Results. Press release, April 2018. URL https://s21.q4cdn.com/600692695/files/doc_financials/2018/Q1/Q1-2018_EarningsRelease-FINAL.pdf.

[6] 2019 CES: Intel Showcases New Technology for Next Era of Computing. Press release, January 2019. URL https://newsroom.intel.com/news/2019-ces-intel-showcases-new-technology-next-era-computing/.

[7] Thomas Abbott. Lessons learned from the integration of MeerKAT array release 1. Plenary talk SKA Engineering meeting, October 2016. URL https://indico.skatelescope.org/event/402/material/1/6.pdf. Stellenbosch, South Africa.

[8] Haroon Ahmed. *Cambridge Computing: The First 75 Years*. Third Millennium Publishing Limited, 2013. URL http://www.cl.cam.ac.uk/downloads/books/CambridgeComputing_Ahmed.pdf.

[9] P. Alexander, V. Allan, U. Badenhorst, C. Broekema, T. Cornwell, S. Gounden, F. Graser, K. Kirkham, B. Mort, R. Nijboer, B. Nikolic, R. Simmonds, J. Taylor, A. Wicenec, and P.Wortmann. SDP System Module Decomposition and Dependency View. Technical report, SDP Consortium, 2019. URL http://ska-sdp.org/sites/default/files/attachments/ska-tel-sdp-0000013_07_sdparchitecture_module_views.pdf.

[10] P. Alexander, B. Nikolic, V. Allan, C. Broekema, M. Deegan, and P.Wortmann. SKA1 SDP High Level Overview. Technical report, SDP Consortium, 2019. URL http://ska-sdp.org/sites/default/files/attachments/ska-tel-sdp-0000180_02_sdpoverview_part_1_-_signed.pdf.

[11] P. Alexander et al. Analysis of requirements derived from the DRM, August 2011. SKA Software and Computing CoDR.

[12] Paul Alexander, Chris Broekema, Simon Ratcliffe, Rosie Bolton, and Bojan Nikolic. SDP Element concept. Technical report, SDP Consortium, 2013. URL https://www.skatelescope.org/wp-content/uploads/2013/09/SDP-PROP-DR-001-1_ElemConc.pdf.

[13] V. Allan, B. Nikolic, M. Farreras, T. Cornwell, and R. Lyon. SKA1 SDP Execution Frameworks Prototyping Report. Technical report, SDP Consortium, 2019. URL http://ska-sdp.org/sites/default/files/attachments/ska-tel-sdp-0000117_02_sdp_executionframeworksreport_part_1_-_signed.pdf.

[14] Antony Antony, Johan Blom, Cees de Laat, and Jason Lee. Exploring practical limitations of TCP over transatlantic networks. *Future Generation Computer Systems*, 21(4):489 – 499, 2005. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2004.10.004. URL http://www.sciencedirect.com/science/article/pii/S0167739X04001645. High-Speed Networks and Services for Data-Intensive Grids: the DataTAG Project.

[15] Audrey Apfel and Michael Smith. TVO methodology: Valuing IT investments via the Gartner business performance framework. *Strategic Analysis Report*, 2003.

[16] Wesley Armour, Aris Karastergiou, Michael Giles, Chris Williams, Alessio Magro, Kimon Zagkouris, Sarah Roberts, Stefano Salvini, Fred Dulwich, and Ben Mort. A GPU-based survey for millisecond radio transients using ARTEMIS. *ASP Conference Series*, 461, Astronomical Data Analysis Software and Systems XXI:33 – 36, 2012. URL http://www.aspbooks.org/a/volumes/article_details/?paper_id=34616.

[17] M. Ashdown, R. Bolton, F. Graser, F. Malan, and R. Nijboer. SKA1 SDP System Sizing, November 2018. URL http://ska-sdp.org/sites/default/files/attachments/ska-tel-sdp-0000038_04_sdpsystemsizing_part_1_-_signed.pdf. SKA SDP CDR deliverable.

[18] Henri Bal, Dick Epema, Cees de Laat, Rob van Nieuwpoort, John Romein, Frank Seinstra, Cees Snoek, and Harry Wijshoff. A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term. *IEEE Computer*, 49(5):54–63, May 2016. doi: 10.1109/MC.2016.127.

[19] C. G. Bassa, Z. Pleunis, and J. W. T. Hessels. Enabling pulsar and fast transient searches using coherent dedispersion. *Astronomy and Computing*, 18:40–46, Jan 2017. doi: 10.1016/j.ascom.2017.01.004.

[20] C. G. Bassa, Z. Pleunis, J. W. T. Hessels, E. C. Ferrara, R. P. Breton, N. V. Gusinskaia, V. I. Kondratiev, S. Sanidas, L. Nieder, C. J. Clark, T. Li, A. S. van Amesfoort, T. H. Burnett, F. Camilo, P. F. Michelson, S. M. Ransom, P. S. Ray, and K. Wood. LOFAR Discovery of the Fastest-spinning Millisecond Pulsar in the Galactic Field. *Astrophysical Journal, Letters*, 846(2):L20, Sep 2017. doi: 10.3847/2041-8213/aa8400.

[21] Andrea Bastianin and Massimo Florio. Social Cost-Benefit Analysis of HL-LHC, 2018. URL `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3202220`.

[22] Motti Beck and Michael Kagan. Performance evaluation of the RDMA over Ethernet (RoCE) standard in enterprise data centers infrastructure. In *Proceedings of the 3rd Workshop on Data Center-Converged and Virtual Ethernet Switching*, pages 9–15. International Teletraffic Congress, 2011.

[23] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on computing*, 26(5):1411–1473, 1997.

[24] Ian Bird. Computing for the Large Hadron Collider. *Annual Review of Nuclear and Particle Science*, 61(1):99–118, 2011. doi: 10.1146/annurev-nucl-102010-130059. URL `https://doi.org/10.1146/annurev-nucl-102010-130059`.

[25] Nanette J Boden, Danny Cohen, Robert E Felderman, Alan E Kulawik, Charles L Seitz, Jakov N Seizovic, and Wen-King Su. Myrinet: A gigabit-per-second local area network. *IEEE micro*, (1):29–36, 1995. doi: 10.1109/40.342015.

[26] Albert-Jan Boonstra and Ronald Nijboer. Radio Telescope Design Optimization Using Costing Constraints: Extrapolating LOFAR Costing to the Square Kilometre Array. *Radio Science*, 53(11):1346–1355, 2018. doi: 10.1029/2018RS006624. URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018RS006624`.

[27] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. P4: Programming Protocol-independent Packet Processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014. ISSN 0146-4833. doi: 10.1145/2656877.2656890. URL `http://doi.acm.org/10.1145/2656877.2656890`.

[28] M. Branicky. Analog computation with continuous ODEs. In *Proceedings Workshop on Physics and Computation. PhysComp '94*, pages 265,266,267,268,269,270,271,272,273,274, Los Alamitos, CA, USA, nov 1994. IEEE Computer Society. doi: 10.1109/PHYCMP.1994.363672. URL

https://doi.ieeecomputersociety.org/10.1109/PHYCMP.1994.363672.

[29] Jaap D. Bregman. Design concepts for a sky noise limited low frequency array. In A.B. Smolders and M.P. van Haarlem, editors, *Perspectives on Radio Astronomy – Technologies for Large Antenna Arrays*, 1999.

[30] Jaap D. Bregman. Concept design for a low-frequency array. In *Proc.SPIE*, volume 4015, pages 4015 – 4015 – 14, 2000. doi: 10.1117/12.390420. URL https://doi.org/10.1117/12.390420.

[31] Jaap D. Bregman. System Optimisation Of Multi-Beam Aperture Synthesis Arrays For Survey Performance. *Experimental Astronomy*, 17:365–380, June 2004. doi: 10.1007/s10686-005-2872-8.

[32] Jaap D. Bregman. *System Design and Wide-field Imaging Aspects of Synthesis Arrays with Phased Array Stations*. PhD thesis, Rijksuniversiteit Groningen, 2012.

[33] Brendan Gregg's Blog. KPTI/KAISER Meltdown Initial Performance Regressions, 2018. URL http://www.brendangregg.com/blog/2018-02-09/kpti-kaiser-meltdown-performance.html.

[34] Michiel A. Brentjens. Cobalt commissioning report. Technical report, ASTRON, September 2014.

[35] C. Broekema. SDP Hardware Decomposition View. URL http://ska-sdp.org/sites/default/files/attachments/ska-tel-sdp-0000013_07_sdparchitecture_hardware.pdf. SKA SDP CDR deliverable.

[36] P. C. Broekema. Compute Platform Element Subsystem Design, February 2015. SKA SDP PDR deliverable.

[37] P. C. Broekema. Improving sensor network robustness and flexibility using software-defined networks, February 2015. URL https://www.astron.nl/~broekema/papers/SDP-PDR/SDP-MEMO%20Software%20Defined%20Networks.pdf. SKA SDP Memo, SKA SDP PDR deliverable.

[38] P. C. Broekema et al. DOME: towards the ASTRON & IBM Center for Exascale Technology. In *Proceedings of the 2012 workshop on High-Performance Computing for Astronomy*, Astro-HPC '12, pages 1–4, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1338-4. doi: 10.1145/2286976.2286978. URL http://doi.acm.org/10.1145/2286976.2286978.

[39] P. Chris Broekema, Rob V. van Nieuwpoort, and Henri E. Bal. Exascale high performance computing in the square kilometer array. In *Proceedings of the*

2012 Workshop on High-Performance Computing for Astronomy Date, Astro-HPC '12, pages 9–16, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1338-4. doi: 10.1145/2286976.2286982. URL http://doi.acm.org/10.1145/2286976.2286982.

[40] P. Chris Broekema, J. Jan David Mol, Ronald Nijboer, Alexander S. van Amesfoort, Michiel A. Brenjens, G. Marcel Loose, and John W. Romein. Cobalt: a GPU-based correlator and beamformer for LOFAR. *Astronomy and Computing*, 23:180–192, April 2018. ISSN 2213-1337. doi: https://doi.org/10.1016/j.ascom.2018.04.006. URL http://www.sciencedirect.com/science/article/pii/S2213133717301439.

[41] A. Brown, T. N. Chan, K. Adamek, F. Dulwich, C. Pearson, C. Broekema, and W. Armour. SKA1 SDP Vertical Prototyping and Compute efficiency report. URL http://ska-sdp.org/sites/default/files/attachments/ska-tel-sdp-0000154_02_sdp_verticalprototypingcomputeefficiency_part_1_-_signed.pdf. SKA SDP CDR deliverable.

[42] Hadrien Bullot, R. Les Cottrell, and Richard Hughes-Jones. Evaluation of advanced tcp stacks on fast long-distance production networks. *Journal of Grid Computing*, 1(4):345–359, 2003. ISSN 1572-9184. doi: 10.1023/B:GRID.0000037555.53402.4f. URL http://dx.doi.org/10.1023/B:GRID.0000037555.53402.4f.

[43] H.R. Butcher. LOFAR: First of a New Generation of Radio Telescopes. *Proceedings of the SPIE*, 5489:537–544, October 2004.

[44] CasaCore. casacore. URL http://code.google.com/p/casacore/.

[45] Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 59–68, New York, NY, USA, 2003. ACM. ISBN 1-58113-674-9. doi: 10.1145/780542.780552. URL http://doi.acm.org/10.1145/780542.780552.

[46] M. A. Clark, P. C. La Plante, and L. J. Greenhill. Accelerating Radio Astronomy Cross-Correlation with Graphics Processing Units. *The International Journal of High Performance Computing Applications*, 27:178–192, May 2012. doi: 10.1177/1094342012444794.

[47] Thijs Coenen, Joeri van Leeuwen, Jason W. T. Hessels, Ben W. Stappers, Vladislav I. Kondratiev, A Alexov, R P. Breton, A Bilous, S Cooper, H Falcke, Richard Fallows, Vishal Gajjar, J M. Grießmeier, T E. Hassall, A Karastergiou, E F. Keane, M Kramer, M Kuniyoshi, A Noutsos, and A Zensus. The LOFAR Pilot Surveys for Pulsars and Fast Radio Transients. *Astronomy and Astrophysics*, 570, 08 2014. doi: 10.1051/0004-6361/201424495.

[48] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[49] T. J. Cornwell, P. Wortmann, B. Nikolic, and J. Farnes. SKA1 SDP Algorithm Reference Library (ARL) Report. Technical report, SDP Consortium, 2019. URL `http://ska-sdp.org/sites/default/files/attachments/ska-tel-sdp-0000150_02_sdparlreport_part_1_-_signed.pdf`.

[50] Mary Croarken. *Early scientific computing in Britain / Mary Croarken.* Oxford science publications. Clarendon Press, 1990. ISBN 9780198537489.

[51] D. Dalessandro, A. Devulapalli, and P. Wyckoff. Design and Implementation of the iWarp Protocol in Software. In *Proceedings of Parallel and Distributed Computing and Systems 2005*. ACTA Press, November 2005.

[52] D. Dalessandro, A. Devulapalli, and P. Wyckoff. iWarp protocol kernel space software implementation. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 8 pp.–, April 2006. doi: 10.1109/IPDPS.2006.1639565.

[53] C. M. de Vos, K. Van der Schaaf, and J. D. Bregman. Cluster computers and grid processing in the first radio-telescope of a new generation. In *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, CCGRID '01, pages 156–, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1010-8. URL `http://dl.acm.org/citation.cfm?id=560889.792400`.

[54] A. T. Deller, W. F. Brisken, C. J. Phillips, J. MMorgan, W. Alef, R. Cappallo, E. Middelberg, J. Romney, H. Rottmann, S. J. Tingay, and R. Wayth. DiFX-2: A More Flexible, Efficient, Robust, and Powerful Software Correlator. *Publications of the Astronomical Society of the Pacific*, 123:275–287, March 2011. doi: 10.1086/658907.

[55] A.T. Deller, S.J. Tingay, M. Bailes, and C. West. DiFX: A software correlator for very long baseline interferometry using multi-processor computing environments. *Publications of the Astronomical Society of the Pacific*, 119:318–336, February 2007. doi: 10.1086/513572.

[56] N. Denman, Mandana Amiri, Kevin Bandura, Jean-François Cliche, Liam Connor, Matt Dobbs, Mateus Fandino, Mark Halpern, Adam Hincks, Gary Hinshaw, Carolin Höfer, Peter Klages, Kiyoshi Masui, Juan Mena Parra, Laura Newburgh, Andre Recnik, J. Richard Shaw, Kris Sigurdson, Kendrick Smith, and Keith Vanderlinde. A GPU-based Correlator X-engine Implemented on the CHIME Pathfinder. In *6th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 35–40. IEEE, July 2015. doi: 10.1109/ASAP.2015.7245702.

[57] Robert H Dennard, Fritz H Gaensslen, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[58] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London Series A*, 400: 97–117, July 1985. doi: 10.1098/rspa.1985.0070.

[59] P. Dewdney et al. SKA phase 1: Preliminary system description, 2010. URL `https://www.skatelescope.org/uploaded/21705_130_Memo_Dewdney.pdf`. SKA Design documentation.

[60] P. E. Dewdney. SKA1 system baseline design, March 2013. URL `https://www.skatelescope.org/wp-content/uploads/2012/07/SKA-TEL-SKO-DD-001-1_BaselineDesign1.pdf`. SKA Design documentation.

[61] Peter E Dewdney, Peter J Hall, Richard T Schilizzi, and T Joseph LW Lazio. The Square Kilometre Array. *Proceedings of the IEEE*, 97(8):1482–1496, 2009.

[62] Lisa M Ellram. Total cost of ownership: an analysis approach for purchasing. *International Journal of Physical Distribution & Logistics Management*, 25(8): 4–23, 1995.

[63] B. Elsmore, S. Kenderdine, and Sir Ryle, Martin. The operation of the Cambridge one-mile telescope. *Monthly Notices of the Royal Astronomical Society*, 134:87, Jan 1966. doi: 10.1093/mnras/134.1.87.

[64] R. P. Feynman. Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21:467–488, June 1982. doi: 10.1007/BF02650179.

[65] FITS. FITS world coordinate systems. URL `http://www.atnf.csiro.au/people/mcalabre/WCS/`.

[66] Massimo Florio, Stefano Forte, and Emanuela Sirtori. Forecasting the socio-economic impact of the Large Hadron Collider: A cost–benefit analysis to 2025 and beyond. *Technological Forecasting and Social Change*, 112:38–53, 2016.

[67] T Freeth, Y Bitsakis, X Moussas, JH Seiradakis, A Tselikas, E Magkou, M Zafeiropoulou, R Hadland, D Bate, A Ramsey, et al. Decoding the antikythera mechanism: investigation of an ancient astronomical calculator. *Nature*, 444 (7119):587–591, 2006.

[68] Steve B Furber, David R Lester, Luis A Plana, Jim D Garside, Eustace Painkras, Steve Temple, and Andrew D Brown. Overview of the SpiNNaker System Architecture. *IEEE Transactions on Computers*, 62(12):2454–2467, 2012.

[69] Ferdl Graser and John Taylor. Ska sdp costing basis of estimate. Technical report, SDP Consortium, 2015. URL `http://www.astron.nl/~broekema/papers/SDP-PDR/PDR07-01%20Costs%20Basis%20of%20Estimate.pdf`.

[70] Green500. The Green500 list. URL `http://www.green500.org`.

[71] Thomas Gross and David Richard O'Hallaron. *iWarp: anatomy of a parallel computing system*. Mit Press, 1998.

[72] SKA Science Working Group. The square kilometre array design reference mission: SKA phase 1 v. 2.0, September 2011.

[73] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

[74] Juan Carlos Guzman, Gianluca Chiozzi, Alan Bridger, and Jorge Ibsen. The cost of developing and maintain the monitoring and control software of large ground-based telescopes. In *Software and Cyberinfrastructure for Astronomy III*, volume 9152, page 91521P. International Society for Optics and Photonics, 2014.

[75] Michael T Heideman, Don H Johnson, and C Sidney Burrus. Gauss and the history of the fast fourier transform. *Archive for history of exact sciences*, 34(3): 265–277, 1985.

[76] H.A. Holties. BP/P Replacement options. Technical report, ASTRON, May 2012.

[77] Ben Humphreys and Chris Broekema. HPC technology roadmap. Technical report, SPDO, December 2011. SKA Software and Computing CoDR.

[78] Giacomo Indiveri, Bernabé Linares-Barranco, Tara Julia Hamilton, André Van Schaik, Ralph Etienne-Cummings, Tobi Delbruck, Shih-Chii Liu, Piotr Dudek, Philipp Häfliger, Sylvie Renaud, et al. Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5:73, 2011.

[79] Kamil Iskra, John W. Romein, Kazutomo Yoshii, and Pete Beckman. ZOID: I/O-Forwarding Infrastructure for Petascale Architectures. In *ACM Symposium on Principles and Practice of Parallel Programming (PPoPP'08)*, pages 153–162, Salt Lake City, UT, February 2008. doi: http://doi.acm.org/10.1145/1345206.1345230.

[80] Justin L Jonas. MeerKAT - The South African array with composite dishes and wide-band single pixel feeds. *Proceedings of the IEEE*, 97(8):1522–1530, 2009.

[81] Lance Joneckis, David Koester, and Joshua Alspector. An initial look at alternative computing technologies for the intelligence community. Technical report, INSTITUTE FOR DEFENSE ANALYSES ALEXANDRIA VA, 2014.

[82] A. Keimpema, M. M. Kettenis, S. V. Pogrebenko, R. M. Campbell, G. Cimó, D. A. Duev, B. Eldering, N. Kruithof, H. J. van Langevelde, D. Marchal, G. Molera Calvés, H. Ozdemir, Z. Paragi, Y. Pidopryhora, A. Szomoru, and J. Yang. The SFXC software correlator for Very Long Baseline Interferometry: algorithms and implementation. *Experimental Astronomy*, 39(2):259–279, 2015. ISSN 1572-9508. doi: 10.1007/s10686-015-9446-1. URL `http://dx.doi.org/10.1007/s10686-015-9446-1`.

[83] Athol J Kemball and TJ Cornwell. A simple model of software costs for the square kilometre array. *Experimental Astronomy*, 17(1-3):317–327, 2004.

[84] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. *ArXiv e-prints*, January 2018.

[85] J. Kocz, L. J. Greenhill, B. R. Barsdell, D. Price, G. Bernardi, S. Bourke, M. A. Clark, J. Craig, M. Dexter, J. Dowell, T. Eftekhari, S. Ellingson, G. Hallinan, J. Hartman, A. Jameson, D. MacMahon, G. Taylor, F. Schinzel, and D. Werthimer. Digital Signal Processing Using Stream High Performance Computing: A 512-Input Broadband Correlator for Radio Astronomy. *Journal of Astronomical Instrumentation*, 4, March 2015. doi: 10.1142/S2251171715500038.

[86] Peter M. Kogge. Energy at exaflops. Supercomputing, 2009. The ExaScale Panel.

[87] Vincent Lariviere and Cassidy R. Sugimoto. The journal impact factor: A brief history, critique, and discussion of adverse effects. In W. Glanzel, H. F. Moed, U. Schmoch, and M. Thelwall, editors, *Springer Handbook of Science and Technology Indicators*. Springer International Publishing, Cham, Switzerland, 2018. URL `https://arxiv.org/abs/1801.08992`.

[88] LINPACK. The LINPACK benchmark. URL `http://www.netlib.org/benchmark/hpl/`.

[89] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *ArXiv e-prints*, January 2018.

[90] Jiuxing Liu, Balasubramanian Chandrasekaran, Jiesheng Wu, Weihang Jiang, Sushmitha Kini, Weikuan Yu, Darius Buntinas, Peter Wyckoff, and Dhabaleswar K Panda. Performance comparison of MPI implementations over InfiniBand, Myrinet and Quadrics. In *Supercomputing, 2003 ACM/IEEE Conference*, pages 58–58. IEEE, 2003.

[91] Jiuxing Liu, Jiesheng Wu, and Dhabaleswar K Panda. High performance RDMA-based MPI implementation over InfiniBand. *International Journal of Parallel Programming*, 32(3):167–198, 2004.

[92] Joe Mambretti, Jim Chen, Fei Yeh, Jingguo Ge, Junling You, Tong Li, Cees de Laat, Paola Grosso, Te-Lung Liu, Mon-Yen Luo, Aki Nakao, Paul Müller, Ronald van der Pol, Martin Reed, Michael Stanton, and Chu-Sing Yang. *Creating a Worldwide Network for the Global Environment for Network Innovations (GENI) and Related Experimental Environments*, pages 589–632. Springer International Publishing, Cham, 2016. ISBN 978-3-319-33769-2. doi: 10.1007/978-3-319-33769-2_24. URL `http://dx.doi.org/10.1007/978-3-319-33769-2_24`.

[93] Steve McConnell. *Code Complete, Second Edition*. Microsoft Press, Redmond, WA, USA, 2004. ISBN 0735619670.

[94] R. McCool and T. C. Cornwell. Miscellaneous corrections to the baseline design, October 2013. URL `https://www.skatelescope.org/wp-content/uploads/2014/11/SKA-TEL-SKO-0000002-AG-BD-DD-01-SKA1_System_Baseline_Design_Miscellaneous_Corrections.pdf`. SKA Design documentation.

[95] Derek McKay-Bukowski, Juha Vierinen, Ilkka I Virtanen, Richard Fallows, Markku Postila, Thomas Ulich, Olaf Wucknitz, Michiel Brentjens, Nico Ebbendorf, Carl-Fredrik Enell, et al. KAIRA: The Kilpisjärvi atmospheric imaging receiver array—System overview and first results. *IEEE Transactions on Geoscience and Remote Sensing*, 53(3):1440–1451, 2014. doi: 10.1109/TGRS.2014.2342252.

[96] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[97] Catalin Meirosu, Piotr Golonka, Andreas Hirstius, Stefan Stancu, Bob Dobinson, Erik Radius, Antony Antony, Freek Dijkstra, Johan Blom, and Cees de Laat. Native 10gigabit ethernet experiments over long distances. *Future Generation Computer Systems*, 21(4):457 – 468, 2005. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2004.10.003. URL `http://www.sciencedirect.com/science/article/pii/S0167739X04001657`. High-Speed Networks and Services for Data-Intensive Grids: the DataTAG Project.

[98] Jan David Mol and John W. Romein. The LOFAR Beam Former: Implementation and Performance Analysis. In *EuroPar'11*, volume LNCS 6853, Part II, pages 328–339, Bordeaux, France, August 2011. doi: 10.1007/978-3-642-23397-5_33.

[99] Gordon E. Moore. Cramming more components in integrated circuits. *Electronics*, 38(8), April 1965.

[100] Gordon E Moore et al. Progress in digital integrated electronics. In *Electron Devices Meeting*, volume 21, pages 11–13, 1975.

[101] Stephen G. Nash. *A history of scientific computing*. ACM Press history series. Addison-Wesley, 1990. ISBN 0201508141.

[102] National Archives. National archives currency converter, 2018. URL `http://www.nationalarchives.gov.uk/currency-converter/`.

[103] Roger M. Needham. Later Developments at Cambridge: Titan, CAP, and the Cambridge Ring. *IEEE Annals of the History of Computing*, 14, 1992. doi: 10.1109/85.194056.

[104] F.D. Neeser, B. Metzler, and P.W. Frey. SoftRDMA: Implementing iWARP over TCP kernel sockets. *IBM Journal of Research and Development*, 54(1):5:1–5:16, January 2010. ISSN 0018-8646. doi: 10.1147/JRD.2009.2036396.

[105] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with CUDA. *Queue*, 6(2):40–53, March 2008. ISSN 1542-7730. doi: 10.1145/1365490.1365500. URL `http://doi.acm.org/10.1145/1365490.1365500`.

[106] R. V. van Nieuwpoort and J. W. Romein. Correlating Radio Astronomy Signals with Many-Core Hardware. *International Journal of Parallel Processing*, 1(39): 88–114, February 2011. doi: 10.1007/s10766-010-0144-3.

[107] R. J. Nijboer et al. Parametric models of SDP compute requirements, February 2015. SKA SDP PDR deliverable.

[108] B. Nunes, Marc Mendonca, Xuan-Nam Nguyen, K. Obraczka, and Thierry Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys and Tutorials*, 2014.

[109] S. M. Ord, B. Crosse, D. Emrich, D. Pallot, R. B. Wayth, M. A. Clark, S. E. Tremblay, et al. The Murchison Widefield Array Correlator. *Publications of the Astronomical Society of Australia*, 32, 2015. ISSN 1448-6083. doi: 10.1017/pasa.2015.5. URL `http://journals.cambridge.org/article_S1323358015000053`.

[110] A. R. Parsons, D. C. Backer, G. S. Foster, et al. The Precision Array for Probing the Epoch of Re-ionization: Eight Station Results. *The Astronomical Journal*, 139:1468–1480, April 2010. doi: 10.1088/0004-6256/139/4/1468.

[111] Peter M. Kogge et al. ExaScale Computing Study: Technology Challenges in Achieving ExaScale Systems, September 2008.

[112] Gregory F Pfister. An introduction to the infiniband architecture. *High Performance Mass Storage and Parallel I/O*, 42:617–632, 2001.

[113] Peeyush Prasad, Folkert Huizinga, Eric Kooistra, Daniel van der Schuur, Andre Gunst, John Romein, Mark Kuiack, Gijs Molenaa, Antonia Rowlinson, John D. Swinbank, and Wijers Ralph A.M.J. The AARTFAAC All Sky Monitor: System Design and Implementation. *Journal of Astronomical Instrumentation*, 05(04), December 2016. doi: 10.1142/S2251171716410087.

[114] Peeyush Prasad, Folkert Huizinga, Eric Kooistra, Daniel van der Schuur, Andre Gunst, John Romein, Mark Kuiack, Gijs Molenaar, Antonia Rowlinson, John D Swinbank, et al. The AARTFAAC All-Sky Monitor: System Design and Implementation. *Journal of Astronomical Instrumentation*, 5(04):1641008, 2016. doi: 10.1142/S2251171716410087.

[115] Mohammad J Rashti and Ahmad Afsahi. 10-Gigabit iWARP Ethernet: comparative performance analysis with InfiniBand and Myrinet-10G. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8. IEEE, 2007.

[116] Mohammad J Rashti and Ahmad Afsahi. 10-Gigabit iWARP Ethernet: comparative performance analysis with InfiniBand and Myrinet-10G. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8. IEEE, 2007.

[117] Ralf Reussner, Peter Sanders, and Jesper Larsson Träff. SKaMPI: A comprehensive benchmark for public benchmarking of MPI. *Scientific Programming*, 10(1): 55–65, 2002. doi: 10.1155/2002/202839.

[118] J. W. Romein. An Efficient Work-Distribution Strategy for Gridding Radio-Telescope Data on GPUs. In *ACM International Conference on Supercomputing (ICS'12)*, pages 321–330, Venice, Italy, June 2012.

[119] John W. Romein. FCNP: Fast I/O on the Blue Gene/P. In *Parallel and Distributed Processing Techniques and Applications (PDPTA'09)*, volume 1, pages 225–231, Las Vegas, NV, July 2009.

[120] John W. Romein. A Comparison of Accelerator Architectures for Radio-Astronomical Signal-Processing Algorithms. In *International Conference on Parallel Processing (ICPP'16)*, pages 484–489, Philadelphia, PA, August 2016. doi: 10.1109/ICPP.2016.62.

[121] John W. Romein and Bram Veenboer. PowerSensor 2: A Fast Power Measurement Tool. In *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 111–113. IEEE, april 2018. doi: 10.1109/ISPASS.2018.00020.

[122] John W. Romein, P. Chris Broekema, Ellen van Meijeren, Kjeld van der Schaaf, and Walther H. Zwart. Astronomical Real-Time Streaming Signal Processing on a Blue Gene/L Supercomputer. In *ACM Symposium on Parallel Algorithms and Architectures (SPAA'06)*, pages 59–66, Cambridge, MA, July 2006. doi: 10.1145/1148109.1148118.

[123] John W. Romein, P. Chris Broekema, Jan David Mol, and Rob V. van Nieuwpoort. The LOFAR Correlator: Implementation and Performance Analysis. In *ACM Symposium on Principles and Practice of Parallel Programming (PPoPP'10)*, pages 169–178, Bangalore, India, January 2010. doi: 10.1145/1693453.1693477.

[124] J.W. Romein, J.D. Mol, R.V. van Nieuwpoort, and P.C. Broekema. Processing LOFAR Telescope Data in Real Time on a Blue Gene/P Supercomputer. In *URSI General Assembly and Scientific Symposium (URSI GASS'11)*, Istanbul, Turkey, August 2011.

[125] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann. Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge. *Micro, IEEE*, 32(2):20–27, March 2012. ISSN 0272-1732. doi: 10.1109/MM.2012.12.

[126] Jayanta Roy, Yashwant Gupta, Ue-Li Pen, Jeffrey B. Peterson, Sanjay Kudale, and Jitendra Kodilkar. A real-time software backend for the GMRT. *Experimental Astronomy*, 28(1):25–60, August 2010. doi: 10.1007/s10686-010-9187-0.

[127] Royal Society: About. About the Royal Society, 2018. URL `https://royalsociety.org/about-us/`.

[128] Royal Society: Elections. Elections, 2018. URL `https://royalsociety.org/fellows/elections/`.

[129] Royal Society Fellows. Fellows of the Royal Society 1660-2007, 2008. URL `http://royalsociety.org/uploadedFiles/Royal_Society_Content/about-us/fellowship/Fellows1660-2007.pdf`.

[130] Winston Royce. Managing the development of large software systems. volume 26 of *WESCON*. IEEE, August 1970.

[131] M. Ryle and A. C. Neville. A radio survey of the North Polar region with a 4.5 minute of arc pencil-beam system. *Monthly Notices of the Royal Astronomical Society*, 125:39, Jan 1962. doi: 10.1093/mnras/125.1.39.

[132] M. Ryle, B. Elsmore, and Ann C. Neville. Observations of radio galaxies with the one-mile telescope at Cambridge. *Nature*, 207:1024–1027, September 1964. URL `https://www.nature.com/articles/2071024a0.pdf`.

[133] Martin Ryle. Radio telescopes of large resolving power. In Sven Lundqvist, editor, *Nobel Lectures, Physics 1971-1980*. World Scientific Publishing Co., Singapore, 1992. URL `https://www.nobelprize.org/uploads/2018/06/ryle-lecture.pdf`.

[134] A. Scaife et al. Imaging pipeline, February 2015. SKA SDP PDR deliverable.

[135] Johannes Schemmel, Andreas Grubl, Karlheinz Meier, and Eilif Mueller. Implementing synaptic plasticity in a VLSI spiking neural network model. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1–6. IEEE, 2006.

[136] G. W. Schoonderbeek, A. Szomoru, A. W. Gunst, L. Hiemstra, and J. Hargreaves. UniBoard^2, A Generic Scalable High-Performance Computing Platform for Radio Astronomy. *Journal of Astronomical Instrumentation*, 8(2), Jan 2019. doi: 10.1142/S225117171950003X.

[137] Herwig Schopper. Some remarks concerning the cost/benefit analysis applied to LHC at CERN. *Technological Forecasting and Social Change*, 112:54–64, 2016.

[138] Ken Schwaber. Scrum development process. In Jeff Sutherland, Cory Casanave, Joaquin Miller, Philip Patel, and Glenn Hollowell, editors, *Business Object Design and Implementation*, pages 117–134. Springer London, 1997. ISBN 978-3-540-76096-2. doi: 10.1007/978-1-4471-0947-1_11. URL http://dx.doi.org/10.1007/978-1-4471-0947-1_11.

[139] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001. ISBN 0130676349.

[140] Sequoia. ASC Sequoia Benchmark Codes. URL http://asc.llnl.gov/sequoia/benchmarks/.

[141] Maciej Serylak, Aris Karastergiou, Chris Williams, Wesley Armour, Michael Giles, LOFAR Pulsar Working Group, et al. Observations of transients and pulsars with LOFAR international stations and the ARTEMIS backend. *Proceedings of the International Astronomical Union*, 8(S291):492–494, 2012. doi: 10.1017/S1743921312024623.

[142] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, 41:303–332, January 1999. doi: 10.1137/S0036144598347011.

[143] Peter W Shor. Why haven't more quantum algorithms been found? *Journal of the ACM (JACM)*, 50(1):87–90, 2003.

[144] SKA. The road to key science observations, 2015. URL https://www.skatelescope.org/news/the-road-to-key-science-observations-with-the-ska-kicks-off-in-stockholm/.

[145] James E Smith. Characterizing computer performance with a single number. *Communications of the ACM*, 31(10):1202–1207, 1988. doi: 10.1145/63039.63043.

[146] Richard Stone. Input-output and demographic accounting: A tool for educational planning. *Minerva*, IV(3), 1966.

[147] Hari Subramoni, Ping Lai, Miao Luo, and Dhabaleswar K Panda. RDMA over Ethernet—A preliminary study. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–9. IEEE, 2009.

[148] S.Y. Tjong Tjin Tai, J. van den Broek, and J. Deuten. De impact van grootschalige onderzoeksinfrastructuren – Een meetmethode voor de return on investment van internationale onderzoeksfaciliteiten, 2019. URL https://www.rathenau.nl/nl/vitale-kennisecosystemen/de-impact-van-grootschalige-onderzoeksinfrastructuren. In Dutch.

[149] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A compound tcp approach for high-speed and long distance networks. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–12, April 2006. doi: 10.1109/INFOCOM.2006.188.

[150] J. Taylor. SKA1 SDP Performance Prototype Platform (P3-Alaska) Prototyping Report. Technical report, SDP Consortium, 2019. URL `http://ska-sdp.org/sites/default/files/attachments/ska-tel-sdp-0000151_02_sdpp3alaskareport_part_1_-_signed.pdf`.

[151] A Richard Thompson, James M Moran, and George W Swenson Jr. *Interferometry and Synthesis in Radio Astronomy*. Springer, 3 edition, 2017. doi: 10.1007/978-3-319-44431-4. URL `https://link.springer.com/content/pdf/10.1007%2F978-3-319-44431-4.pdf`.

[152] Top500. The Top500 list. URL `http://www.top500.org`.

[153] top500. The 25th TOP500 list. https://www.top500.org/lists/2005/06/, June 2005.

[154] Damiaan Twelker. On the Feasibility of Software-Defined Networking in the Square Kilometre Array Science Data Processor. Bachelors thesis, University of Amsterdam, 2017. URL `https://esc.fnwi.uva.nl/thesis/centraal/files/f437594635.pdf`.

[155] Jan van den Ende. Tidal Calculations in The Netherlands. *IEEE Annals of the History of Computing*, 14(3):23–33, 1992. doi: 10.1109/85.150066.

[156] K. van der Schaaf, J. D. Bregman, and C. M. de Vos. Hybrid cluster computing hardware and software in the LOFAR radio telescope. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA*, volume 2, pages 695–701, Las Vegas, Nevada, USA, June 2003.

[157] Kjeld Van Der Schaaf and Ruud Overeem. COTS correlator platform. *Experimental Astronomy*, 17(1):287–297, June 2004. ISSN 1572-9508. doi: 10.1007/s10686-005-2864-8. URL `https://doi.org/10.1007/s10686-005-2864-8`.

[158] M. P. van Haarlem, M. W. Wise, A. W. Gunst, et al. LOFAR: The LOw-Frequency ARray. *Astronomy and Astrophysics*, 556, August 2013. doi: 10.1051/0004-6361/201220873.

[159] M.P. van Haarlem, M.W. Wise, A.W. Gunst, G. Heald, J.P. McKean, et al. LOFAR: The LOw-Frequency ARray. *Astronomy and Astrophysics*, 556, August 2013. doi: 10.1051/0004-6361/201220873.

[160] Rob V. van Nieuwpoort and John W. Romein. Building Correlators with Many-Core Hardware. *IEEE Signal Processing Magazine (special issue on "Signal Processing on Platforms with Multiple Cores: Part 2 – Design and Applications")*, 27(2):108–117, March 2010. doi: 10.1109/MSP.2009.935385.

[161] J. van Veen. Analogy between tides and AC electricity. *Engineering*, 184:498, 1947.

[162] J. Wagg, T. Bourke, J. Green, R. Braun, et al. SKA1 scientific use cases, March 2014. SKA Design documentation.

[163] J. Wagg, R. Braun, T. Bourke, and J. Green. A model schedule for five years of SKA1 observations, October 2014. SKA Design documentation.

[164] David D Walden, Garry J Roedler, Kevin Forsberg, R Douglas Hamelin, and Thomas M Shortell. *Systems engineering handbook: A guide for system life cycle processes and activities*. John Wiley & Sons, 2015.

[165] Jonas Weiss and Peter Maat. Analog optical signaling for large scale radio telescopes in harsh environments. In *2018 Optical Fiber Communications Conference and Exposition (OFC)*, March 2018.

[166] M. V. (Maurice Vincent) Wilkes. *Memoirs of a computer pioneer / by Maurice V. Wilkes.* The MIT Press series in the history of computing. MIT Press, Cambridge, Mass. ; London, 1985. ISBN 0262231220.

[167] Marc Wouters, James C. Anderson, and Finn Wynstra. The adoption of total cost of ownership for sourcing decisions—a structural equations analysis. *Accounting, Organizations and Society*, 30(2):167 – 191, 2005. ISSN 0361-3682. doi: https://doi.org/10.1016/j.aos.2004.03.002. URL `http://www.sciencedirect.com/science/article/pii/S0361368204000327`.

[168] Kazutomo Yoshii, Kamil Iskra, Harish Naik, Pete Beckman, and P. Chris Broekema. Characterizing the performance of "Big Memory" on Blue Gene Linux. In *Parallel Processing Workshops, 2009. ICPPW'09. International Conference on*, pages 65–72. IEEE, 2009. doi: 10.1109/ICPPW.2009.35.

[169] Kazutomo Yoshii, Kamil Iskra, Harish Naik, Pete Beckman, and P. Chris Broekema. Performance and Scalability Evaluation of 'Big Memory' on Blue Gene Linux. *International Journal of High Performance Computing Applications*, 25:148–160, May 2011. doi: doi:10.1177/1094342010369116. first published online on May 12, 2010.

[170] P. Zarka, Mohammed Nabil El Korso, Remy Boyer, Pascal Larzabal, et al. NenUFAR: Instrument description and science case. *2015 International Conference on Antenna Theory and Techniques (ICATT)*, pages 1–6, April 2015. doi: 10.1109/ICATT.2015.7136773.

Acknowledgements / Dankwoord

The fact that I'm currently writing an acknowledgement section drives home the fact that my thesis really is finaly finished. It has taken quite a while, I remember discussing the initial idea of doing a PhD with Marco, about twelve years ago according to my notes. That said, the content of this thesis is based on professional experience from my entire professional career so far. That means, almost by definition, that this section is going to be incomplete.

Looking back even further, I remember having weekly dinners with Titus and Geeske during my time in Delft, around 1995. What I remember most about those evenings is Titus asking *why?* or *how?* on some obscure technical topic he had somehow convinced himself I was knowledgeable on. It is these discussions that were my first attempts at explaining a complex subject in a clear and concise manner and answering difficult questions. I would have enjoyed explaining the content of this thesis, but unfortunately he passed away years ago. I do want to thank him for challenging me and inspiring me to, eventually, go into research and keep wondering about the *why* or *how*.

I want to thank my supervisors, Henri and Rob, for their support and help. While I wasn't necessarily around at the university very often, we worked together well using modern communication technologies. Thank you for your prompt replies to my questions, your support and encouragements especially when I was swamped with project work and was finding it hard to make time for research. In the end it all worked out.

Also my thanks to Mark Bentum, Cees de Laat, Jeff Templon, Paul Alexander and Herbert Bos for reading my thesis and being part of my thesis committee. Your comments were very useful and generally improved the quality of the work as it lies before you today.

Over the course of this project I have worked with colleagues from all around the world. My experience and research ambitions found a natural place in the SDP consortium, and this resulted in some excellent work, some of which is documented in this thesis. I want to thank all of my SDP colleagues that I have worked with over the last couple of years. There are too many to single out individually, but special thanks to Verity, Paul Alexander, Paul Calleja, Jeremy, Peter, Andreas, Markus, Rob, Kechil, Rosie, Bojan, Ferdl and Simon. Also my thanks to the excellent people at the SKA office, in particular Miles, Nick and (previously) Tim.

Many thanks too my co-workers in the DOME project. During the course of that project we established a really nice academic environment at ASTRON that I enjoyed

tremendously. Yan, Bram, Przemek, Rik, Erik, Leando, Yusik, Matthias, Giovanni and Liying, thanks for the coffees, beers, dinners and discussions on scientific and not so scientific matters. I would also like to thank our counterparts at IBM research in Zürich for their collaboration and hospitality. In particular my thanks to Ton, Ronald and Bernard for their contributions to my work.

I would also like to thank my colleagues from the LOFAR team, in particular those involved with the evolution of the Central Processor over the years. With Teun, Harm, Hopko, Wietze, Hans, Arjen and many others, I learned a lot about what it takes to design a useful system for an operational telescope. Many of the insights collected in this thesis can be traced back to the early experiences with the LOFAR Central Processor systems.

ASTRON has supported my research ambitions from the very start and I was fortunate enough to be able to pursue my PhD as part of my normal job. Special thanks to Marco de Vos who's immediate positive response to my wild idea of getting a PhD gave me the confidence to start on this journey. My thanks to all my colleagues at ASTRON over the years, in particular Ronald, John, Stefan, Albert-Jan, Jan David, Yan and Bram for all the things we have explored together over the last couple of years. Ágnes deserves a special acknowledgement here, since I've relied on her proof-reading skills more often than I can count. My thanks to you and your trusty red pen!

I can rely on a small but close group of friends. We've generally known each other since university and still communicate on a daily basis. Anton (x2), Torsten, Barry, Gijs, HW, Marjolein, Witske, Lia bedankt voor jullie vriendschap en ondersteuning. Onze gezamelijke biertjes in MOUT (bedankt Susan!), kerstdiners, dagtripjes en vakanties houden me stevig met beide benen op de grond. Een speciaal woord van dank voor mijn paranimfen, Torsten en Anton, voor de hulp bij het organiseren van de ceremonie en het feest.

En tot slot mijn dank aan mijn familie. Elke dinsdagavond ben ik welkom bij mijn zus en haar famillie thuis, waar mijn ouders op de kleinkinderen Ella en Tim hebben gepast en een maaltijd hebben gekookt. Dit zijn dit soort simpele dingen die het soms saaie schrijven van wetenschappelijke tekst leuk houden. Ik zeg het niet vaak genoeg, maar ik waardeer dit enorm en ik hoop dat jullie ook een beetje trots zijn op het werk dat hier nu voor je ligt.

Curriculum vitae

Chris Broekema has been employed by ASTRON, the Netherlands institute for radio astronomy, since 2003, first as a scientific programmer, now as a research staff member. Over the years he has focused most of his work on compute hardware design for radio astronomy. He has designed, built, procured, operated and worked on high-performance computing systems for the LOFAR telescope in the Netherlands, including some of the fastest supercomputers in the world at the time, since 2004. More recently his focus shifted to the computational and data-transport challenges in the Square Kilometre Array (SKA), where he was responsible for the hardware platform design of the SKA Science Data Processor (SDP).

Apart from the direct work on telescopes described above, he has been doing applied research into compute- and data-transport systems for radio astronomy applications. His focus is on the various system aspects that impact the suitability and usability of particular compute- and data-transport systems for radio astronomy. He has published several papers on the subject, some of which are collected in this thesis.

Finally, he served in various panels and review committees, most notably for the ASKAP science data processor and the ARTS system, and has presented my work numerous times, both nationally and internationally. Some highlights of his publications, both peer-reviewed and design work, are detailed below:

Peer-reviewed publications (2012 - 2019)

DOME: Towards the ASTRON & IBM Center for ExaScale Technology,
P. Chris Broekema, Albert-Jan Boonstra, Victoria Caparrós Cabezas, Ton Engbersen, Robert Haas, Hanno Holties, Jens Jelitto, Ronald P. Luijten, Peter Maat, Rob V. van Nieuwpoort, Ronald Nijboer, John W. Romein, and Bert Jan Offrein, *Proceedings of the 2012 workshop on High-Performance Computing for Astronomy (AstroHPC'12), Delft, the Netherlands, June, 2012*

ExaScale high performance computing in the Square Kilometer Array,
P. Chris Broekema, Rob V. van Nieuwpoort, and Henri E. Bal, *Proceedings of the 2012 workshop on High-Performance Computing for Astronomy (AstroHPC'12), Delft, the Netherlands, June 2012*

LOFAR: the Low Frequency Array,
MP van Haarlem, MW Wise, AW Gunst, George Heald, JP McKean, JWT Hessels, AG de Bruyn, Ronald Nijboer, John Swinbank, Richard Fallows, M Brentjens, A Nelles, Rainer Beck, H Falcke, R Fender, J Hörandel, LVE Koopmans, G Mann, G Miley, H Röttgering, BW Stappers, RAMJ Wijers, S Zaroubi, M van Den Akker, A Alexov, J Anderson, K Anderson, Arnold van Ardenne, M Arts, A Asgekar, IM Avruch, Fabien Batejat, L Bähren, ME Bell, MR Bell, I van Bemmel, P Bennema, Marinus Jan Bentum, G Bernardi, P Best, L Bîrzan, A Bonafede, A-J Boonstra, R Braun, J Bregman, F Breitling, RH Van de Brink, J Broderick, PC Broekema, WN Brouw, M Brüggen, HR Butcher, W Van Cappellen, B Ciardi, T Coenen, John Conway, A Coolen, A Corstanje, S Damstra, O Davies, AT Deller, R-J Dettmar, G Van Diepen, K Dijkstra, P Donker, A Doorduin, J Dromer, M Drost, A Van Duin, J Eislöffel, J Van Enst, C Ferrari, W Frieswijk, H Gankema, MA Garrett, F De Gasperin, M Gerbers, E De Geus, J-M Grießmeier, T Grit, P Gruppen, JP Hamaker, T Hassall, M Hoeft, HA Holties, A Horneffer, A Van Der Horst, A Van Houwelingen, A Huijgen, M Iacobelli, H Intema, N Jackson, V Jelic, A De Jong, E Juette, D Kant, A Karastergiou, A Koers, H Kollen, VI Kondratiev, E Kooistra, Y Koopman, A Koster, M Kuniyoshi, M Kramer, G Kuper, P Lambropoulos, C Law, J Van Leeuwen, J Lemaitre, M Loose, P Maat, G Macario, S Markoff, J Masters, RA McFadden, D McKay-Bukowski, H Meijering, H Meulman, M Mevius, E Middelberg, R Millenaar, JCA Miller-Jones, RN Mohan, JD Mol, J Morawietz, R Morganti, DD Mulcahy, E Mulder, H Munk, L Nieuwenhuis, R van Nieuwpoort, JE Noordam, M Norden, A Noutsos, AR Offringa, H Olofsson, A Omar, E Orrú, R Overeem, H Paas, M Pandey-Pommier, VN Pandey, R Pizzo, A Polatidis, D Rafferty, S Rawlings, W Reich, J-P de Reijer, J Reitsma, *Astronomy & Astrophysics, Volume 556, August 2013*.

The Square Kilometre Array Science Data Processor - Preliminary Compute Platform Design,
P. Chris Broekema, Rob V. van Nieuwpoort, and Henri E. Bal, *Journal of Instrumentation, Volume 10, Number 07, July, 2015*

Energy-Efficient Data Transfers in Radio Astronomy with Software UDP RDMA,
Przemyslaw Lenkiewicz, P. Chris Broekema, and Bernard Metzler, *Future Generation Compute Systems, February 2018*

Software-defined networks in large-scale radio telescopes,
P. Chris Broekema, Damiaan R. Twelker, Daniel C. Romão, Paola Grosso, Rob V. van Nieuwpoort, Henri E. Bal, *Computing Frontiers, May 15-17 2017, Sienna, Italy*

Cobalt: A GPU-based correlator and beamformer for LOFAR,
P. Chris Broekema, J. Jan David Mol, Ronald Nijboer, Alexander S. van Amesfoort, Michiel A. Brentjens, G. Marcel Loose, John W. Romein, *Astronomy and computing, Volume 23, April 2018, pages 180-192*

On optimising cost and value in compute systems for radio astronomy,
P. Chris Broekema, Verity L. Allan, Rob V. van Nieuwpoort, Henri E. Bal, *Astronomy and Computing, Volume 30, January 2020*

Architecture- and design documentation (2012 - 2019)

COBALT - Requirements and Specifications,
Chris Broekema, Jan David Mol, Ronald Nijboer, *COBALT Critical Design Review deliverable, February 2013*.

COBALT - Top Level Hardware Design,
Chris Broekema, *COBALT Critical Design Review deliverable, February 2013*.

Compute Platform Element Subsystem Design,
P. C. Broekema, *SKA SDP Preliminary Design Review deliverable, February 2015*.

Compute platform: Hardware alternatives and developments,
J. Bancroft, A. Ensor, J. Taylor, S. Wu, Y. Zhu, P. C. Broekema, Á. Mika, *SKA SDP Preliminary Design Review deliverable, February 2015*.

Compute platform: Software stack developments and considerations,
D. Christie, P. Crosby, A. Ensor, N. Erdody, P. C. Broekema, B. A. do Lago, M. Mahmoud, R. O'Brien, R. Rocha, T. Stevenson, A. St John, J. Taylor, Y. Zhu, Á. Mika, *SKA SDP Preliminary Design Review deliverable, February 2015*.

Analysis of SDP system scaling to SKA phase 2,
P. Alexander, P.C. Broekema, Á. Mika, R. Bolton, *SKA SDP Preliminary Design Review deliverable, February 2015*.

Improving sensor network robustness and flexibility using software-defined networks,
P.C. Broekema, *SKA SDP Preliminary Design Review deliverable, February 2015*.

SDP Architecture,
P. Alexander, V. Allan, R. Bolton, P. C. Broekema, G. van Diepen, S. Gounden, Á. Mika, R. Nijboer, B. Nikolic, S. Ratcliffe, A. Scaife, R. Simmonds, J. Taylor, A. Wicenec, *SKA SDP Delta Prelimininary Design Review deliverable, March 2016*.

SDP Data Processor Platform Design,
P.C. Broekema, *SKA SDP Delta Prelimininary Design Review deliverable, April 2016*.

SDP Preservation Design,
Markus Dolensky, Chris Broekema, Patrick Dowler, Iain Emsley, Julián Garrido, Kevin Vinsen, Andreas Wicenec, *SKA SDP Delta Prelimininary Design Review deliverable, March 2016*.

Parametric models of SDP compute requirements,
R. Bolton, P.C. Broekema, T.J. Cornwell, G. van Diepen, C. Hollitt, M. Johnston-Hollitt, L. Levin Preston, Á. Mika, R. Nijboer, B. Nikolic, S. Salvini, A. Scaife, B. Stappers, *SKA SDP Delta Prelimininary Design Review deliverable, April 2016*.

Analysis of SDP system scaling to SKA phase 2,
P. Alexander, P.C. Broekema, Á. Mika, R. Bolton, *SKA SDP Delta Prelimininary De-*

sign Review deliverable, March 2016.

SKA1 SDP High Level Overview,
P. Alexander, B. Nikolic, V. Allan, C. Broekema, M. Deegan, P. Wortmann, *SKA SDP Critical Design Review deliverable, October 2018*.

SKA1 SDP Architecture Reading Guide,
P. Alexander, V. Allan, M. Ashdown, U. Badenhorst, R. Bolton, C. Broekema, L. Christelis, J. Coles, T. Cornwell, M. Deegan, G. van Diepen, F. Dulwich, A. Ensor, D. Fenech, S. Gounden, J. Garbutt, S. Goliath, F. Graser, J.C. Guzman, P. Harding, K. Kirkham, L. Levin Preston, R. Lyon, Á. Mika, D. Mitchell, B. Mort, B. Nikolic, R. Nijboer, S. Sanchez, R. Simmonds, B. Stappers, J. Taylor, A. Wicenec, P. Wortmann, *SKA SDP Critical Design Review deliverable, October 2018*.

Parametric models of SDP compute requirements,
R. Bolton, P.C. Broekema, T.J. Cornwell, G. van Diepen, C. Hollitt, M. Johnston-Hollitt, L. Levin Preston, Á. Mika, R. Nijboer, B. Nikolic, S. Salvini, H. Rampadarath, A. Scaife, B. Stappers, P. Wortmann, *SKA SDP Critical Design Review deliverable, October 2018*.

SDP Hardware Decomposition View,
C. Broekema, *SKA SDP Critical Design Review deliverable, October 2018*.

SDP System Module Decomposition and Dependency view,
P. Alexander, V. Allan, U. Badenhorst, C. Broekema, T. Cornwell, S. Gounden, F. Graser, K. Kirkham, B. Mort, R. Nijboer, B. Nikolic, R. Simmonds, J. Taylor, A. Wicenec, P. Wortmann, *SKA SDP Critical Design Review deliverable, October 2018*.

SKA1 SDP Construction and Verification Plan,
F. Graser, B. Nikolic, P. Alexander, J. Coles, C. Broekema, *SKA SDP Critical Design Review deliverable, October 2018*.

SKA1 SDP Vertical Prototyping and Compute Efficiency Report,
A. Brown, T. N. Chan, K. Adamek, F. Dulwich, C. Pearson, C. Broekema, W. Armour, *SKA SDP Critical Design Review deliverable, October 2018*.