The LOFAR Beam Former: Implementation and Performance Analysis

Jan David Mol and John W. Romein

Stichting ASTRON (Netherlands Institute for Radio Astronomy)
Oude Hoogeveensedijk 4, 7991 PD Dwingeloo, The Netherlands
{mol,romein}@astron.nl

Abstract. Traditional radio telescopes use large, steel dishes to observe radio sources. The LOFAR radio telescope is different, and uses tens of thousands of fixed, non-movable antennas instead, a novel design that promises groundbreaking research in astronomy. The antennas observe omnidirectionally, and sky sources are observed by signal-processing techniques that combine the data from all antennas.
Another new feature of LOFAR is the elaborate use of software to do signal processing in real time, where traditional telescopes use custom-built hardware. The use of software leads to an instrument that is inherently more flexible. However, the enormous data rate (198 Gb/s of input data) and processing requirements compel the use of a supercomputer: we use an IBM Blue Gene/P.
This paper presents a collection of new processing pipelines, collectively called the beam-forming pipelines, that greatly enhance the functionality of the telescope. Where our first pipeline could only correlate data to create sky images, the new pipelines allow the discovery of unknown pulsars, observations of known pulsars, and (in the future), to observe cosmic rays and study transient events. Unlike traditional telescopes, we can observe in hundreds of directions simultaneously. This is useful, for example, to search the sky for new pulsars. The use of software allows us to quickly add new functionality and to adapt to new insights that fully exploit the novel features and the power of our unique instrument. We also describe our optimisations to use the Blue Gene/P at very high efficiencies, maximising the effectiveness of the entire telescope. A thorough performance study identifies the limits of our system.

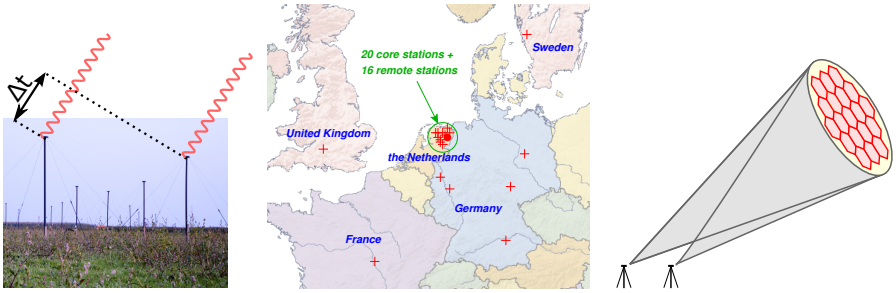
1 Introduction

The LOFAR (LOw Frequency ARray) telescope is the first of a new generation of radio telescopes. Instead of using a set of large, expensive dishes, LOFAR uses many thousands of simple antennas. Every antenna observes the full sky, and the telescope is aimed through signal-processing techniques. LOFAR's novel design allows the telescope to perform wide-angle observations as well as to observe in multiple directions simultaneously, neither of which are possible when using traditional dishes. In several ways, LOFAR will be the largest telescope in the world, and will enable groundbreaking research in several areas of astronomy and particle physics [1].

Another novelty is the elaborate use of software to process the telescope data in real time. Previous generations of telescopes depended on custom-made hardware to combine data, because of the high data rates and processing requirements. The availability

of sufficiently powerful supercomputers however, allow the use of software to combine telescope data, creating a more flexible and reconfigurable instrument. Because LOFAR is driven by new science, flexibility in the design is essential in order to explore the possibilities and limits of our telescope.

For processing LOFAR data, we use an IBM BlueGene/P (BG/P) supercomputer. The LOFAR antennas are grouped into stations, and each station sends its data (up to 198 Gb/s for all stations) to the BG/P. Inside the BG/P, the data are split and combined using both real-time signal-processing routines as well as two all-to-all exchanges. The output data streams are sufficiently reduced in size in order to be able to stream them out of the BG/P and store them on disks in our storage cluster.



(a) The left antenna receives the wave later.

(b) Locations of the stations.

(c) Tied-array beams (hexagons) formed within two station beams (ellipse).

Fig. 1. LOFAR antennas

In this paper, we will present the LOFAR *beam former*: a collection of software pipelines that allow the LOFAR telescope to be aimed at hundreds of directions simultaneously. This feat is made possible by LOFAR's unique design and the resources offered by the BG/P. Traditional radio telescopes are aimed by focussing their dishes on a source, resulting in a sharp but narrow field-of-view per dish. The LOFAR antennas are omnidirectional, and do not have to be moved or rotated. A station is focussed on a source by taking advantage of the fact that the speed of electromagnetic waves is finite, causing an electromagnetic wave to arrive at different antennas at different times (see Figure 1(a)). A process called *delay compensation* counters the differences in arrival times by delaying the signals from different antennas such that they are synchronised with respect to the observed source. The synchronised signals from all antennas are accumulated (by the station *beam former*) and sent to the BG/P. The resulting *station beam* has a wide field-of-view around the source. It contains samples for both the X and the Y polarisations as 16-bit complex integers, resulting in 3.1 Gb/s of data per station.

The BG/P, which receives the signals from all stations, again performs delay compensation and beam forming, this time in software. The BG/P beam former can focus on sources anywhere in the fields of view of the station beams, creating *tied-array beams* (beams). An example is shown in Figure 1(c), in which station beams (represented

by an ellipse) contains several tied-array beams (represented by hexagons). The actual width of the station beams, as well as the width of the tied-array beams, depends on the number as well as the locations of the stations used. Hundreds of tied-array beams are typically needed to fully cover the field of view of a station beam. Different tied-array beams are created by adding the signals from the individual stations using different delays, which depend on the relative positions of the stations and the relative direction of the tied-array beam with respect to the station beam. The BG/P applies delay compensation in two steps. First, coarse-grain compensation is performed by shifting the samples from different stations with respect to each other. Then, for each tied-array beam, the remaining sub-sample delays are compensated for by shifting the phases of the signals. The phase of each sample is changed through a complex multiplication with a precomputed weight. Tied-array beams are thus a linear combination of the (shifted) signals from the stations.

Our beam former supports several pipelines: *XY polarisations*, *Stokes IQUV*, and *Stokes I*. The XY polarisations pipeline outputs the raw tied-array beams, which consist of two 3.1 Gb/s streams of 32-bit complex floating points numbers (floats), one stream for each polarisation. The Stokes IQUV pipeline applies a domain transformation to each sample of the raw tied-array beams, which is useful for polarisation-related studies. The four Stokes parameters, calculated through $I = X\overline{X} + Y\overline{Y}$, $Q = X\overline{X} - Y\overline{Y}$, $U = 2\mathrm{Re}(X\overline{Y})$, $V = 2\mathrm{Im}(X\overline{Y})$, with each parameter being a 32-bit float, resulting in four 1.5 Gb/s streams. The Stokes I pipeline calculates only the first Stokes parameter, which represents the signal strength in both polarisations. The Stokes I pipeline supports temporal integration to trade time resolution for a reduced bandwidth per beam, allowing more beams to be created.

Finally, our software can produce the Stokes parameters (I or IQUV) of an *incoherent* beam, which is an accumulation of the uncompensated station signals. The incoherent beam is less sensitive than a tied-array beam, but it maintains the wide field-of-view of the stations. The incoherent beam is used to detect the presence of sources, but does not reveal their location within the station beams.

The primary scientific use case driving the work presented in this paper is pulsar research. A pulsar is a rapidly rotating, highly magnetised neutron star, which emits electromagnetic radiation from its poles. Similar to the behaviour of a lighthouse, the radiation is visible to us only if one of the poles points towards Earth, and subsequently appears to us as a very regular series of pulses, with a period as low as 1.4 ms [4]. Pulsars are weak radio sources, and their individual pulses often do not rise above the background noise that fills our universe. Our beam former can focus on several pulsars at LOFAR's full observational bandwidth, producing either XY polarisation or Stokes IQUV data. Alternatively, the beam former is capable of efficiently performing sky surveys to discover new pulsars (or other radio sources) by covering the sky with hundreds of tied-array beams.

In this paper, we will show how a software solution and the use of a massively parallel machine allows us to achieve these feats. We provide an in-depth study on all performance aspects, real-time behaviour, and scaling characteristics. This paper is organised as follows. First, we will describe the key characteristics of the IBM BlueGene/P supercomputer in Section 2. Section 3 describes the implementation of our pipelines,

followed by the performance analysis in Section 4. We briefly discuss related work in Section 5, and conclude in Section 6.

2 IBM BlueGene/P

We use an IBM BlueGene/P (BG/P) supercomputer for the real-time processing of station data. We will describe the key features of the BG/P, but more information can be found elsewhere [8]. Furthermore, we will describe how our BG/P is connected to its input and output systems.

2.1 System Description

Our system consists of 3 racks, with 12,480 processor cores that provide 42.4 TFLOPS peak processing power. One chip contains four PowerPC 450 cores, running at a modest 850 MHz clock speed to reduce power consumption and to increase package density. Each core has two floating-point units (FPUs) that provide support for operations on complex numbers. The chips are organised in *psets*, each of which consists of 64 cores for computation (*compute cores*) and one chip for communication (*I/O node*). Each compute core runs a fast, simple, single-process kernel, and has access to 512 MiB of memory. The I/O nodes consist of the same hardware as the compute nodes, but additionally have a 10 Gb/s Ethernet interface connected. They run Linux, which allows the I/O nodes to do full multitasking. One rack contains 64 psets, which is equal to 4096 compute cores and 64 I/O nodes.

The BG/P contains several networks. A fast *3-dimensional torus* connects all compute nodes and is used for point-to-point and all-to-all communications over 3.4 Gb/s links. The torus uses DMA to offload the CPUs and allows asynchronous communication. The *collective network* is used for communication within a pset between an I/O node and the compute nodes, using 6.8 Gb/s links. In both networks, data is routed through compute nodes using a shortest path. Additional networks exist for fast barriers, initialisation, diagnostics, and debugging.

2.2 External I/O

We customised the I/O node software stack [9] and run a multi-threaded program on each I/O node which is responsible for the handling of both the input and the output. Even though the I/O nodes each have a 10 Gb/s Ethernet interface, they do not have enough computation power to handle 10 Gb/s of data. The overhead of handling IRQs, IP, and UDP/TCP put a high load on the 850 MHz cores of the I/O nodes, limiting performance. An I/O node can output at most 3.1 Gb/s, unless it has to handle station input (3.1 Gb/s), in which case it can output at most 1.1 Gb/s. We implemented a low-overhead communication protocol called FCNP [6] to efficiently transport data to and from the compute nodes, which perform the required signal processing. The I/O nodes forward the results to our storage cluster, which can sustain a throughput up to 80 Gb/s.

3 Beam Former Pipelines

In this section, we will describe in detail how the full signal-processing pipeline operates, in and around the beam former. The use of a software pipeline allows us to reconfigure the components and design of our standard imaging pipeline, described in [7]. Due to the flexibility of software, we can run several pipelines in parallel on the same data, as long as resource requirements are met. Figure 2 gives an overview of our system. Our software is written in C++, with core routines ported to assembly to obtain maximal performance.

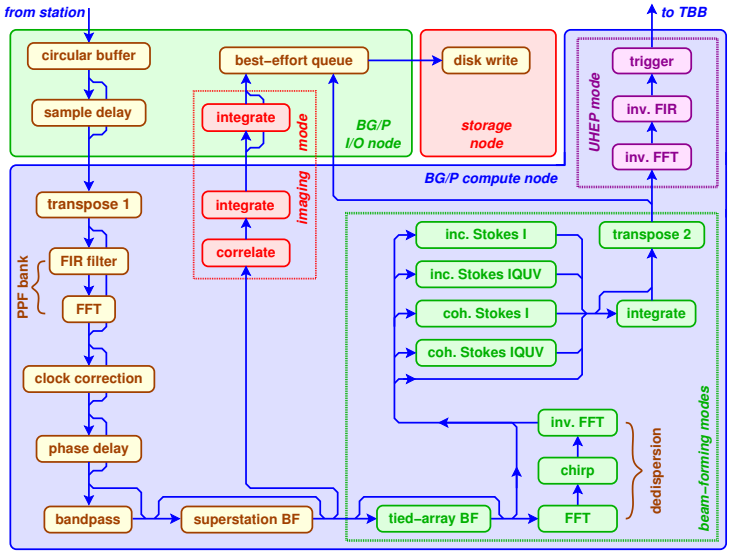


Fig. 2. Data flow diagram describing the on-line pipelines of LOFAR. The imaging and UHEP modes are outside the scope of this work.

3.1 Input from Stations

The BG/P receives data from up to 64 stations. The stations are strategically placed, with 20 stations acting as its centre (the *core*) and 24 stations at increasing distances from the core (see Figure 1(b)). A core station can act as two individual stations in some observational modes. A station is able to produce 248 frequency subbands of 195 kHz out of the sensitivity range of 10 MHz to 250 MHz. Each sample consists of two complex 16-bit integers, representing the amplitude and phase of the X and Y polarisations of the antennas. The resulting data stream from a station is a 3.1 Gb/s UDP stream, which is sent to an I/O node in our BG/P.

At the I/O nodes, the station data are split into chunks of one subband and 0.25 seconds. The chunk size is chosen such that the compute cores have enough memory

to perform all of the necessary processing. Due to the BG/P design, an I/O node sends chunks to its own compute cores only. The compute cores exchange the chunks they obtain from their I/O node using an all-to-all exchange.

3.2 First All-to-all Exchange

The first all-to-all exchange in the pipeline allows the compute cores to distribute the chunks from a single station, and to collect all the chunks of the same subband from all of the stations. The exchange is performed over the fast 3D-torus network, but with up to 198 Gb/s of station data to be exchanged, special care still has to be taken to avoid network bottlenecks. It is impossible to optimise for short network paths due to the physical distances between the different psets across a BG/P rack. Instead, we optimised the data exchange by creating as many paths as possible between compute cores that have to exchange data. Within each pset, we employ a virtual remapping such that the number of possible routes between communicating cores in different psets is maximised.

The communications in the all-to-all exchange are asynchronous, which allows a compute core to start processing a subband from a station as soon as it arrives, up to the point that data from all stations are required. Communication and computation are thus overlapped as much as possible.

3.3 Signal Processing

Once a compute core receives a chunk, it performs a sequence of processing steps:

Conversion of the data from 16-bit little-endian integers to 32-bit big-endian floats, in order to be able to do further processing using the powerful dual FPU units present in each core. The data doubles in size, which is the main reason why we implement it *after* the exchange.

Poly-Phase Filter (PPF) bank filters the data, which consists of a Finite Impulse Response (FIR) filter and a Fast Fourier Transform (FFT). The FFT allows the chunk, which represents a subband of 195 kHz, to be split into narrower subbands (*channels*). A higher frequency resolution allows more precise corrections in the frequency domain, such as the removal of radio interference at specific frequencies.

Clock correction compensates for known clock offsets between stations.

Phase (fine-grain) delay compensation is performed to align the chunks from the different stations. The fine-grain delay compensation is performed as a phase rotation, which is implemented as one complex multiplication per sample. The delays are both frequency and time dependent.

Band pass correction is applied to adjust the signal strengths in all channels, because the stations introduce a bias in the signal strengths across the channels within a subband.

Up to this point, processing chunks from different stations can be done independently, but from here on, the data from all stations are required. The first all-to-all exchange thus ends here.

3.4 Beam Forming and Stokes Calculations

First, the different weights required for the different tied-array beams are computed, based on the station positions and the beam directions. Note that the data in the chunks are already delay compensated with respect to the source at which the stations are pointed. Any delay compensation performed by the beam former is therefore to compensate the delay differences between the desired beams and the station's source. The reason for this two-stage approach is flexibility. By already compensating for the station's source in the previous step, the resulting data can not only be fed to the beam former pipelines, but also to other pipelines, such as the imaging pipeline. Because we have a software pipeline, we can implement and connect different processing pipelines with only a small increase in complexity.

The delays are applied to the station data through complex multiplications and additions, programmed in assembly. In order to take full advantage of the L1 cache and the available registers, data is processed in sets of 6 stations, producing 3 beams, in portions of 128 samples, or a subset thereof to cover the remainders. While the exact ideal set size in which the data is to be processed is platform specific, we have shown in previous work that similar tradeoffs exist for similar problems across different architectures [5].

Because each beam is an accumulation of the data from all stations, the bandwidth of each beam is equal to the bandwidth of data from a single station, which is 6.2 Gb/s now that the samples are 32-bit floats. Once the beams are formed, they are kept as XY polarisations or transformed into the Stokes IQUV or the Stokes I parameters. In the latter case, the beams can also be integrated temporally to reduce the resulting data rate. Finally, an incoherent beam can be created in parallel, and converted into either Stokes I or Stokes IQUV parameters.

The beam former transforms chunks representing station data into chunks representing beam data. Because a chunk representing station data contained data for only one subband, the chunks representing different subbands of the same beam are still spread out over the full BG/P. Chunks corresponding to the same beam are brought together using a second all-to-all exchange.

3.5 Channel-level Dedispersion

Another major component in the pulsar-observation pipeline is real-time dedispersion. Since light of a high frequency travels faster through the interstellar medium than light of a lower frequency, the arrival time of a pulse differs for different wave lengths. To combine data from multiple frequency channels, the channels must be aligned (shifted in time). Otherwise, the pulse will be smeared or even overlap with the next pulse, causing many details to be lost. This process, called *dedispersion*, is done by post-processing software that runs after the observation has finished. However, to observe at the lowest frequencies, or to observe fast-rotating millisecond pulsars, dedispersion must also be performed *within* a channel, since our channels (typically 12 kHz) are too wide to ignore dispersion.

Figure 3 illustrates pulses of pulsar J0034-0534 at four frequencies. The pulse period is 1.88 ms. On the left is the original dispersed signal, which results in a smeared

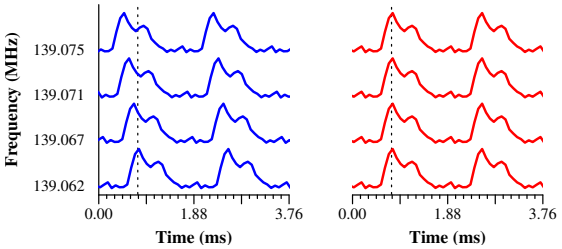


Fig. 3. Pulse arrival times within a 12 kHz channel before (left) and after (right) channel-level dedispersion.

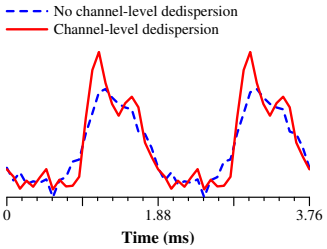


Fig. 4. Pulse profiles with and without channel-level dedispersion.

pulse when the frequencies are collapsed into a 12 kHz channel. On the right is the dedispersed signal, which results in a sharp pulse when collapsed.

Dedispersion is performed in the frequency domain, effectively by doing a 4096-point FFT that splits a 12 kHz channel into 3 Hz subchannels. The phases of the observed samples are corrected by applying a chirp function, i.e., by multiplication with precomputed, channel-dependent, complex weights. These multiplications are programmed in assembly, to reduce the computational costs. A backward FFT is done to revert to 12 kHz channels.

Figure 4 shows the observed effectiveness of channel-level dedispersion, which improves the effective time resolution from 0.51 ms to 0.082 ms, revealing a more detailed pulse and a better signal-to-noise ratio. Dedispersion thus contributes significantly to the data quality, but it also comes at a significant computational cost due to the two FFTs it requires. It demonstrates the power of using a *software* telescope: the pipeline component was implemented, verified, and optimised in only one month time.

3.6 Second All-to-all Exchange

In the second all-to-all exchange, the chunks made by the beam former are again exchanged over the 3D-torus network. Due to memory constrains on the compute cores, the cores that performed the beam forming cannot be the same cores that receive the beam data after the exchange. We assign a set of cores (*output cores*) to receive the chunks. The output cores are chosen before an observation, and are distinct from the *input cores* which perform the earlier computations in the pipeline.

An output core gathers the chunks that contain different subbands but belong to the same output stream. An output stream consists of all 248 subbands belonging to the same polarisation or Stokes parameter. If the full 248 subbands cannot be exported by the I/O node due to data rate limitations, the polarisation or Stokes parameter is split into multiple

Then, it rearranges the dimensions of the data into their final ordering, which is necessary, because the data order that will be written to disk is not the same order that can be produced by our computations without taking heavy L1 cache penalties. We hide this reordering cost at the output cores by overlapping computation (the reordering of

a chunk) with communication (the arrival of other chunks). Once all of the chunks are received and reordered, they are sent back to the I/O node.

For the distribution of the workload over the available output cores, three factors have to be considered. First, all of the data belonging to the same beam has to be processed by output cores in the same pset, to ensure that one I/O node can concatenate all of the 0.25 second chunks that belong to the beam. Second, the maximum output rate per I/O node has to be respected. Finally, the presence of the first all-to-all exchange, which uses the same network at up to 198 Gb/s. The second exchange uses up to 80 Gb/s. Even though each link sustains 3.4 Gb/s, it has to process the traffic from four cores, as well as traffic routed through it between other nodes. The network links in the BG/P become overloaded unless enough output cores are used to spread the load.

3.7 Transport to Disks

Once an output core has received and reordered all of its data, the data are sent to the core's I/O node. The I/O node forwards the data over TCP/IP to the storage cluster. To avoid any stalling in our pipeline due to network congestion or disk issues, the I/O node uses a best-effort buffer which drops data if it cannot be sent.

4 Performance Analysis

We will focus our performance analysis on edge cases that are of astronomical interest. In all cases, we respect the real-time nature of our system by limiting the load such that there is at most 0.1% of data loss. Almost all variance occurs in the networks within the BG/P due to clashes caused by scheduling intricacies.

4.1 Overall Performance

Figure 4.2 shows the maximum number of beams that can be created when using a various number of stations, in each of the three modes: XY polarisations, Stokes IQUV, and Stokes I. In both the XY polarisations and the Stokes IQUV modes, the pipeline is I/O bound. Each beam is 6.2 Gb/s wide. We can make at most 12 beams without exceeding the available 80 Gb/s to our storage cluster. The available bandwidth decreases down to 70 Gb/s due to the fact that an I/O node can only output 1.1 Gb/s if it also has to process station data. The granularity with which the output can be distributed over the I/O nodes, as well as scheduling details, determine the actual number of beams that can be created, but in all cases, the beam former can create at least 10 beams at LOFAR's full observational bandwidth.

In the Stokes I mode, we applied several integration factors (1, 2, 4, 8, and 12) in order to show the trade-off between beam quality and the number of beams. Integration factors higher than 12 does not allow significantly more beams to be created, but could be used in order to further reduce the total output rate. For low integration factors, the beam former is again limited by the available output bandwidth. Once the Stokes I streams are integrated sufficiently, the system becomes bounded by the compute nodes: if only signals from a few stations have to be combined, the beam former is limited

by the amount of available memory required to store the beams. If more input has to be combined, the beam former becomes limited by the CPU power available in the compute cores. For observations for which a high integration factor is acceptable, the beam former is able to create 155 up to 543 tied-array beams, depending on the number of stations used. For observations which need a high time resolution and thus a low integration factor, the beam former is still able to create at least 42 tied-array beams.

4.2 System Load

We further analyse the workload of the compute cores by highlighting a set of cases, summarised in Table 1. We will focus on a memory-bound case (A), which also creates the highest number of beams, on CPU-bound cases interesting for performing surveys, with either 24 stations (B) or 64 stations (C) as input. Cases D and E focus on high-resolution observations of known sources, and are I/O bound configurations with 24 and 64 stations, respectively. Case F focusses on the observations of known sources as well, using Stokes I output, which allows more beams to be created. Channel-level dedispersion is applied for all cases that observe known sources.

Case	Mode	Channel dedisp.	Int. factor	Stations	Beams	Input rate	Output rate	Bound	Used for
A	Stokes I	N	12	4	543	12 Gb/s	70 Gb/s	Memory	Surveys
B	Stokes I	N	8	24	327	74 Gb/s	63 Gb/s	CPU	Surveys
C	Stokes I	N	8	64	155	198 Gb/s	30 Gb/s	CPU	Surveys
D	Stokes IQUV	Y	-	24	13	74 Gb/s	80 Gb/s	I/O	Known sources
E	Stokes IQUV	Y	-	64	10	198 Gb/s	62 Gb/s	I/O	Known sources
F	Stokes I	Y	1	64	42	198 Gb/s	65 Gb/s	CPU	Known sources

Table 1. Several highlighted cases.

The workload of the compute cores for each case is shown in Figure 4.2, which shows the average workload per core. For the CPU-bound cases B and C, the average load has to be lower than 100% in order to prevent fluctuations from slowing down our real-time system. These fluctuations typically occur due to clashes within the BG/P 3D-torus network which is used for both all-to-all-exchanges, and cannot be avoided in all cases.

The cases which create many beams (A-C) spend most of the cycles performing beam forming and calculation the Stokes I parameters. The beam forming scales with both the number of stations and the number of beams, while the Stokes I calculation costs depends solely on the number of beams. Case A has to beam form only four stations, and thus requires most of its time calculating the Stokes I parameters. Case B and C use more stations, and thus need more time to beam form.

The costs for both the first and the second all-to-all exchange are mostly hidden due to overlaps with computation. The remaining cost for the second exchange is proportional to the output bandwidth required in each case.

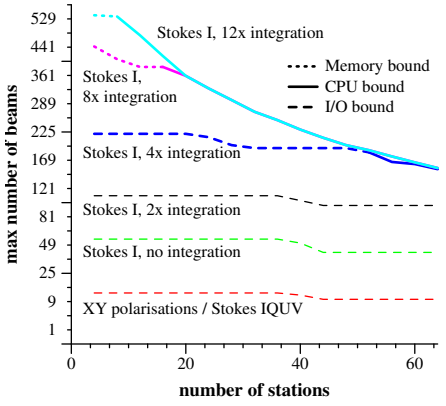


Fig. 5. The maximum number of beams that can be created in various configurations.

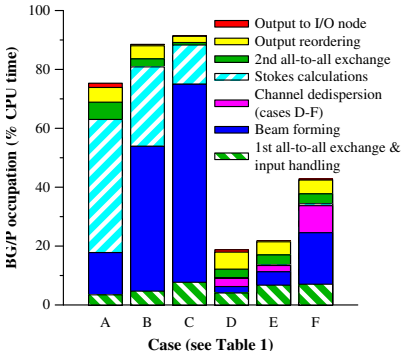


Fig. 6. The time spent in the processing steps.

For the I/O-bound cases D-F, only a few tied-array beams are formed and transformed into Stokes I(QUV) parameters, which produces a lot of data but requires little CPU time. Enough CPU time is therefore available to include channel-level dedispersion, which scales with the number of beams and, as Figure 4.2 shows, is an expensive operation.

5 Related Work

The LOFAR beam former is the only beam former capable of producing hundreds of tied-array beams. Traditional radio dishes are unsuitable for beam forming due to their narrow field-of-view. A radio dish can be extended to focus on multiple sources by placing additional receivers in its focal point (a *focal plane array*) [3], but such a solution does not scale. The Murchison Widefield Array (MWA) uses a design similar to LOFAR, and plans to build a beam former, but is still under construction [2].

6 Conclusions

We have shown the capabilities of our beam former pipelines, running in software on an IBM BlueGene/P supercomputer. Our system is capable of producing 13 tied-array beams at LOFAR's full observational bandwidth before our output limit of 80 Gb/s is met. Alternatively, it can form hundreds of beams at a reduced resolution, the exact number depending on the number of stations and the pipeline used. Finally, an incoherent beam can be created, which retains the wide field-of-view offered by our stations.

The use of a software solution on powerful interconnected hardware is a key aspect in the development and deployment of our pipeline. Because we use software, rapid prototyping is cheap, allowing novel features to be tested to aid the exploration of the design space of a new instrument. The resulting pipelines retain the flexibility

that software allows. The control flow and bookkeeping can become complex while remaining manageable through software abstraction. We are able to run the same station data through multiple pipelines in parallel, and even multiple independent observations in parallel, as long as there are enough available resources. The science which drives LOFAR, and which is driven by it, is greatly accelerated through the use of an easily reconfigurable instrument.

The BlueGene/P supercomputer provides us with enough computing power and powerful networks to be able to implement the signal processing and all-to-all-exchanges that we require, without having to resort to a dedicated system which inevitably curbs the design freedom that the supercomputer provides. As with any system, platform-specific parameters nevertheless become important when maximal performance is desired. We tuned the distribution of the workload over the cores to avoid network collisions, and implemented our core routines in assembly in order to maximise the throughput.

References

1. A.G. de Bruyn et al. Exploring the Universe with the Low Frequency Array, A Scientific Case, September 2002.
   http://www.lofar.org/PDF/NL-CASE-1.0.pdf.
2. C.J. Lonsdale et al. The Murchison Widefield Array: Design Overview. *Proceedings of the IEEE*, 97(8):1497–1506, August 2009.
3. L. Staveley-Smith et al. The Parkes 21cm Multibeam Receiver. 13(3):243–248, November 1996.
4. J.W.T. Hessels, S.M. Ransom, I.H. Stairs, P.C.C. Freire, V.M. Kaspi, and F. Camilo. A Radio Pulsar Spinning at 716 Hz. *Science*, 311(5769):1901–1904, March 2006.
5. R.V. van Nieuwpoort and J.W. Romein. Using Many-Core Hardware to Correlate Radio Astronomy Signals. In *ACM International Conference on Supercomputing (ICS'09)*, pages 440–449, New York, NY, June 2009.
6. J.W. Romein. FCNP: Fast I/O on the Blue Gene/P. In *Parallel and Distributed Processing Techniques and Applications (PDPTA'09)*, Las Vegas, NV, July 2009.
7. J.W. Romein, P.C. Broekema, J.D. Mol, and R.V. van Nieuwpoort. The LOFAR Correlator: Implementation and Performance Analysis. In *ACM SIGPLAN Symposium on Principles and Practice on Parallel Programming (PPoPP'10)*, Bangalore, India, January 2010. To appear.
8. IBM Blue Gene team. Overview of the IBM Blue Gene/P Project. *IBM Journal of Research and Development*, 52(1/2), January/March 2008.
9. K. Yoshii, K. Iskra, H. Naik, and P.C. Broekema P. Beckman. Performance and Scalability Evaluation of "Big Memory" on Blue Gene Linux. *International Journal of High Performance Computing*. To appear.