The Square Kilometre Array Science Data Processor Preliminary Compute Platform Design

P. Chris Broekema
Netherlands Institute for Radio Astronomy (ASTRON)
broekema@astron.nl

Rob V. van Nieuwpoort
Netherlands e-Science center
r.vannieuwpoort@esciencecenter.nl

Henri E. Bal
VU University Amsterdam
bal@cs.vu.nl

***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 paper 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.**

I. 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 than 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 paper starts with an analysis of the requirements and contraints 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 paper 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.

A. 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 [6] and the corrections thereof [8]. Here we suffice with a summary of the characteristics of the SKA phase one telescopes as shown in Table I. In March 2015, the SKA underwent a major rebaselining [2], 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.

The SKA, in addition to being one of the premier science instruments of this century, is considered a major IT challenge. Table I shows the input bandwidth expected into the SDP facilities and the compute capacity required as indicated by our initial parametric modelling efforts [7], [9]. The required compute capacity is a work in progress and does not take computational efficiency into account, which means that in reality the installed system (peak) capacity needs to be several times larger. We expand on this in section II-A.

Figure 1 gives a high-level overview of the SKA1 system, showing the two (distributed) telescope receivers, the Central Signal Processor (CSP, see Section I-B) systems and the Science Data Processors. This paper will concentrate on the Science Data Processor.

B. 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 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.

	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 [7]	24 PFLOPS	5.7 PFLOPS

TABLE I

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.

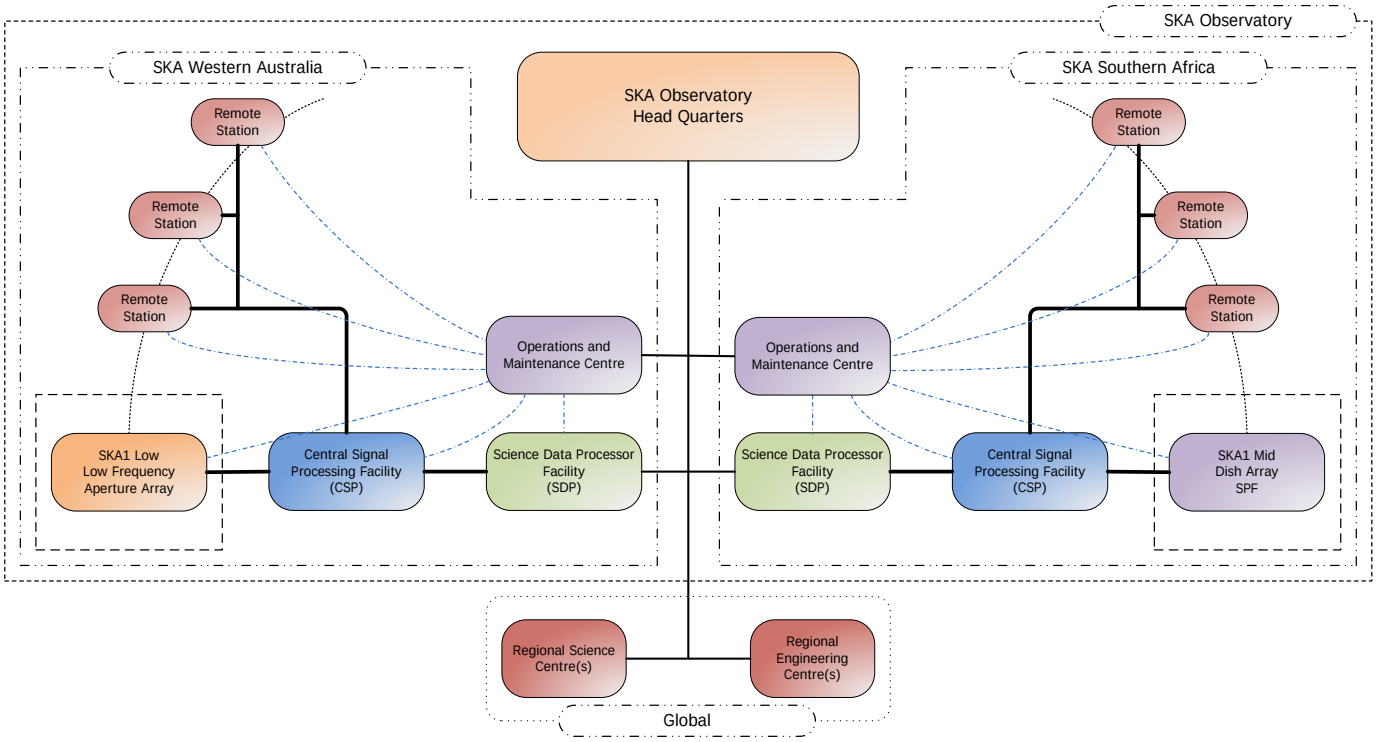

Fig. 1. The Square Kilometre Array top-level system overview for phase one of the project. This figure is based on similar images by the SKA Office.

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 paper. The interested reader is referred to the wealth of information available on the subject, in particular [9], [11].

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 I, compute charateristics 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.

II. 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 [14], but these are only described in limited detail [13]. 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 [7], [9].

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.

A. Defining the required SDP capacity

The required aggregate compute capacity of the SDP (R_{SDP}), assumed to be in double precision[1] floating point operations per second (FLOPS), is defined by:

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

where I_{bw} is the input bandwidth which is given in the baseline design [6]. q is the computational intensity[2] of the processing required in FLOP/byte, an estimate of which for each pipeline component is given in our parametric models [9]. 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 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 undestanding of the required processing, on current day best-of-breed hardware. This shows very poor efficiency of **at most** 20% of R_{peak} [1], [10].

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

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

B. 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 2 shows the preliminary timeline for the SDP roll-out for the three systems.

III. 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
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. Note that these priorities are not limited to the hardware design, but span the entirety of the SDP design.

A. SDP top-level design considerations

Taking into account the design principles introduced above, we make some key observations. The SKA SDP will be

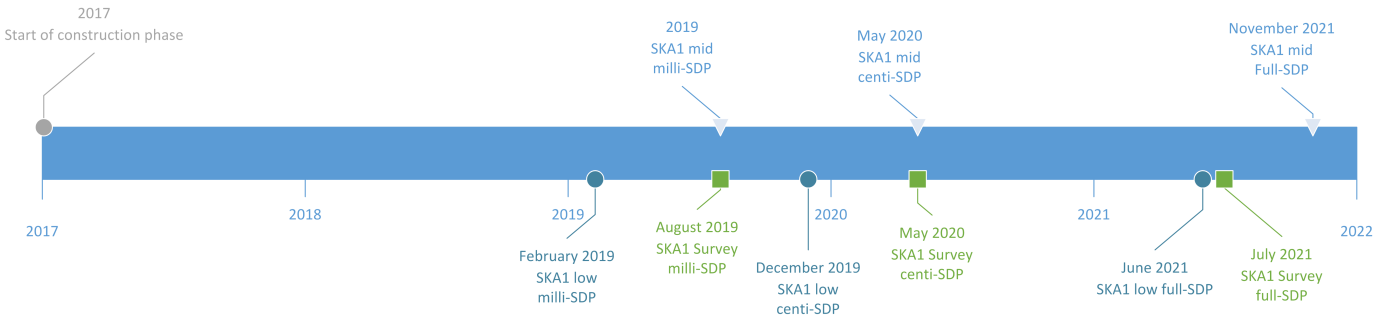

Fig. 2. Preliminary roll-out schedule for both SDP systems, based on the preliminary roll-out schedule of the antennas.

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 [12], 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. 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.

IV. 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 VI, that are sized to support end-to-end processing of the data directed to them. Figure 3 shows a high-level overview of the SKA SDP data flow.

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.

V. Top-level network design

The top-level SDP network architecture is shown in Figure 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.

A. 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 [4]. The classic split between

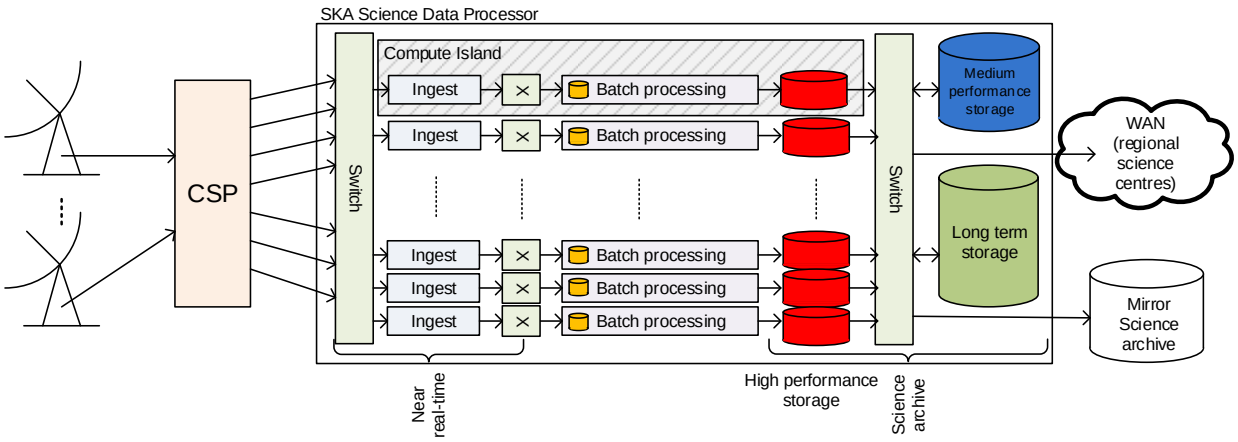

Fig. 3. The SDP top-level data flow. Data flows into the SDP switches from CSP, where it is directed to the subscribed SDP component(s). After ingest and optional reordering of data through the Compute Island switch, identified by a X, data are buffered for iterative batch processing. Science ready outputs are stored in the science archive and exported to the world outside of SKA SDP.

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 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 [4].

VI. Compute Islands

In this paper 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 only to processes 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 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 VII.

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 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

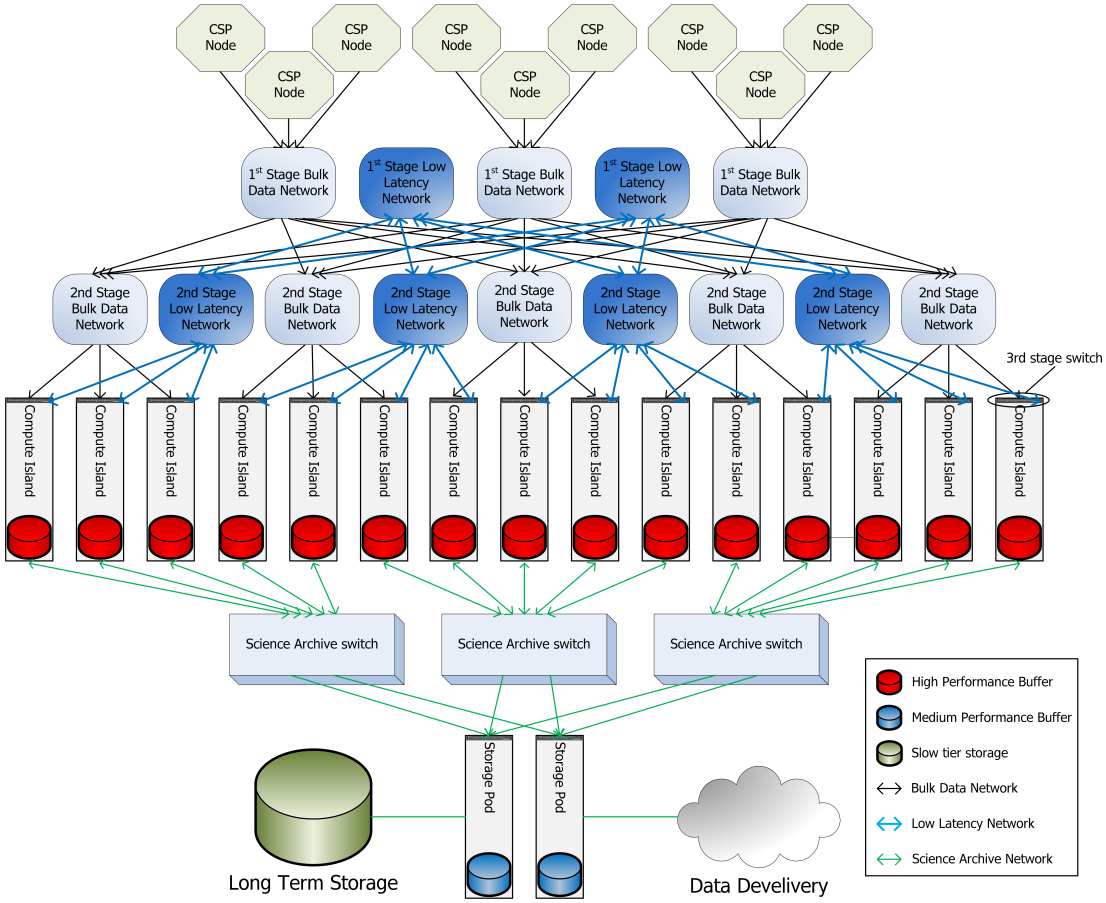

Fig. 4. The SDP top-level network design

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.

VII. 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).

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 [9] 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

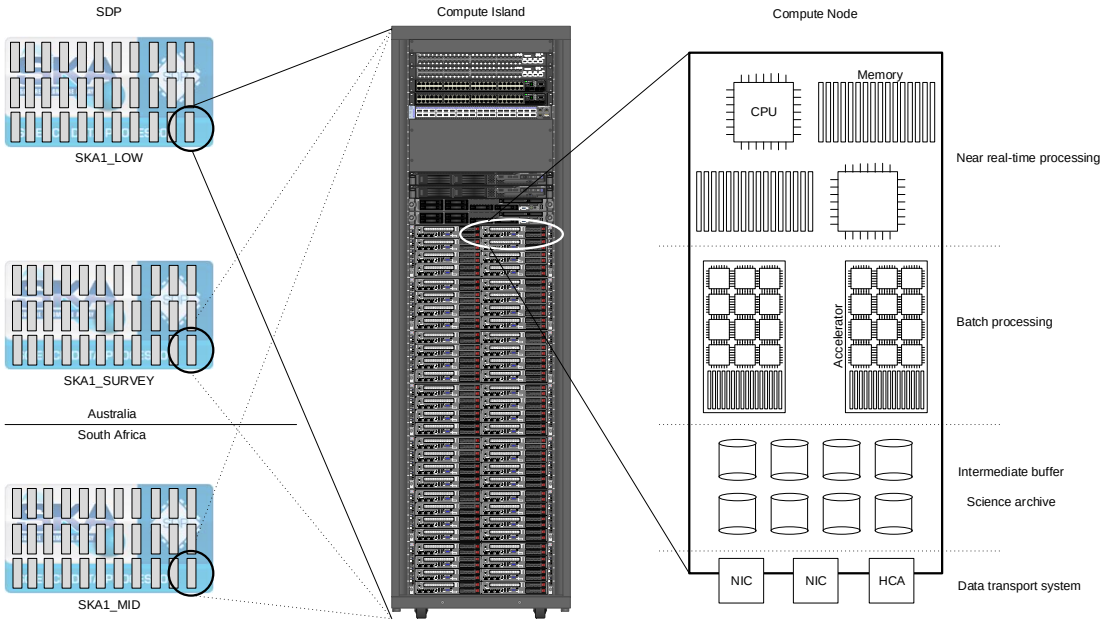

Fig. 5. SDP scaling using compute Islands. Each SDP instance contains many self-contained Compute Islands, each containing multiple compute nodes.

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.

VIII. Conclusion and discussion

In this paper we present an overview of the design considerations and constraints for the SKA Science Data Processor. This paper 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 paper, 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, 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 paper 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.

Acknowledgements

This work is based heavily on the SKA SDP Preliminary Design Review documentation, specifically the compute platform component [3]. In addition, many of the concepts introduced in this paper, in particular the four design priorities in Section III, evolved during discussions with the rest of the SDP consortium. This work is supported by the ASTRON/IBM Dome project [5], funded by the province Drenthe and the Dutch Ministry of EL&I.

References

[1] Cuda 6.5 performance report. https://developer.nvidia.com/cuFFT, September 2014.
[2] The world's largest radio telescope takes a major step towards construction. https://www.skatelescope.org/news/worlds-largest-radio-telescope-near-construction, March 2015.
[3] P. C. Broekema. Compute platform element subsystem design, February 2015. SKA SDP PDR deliverable.
[4] P. C. Broekema. Improving sensor network robustness and flexibility using software-defined networks, February 2015. SKA SDP Memo.
[5] 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.
[6] P. E. Dewdney. SKA1 system baseline design, March 2013.
[7] F. Malan et al. SKA SDP performance model. https://github.com/SKA-ScienceDataProcessor/sdp-par-model.
[8] R. McCool and T. C. Cornwell. Miscellaneous corrections to the baseline design, October 2013.

[9] R. J. Nijboer et al. Parametric models of SDP compute requirements, February 2015. SKA SDP PDR deliverable.
[10] 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.
[11] A. Scaife et al. Imaging pipeline, February 2015. SKA SDP PDR deliverable.
[12] M. P. van Haarlem, M. W. Wise, A. W. Gunst, G. Heald, J. P. McKean, et al. LOFAR: The LOw-Frequency ARray. *Astronomy & Astrophysics*, 556, 2013.
[13] J. Wagg, T. Bourke, J. Green, R. Braun, et al. SKA1 scientific use cases, March 2014.
[14] J. Wagg, R. Braun, T. Bourke, and J. Green. A model schedule for five years of SKA1 observations, October 2014.