Article

International Journal of HIGH PERFORMANCE COMPUTING APPLICATIONS

The International Journal of High Performance Computing Applications
2015, Vol. 29(1) 37–50

DOI: 10.1177/1094342014549059
hpc.sagepub.com
SAGE

Challenges in exascale radio astronomy: Can the SKA ride the technology wave?

Erik Vermij[1], Leandro Fiorin[1], Rik Jongerius[1], Christoph Hagleitner[2] and Koen Bertels[3]

Abstract

The Square Kilometre Array (SKA) will be the most sensitive radio telescope in the world. This unprecedented sensitivity will be achieved by combining and analyzing signals from 262,144 antennas and 350 dishes at a raw datarate of petabits per second. The processing pipeline to create useful astronomical data will require hundreds of peta-operations per second, at a very limited power budget. We analyze the compute, memory and bandwidth requirements for the key algorithms used in the SKA. By studying their implementation on existing platforms, we show that most algorithms have properties that map inefficiently on current hardware, such as a low compute–bandwidth ratio and complex arithmetic. In addition, we estimate the power breakdown on CPUs and GPUs, analyze the cache behavior on CPUs, and discuss possible improvements. This work is complemented with an analysis of supercomputer trends, which demonstrates that current efforts to use commercial off-the-shelf accelerators results in a two to three times smaller improvement in compute capabilities and power efficiency than custom built machines. We conclude that waiting for new technology to arrive will not give us the instruments currently planned in 2018: one or two orders of magnitude better power efficiency and compute capabilities are required. Novel hardware and system architectures, to match the needs and features of this unique project, must be developed.

Keywords

Exascale, Square Kilometre Array, supercomputers, power efficiency, astronomy

1. Introduction

The Square Kilometre Array (SKA) (SKA Organisation, n.d.) will be the largest radio telescope in the world, and it will have an unprecedented sensitivity, angular resolution and survey speed. Most specifications are 10 to a 100 times better than any existing telescope. Because of the size of the project, its construction has been divided into two phases: SKA1, and its extension SKA2. SKA1 is currently being designed, and construction will start in 2018. In the same year, the design of SKA2 will start. This paper will only look at the SKA1, because the specifications for SKA2 have yet to be finalized. SKA1 will deploy 262,144 antennas and 350 dishes in remote areas in South Africa and Australia, together producing several petabits of data per second. Realizing the SKA1 will face many challenges in diverse fields like data transport, algorithms, data storage, and system design. In this paper we will look at the computational challenges of the project: the absolute performance and power efficiency required. Power efficiency has special attention, since several subsystems of the SKA1 will be located far away from any human infrastructure.

The main contributions of this paper are as follows. We present a detailed computational profile of SKA1 and its main algorithms, analyze the algorithms on existing hardware, and discuss points for improvement. We show relevant trends in high-performance computing and introduce the innovation metric to compare generations of supercomputers. Finally, we argue about the feasibility of deploying the SKA1 using commercial off-the-shelf (COTS) hardware.

[1]IBM Research, The Netherlands
[2]IBM Research, Zurich, Switzerland
[3]Delft University of Technology, Delft, The Netherlands

Corresponding author:
Erik Vermij, IBM Research, PO Box 2 Dwingeloo, AA 7990, The Netherlands.
Email: erik.vermij@nl.ibm.com

2. SKA1 project description

SKA1 (SKA Organisation, 2013) will consist of three instruments: SKA1-low, SKA1-mid, and SKA1-survey.

SKA1-low is an aperture-array instrument (Perley, 1984) consisting of 1024 stations, each containing 256 dual-polarized antennas, which will receive signals between 50 and 350 MHz. The antenna signals are summed per station into a single beam, which is transported to a central signal-processing facility. The stations will be 35 m in diameter, and the maximum distance between any two stations is 70 km. This instrument will be very much like a big version of LOFAR, the low-frequency aperture array built in the Netherlands (Van Haarlem et al., 2013).

SKA1-mid will use 254 single-pixel feed dishes capable of receiving signals between 350 MHz and 13.8 GHz. From this frequency range, a 2.5 GHz band can be selected for measurements. The distance between any two dishes will be at most 200 km.

The SKA1-survey instrument will use 96 dishes, each containing phased-array receivers. Every receiver will have 94 antennas and can point 36 beams onto the sky. In this way, a single dish has a huge field of view, compared with the SKA1-mid dishes. The frequency ranges between 250 MHz and 4 GHz, with an instantaneous bandwidth of 500 MHz. The maximum distance between any two dishes will be 50 km.

2.1 Science cases for the SKA1

In the early stages of the SKA(1) project, two major science cases where identified (SKA Organisation, 2014):

- understanding the history and role of neutral hydrogen in the Universe from the dark ages to the present-day;
- detecting and timing binary pulsars and spin-stable millisecond pulsars in order to test theories of gravity (including general relativity and quantum gravity), to discover gravitational waves from cosmological sources, and to determine the equation of state of nuclear matter.

Besides these two main science cases, there are about 10 others, like searching for exo-planets and studying cosmic magnetism. From these science cases, various use-cases have been defined, which give hints towards the instrument and processing requirements of what is needed to produce relevant scientific results. Two main categories can be identified: imaging and non-imaging use-cases. In the imaging mode we create images of the sky, while for the non-imaging mode, we are interested in, for example, time-series. Besides the very first processing steps, these two modes have not much in common.

The first science case stated above (also known as Epoch of Reionization), has an imaging use-case, and some requirements are listed below:

- use the SKA1-low instrument, in a frequency range from 50 to 300 MHz;
- 208,333 image frequency channels, about 1.2 kHz each;
- image resolution between 7 arcseconds and 1 arcminute, for respectively calibration and actual imaging;
- image dynamic range larger than 2.5×10^6.

From these requirements we can, to some extent, dimension the telescope and processing pipeline. For example, the image frequency channel requirement translates to the Fast Fourier Transform (FFT) size needed in the correlator (as explained in the next section). The dynamic range requirement translates to the quality of the entire calibration and imaging pipeline.

As we cannot evaluate all science cases in this work, we will focus on an overall use-case: creating sky images for the entire frequency range of the instrument, and no channel integration. Based on experience with existing telescopes, this is expected to be the most compute intensive workload. In Figure 1 we show an example of a sky image (for a single frequency channel). In this image we see the distribution of hydrogen in the M81 galaxy, which shows a far more extended structure than images made in the human-visible light spectrum.

2.2 Processing pipeline and applications

In Figure 2, we show the simplified processing flow for the SKA1-low, using the continuum imaging science case. For SKA1-mid and SKA1-survey, the antenna/station subsystem is replaced by a single dish.

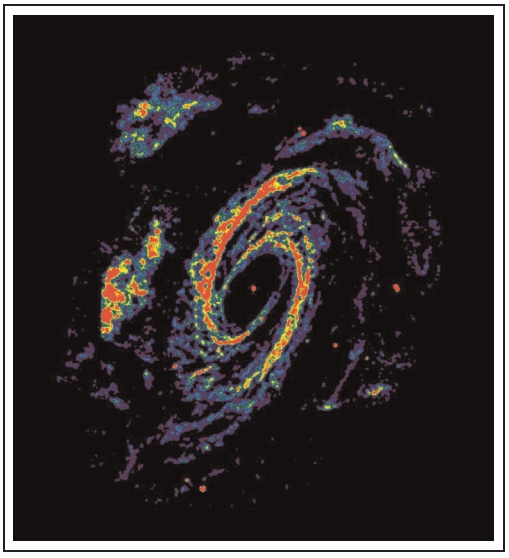

Figure 1. Cold atomic hydrogen gas in the M81 galaxy, measured at 1420.4 MHz. As a reference, human-visible red light has a frequency of 430 THz.
Source: Image courtesy of NRAO, DS Adler, DJ Westpfahl.

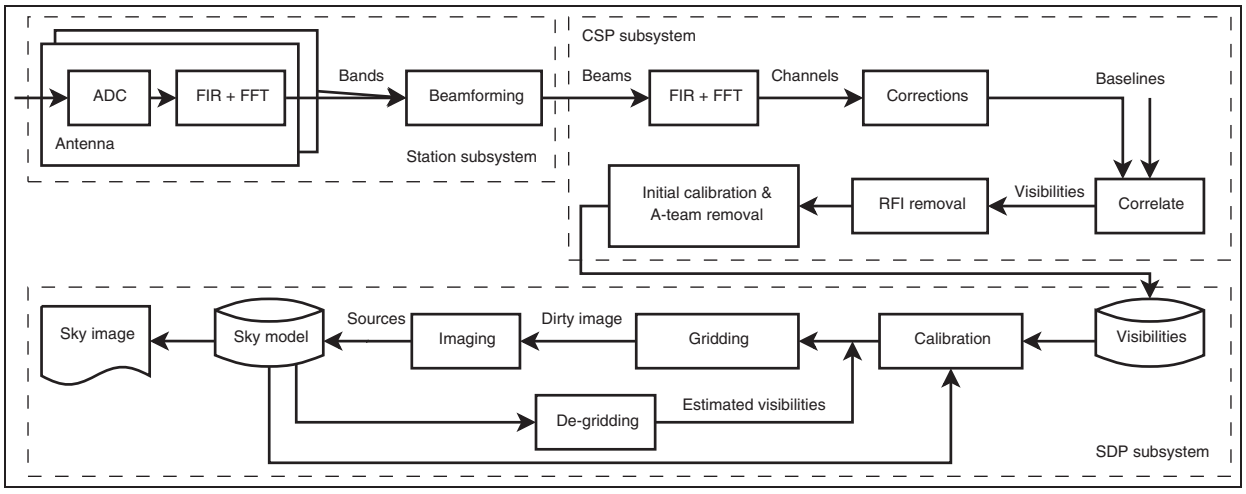

Figure 2. Overview of the processing steps for the SKA1-low, using imaging pipeline. For SKA1-mid and SKA1-survey, the antenna/station subsystem will be replaced by a dish.

SKA1-survey phased-array feed processing is not analyzed in this article.

2.2.1 Station processing for SKA1-low. Because the SKA1-low uses aperture arrays instead of dishes, extra processing is required to synthesize a dish. The digitized antenna signal from the analog-to-digital converter is sent to a polyphase channelizer consisting of several finite impulse response (FIR) filter banks and an FFT, creating a number of frequency bands. These are fed into the beamformer, in which every band is multiplied with a complex phase shift to delay the signal, and added to the corresponding band from the other antennas. By delaying signals between antennas and summing them, the instrument focuses its sensitivity into a specific direction, creating a so-called *beam*. The channelization before beamforming is needed because the beamforming concept only works on small frequency bandwidths (Jeffs, n.d.).

2.2.2 Central signal processing (CSP). The beam from every station or dish undergoes a second channelization, generating finer frequency channels, after which the beams are aligned in time and phase and, in the case of the SKA1-low, undergo a gain correction to offset filter artifacts from the station processing. The beam is correlated (multiplied) (Thompson et al., 2001) with the data from all other stations or dishes, and integrated over a small period of time (the dump time). By correlating, the signal-to-noise ratio of the data improves. A pair of stations/dishes is called a *baseline*. The result is a visibility, which is a sample of the Fourier-transformed sky. The visibilities are processed by removing RFI signals and by performing a calibration step, to correct for known system inequalities. Furthermore, a set of well-known very bright sources (the A-team, sources such as Cassiopeia A or Centaurus A) is demixed from the dataset. After these steps, the data is often integrated again in time and frequency, depending on frequency smearing (Bridle and Schwab, 1989) and other science requirements.

2.2.3 Science data processor (SDP). From the visibilities, a sky image can be constructed. The calibration works on a station/dish basis, and accounts for both direction-independent effect (gains, crosstalks) and direction-dependent effects, such as ionospheric distortion. The corrected visibilities together with the calibration solution are passed to the imaging pipeline. From a telescope model and the calibration parameters, we can create a small map representing the complex gain function for a beam, called A-projection (analog to the lens behavior in an optical camera) (Tasse et al., 2012). Two maps of a single baseline are multiplied together, and then multiplied with a W-term to account for the non-coplanar baselines effect (Cornwell et al., 2008) (the earth is not flat), and scaled up. The resulting map, or convolution matrix, is multiplied with a visibility and added (gridded) onto a Fourier grid. This gridding process happens in various time-steps, called W-snapshots (Cornwell et al., 2012). All snapshots are later refitted into a single grid. A Fourier transform of the grid results in a dirty image, and the CLEAN deconvolution algorithm (Högbom, 1974) is used to extract bright sky sources. After a certain threshold has been reached, the extracted sources are converted back into visibilities (de-gridding), which are subtracted from the original dataset. This gives us a visibilities dataset with only weak sources, and the gridding process starts over until only noise is left. All extracted sources are kept in a sky model. This sky model is used to make better estimations about certain calibration parameters, resulting in another feedback loop back into the calibration step.

After a sufficient amount of iterations, the sky model is converted into a sky image.

3. SKA1 computational profile

In this section we analyze the compute requirements of various applications in the SKA1 processing chain. The key element of this research are the scaling rules, or how the compute for an application relates to telescope-parameters. The analysis is based on work performed by Jongerius et al. (2014), internal project documents, and our own research. We would like to point out that the numbers in this section are estimations. Furthermore, some algorithms, like calibration, are missing, since they are at this moment not well defined enough to include.

3.1 Station processing application set

3.1.1 Channelization. The channelization consists of a fixed-size bank of FIR filters and a fixed-size (real to complex) FFT. The compute requirements per second per single-polarized antenna are given in equation (1). N_{taps} is the number of filter taps and N_{bands} is the number of frequency bands. $\text{Samples}_{\text{sec}}$ is the incoming sample rate.

$$\begin{aligned}\text{Ops}_{\text{channelization}} &= \text{Samples}_{\text{sec}} \\ &\times (2N_{\text{taps}} + 5 \times 0.5 \log_2 (2N_{\text{bands}}))\end{aligned} \tag{1}$$

3.1.2 Beamforming. The beamforming step multiplies every dual-polarized antenna sample with 2×2 matrix holding complex weights. Thus every sample undergoes 14 operations, 8 multiplications and 6 additions.

3.2 CSP application set

3.2.1 Channelization. The compute requirements per second per beam for this step is a variation on equation (1). In this case the inputs are complex numbers, and we generate frequency channels instead of frequency bands.

3.2.2 Correlation. The correlator multiplies two signals together and adds the result to a sum. The compute for the correlator per second per channel is given in equation (2). Here, N_{stat} is the number of stations or dishes.

$$\text{Ops}_{\text{correlator}} = 8 \times \text{Samples}_{\text{sec}} \times 0.5 \times N_{\text{stat}}^2 \tag{2}$$

An implementation challenge lies in the fact that data arrives per station, containing all the frequency channels of that station, whereas the algorithm wants the data per frequency channel, containing all the stations. This data rearrangement is often called the 'corner turn', and frustrates practical implementations of the correlator.

3.2.3 RFI removal. Based on experiences from LOFAR, we see that good RFI removal costs 278 operations per input sample (Offringa et al., 2010). This is not shown in an equation.

3.3 Imaging application set

3.3.1 Convolution matrix generation. Creating the convolution matrices as described in Section 2.2.3 involves several 2D FFTs and point-wise matrix multiplications. The compute is however dominated by a single 2D FFT, often not a power-of-two in size. The compute requirements per second per channel and baseline are given by equation (3). W is the average W-matrix size, and O is a scaling factor. S_{sec} indicates how many seconds the projection matrices are valid because of time dependent effects. C_{chan} is the channel compression, indicating how many channels can be served by the same W-A combination. N_{iter} is the number of iterations following the feedback loops described in Section 2.2.

$$\text{Ops}_{\text{Conv-matr-gen}} = 5 \times \frac{N_{\text{iter}}}{S_{\text{sec}} C_{\text{chan}}} \times W^2 O^2 \log_2 (W^2 O^2) \tag{3}$$

The up-scaling introduces a form of interpolation in the gridder. This is necessary because the location of the visibilities is of much higher precision than the Fourier plane gridpoints.

3.3.2 W-snapshots gridding. As de-gridding and gridding are very much the inverse of each other with the same kernels and properties, we will only focus on gridding in this section. Based on the location of the visibility with respect to the grid, a 1/64th subset of the convolution matrix is selected. The visibility is multiplied with this matrix and added to the Fourier plane. The compute per second per channel is given in equation (4). T_{dump} is the correlator dump time, and N_{bl} the number of baselines.

$$\text{Ops}_{\text{gridding}} = (6 + 2) \times \frac{N_{\text{iter}} N_{\text{bl}}}{T_{\text{dump}}} \times W^2 \tag{4}$$

A practical aspect of this algorithm is that additional parallelism in the baselines exists. However, exploiting that can be difficult because of the addition to a final grid, which has to happen in an atomic way.

3.3.3 W-snapshots re-projection and 2D FFT. Refitting a gridding snapshot consists of a coordinate transformation, and a 2D FFT. The re-projection is estimated to

be 50 operations per pixel per channel and is not shown in an equation. The compute requirements per channel for the FFT follow equation (5), where R is the amount of pixels in one dimension of the image. $\text{Snapshots}_{\text{sec}}$ indicates the amount of seconds a snapshot is valid.

$$\text{Ops}_{\text{snapshot-refitting}} = 5 \times \frac{N_{\text{iter}}}{\text{Snapshots}_{\text{sec}}} \times R^2 \log_2(R^2) \tag{5}$$

3.3.4 Deconvolution. For this application we assume Cotton-Schwab CLEAN (Schwab, 1984), together with the W-snapshots described earlier. An estimate for the compute requirements per channel for this algorithm are shown in equation (6). N_{mc} is the number of minor cycles, N_{pp} is the amount of pixels along one axis of the dirty-beam patch, N_{pi} is the amount of pixels along one axis of the image, N_{al} is the number of pixels in the active-list, and $\text{Measurement}_{\text{sec}}$ is the measurement time. A detailed explanation of these parameters is beyond the scope of this work, but can be found in Taylor et al. (1999).

$$\text{Ops}_{\text{deconvolution}} = 3 \times N_{\text{mc}} \times \frac{N_{\text{iter}}}{\text{Measurement}_{sec}} \times \left(\frac{N_{\text{pp}}}{N_{\text{pi}}}\right)^2 \times N_{\text{al}} \tag{6}$$

3.4 Compute requirements

With the analysis performed in the previous Sections, and the specific parameters of all the instruments, we can calculate the compute requirements. The result is shown in Figure 3. From this figure it is clear that several applications have very high compute requirements, up to several hundreds of peta-operations per second. To make this work as relevant as possible, we focus on four compute intensive kernels, namely:

- fir + 1D FFT (channelization), used in the SKA1-low stations and the CSP;
- correlation;
- 2d FFT, used in creating the convolution matrices, the W-snapshots re-projection, and several places not explicitly mentioned in this work;
- gridding.

In the remainder of this work we will discuss these four kernels.

3.4.1 Parallelism and datatypes in the applications. The compute requirements for SKA1 are high, but there are also trivial parallelizations possible. Already recognizable in the previous section, most SDP applications can be parallelized over frequency channels. These channels do not interact with each other, and they can therefore be processed completely independent of each other. Another parallelization possibility are the beams, which is especially relevant for SKA1-Survey.

Another interesting point of our application set is that the datatypes are not necessarily 32 or 64-bit floating-point. For example, the input for the correlator are 8-bit integers. The applications in the SDP do not have requirements like this, but it is clear that some steps need more precision and/or dynamic range than others. For example, the multiplication of a visibility with a convolution matrix can be done with a 'small' datatype, compared to the full Fourier plane.

3.5 Visibility buffer and dataflows

The visibility stream out of the correlator must be stored in a visibility buffer (or UV buffer). The various major and calibration cycles can then be executed on

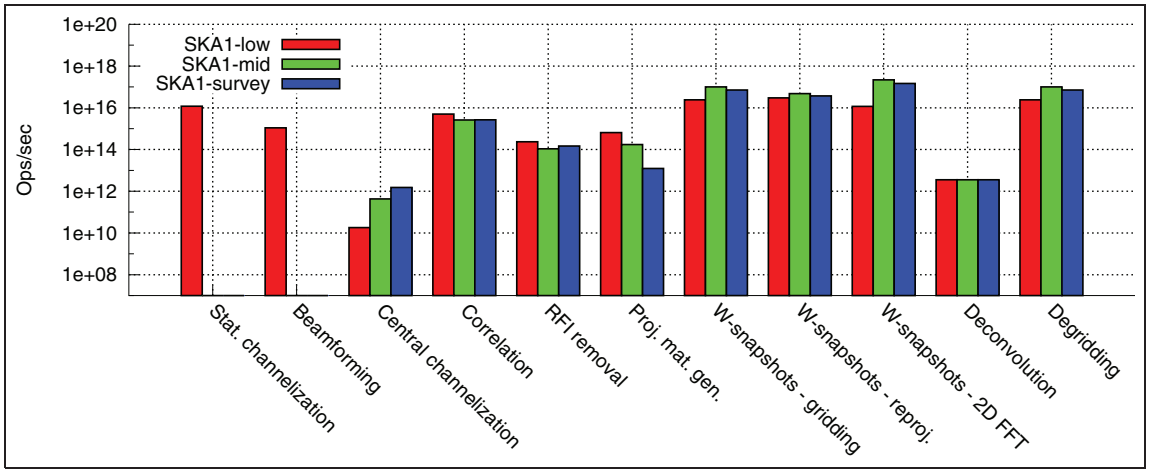

Figure 3. Compute requirement estimates for the various applications for the three SKA1 instruments.

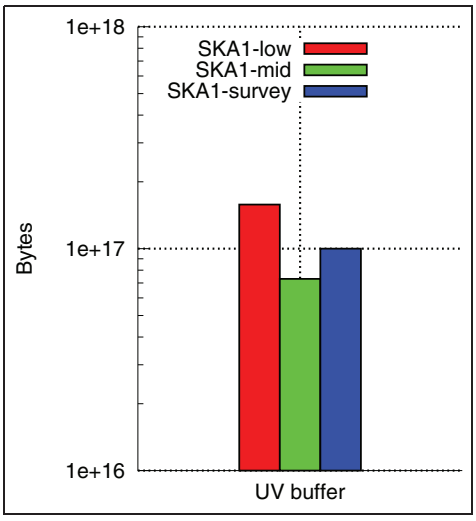

Figure 4. UV buffer requirements for the SKA1.

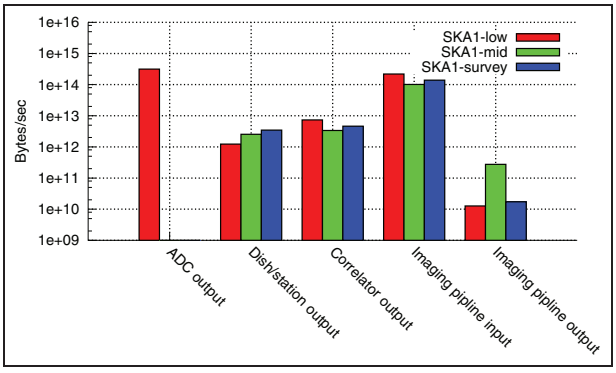

Figure 5. Bandwidths throughout the SKA1 instruments.

this dataset. The visibilities need to be stored in a ping-pong fashion, one buffer for the incoming data, and one buffer for the dataset currently being processed. Here we assume that the telescope will always be 'on'. In Figure 4 we show the amount of storage needed for the visibilities, for the three instruments of the SKA1. It can be observed that the buffer requirements are very high, over 100 PB.

The input and output bandwidths for several steps for the three SKA1 instruments are shown in Figure 5. For the SKA1-low we see a huge bandwidth reduction from the AD-converters to the station output, due to the beamforming, where we sum all the antennas together. For all the instruments, we see an increase in data rates in the correlator. This is due to the large number of stations/dishes we have in SKA1. Most existing, smaller, telescopes reduce the data rates in the correlator, due to the summation happening there. The input data rate for the imaging pipeline is again much higher than the correlator output because of the various calibration and major loops that need to happen here. Finally, the output data rates of the imaging process (a set of images), is much smaller, as a large stream of visibilities is converted into an image.

4. The SKA1 on today's technology

As shown in the preceding sections, SKA1 will require significant computational power. In this section, we will be analyzing state-of-the-art implementations of the key algorithms, and take a look at how we could optimize current technology for SKA1.

4.1 Core technologies

A decade ago, the first *multi-core* CPU was introduced, which sacrificed single-core peak performance for parallelism. With *manycore* architectures, this concept is taken to the extreme. Besides using more cores, *specialized* logic designed for a specific task can be added: using additional area is traded off for better power efficiency or throughput/latency (Venkatesh et al., 2010). Performance and/or power efficiency can be further improved by using *heterogeneity*. This can be done on a node level by using multiple types of devices (CPUs and GPUs for example), or within a single device (ARM® big.LITTLE (ARM, n.d.) for example). Another kind of heterogeneity comes in the form of attached FPGAs, which are *reconfigurable*. Examples of this are the Molen polymorphic processor (Vassiliadis et al., 2004), and the systems of Convey Computer (Convey Computer™, n.d.).

Most advances in power efficiency and throughput we see today are based on these technologies. For example, GPUs employ the manycore paradigm, have thousands of small cores, and are often used in a heterogeneous setup. CPUs become faster by adding wider and more specialized instructions (Intel, n.d.(b)) or small accelerators (Intel, n.d.(a)).

4.2 The SKA1 kernels on existing products

4.2.1 Channelization. The work performed by Shahbahrami et al. (2005) shows that FIR filters using real numbers are well suited for SIMD parallelization. For complex numbers additional data-shuffling hardware is required. Jongerius et al. (2012) show that a modern CPU can only run a real FIR filter at 10–15% of its peak performance, because of the low number of operations per byte of I/O. Romein (2013) shows that this also holds true for complex FIR filters on GPUs.

The FFT algorithm features an irregular data access pattern with low computational intensity, which make it hard to run this algorithm efficiently on almost any architecture. Jongerius et al. (2012) show that the FFTW library (Frigo and Johnson, 1998) reaches 17% of the peak performance of a modern CPU. The

research by Xu et al. (2011) shows in detail how the FFT can be optimized for SIMD processing on a modern CPU. Romein (2013) shows that FFTs on GPUs are heavily IO bound, and achieve around 20% of the peak performance, comparable with CPUs. Research by Lobeiras et al. (2011) and nVidia's own CUFFT library confirm this.

LOFAR shows us that using FPGAs on the Uniboard (Szomoru, 2011) is a good match for the fixed-sized FIR and FFT. The multiple bitwidth modes LOFAR can use are all supported on a single FPGA image.

For the channelization we can conclude that SIMD and SIMT models both work fine, and that the bottlenecks are in the memory bandwidth and memory access pattern.

4.2.2 Correlator. Nieuwpoort and Romein (2011) implemented the correlation algorithm on several architectures, namely CPUs, GPUs, the Blue Gene® /P (Romein et al., 2010), and the Cell processor. They conclude that having sufficient local storage is key to good performance. Both Cell and the Blue Gene® achieve close to peak performance, whereas CPUs and GPUs reach significantly lower numbers (70 and 40% respectively). Work by (Clark et al., 2011) and (Romein, 2013) shows that, for modern GPUs, a peak performance of up to 80% can be achieved for a large number of stations, provided that significant optimization effort is put into register and IO usage.[1]

Research by Woods (2010) and De Souza et al. (2007) shows that FPGAs are a suitable candidate for running the correlation algorithm, because of its regular and simple structure. Utilizing custom datawidths is a major, and realistic, advantage over other architectures.

In conclusion, the correlator runs well on most architectures, but a big local storage is important. Using custom datatypes gives an advantage.

4.2.3 2D FFT. A white paper published by Intel® (Intel, 2011) reports CPU utilizations ranging from 70% for a 64 × 64 dataset down to 50% for a 256 × 256 dataset, using their commercial math packages. For larger datasets the utilization drops, but one dimension is always kept at or below 256, which makes the results less valuable for this analysis. For the utilization on a GPU we run some small benchmarks. Using an nVidia® K20 GPU and CUDA® 6.0, we see rather flat utilizations in the range of 7.5 and 10%, for input sizes from 64 × 64 to 8192 × 8192. The datasets for 2D FFTs on full sky images (10,000–50,000 pixels across) will not fit on current day discrete accelerator devices, thereby frustrating efficient implementation.

Despite vendor-optimized codes, we see that (large) 2D FFTs do not run very well on CPUs and GPUs. Their computational intensity is somewhat better than 1D FFTs, but is still low.

4.2.4 Gridding. The naive way to implement this algorithm gives a very low computational intensity (Van Amesfoort et al., 2009), and a lot of atomic add operations. This is shown in Humphreys and Cornwell (2011) for CPUs and GPUs. Both platforms achieve around 5% of their peak performance because of bandwidth limitations. An implementation on the Cell (Varbanescu et al., 2009) achieves around 25% of the peak performance, but only after considerable optimizations in the memory bandwidth usage. GPU research performed in Romein (2012) tries to eliminate atomic add instructions as much as possible, and the GPU reaches 25% of its peak performance. Even when there are hardly any atomic add operations left (0.23% of all grid updates), they still take up 26% of the time.

For the gridding, we can conclude that the simple kernel will run well on most architectures; bottlenecks are in the memory bandwidth and atomic additions.

4.3 Hints towards optimized architectures

This section presents and analyzes two aspects of the key SKA1 algorithms: first, the power breakdown on CPUs and GPUs, and second, the cache performance on CPUs. Together with the work presented in Section 4.2, these analyses can be seen as a starting point for research into architectures that can run the algorithms at higher throughputs and with improved power efficiency. The CPU tests are run on a model of an Intel® Xeon® E5-2630 implemented by using the Gem5 simulator (Binkert et al., 2011) and McPAT (Li et al., 2009), while the GPU tests are run on a generic model of a NVIDIA® GTX480 GPU implemented by using an enhanced version of GPUWattch (Leng et al., 2013), a microarchitecture-level GPU power simulator based on GPGPU-Sim (Bakhoda et al., 2009) and McPAT. This GPU architecture is not the most modern one, and is already superseded by two architectures. The general organization of GPUs has however not changed, therefore we believe the results shown are still relevant. The CPU performance numbers are obtained by using the Intel® Vtune™ Amplifier and again a Xeon® E5-2630 processor.

For the channelization, we used the algorithm described by Romein (2013). The correlator implementations are based on code presented by Nieuwpoort and Romein (2011), and the gridding implementation described by Romein (2012). For the 2D FFT kernel, we used a 128 × 128 complex matrix as input. This is a realistic size for the convolution matrix generation, but somewhat small for the *W*-snapshot refitting.

4.3.1 Power breakdown. Figure 6 shows the results of our power breakdown experiments. As can be seen, the CPU shows very similar power distributions for all the algorithms. Only a very small portion of the energy

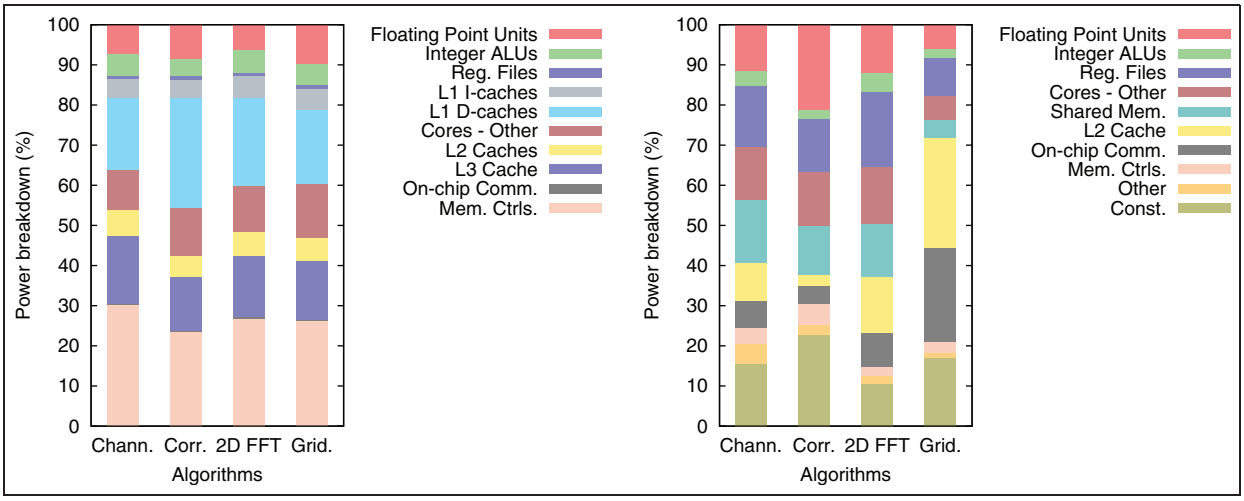

Figure 6. Power breakdown on CPUs (*left*) and GPUs (*right*) for the four SKA1 algorithms considered in this work: channelization (Chann.), correlator (Corr.), 2D FFT, and gridding (Grid.).

Table 1. CPU cache performance and ALU utilization for the four SKA1 algorithms considered in this work. Here 'Chann.' is the station channelization.

	Chann. (%)	Correlator (%)	2D FFT (%)	Gridding (%)
Cycles under L1$ miss	29	32	46	1
Cycles under L1$ miss, L2$ hit	3	29	26	1
Cycles under L2$ miss, L3$ hit	4	2	7	0
Cycles under L3$ miss	23	1	13	0

goes to the actual computations. Around 50% of the energy goes to the three levels of cache, and another 20 to 30% is spent in the memory controllers. Although not shown in the graph, almost all the power for the level two and level three caches is due to the static power component. For the level one cache, the static and dynamic power distribution is about equal. This means that a significant part of a CPUs power usage is due to the presence of these big caches, which consume energy even when they are idle. From the graph it is possible to notice the slightly higher usage of the memory controllers in the channelization, due to the streaming nature of the algorithm. Moreover, the correlator uses more power in the level one cache, which corresponds with its intensive use of a local storage, as indicated in Section 4.2.2.

The breakdown for the GPU shows a bigger power percentage for the functional units than the CPU, as one would expect with GPUs being compute oriented platforms. This difference is especially clear when running a compute-bound algorithm like the correlator. For the 2D FFT the difference is much smaller. This shows that compute bound problems run more efficiently on a GPU, and this is of course the reason why they are used in HPC. Another interesting observation is the large amount of energy GPUs spend in the register file (especially compared to CPUs), around 15%. This corresponds with the fact that the register file in a GPU is large, and can be compared with the level-one cache in CPUs. The constant power (around 15%) is mainly caused by processor and memory leakage power and peripheral circuits' power (Leng et al., 2013). In the case of GPUs, it can be noticed that we have a high power consumption of the floating point units while running the correlator, as we would expect for a compute bound problem. Furthermore, the 2D FFT consumes a lot of energy in the local storage (register files, shared memory and level two cache), corresponding with the large datasets it has to process in a relatively inefficient way.

4.3.2 CPU cache performance. The results for the CPU cache performance are shown in Table 1, and correspond with what we know about the algorithms. The channelization does not use the level two and level three cache: data is streamed from external RAM into level one, processed and evicted again. For the correlator we know that it appreciates large amounts of local storage, this is reflected in the numbers: level one misses almost always hit in level two. The 2D FFT behaves in the same way as the channelization, but also uses the level

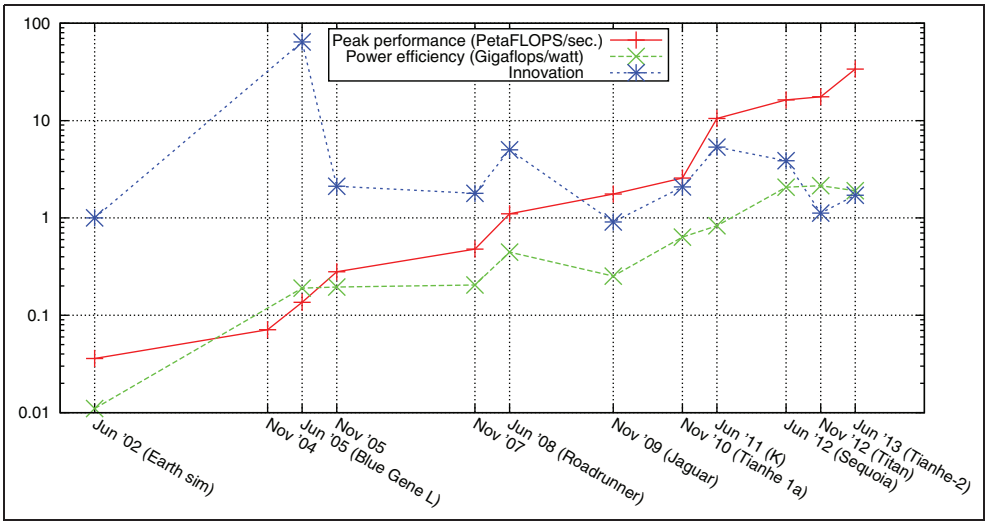

Figure 7. Trends for the new number one systems in the Top 500. Innovation is explained in Section 5.0.6.

two cache, since it uses larger datasets. The gridding exhibits very good cache behavior.

4.3.3 Considerations. From these numbers, we can extract some general guidelines on how to improve the power efficiency and throughput of these four algorithms.

GPUs spend a significant amount of energy in the floating point units, the register file and the rest of the core. We see a similar picture for the corresponding CPU categories. Adding macro instructions to the core-architecture for executing FFT butterflies (beyond existing shuffle instructions), complex multiplications, or even bigger instruction blocks, would result in less register file accesses and more efficient execution units. This could reduce the power consumption and improve the throughput. Furthermore, core-architectures could be optimized to capture the simplicity and regularity of the algorithms. The correlator for example, has basically a single type of instruction (complex multiply accumulate) inside several loops, but still, when running this algorithm on a CPU, 5% of the power goes to the instruction cache. Exploiting this regularity can also improve the power efficiency and throughout.

Besides the computational part of the chips, the local memories use large amounts of energy, and are often also a performance bottleneck. Making the memories deeper and wider will make performance better, but the energy consumption worse. A possible improvement would be to be able to exploit the regularity and predictability of the data needs. With carefully arranged accesses, specific to our SKA1 algorithms, we could utilize the available bandwidth to the largest extent. Similarly an improved, more SKA1 specific, organization of the local memories would increase data locality, reducing the number of accesses to a lower level or the external memory. The numbers presented in Table 1 and the power breakdown observations in Section 4.3.1 can serve as a guideline here.

5. HPC trends

In this section, we will look at several HPC trends, foremost based on the Top500 (n.d.) and Green 500 (n.d.). With the petaFLOPS (PFLOPS) well achieved, the high-performance computing community is looking at the next big step: exaFLOPS of sustained Linpack performance. In this section, we use the data from the June 2014 lists.

5.0.4 Peak performance analysis

Figure 7 shows the measured performance of the number one systems of the Top 500 for the past decade. Note we only include new number one systems: including the number one from every list would make the figure less clear, while not changing the result. It can be seen that on a logarithmic scale, this dataset fits a straight line, which goes back all the way back to 1993. The performance of the number one systems increases with roughly a factor of two every year. These performance projections suggest that it could be possible to build an exaFLOP system in 2018.

5.0.5 Power efficiency lags behind peak performance

Figure 7 shows, in FLOPS per watt, the power efficiency of the system. Also in this case, we can observe a straight line. The slope of the power efficiency is lower than that of the peak performance, meaning that peak performance is growing exponentially faster than power efficiency. This means that new systems use exponentially more absolute power than their predecessors. We

can generalize these Linpack numbers for general-purpose workloads (Kamil et al., 2008): power bills are becoming higher and cooling will become more impractical. When the practical absolute power usage limit is reached in the not so distant future, the growth in peak performance will have to slow down to follow the trend in power efficiency.

5.0.6 Innovation analysis

Every new number one supercomputer is faster than its predecessor, and, most of the time, also more power efficient. We are interested in the level of innovation that every new generation brings, i.e. how much closer it brings us to high-performance and power-efficient computing. To investigate this, we create the Innovation metric, shown in Equation (7). Perf, Eff and n are the measured peak performance, power efficiency and system index, respectively. The results are shown in Figure 7. The higher this number the better.

$$\text{Innovation}_n = \frac{\text{Perf}_n}{\text{Perf}_{n-1}} \times \frac{\text{Eff}_n}{\max(\text{Eff}_0 : \text{Eff}_{n-1})} \quad (7)$$

Over the past decade, an interesting observation is that, out of all the systems, the best scoring ones are all based on custom-designed hardware (Blue Gene®/L, PowerXCell®, K, Blue Gene® /Q). The lower scoring systems are all based on commercial off-the-shelf (COTS) products, or are extensions of existing systems. This analysis shows us that although the use of COTS products is convenient for many reasons, they apparently do not give us the big steps forward we need.

5.1 Heterogeneity and power efficiency in HPC

In recent years, accelerators have gained in popularity in supercomputers. These products are often celebrated for their power efficiency and peak performance.

For power efficiency, we look at the Green 500: the Top 500 sorted by power efficiency. The current, GPU based and oil-immersed, green number one system achieves an impressive 4.38 GFLOPS/W (1.9 for the Top 500 number one), but is a factor 157 slower in peak performance. This difference in peak performance makes comparing the efficiency hard, because small systems can always be more power efficient than big ones, because of less communication overhead and other scaling rules.

The first big supercomputer in the green list is the Piz Daint. This system is number six in the Top 500 (6.2 PFLOPS), and number four in the Green 500 (3.18 GFLOPS/W). This shows that it is possible to build big heterogeneous supercomputers, with around 50% better performance per watt than the homogeneous Blue Gene® /Q systems. These results might, however, not hold for a wider set of benchmarks. This is reflected in Dongarra (n.d.), noting that Linpack no longer represents real workloads, which are often not as GPU-friendly as the DGEMM kernel in Linpack.

In the Top 500, only a little over 10% of the systems use accelerators, a number that hardly changed in the last four lists. Important factors for this might be the extra development effort they require, the narrower application space that will run well on them, and the only modest improvement in peak performance (Lee et al., 2010) and power efficiency as shown above.

5.1.1 Future of heterogeneity. Although the acceptance of accelerators as we know them today is still slow (or even halted if we look at the Top 500), we have to realize that specialization is, besides integration, one of the few things left to increase the performance of (silicon based) computer systems. Thus, the concept of accelerators is very valid, and in the upcoming years we will see two trends emerging: one, the host—PCIe—accelerator model will change, to remove the PCIe bottleneck, give accelerators fast access to large amounts of memory, and remove the master–slave execution model. And two, a much deeper form of specialization, towards workload-optimized systems instead of a one-size-fits-all approach. In this direction, both Microsoft® (Putnam et al., 2014) and Google® (2014) recently announced (partly) custom hardware to drive their data-centers.

6. Technology Challenges For the SKA1

6.1 Traditional walls

The traditional technology walls have been power, memory, and instruction-level parallelism. Dennard scaling (Dennard et al., 2007) no longer works; thus, with every technology scaling of S, the amount of transistors that can be switched on while staying within the power envelope shrinks by a factor of S (Esmaeilzadeh et al., 2011). Furthermore, moving data costs a lot of power: moving data from external RAM to the compute core can be a factor of 1000 more expensive than doing a computation on that data (Keckler et al., 2011).

Regarding memory, we see that compute performance is growing exponentially faster than bandwidth (Wulf and McKee, 1995), and latency lags bandwidth quadratically (Patterson, 2005). This means that getting enough data to feed the cores on time is becoming increasingly hard. As an example, for nVidia's® Tesla® product line, the bandwidth-to-compute ratio has worsened by a factor of 2.15 in the past 5 years (C870 to K40) (Cook, 2012).

Lastly, instruction-level parallelism is limited in every real application, thereby limiting the return on investments regarding wider execution machines (Hennessy and Patterson, 2006).

6.2 SKA1 challenges

For SKA1, the power and memory walls can be summarized into a data-locality problem. The instrument generates so much data (Section 3.5), that it is impossible to move the data to an appropriate processing device. As a node-level example: a modern PCIe-attached device needs hundreds of operations per incoming byte to run at a high utilization. We do not have that amount of operations, as indicated in Section 4. Between nodes we see a similar problem: a significant portion of supercomputing power goes to the interconnect, and with the rapidly increasing node peak performance, data communication cannot keep up. Managing where the data is with respect to the compute resources will be the key challenge.

As discussed in the introduction, power efficiency is of special importance to the SKA1, since several subsystems will be located in remote areas. Power will have to be transported there or generated on the spot. This gives another dimension to the power efficiency challenge.

7. Can SKA1 ride the technology wave?

7.1 At node level

The algorithms discussed and analyzed in Sections 3 and 4 exhibit features that do not map well onto existing technologies. In Section 4.3, we already briefly discussed some of them: predictable data-access patterns, high bandwidth-to-compute ratio, and complex arithmetic. Other exploitable features are narrow and/or flexible datawidths and simple highly repetitive kernels.

Given the challenging requirements of the SKA1, we cannot afford to either waste computational resources or energy, or not to benefit from some specific features the algorithms offer us. An extra level of specialization, such as the industry examples indicated in Section 5.1.1, will be needed.

7.2 On system level

In Figure 8, we show the compute versus power efficiency operation points for the several subsystems of SKA1. The SKA1 systems should be built towards the end of this decade. Power requirements are found in the SKA1 baseline design, and the compute is based on Section 3. The graph furthermore shows several point sets and a trend based on several Top 500 top-10 systems. We would like to emphasize that our kernels will run on a lower utilization than the Linpack kernel for the Top/Green 500 datapoints. For this figure we do not use the latest Top 500 list, since the number 10 in that list (the only new one), does not have a reported power number.

Two observations can be made. First, the power efficiency of the subsystems are orders of magnitude apart. This means that the power budgets are not balanced very well. As illustrated we also included a weighted power efficiency in Figure 8. Secondly, it is clear that riding the technology wave and waiting for systems available at the end of the decade will not give us the hardware needed to realize some subsystems (foremost the SDPs) of SKA1.

Given the data challenges indicated in Section 6, novel data-centric approaches should be considered. A high-level example of this could be to split the system physically into frequency channels (as described in Section 3.4). Data can stay at rest within a 'channel-node', while the node does all consecutive processing steps. This would significantly reduce the load on the system interconnect, and thereby reduce the power consumption. With such an approach, we would end up with a true data-centric supercomputer.

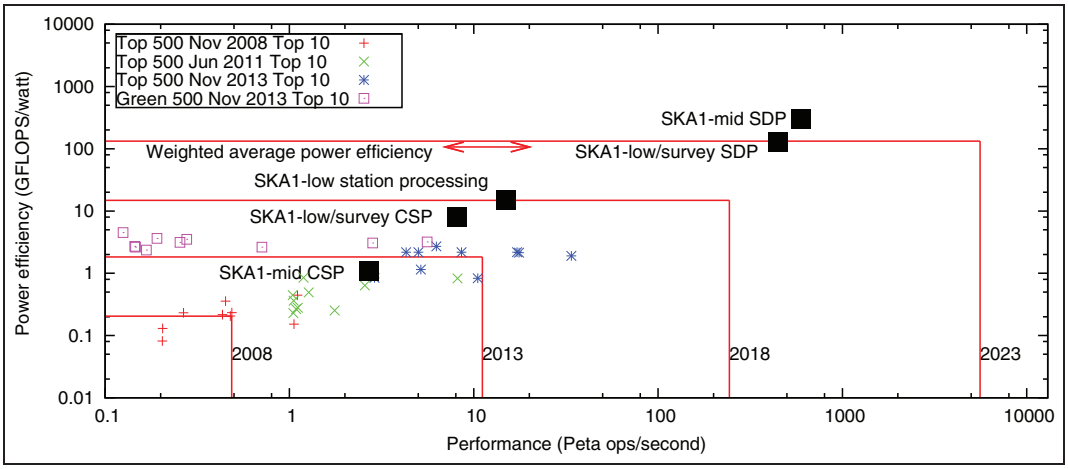

Figure 8. The compute and power efficiency points for the SKA1 subsystems.

8. Conclusion—Realizing the SKA1

In this paper, we presented an overview of SKA1 from a compute perspective, and analyzed whether the instrument can be built using COTS hardware in 2018. Three observations can be made: (1) SKA1 will require large amounts of computational power at a very high power efficiency. (2) Most SKA1 algorithms will only run at efficiencies of around 1 to 50% on COTS hardware, with clear points for improvement being smarter memory accesses and specialized execution units. (3) The HPC innovation and trend analysis shows that custom-build machines achieve two to three times larger improvement in compute capabilities *and* power efficiency than systems employing COTS hardware.

We conclude that waiting for new technology to arrive will not give us the instruments currently planned at the end of the decade: an order of magnitude better power efficiency and compute capabilities are required. Developing new, workload specific technology, specialized for the tasks at hand must be considered. Solving the data-locality problem will be key: there should always be a suitable compute element near the data, on every level.

Acknowledgement

This work is conducted in the context of the joint ASTRON and IBM DOME project.

Note

1. Blue Gene is a trademark of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product or service names may be trademarks or service marks of IBM or other companies.

Funding

This work was supported by the Netherlands Organization for Scientific Research (NWO), the Dutch Ministry of EL&I, and the Province of Drenthe, the Netherlands.

References

ARM (n.d.) big.LITTLE processing. Available at: http://www.arm.com/products/processors/technologies/biglittleprocessing.php.

Bakhoda A, Yuan GL, Fung WWL, et al. (2009) Analyzing CUDA workloads using a detailed GPU simulator. In: *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS, Boston, 26-28 April 2009)*, pp.163–174. doi:10.1109/ISPASS.2009.4919648.

Binkert N, Beckmann B, Black G, et al. (2011) The gem5 simulator. *SIGARCH Computer Architecture News* 39(2): 1–7. doi:10.1145/2024716.2024718.

Bridle AH and Schwab FR (1989) Wide field imaging I: Bandwidth and time-average smearing. *Synthesis Imaging in Radio Astronomy* 6: 247.

Clark MA, Plante PCL and Greenhill LJ (2011) Accelerating radio astronomy cross-correlation with graphics processing units. *Computing Research Repository* abs/1107.4264.

Convey Computer™ (n.d.) Convey Computer website. Available at: http://www.conveycomputer.com.

Cook S (2012) *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*. 1st edition. San Francisco, CA: Morgan Kaufmann Publishers.

Cornwell T, Golap K and Bhatnagar S (2008) The noncoplanar baselines effect in radio interferometry: The W-Projection algorithm. *IEEE Journal of Selected Topics in Signal Processing* 2(5): 647-657.

Cornwell TJ, Voronkov MA and Humphreys B (2012) Wide field imaging for the square kilometre array. *Proceedings of SPIE 8500, Image Reconstruction from Incomplete Data VII* 8500L: 1–12. doi:10.1117/12.929336.

De Souza L, Bunton J, Campbell-Wilson D, et al. (2007) A radio astronomy correlator optimized for the Xilinx Virtex-4 SX FPGA. In: *Proceedings of the international conference on field programmable logic and applications*, Amsterdam, 27-29 August, pp.62–67. doi:10.1109/FPL.2007.4380626.

Dennard R, Gaensslen F, and Yu HN (2007) Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Solid-State Circuits Society Newsletter* 12(1): 38–50. doi:10.1109/N-SSC.2007.4785543.

Dongarra J (n.d.) HPCG benchmarking. Available at: http://www.sandia.gov/maherou/docs/HPCG-Benchmark.pdf.

Esmaeilzadeh H, Blem E, St Amant R, et al. (2011) Dark silicon and the end of multicore scaling. *SIGARCH Computer Architecture News* 39(3): 365–376. doi:10.1145/2024723.2000108.

Frigo M and Johnson SG (1998) FFTW: An adaptive software architecture for the FFT. In: *Proceedings of the 1998 IEEE international conference on acoustics speech and signal processing*, Vol. 3, Seattle, 12–15 May, pp.1381–1384.

Google® (2014) Custom motherboard announcement. Available at: https://plus.google.com/111282580643669107165/posts/Uwh9W3XiZTQ.

Green 500 (n.d.) Green500 website. Available at: http://www.green500.org/.

Hennessy JL and Patterson DA (2006) *Computer Architecture: A Quantitative Approach*. 4th edition. San Francisco, CA: Morgan Kaufmann Publishers.

Högbom JA (1974) Aperture synthesis with a non-regular distribution of interferometer baselines. *Astronomy and Astrophysics Supplement* 15: 417.

Humphreys B and Cornwell T (2011) Analysis of convolutional resampling algorithm performance. Available at: http://www.skatelescope.org/uploaded/59116_132_memo_humphreys.pdf.

Intel (n.d.(a)) Intel random number generator. Available at: http://software.intel.com/sites/default/files/m/d/4/1/d/8/441_Intel_R__DRNG_Software_Implementation_Guide_final_Aug7.pdf.

Intel (n.d.(b)) Intel SSE and AVX extensions. Available at: http://software.intel.com/en-us/intel-isa-extensions.

Intel (2011) Signal processing on Intel® architecture: Performance analysis using Intel® performance primitives. Technical report, Intel®. Available at: http://www.intel.nl/content/dam/doc/white-paper/signal-processing-on-intel-architecture.pdf.

Jeffs B (n.d.) Beamforming presentation. Available at: http://ens.ewi.tudelft.nl/Education/courses/et4235/Beamforming.pdf.

Jongerius R, Corporaal H, Broekema C, et al. (2012) Analyzing LOFAR station processing on multi-core platforms. In: *Proceedings of the ICT Open 2012*, 22–23 October 2012, pp. 71-76. Rotterdam, The Netherlands,. Available at: http://www.ictopen2013.nl/content/proceedings + 2012.

Jongerius R, Wijnholds S, Nijboer R, et al. (2014) An End-to-End Computing Model for the Square Kilometre Array. *IEEE Computer* 47(9).

Kamil S, Shalf J and Strohmaier E (2008) Power efficiency in high performance computing. In: *Proceedings of the IEEE international symposium on parallel and distributed processing*, Miami, 14–18 April, pp.1–8. doi:10.1109/IPDPS.2008.4536223.

Keckler S, Dally W, Khailany B, et al. (2011) GPUs and the future of parallel computing. *IEEE Micro* 31(5): 7–17. doi:10.1109/MM.2011.89.

Lee VW, Kim C, Chhugani J, et al. (2010) Debunking the 100 × GPU vs. CPU myth: An evaluation of throughput computing on CPU and GPU. *SIGARCH Computer Architecture News* 38(3): 451–460. doi:10.1145/1816038.1816021.

Leng J, Hetherington T, ElTantawy A, et al. (2013) GPUWattch: Enabling energy optimizations in GPGPUs. In: *Proceedings of the 40th annual international symposium on computer architecture*, Tel-Aviv, 23–27 June. doi:10.1145/2485922.2485964.

Li S, Ahn JH, Strong R, et al. (2009) McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In: *Proceedings of the 42nd annual international symposium on microarchitecture*, New York, 12–16 December, pp.469–480.

Lobeiras J, Amor M and Doallo R (2011) FFT implementation on a streaming architecture. In: *Proceedings of the 19th Euromicro international conference on parallel, distributed and network-based processing (PDP)*, Ayia Napa, 9–11 February, pp.119–126. doi:10.1109/PDP.2011.31.

Nieuwpoort R and Romein J (2011) Correlating radio astronomy signals with many-core hardware. *International Journal of Parallel Programming* 39(1): 88–114. doi:10.1007/s10766-010-0144-3.

Offringa AR, de Bruyn AG, Zaroubi S, et al. (2010) A LOFAR RFI detection pipeline and its first results. In: Groningen U (ed.) *RFI mitigation workshop—RFI 2010, Groningen. Proceedings of Science*, POS(RFI2010)036. SISSA.

Patterson PD (2005) Latency lags bandwidth. In: *Proceedings of the 2005 international conference on computer design.*, San Jose, 2–5 October. pp.3–6. doi:10.1109/ICCD.2005.67.

Perley RAE (1984) A proposal for a large, low frequency array located at the VLA site. *VLA Scientific Memorandum*: 146.

Putnam A, Caulfield A, Chung E, et al. (2014) A reconfigurable fabric for accelerating large-scale datacenter services. In: *Proceedings of the 41st annual international symposium on computer architecture*, Minneapolis, 14–18 June, pp. 13–24. DOI: 10.1109/ISCA.2014.6853195. Available at http://research.microsoft.com/apps/pubs/default.aspx?id=212001.

Romein J (2013) Signal processing on GPUs for radio telescopes. In: *Proceedings of the GPU technology conference*. San Jose, 18-21 March.

Romein JW (2012) An efficient work-distribution strategy for gridding radio-telescope data on GPUs. In: *Proceedings of the ACM international conference on supercomputing*, Venice, Italy, 25–29 June pp.321–330.

Romein JW, Broekema PC, Mol JD, et al. (2010) The LOFAR correlator: Implementation and performance analysis. In: *Proceedings of the 15th ACM SIGPLAN symposium on principles and practice of parallel programming*, Bangalore, 9–14 January, pp.169–178. doi:10.1145/1693453.1693477.

Schwab FR (1984) Relaxing the isoplanatism assumption in self-calibration; Applications to low-frequency radio interferometry. *Astronomical Journal* 89: 1076–1081.

Shahbahrami A, Juurlink B and Vassiliadis S (2005) Efficient vectorization of the FIR filter. In: *Proceedings of the 16th annual workshop on circuits, systems and signal processing*, Veldhoven, 17–18 November, pp.432–437.

SKA Organisation (n.d.) Square kilometer array. Available at: http://www.skatelescope.org/.

SKA Organisation (2013) SKA1 Baseline design. Available at: https://www.skatelescope.org/wp-content/uploads/2012/07/SKA-TEL-SKO-DD-001-1_BaselineDesign1.pdf.

SKA Organisation (2014) SKA phase 1 science (level 0) requirements specification. Available at: https://www.skatelescope.org/wp-content/uploads/2014/03/SKA1-Level0-Requirements.pdf.

Szomoru A (2011) The UniBoard: A multi-purpose scalable high-performance computing platform for radio-astronomical applications. In: *Proceedings of the URSI General Assembly and Scientific Symposium*, Istanbul, 13–20 August, pp.1–4. doi:10.1109/URSIGASS.2011.6051281.

Tasse C, van der Tol B, van Zwieten J, et al. (2012) Applying full polarization A-Projection to very wide field of view instruments: An imager for LOFAR. *Astronomy & Astrophysics*, Article number A105, Volume 553, May 2013. Available at: http://arxiv.org/abs/1212.6178.

Taylor GB, Carilli CL and Perley RA (eds) (1999) *Synthesis Imaging in Radio Astronomy II*Astronomical Society of the Pacific Conference Series, Vol. 180). (San Francisco, CA: Astronomical Society of the Pacific.

Thompson AR, Moran JM and Swenson GW (2001) *Interferometry and Synthesis in Radio Astronomy*. 2nd edition. Weinheim, Germany: Wiley-VCH.

Top500 (n.d.) Top500 website. Available at: http://www.top500.org/.

Van Amesfoort AS, Varbanescu AL, Sips HJ, et al. (2009) Evaluating multi-core platforms for HPC data-intensive kernels. In: *Proceedings of the 6th ACM conference on computing frontiers*, Ischia, 18–20 May, pp.207–216. doi:10.1145/1531743.1531777.

Van Haarlem M, Wise M, Gunst A, et al. (2013) LOFAR: The LOw-Frequency ARray. *Astronomy and Astrophysics*. 556(August 2013). Article number A2.

Varbanescu AL, van Amesfoort AS, Cornwell T, et al. (2009) Building high-resolution sky images using the Cell/B.E. *Science of Computer Programming* 17(1–2): 113–134. Available at: http://dl.acm.org/citation.cfm?id=1507443.1507454.

Vassiliadis S, Wong S, Gaydadjiev G, et al. (2004) The MOLEN polymorphic processor. *IEEE Transactions on Computers* 53(11): 1363–1375. doi:10.1109/TC.2004.104.

Venkatesh G, Sampson J, Goulding N, et al. (2010) Conservation cores: Reducing the energy of mature computations. *SIGARCH Computer Architecture News* 38(1): 205–218. doi:10.1145/1735970.1736044.

Woods A (2010) *Accelerating software radio astronomy FX correlation with GPU and FPGA co-processors*. Master's Thesis, University of Cape Town, South Africa. http://books.google.nl/books?id=ANHKXwAACAAJ.

Wulf WA and McKee SA (1995) Hitting the memory wall: Implications of the obvious. *SIGARCH Computer Architecture News* 23(1): 20–24. doi:10.1145/216585.216588.

Xu W, Yan Z and Shunying D (2011) A high performance FFT library with single instruction multiple data (SIMD) architecture. In: *Proceedings of the international conference on electronics, communications and control (ICECC)*, Ningbo, 9–11 September, pp.630–633. doi:10.1109/ICECC.2011.6066463.

Author biographies

Erik Vermij is a predoctoral researcher at IBM Research. He is pursuing his PhD at the Delft University of Technology, where he earlier received his BSc and MSc in electrical engineering and computer engineering. His research interests include high-performance computing, computer architecture, and reconfigurable computing.

Leandro Fiorin received an MS degree in electronic engineering from the University of Cagliari, Italy, a Master of Engineering degree in embedded system design from the University of Lugano (USI), Switzerland, and a PhD degree from the Faculty of Informatics USI, in 2001, 2004, and 2012, respectively. He is currently research scientist at IBM Research, at the ASTRON and IBM Center for Exascale Technology, the Netherlands. Previously, he was a research associate at the Advanced Learning and Research Institute (ALaRI) on Embedded System Design, USI, working on networks-on-chip and embedded systems architectures. His research interests focus on energy efficient computing architectures, fault-tolerant and secure networks-on-chip and embedded systems, on-chip multiprocessors, and reconfigurable systems. He is coauthor of several scientific papers on networks-on-chip, design methodologies for systems-on-chip, and embedded system security, and of two patents on networks-on-chip security. He is a member of the IEEE.

Rik Jongerius is a predoctoral researcher at IBM Research and a PhD candidate at the department of Electrical Engineering at the Eindhoven University of Technology. His research interests include high-performance computing, computer architecture, and accelerators. He received his MSc degree in Electrical Engineering from the Eindhoven University of Technology in The Netherlands. He is a student member of the IEEE.

Christoph Hagleitner currently manages the accelerator technologies group at the IBM Research Zurich Lab in Ruschlikon, Switzerland. The research includes all aspects of hardware accelerators from system architecture and compilers to new nano-electro-mechanical (NEM) devices and circuits. He obtained a diploma degree and a PhD degree in Electrical Engineering from the Swiss Federal Institute of Technology (ETH), Zurich, in 1997 and 2002, respectively. During his PhD work, Christoph specialized in interface circuitry and system aspects of CMOS integrated micro- and nano-systems. He worked on CMOS integrated probes for parallel AFM imaging and chemical sensors. After receiving his PhD degree for a thesis on a CMOS single-chip gas detection system, he headed to the circuit-design group of the Physical Electronics Laboratory at ETH Zurich. In 2003 he joined IBM Research, where he worked on the system architecture and analog-frontend design of a novel probe-storage device (the 'Millipede' Project). Since 2008, he manages the accelerator technologies team. Christoph has served as a TPC member for the ISSCC as well as the ESSCIRC conference, and has authored and co-authored six book chapters and 80 plus papers in journals and conference proceedings.

Koen Bertels is a Professor of computer engineering and leads the CE lab at Delft University of Technology, The Netherlands. His research interests focus on heterogeneous multicore architectures with an emphasis on system design aspects and programmability issues involving retargetable compilers.