The LOFAR Beamformer: Implementation and Performance Analysis

Jan David Mol, John W. Romein and Rob V. van Nieuwpoort

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

Abstract. Traditional radio telescopes use one or several large steel dishes to observe a single source. The LOFAR telescope is different and features a novel design in several ways. It uses tens of thousands of fixed omnidirectional antennas, the signals of which are centrally combined in real time. The LOFAR telescope focusses on a source by performing a weighted addition of the signal streams originating from groups of antennas (stations). We can focus our telescope in multiple directions in parallel by combining the signal streams from the stations multiple times using different weights. In fact, the parallel processing power and high-speed interconnect in our supercomputer allows us to look at hundreds of sources at the same time. The power to observe many sources in parallel serves a broad range of scientific astronomical interests, and creates novel opportunities for performing astronomical observations.
LOFAR is also the first major telescope to process its data in software, instead of needing a dedicated hardware design. By using software, the processing remains flexible and scalable, and new features are easier to implement and to maintain. It is through the use of software that we can fully explore the novel features and the power of our unique instrument.
In this paper, we present the processing pipeline in our supercomputer which enables our parallel observations. Our so-called *Pulsar Pipeline*, named after the use case that pushed its development, is implemented on a supercomputer, and receives up to 64 data streams from the stations at 3.1 Gb/s each. Inside the supercomputer, signal-processing techniques and two all-to-all exchanges are performed. Our pulsar pipeline further expresses the power of a software telescope implemented using parallel processing techniques. We present the trade-offs in our design, the CPU and I/O performance bottlenecks that we encounter, as well as the scaling characteristics and its real-time behaviour.

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 can be 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 ground-breaking 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.

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 200 Gb/s for all stations) to the BG/P super computer. Inside the BG/P, the data are split and recombined 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.

The stations can be configured to observe in several directions in parallel, but have to divide their output bandwidth among them. In this paper, we present the *beamformer*, an extension to the LOFAR software which allows the telescope to be aimed in tens of directions simultaneously at LOFAR's full observational bandwidth, and in hundreds of directions at reduced bandwidth. Both feats cannot be matched by any other telescope. The data streams corresponding to each observational direction, called *beams*, are generated through (weighted) summations of the station inputs, which are demultiplexed using an all-to-all exchange, and routed to the storage cluster.

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 [3]. Pulsars are relatively weak radio sources, and their individual pulses often do not rise above the background noise that fills our universe. LOFAR is one of the few telescopes which operates in the frequency range (10 – 240 MHz) in which pulsars are typically at their brightest. Our beamformer also makes LOFAR the only telescope that can observe in hundreds of directions simultaneously with high sensitivity. These aspects make LOFAR an ideal instrument to discover unknown pulsars by doing a sensitive sky survey in a short amount of time, as well as an ideal instrument to study known pulsars in more detail. Astronomers can also use our beamformer to focus on planets, exoplanets, the sun, and other radio objects, with unprecedented sensitivity. Furthermore, our pipeline allows fast broad-sky surveys to discover not only new pulsars but also other radio sources.

In this paper, we will show how a software solution and the use of massive parallellism allows us to achieve this feat. We provide an in-depth study on all performance aspects, real-time behaviour, and scaling characteristics. The paper is organised as follows.

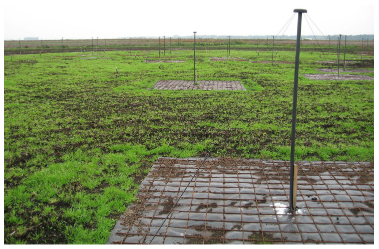
2 Related Work

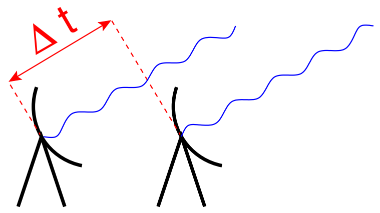
MWA.

LOFAR imaging pipeline [5]

3 LOFAR

The LOFAR telescope consists of many thousands of simple dipole antennas (see Figure 1(a)), grouped in *stations*. The stations are strategically placed, with six stations acting as its center (the *core*) and TODO stations at increasing distances from that core. Every station collects and combines the signals from its antennas, and sends the resulting data stream to our IBM BlueGene/P (BG/P) supercomputer at our central processing facility. The BG/P combines the data streams from one or more stations and reduces the resulting stream in size sufficiently to be able to store it on disks in our storage cluster. Both the stations and the BG/P perform hard-real-time signal processing. Once the data has been stored on disk, off-line processing takes over. The off-line processing transforms and further reduces the data produced by the BG/P into data products such as images or time series, and are made available to the astronomer(s) that requested the observation.


(a) A field with low-band antennas (dipoles).



(b) The left antenna receives the wave later.

Fig. 1. LOFAR antennas

The antennas are omnidirectional and have no moving parts. Instead, all telescope functions are performed electronically through signal processing done at the stations and on the BG/P. The telescope can be aimed because the speed of light is finite: the light emitted by a source will arrive at different antennas at different times (see Figure 1(b)). By adding appropriate delays to the signals from individual antennas before accumulating them, the signals from the source will be amplified with respect to signals from other directions. Once the samples from all antennas are combined, the data are transmitted to the BG/P, which uses the same technique to combine the data from the individual stations. The latter will be explained further in Section 5.

A LOFAR station is able to produce 248 frequency subbands of 195 kHz out of the sensitivity range of 80 MHz to 250 MHz. Each sample consists of two complex 16-bit integers, representing the amplitude and phase of the X and Y polarizations of the antennas. The resulting data stream from a station is a 3.1 Gb/s UDP stream.

4 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 [6]. Furthermore, we will describe how our BG/P is connected to its input and output systems.

4.1 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 (the Compute Node Kernel, or CNK), 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. Also, 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. The torus uses DMA to offload the CPUs and allows asynchronous communication. The *collective network* is used for MPI collective operations, but also for external communication. External communication is routed within each pset, through its I/O node using a tree configuration. The I/O node is capable of transporting 6.8 Gb/s bidirectionally to and from one or more compute nodes. Additional networks exist for fast barriers, initialization, diagnostics, and debugging.

4.2 External Connections

We run a multi-threaded program on each I/O node which is responsible for two tasks: the handling of input, and the handling of output (see Figure 2). 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 such a high load on the 850 MHz cores of the I/O nodes, that the performance seems limited to a total data rate of roughly 4 Gb/s. To achieve these data rates, we installed our own software on the I/O nodes, augmenting IBM's software stack [7], and we implemented a low-overhead communication protocol called FCNP [4] to efficiently transport data to and from the compute nodes. Recall that at full observational bandwidth, a station produces 3.1 Gb/s of data. Each I/O node can thus receive data from at most one station. The I/O nodes forward the station data to the compute nodes. The compute nodes convert the (complex) integer samples to the (complex) float domain, and perform all of the necessary on-line signal processing.

Once the compute nodes have finished processing the data, the results are sent back to the I/O nodes. The I/O nodes forward these results to our 24-node storage cluster.

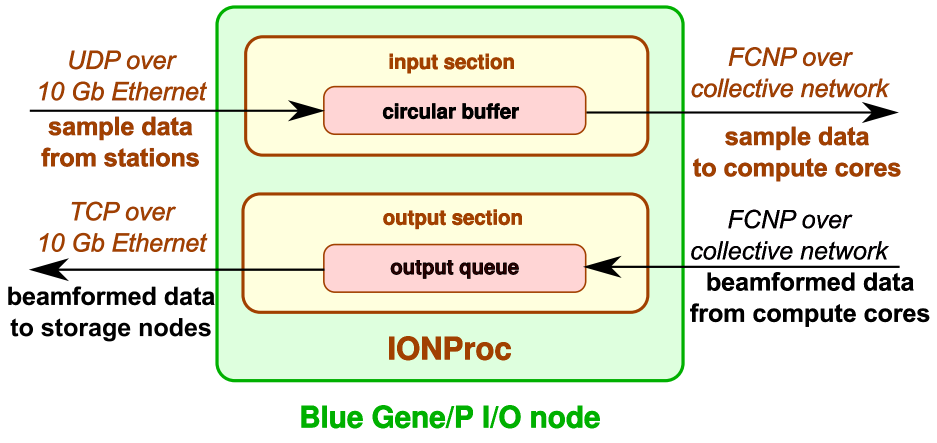


Fig. 2. Data flow diagram for the I/O nodes.

Each I/O node can send up to 1.1 Gb/s if it receives station data at the same time, and up to 3.1 Gb/s if the I/O node is not receiving station data. The storage cluster itself can handle up to 80 Gb/s of sustained throughput. The output queue which is maintained at the I/O node for all data uses a best-effort policy and drops data if it cannot be sent, in order to keep the BG/P running at real time.

5 Beamforming

As mentioned in Section 3, a LOFAR station is aimed at a source by applying different delays to the signals from the antennas, and subsequently adding the delayed signals. Delaying the signals is known as *delay compensation*, and subsequently adding them as *beamforming*, which is performed at several stages in our processing pipeline. The station beamformer is implemented in hardware, in which the signal is delayed by switching it over wires of different lengths. The signals from the different antennas are subsequently added using an FPGA. The resulting beam, even though it is aimed at a certain source, is nevertheless still sensitive to signals from a large area around the source.

The BG/P, in which the signals from all the LOFAR stations come together, again performs delay compensation by adding the appropriate delays to the signals from the various stations, this time in software. Because the beam produced by each station has a large sensitive area around the source, the BG/P beamformer can not only point at the source for which the stations are configured, but also at sources around it. Different beams can then be formed by adding the signals from the individual stations over and over again, each time using different delays. The delay that has to be applied depends on the relative positions of the stations and the relative direction of the beam with respect to the source. The delays are applied in software in two phases. First, the streams are aligned by shifting them a whole number of samples with respect to each other, which resolves delay differences up to the granularity of a single sample. Then, the remaining sub-sample delays are compensated for by shifting the phase of the signal. Even though a phase correction is not a proper shift because information does not shift between one sample and the next, the difference proves to be sufficiently small in practice.

This approximation is in fact good enough to limit the generation of different beams through phase corrections alone. The samples coming from different stations are shifted the same amount for all beams that are created. Because a phase shift can be performed using a complex multiplication, the beam former in the BG/P only has to accumulate the weighted vectors of samples from each station. Let $\overrightarrow{S_i}$ be the stream of samples from station i, $\overrightarrow{B_j}$ the stream of samples of beam j, and w_{ij} the phase correction to be applied on station i to represent the required delay for beam j. Then, the beam former performs the following calculation on the samples of both the X and Y polarisations independently, to obtain beam j:

$$\overrightarrow{B_j} = \sum_{i \in \text{stations}} w_{ij} \overrightarrow{S_i}. \tag{1}$$

Note that this equation allows for easy parallellisation. As mentioned in Section 3, the station data consist of a sample stream consisting of up to 248 subbands. The subbands are independent and can thus be processed in parallel. Also, the streams can quite trivially be split along the time dimension in order to increase parallelisation.

A beam $\overrightarrow{B_j}$ formed at the BG/P consists of a stream of complex 32-bit floating point numbers, two for each time sample (representing the X and Y polarisations), which is equal to 6.2 Gb/s at LOFAR's full observational bandwidth. For some observations however, such a precision is not required, and the beams can be reduced in size in order to be able to output more beams in parallel. In this paper, we consider two types of observations. First, we consider *high-resolution* observations, in which an astronomer wants to look at sources at the highest resolution possible. Such observations will typically produce 6.2 Gb/s per beam. The second type of observation is a *many-beams* observations, in which an astronomer wants to survey the sky with as many beams as possible, given a lower bound on the acceptable resolution. Because each individual beam can be recorded with a much lower resolution than in the high-resolution observations, bandwidth becomes available to create more beams.

The two types of observations translate into three modes in which our pipeline can run:

Complex Voltages are the untransformed beams as produced by the beamformer. For each beam, the complex 32-bit float samples for the X and Y polarisations are split and stored in two separate files in disk, resulting in two 3.1 Gb/s streams to disk per beam.

Stokes IQUV parameters represent the polarisation aspects of the signal, and are the result of a domain transformation performed on the complex voltages. The transformation is useful for polarisation-related studies. The Stokes parameters consists of four real 32-bit float samples which represent the Stokes I, Q, U and V values for each time sample, which are computed from the complex X and Y polarisations using the following formulas:

$$I = X\overline{X} + Y\overline{Y}, \tag{2}$$
$$Q = X\overline{X} - Y\overline{Y}, \tag{3}$$
$$U = 2\mathrm{Re}(X\overline{Y}), \tag{4}$$
$$V = 2\mathrm{Im}(X\overline{Y}). \tag{5}$$

The four Stokes values are stored in separate files, resulting in four 1.5 Gb/s streams to disk per beam.

Stokes I represents the power of the signal in the X and Y polarisations combined, and is equal to computing just the Stokes I stream in Stokes IQUV mode. This mode thus results in one 1.5 Gb/s stream to disk per beam. In this mode, which has the smallest bandwidth per beam, we support integrating the samples over time. Time integration reduces the bandwidth per beam by an integer factor, but reduces the time resolution as well.

The BG/P is able to produce tens to hundreds of beams, depending on the mode used. As our measurements will show, the Complex Voltages and Stokes IQUV modes will typically hit an I/O bottleneck when transporting the created beams towards the storage cluster. In the Stokes I mode, the bandwidth per beam is lower and can be further reduced using integration. The I/O bottleneck can thus be avoided in Stokes I mode. If the number of beams is increased, an upper limit on the available memory and computational power will be reached instead.

6 Pulsar Pipeline

In this section, we will describe in detail how the full signal-processing pipeline operates, in and around the beamformer. Much of the pipeline's operation and design is similar to our standard imaging pipeline, described in [5].

6.1 Input from Stations

The first step in the pipeline is receiving and collecting from the stations on the I/O nodes. Each I/O node receives the data of (at most) one station, and stores the received data in a circular buffer (recall Figure 2). If necessary, the read pointer of the circular buffer is shifted a number of samples to reflect the coarse-grain delay compensation that will be necessary to align the streams from different stations, based on the location of the source at which the stations are pointed.

The station data are split into chunks of one subband and approximately 0.25 seconds. Such a chunk is the unit of data on which a compute node will operate. The size of a chunk is chosen such that the compute nodes will have enough memory to perform the necessary operations on them.

To perform beamforming, the compute nodes need chunks from all stations. Unfortunately, an I/O node can only communicate (efficiently) with the compute nodes in its own pset, which makes it impossible to send the chunks directly to the compute cores that will process them. Instead, the I/O node distributes its chunks over its compute cores in a round-robin fashion, after which the compute cores obtain trade different chunks from the same station for the same chunk from different stations, using an all-to-all exchange.

6.2 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 frequency band produced by all of the stations. The exchange is performed over the fast 3D-torus network, but with up to 200 Gb/s of station data to be exchanged (=64 stations producing 3.1 Gb/s), 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 communicating cores in different psets are colinear (see Figure **??**), while the I/O nodes can still address the compute cores in a round-robin fashion.

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.

6.3 Pre-beamforming Signal Processing

Once a compute core receives a chunk, it can start processing. First, we convert the station data from 16-bit little-endian integers to 32-bit big-endian floating point numbers, 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.

Next, the data are filtered by applying a Poly-Phase Filter (PPF) bank, 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 very specific frequencies.

Next, fine-grain delay compensation is performed to align the chunks from the different stations to the same source at which the stations are pointed. The fine-grain delay compensation is performed as a phase rotation, which is implemented as one complex multiplication per sample. The exact delays are computed for the begin time and end time of a chunk, and interpolated in frequency and time for each individual sample.

Next, a band pass correction is applied to adjust the signal strengths in all channels. This is necessary, because the stadions 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.

6.4 Beamforming

The beamformer creates the beams as described in Section 5. First, the different weights required for the different 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 beamformer 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 beamformer, 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. We implemented our beamformer in assembly as it is the main performance bottleneck in our pipeline. 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, or a subset thereof to cover the remaining stations and beams. Because each beam is an accumulated combination 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 floating point numbers.

7 Results

[2]

8 Discussion

9 Conclusions

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. J. Hessels, B. Stappers, and J. van Leeuwen. The Radio Sky on Short Timescales with LOFAR: Pulsars and Fast Transients. In *The Low-Frequency Radio Universe*, volume 407 of *ASP Conference Series*, pages 318–322, 1999.
3. 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.
4. 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.

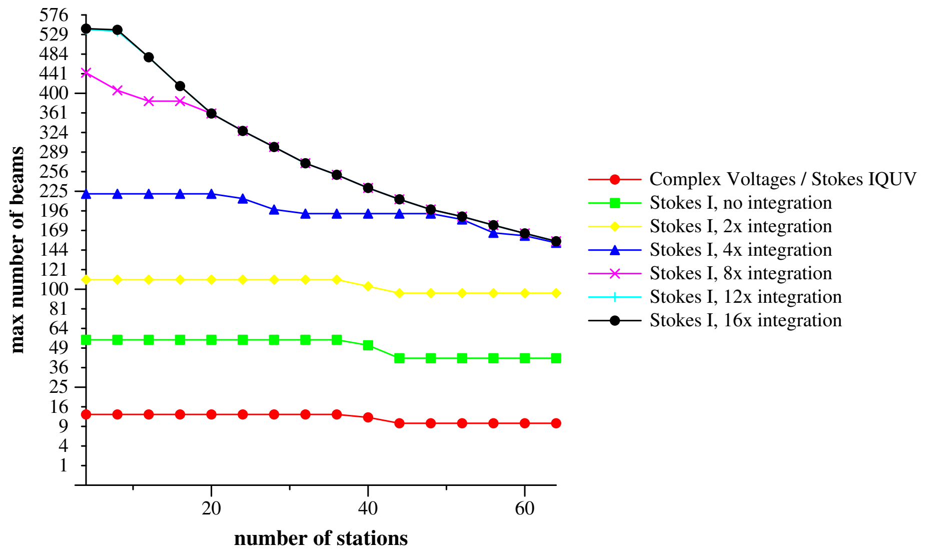


Fig. 3. The maximum number of beams that can be created in various configurations.

5. 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.
6. IBM Blue Gene team. Overview of the IBM Blue Gene/P Project. *IBM Journal of Research and Development*, 52(1/2), January/March 2008.
7. 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.