Astronomical Real-Time Streaming Signal Processing on a Blue Gene/L Supercomputer

John W. Romein[*]
romein@astron.nl

P. Chris Broekema
broekema@astron.nl

Ellen van Meijeren
meijeren@astron.nl

Kjeld van der Schaaf
schaaf@astron.nl

Walther H. Zwart
zwart@astron.nl

Stichting ASTRON (Netherlands Foundation for Research in Astronomy)
Dwingeloo, The Netherlands

ABSTRACT

LOFAR is the first of a new generation of radio telescopes, that combines the signals from many thousands of simple, fixed antennas, rather than from expensive dishes. Its revolutionary design and unprecedented size enables observations in a frequency range that could hardly be observed before, and allows the study of a vast amount of new science cases.

In this paper, we describe a novel approach to process real-time, streaming telescope data in software, *using a* supercomputer. *The desire for a flexible and reconfigurable instrument demands a software solution, where traditionally customized hardware was used. This, and LOFAR's exceptional real-time, streaming signal-processing requirements compel the use of a supercomputer. We focus on the LOFAR CEntral Processing facility (CEP), that combines the signals of all LOFAR stations. CEP consists of a 12,288-core IBM Blue Gene/L supercomputer, embedded in several conventional clusters.*

We describe a highly optimized implementation that will do the bulk of the central signal processing on the Blue Gene/L, namely PolyPhase Filtering, Delay Compensation, *and* Correlation. *Measurements show that we reach exceptionally high computational performance (up to 98% of the theoretical floating-point peak performance). We also discuss how we handle external I/O performance limitations into and out of the Blue Gene/L, to obtain sufficient bandwidth for LOFAR.*

Categories and Subject Descriptors

J.2 [**Physical Sciences and Engineering**]: Astronomy; C.3 [**Special-Purpose and Application-Based Systems**]: Signal Processing Systems; C.5.1 [**Computer System Implementation**]: Large and Medium ("Mainframe") Computers—*Super (very large) computers*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*; D.1.3 [**Programming Techniques**]: Concurrent Programming; J.7 [**Computers in Other Systems**]: Real time

General Terms

Algorithms, Measurement, Performance

Keywords

Blue Gene/L, Correlator, Filtering, LOFAR

[*]To whom correspondence should be directed.

1. INTRODUCTION

LOFAR is an acronym for ***LO****w* ***F****requency* ***AR****ray*, a phased-array radio telescope operating over the 20 to 240 MHz frequency range. Its design breaks radically with conventional telescopes: rather than using large, expensive dishes, LOFAR is built as a distributed sensor network of simple antenna receivers [1]. Their data are centrally collected and processed on the CEntral Processing facility (CEP), that consists of a Blue Gene/L supercomputer and a number of conventional cluster computers [7, 6].

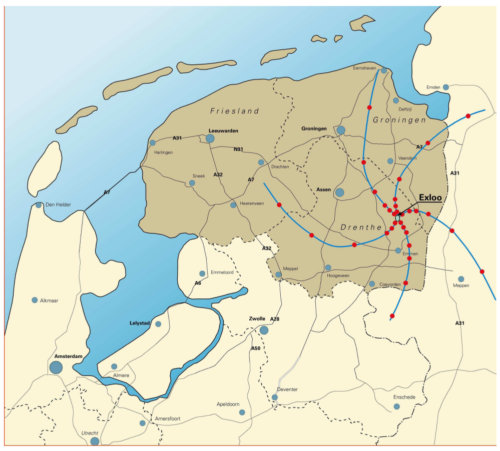

Figure 1: Possible LOFAR configuration.

LOFAR is built in a three-level hierarchy. One hundred co-located antennas form a *station*, i.e., a virtual telescope. Together,

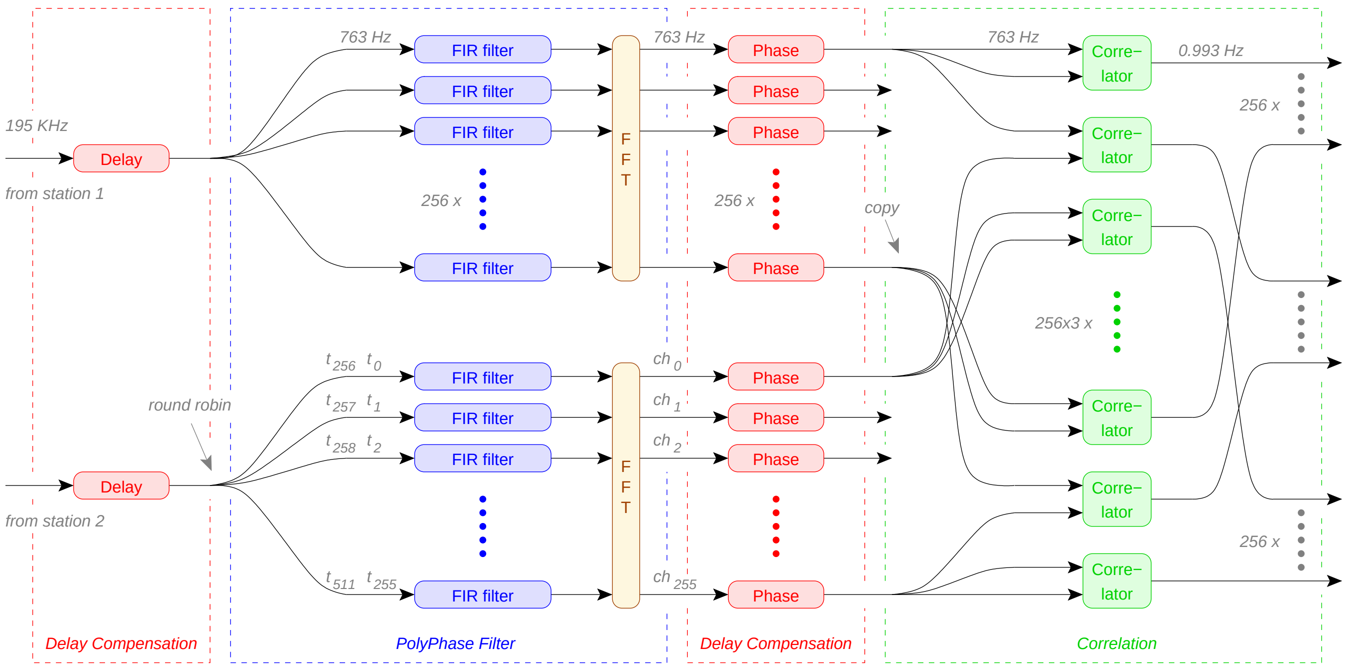

Figure 2: Signal processing on CEP for one subband.

the stations form one, large instrument. A possible topology for LOFAR stations is shown in Figure 1; it covers the northern part of the Netherlands and crosses the German border. Initially, LOFAR will contain 32 stations in the two-kilometer-wide *central core* (marked “Exloo” on the map), plus five remote stations. Construction of the stations will start in the summer of 2006. In the future, more remote stations will be added along five spiral arms, although no decision has been made on their exact positions yet.

LOFAR is driven by the astronomical community, that needs a new instrument to study an extensive amount of new science cases; advances in computer science enabled this. Five key science projects have been defined.[1] First, we expect to see the *Epoch of Reionization* (EoR), the time that the first star galaxies and quasars were formed. The 1.42 GHz emission line of hydrogen is expected to be redshifted into the LOFAR sensitivity range. Second, LOFAR offers a unique possibility in particle astrophysics for studying the origin of high-energy (10^{15}–$10^{20.5}$ eV) *cosmic rays*. Neither the source, nor the physical process that accelerates such particles is known. Third, LOFAR's ability to continuously monitor a large fraction of the sky makes it uniquely suited to study *transient sources*. Since LOFAR has no moving parts, it can instantaneously switch focus to some galactic event. Fourth, *Deep Extragalactic Surveys* will be carried out to find the most distant radio galaxies and study star-forming galaxies. Fifth, it is expected that LOFAR will find many new *pulsars*, that can only be observed in LOFAR's low frequency range. For a more extensive description of the astronomical aspects of the LOFAR system, see de Bruyn et. al. [2].

LOFAR is unique in a number of other aspects, both from an astronomical and from a computer science perspective. Except at the stations, all real-time, streaming processing is done in software. This provides high flexibility and on-the-fly reconfigurability. For example, the observation direction can be changed instantaneously and multiple directions can be handled simultaneously; something that cannot be achieved with conventional dishes. Once constructed, LOFAR will be the world's largest telescope. Its unprecedented data volumes compel processing on a powerful machine. A dedicated Blue Gene/L system, currently ranked ninth in the SuperComputer top 500,[2] and several surrounding clusters provide the necessary resources.

The main contribution of this paper is the description of a new approach to process real-time, streaming data from an astronomical instrument on a Blue Gene/L supercomputer. The idea to develop a real-time software correlator on commodity hardware is also adopted by the eVLBI community [8, 4]; first to quickly ascertain correct operation of the involved telescopes during an observation, and in the future, to generate end product data sets.

We focus on LOFAR's most common processing mode, that filters and correlates the station's data. We present a highly optimized implementation that achieves very high computational performance: the correlator sustains 98% of the theoretical floating point peak performance of a Blue Gene/L compute core. We also show that it is hard to obtain sufficient network bandwidth on the external links of the Blue Gene/L, and discuss how we handle this.

This paper is structured as follows. Section 2 explains the signal-processing steps for the most common observation mode. Section 3 describes the Blue Gene/L, which forms the heart of CEP. In Section 4, we show how we implemented the signal-processing steps on the Blue Gene/L. Section 5 shows results, after which we discuss several issues in Section 6.

2. SIGNAL PROCESSING ON CEP

CEP receives its data via a dedicated wide-area network from the stations. Each station locally combines the signals from its antennas and sends samples to CEP. A sample is a (2×16-bit) complex number that represents the amplitude and phase of a signal at a particular time. The receivers are polarized; they take separate samples from orthogonal (X and Y) directions.

[1]See `http://www.lofar.org/`.

[2]See `http://www.top500.org/`.

By filtering, the stations divide the spectrum into 195 KHz wide *subbands*; signals outside the frequency range of the subband are suppressed. Each subband is sampled at 195 KHz, yielding 195,000 samples per polarization per subband per station. Each station can send up to 160 independent subbands, possibly from multiple observations. Thus, a (noncontiguous) band of up to 32 MHz wide can be monitored during an observation.

Figure 2 illustrates the streaming-data property of our application. The data undergo several transformations, as explained in the remainder of this section. The figure shows the data flow for two stations and a single subband. Only one polarization per station is shown; both polarizations are processed independently, except in the correlator.

2.1 Delay compensation

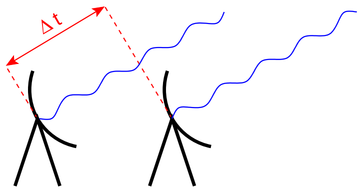

Figure 3: The left antenna receives the signal later.

Since light travels at a finite speed, two antennas do not receive a wave at the same time, as illustrated by Figure 3. To correlate two signals (see Section 2.3), the signal from one of the receivers must be delayed to compensate for the difference in travel time. The delay depends on the distance of the receivers and the direction in which the receivers observe. This is complicated by the rotation of the earth, which alters the orientation of the stations with respect to the observed sky continuously.

Delay compensation is done in two stages. The first stage delays the stream of one of the station samples by an integer amount, such that most of the delay time is compensated for (see the "Delay" boxes in Figure 2). The second stage is performed later (see the "Phase" boxes), and compensates for the fraction of time that remains. It essentially shifts the phase of the signal, by multiplying the PolyPhase Filter output (see Section 2.2) by $e^{-i2\pi\Delta\tau}$. Since the multiplication factor depends on frequency and time (as the earth rotates), we compute the factor once each second for the base frequency of a subband, and interpolate in frequency and time for each sample.

2.2 The PolyPhase Filter

Each 195 KHz wide subband, that comes from the stations as a continuous stream of samples, is split into 256 consecutive frequency channels by a *PolyPhase Filter* (PPF). The PPF itself consists of 256 Finite Impulse Response (FIR) filters, of which the outputs are Fourier transformed (see the respective boxes in Figure 2), as explained below.

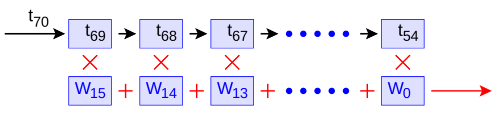

Figure 4: A 16-tap FIR filter.

A *FIR filter* is essentially a time-delay filter with a small (in our case 16) number of history buffers, i.e., *taps* (see Figure 4). Each clock tick, a sample goes in, shifting history data to the right. A weighted sum of the 16 taps goes out. We use the FIR filter as a band pass filter, i.e., a filter that attenuates high and low frequencies. The filter characteristics are determined by the weights. The PPF contains 256 FIR filters, each with its own weight vector. The delayed stream of station samples is round-robin distributed over the FIR filters (see Figure 2), and a FFT of the FIR filter outputs yields 256 frequency channels, each 763 Hz wide.

2.3 Correlation

The received signals from sky sources are so weak, that the antennas mainly receive noise. To see if there is statistical coherence in the noise, simultaneous samples of each pair of stations are correlated, by multiplying the sample of one station with the complex conjugate of the sample of the other station. To reduce the output size, the products are integrated, by accumulating all products. We accumulate 768 correlations at 763 Hz, so that the integration time is approximately one second. This is much shorter than for current telescopes. Correlation is done for each pair of stations, and for each channel separately. Since the correlation of station A and B is the complex conjugate of the correlation of station B and A, only one pair is computed. Stations are also autocorrelated, i.e., with themselves (see, for example, the top "Correlator" box in Figure 2). Both polarizations of station A are correlated with both polarizations of station B, yielding correlations in XX, XY, YX, and YY directions.

2.4 Flagging

Each individual station sample, channel, or correlation can be flagged if its value is not trusted, e.g., as the result of external interference. A flagged sample will not contribute to the final result. The decision whether or not to flag a sample can be based on exceeding some threshold or on a list of known interfering sources (e.g., a TV station). Lost network packets etc. also result in flagged data. Statistics on the flagged samples are kept and provide a measure for the signal quality that can be used for calibration.

3. DESIGN OF THE BLUE GENE/L

In this section, we briefly describe the IBM Blue Gene/L system. Blue Gene/L is a massively parallel supercomputer based on System-on-a-Chip (SoC) components. Each Blue Gene/L compute node consists of two PowerPC 440 cores running at 700 MHz. Each of these cores is extended with two 64-bit FPUs. Each FPU can sustain one fused multiply-add instruction per cycle, giving the core a theoretical peak performance of 2.8 GFlop/sec. The FPUs can read each others registers and can execute instructions that operate on complex numbers. Each core has a 32-KB (noncoherent) L1 cache and a L2 prefetch buffer. Two cores share a 4-MB L3 cache and 512 MB DRAM. Two dual-core compute nodes are located on a compute card, sixteen of which form a node card. Sixteen of these node cards form a midplane, and two midplanes are stacked to form a rack of 1024 dual-core CPUs. Our system consists of six of such racks.

There are two modes in which applications can run: *virtual node mode* and *coprocessor mode*. In the former mode, both cores in a compute node can be used for computations and for synchronous communication; the L3 cache and main memory are split. In the latter mode, one of the nodes is used for computations, and the other is used for (asynchronous) communication. We prefer virtual node mode, because it doubles the floating point performance, although in coprocessor mode, the user can offload computational code onto the communication core.

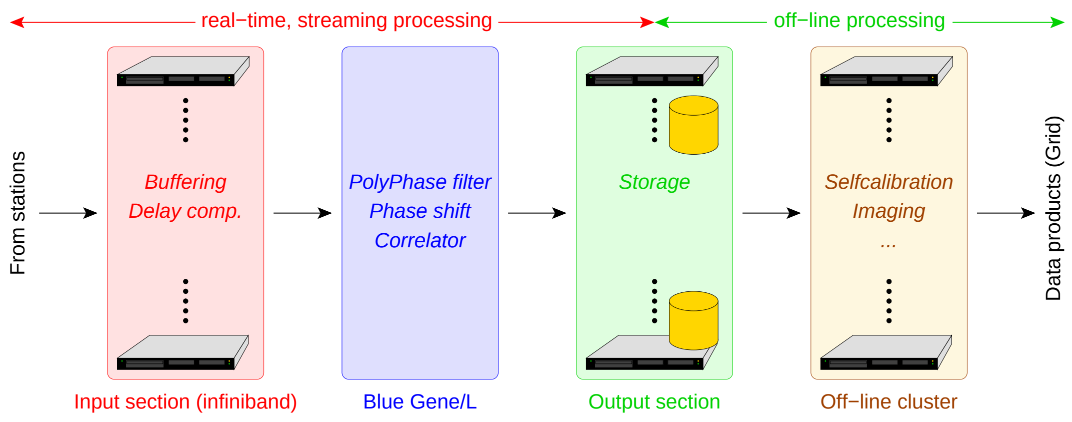

Figure 6: Simplified diagram of the Central Processing Facility.

3.1 I/O to and from the Blue Gene/L

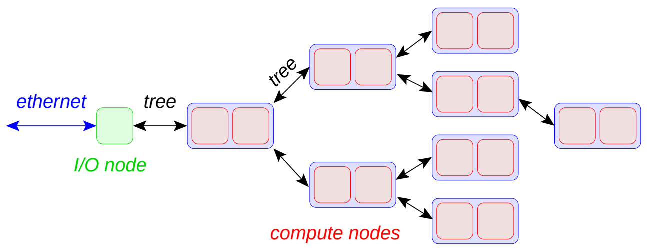

Figure 5: A *Pset*: eight compute nodes behind one Ethernet interface.

The Blue Gene/L is equipped with several types of networks, of which the external Ethernet interfaces and the tree network are the most important ones to us, since they are used to stream data into and out of the Blue Gene/L. Compute nodes communicate externally via an I/O node, that bridges between the tree and Ethernet networks. An I/O node uses the same hardware as a compute node, but has its Ethernet interface enabled, runs another operating system, and does not use the second core. Blue Gene/L was not specifically designed for streaming data communication, but the atypical high number of external network interfaces made it a suitable candidate. To sustain the required high bandwidths, our Blue Gene/L system is configured with the maximum possible number of Ethernet interfaces: each group of 8 compute nodes (16 cores) is connected to one I/O node, as shown in Figure 5. Note that this figure does not reflect the physical structure of the tree, which is in practice irregular and unbalanced. The group of compute nodes behind one I/O node is called a *Pset*. Careful node allocation is necessary to schedule work on the right core in the right Pset. The system has 768 gigabit-Ethernet interfaces in total. For the programmer, the I/O nodes are transparent; the only way to establish communication to an external system is to create a TCP socket on a compute node to a server outside the Blue Gene/L.

Another network, the three-dimensional torus, connects all compute nodes, but not the I/O nodes. We currently do not use the torus.

There are many papers that provide more information on the Blue Gene/L; a special issue of IBM's Journal of Research and Development [5] is an excellent starting point.

4. IMPLEMENTATION

A 12,288-core Blue Gene/L system forms the heart of the Central Processing Facility (see Figure 6). The Blue Gene/L is surrounded by several Linux clusters. Data from the stations flows via a dedicated WAN with hundreds of 1-Gbit links into the *input section*, that runs on a dedicated cluster. The input section buffers the data for ten to twenty seconds (to handle network delays and to be able to react to galactic events) and performs the sample-based delay compensation. An all-to-all transpose over infiniband is performed to reorder the data: the input section receives from each station all subbands, and outputs for each subband all stations. The hardware for the full input section will be installed when the LOFAR stations are built; a small cluster is available for development.

The input section is connected to the Blue Gene/L by gigabit Ethernet. The Blue Gene/L runs the PPF, the phase shift on behalf of the delay compensation, and the correlator. In the original LOFAR design, the PPF was planned to run on dedicated hardware at the stations, but the Blue Gene/L provides sufficient computational power to move it there. With 37 stations (the LOFAR central core plus the first five remote stations), there will be one dedicated Ethernet connection per subband, so that about 20% of the entire Blue Gene/L cluster is needed for a standard, full-band observation.

The correlator output will be stored on a cluster of disks by the output section (a five-hour observation with 37 stations yields 17.5 TB of data; with 77 stations, 75 TB). These data will be calibrated and imaged in the hours that follow by a separate cluster (a computational challenge in itself), but are not part of the real-time processing. Like the input section, the hardware for the output section will be installed when the stations are built.

As can been seen from Figure 2, the data are transposed between all functional units: the data are round-robin distributed over the inputs of the FIR filters; the FFT transforms channels (note the orthogonal direction); each correlator integrates over time, and in the output all channels are combined to facilitate postprocessing. The transposes are hard to implement efficiently, due to cache effects. For each transpose, we tried several implementations to determine the most efficient one; usually it is best to transpose a couple of (but not all) rows at one go and to interleave it with other work. The FIR filters transpose the incoming data on the fly, and the phase shift is done in the transpose between the FFT and correlator, in the latency of the memory reads.

Our application is built on the CEntral Processing Framework *CEPframe* [6] software. CEPframe supports streaming-data applications by providing transparent communication between different types of hosts, as well as configuration management, monitoring, and fault tolerance.

4.1 Work distribution

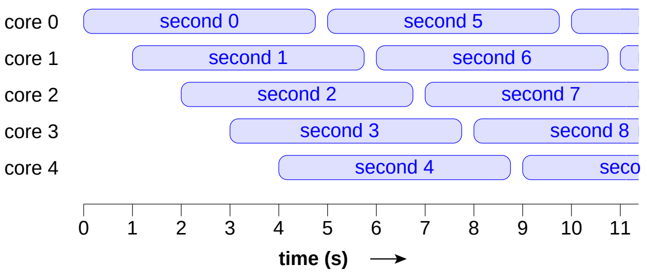

Figure 7: Round-robin work distribution.

We considered several schemes to distribute the work over the Blue Gene/L cores. All schemes share the idea that the subbands are processed independently and that one subband is processed by a single Pset. Since a single core cannot process one second of sampled subband data in one second real time, the work must be divided somehow. We currently use a scheme where each second of sampled subband data is assigned to one core that does all processing for it and runs to completion. Subsequent seconds of sampled data are round-robin distributed over the available cores, as illustrated by Figure 7, where each second of sampled data is sent to one of five cores. In this example, each core needs 4.75 seconds of processing and communication time, and is 0.25 seconds idle. To process the data in real time, the minimum number of required compute cores equals the execution plus communication time on a single core, plus a bit of headroom to recover from temporary stalls (see Section 6). The clear advantage of this scheme is that it is simple. The disadvantage is the relatively high latency to process the data, especially for larger amounts of stations, since the correlator has $O(\#\text{stations}^2)$ time complexity. Low latency is important to react to transient galactic events. Also, the history buffers in the FIR filters have to be filled again for each new second of data, since the history buffers from a previous second reside on a different core, increasing the amount of communication into the Blue Gene/L by almost 2%.

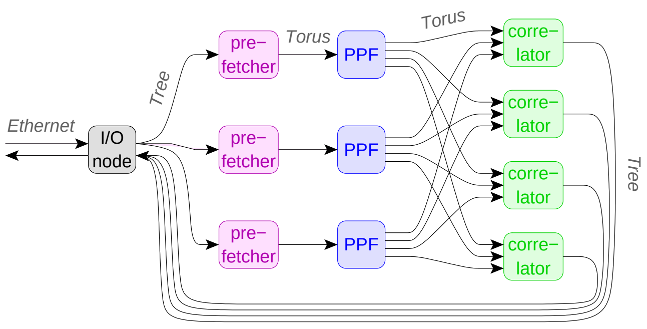

Figure 8: Alternative distribution with specialized workers.

An earlier version of the software distributed the work differently, where each compute core had a specialized task (see Figure 8). Each second of sampled data was partitioned over a number of cores that did the PolyPhase Filtering for their share of the work, and sent their results to another set of cores that did the correlations. The number of PPF cores and correlator cores was chosen in such a way that all PPF cores together required less than one second to receive, process, and send the data, and the correlator cores required another second to receive, process, and send the data. The advantage of this scheme is that the latency is smaller than in the round-robin distribution scheme. However, the scheme has many disadvantages. First, we found that extra prefetcher cores were necessary to read the data from the slow tree network (we will discuss the performance of the tree network in Section 5.2), wasting many cores. Second, the scheme requires a multitude of extra communication, albeit over the fast torus network, wasting even more cores. Moreover, the communication is all to all, and requires senders and receiver to be synchronized. And finally, the software became so complex that it was hard to add new features.

We also considered schemes that can be seen as intermediates of the previously described schemes, but they combine both their advantages and their disadvantages. The decisive argument to continue with the round-robin scheme was the fact that for the EoR observation mode, we cannot afford to waste compute cycles, while latency is not important then. In this observation mode, we strive to observe the sky with 24 bundles from the 32 stations of the central core (possibly extended by the first five remote stations), demanding much more computational power than in the standard observing mode. Thus, we need the round-robin scheme anyway and will use it for the other observation modes as long as there are not that many stations that the latency becomes prohibitive. Moreover, latency can be halved by using the processor as a SMP and having both cores work on one second of sampled data, but we have not found the need to implemented this yet. The granularity of the parallelism is so coarse that the incoherency of the L1 caches will not pose a real problem.

The scheme depicted in Figure 7 shows how a single subband is distributed over five cores. To effectively use all compute cores in a Pset, we extended the round-robin scheduling model so that a Pset can process multiple subbands. The subbands are also round-robin distributed over the cores. For example, in a Pset that processes five subbands, the first second of data from each of the five subbands go to cores 0–4; the second second go to cores 5–9, the third second to 10–14, and the fourth to core 15 and 0–3. Although multiple subbands can be processed by one Pset, the total bandwidth required by these subbands obviously cannot exceed the bandwidth of the Pset's gigabit Ethernet interface.

4.2 Implementation details

For optimal performance, most time-intensive code is written in assembly, since we could not get satisfactory performance from compiled C++ code. We maintain equivalent C++ reference code for testing and portability. The assembly version hides load and instruction latencies, issues concurrent floating point, integer, and load/store instructions, and uses the L2 prefetch buffers in the most optimal way. Most instructions are parallel fused multiply-adds, that sustain four operations per cycle. We optimally exploited the large, 2×32 FPU register file. We also use the PowerPC's ability to influence cache behavior (e.g., the `dcbz` instruction that zeros an entire cache line without reading it from memory); this turned out to be useful in the memory transposes that are performed. The only time-consuming part that is not written in assembly is the FFT; we use the vectorized "Vienna" implementation of FFTW for the Blue Gene/L [3].

An example of an optimization that we implemented is the reduction of memory references by the correlator. This is achieved by keeping correlations that are being accumulated in registers, and by reusing samples that are loaded from memory as many times as possible. A sample can be used multiple times by correlating it

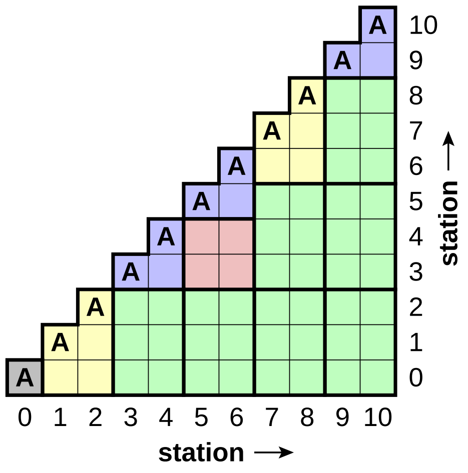

Figure 9: The correlation triangle is divided into as many as possible 2×3 tiles.

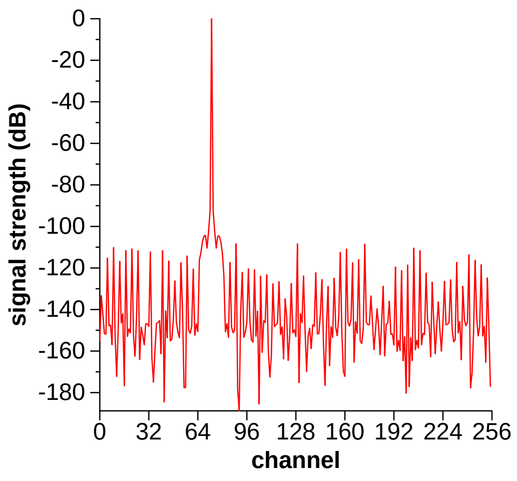

Figure 10: Correlated signal strength of a monochrome signal.

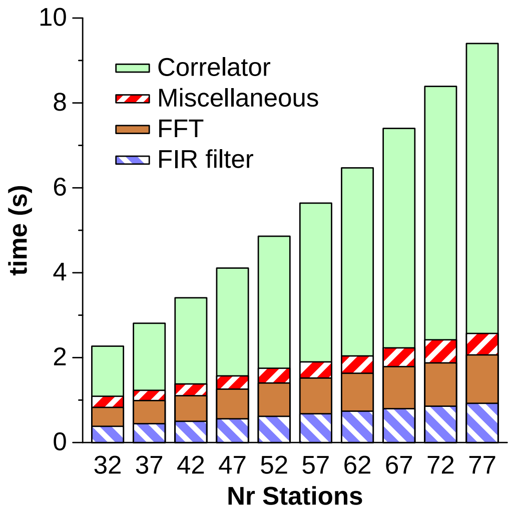

Figure 11: Execution times for several program parts.

with the samples from multiple other stations in the same loop iteration. For example, a sample from station A in the X polarization that is loaded into a register pair can be correlated with the X and Y polarizations of stations B, C and D, using it 6 times. In fact, it is used 12 times, since a correlation requires 2 complex, fused multiply-add instructions. Figure 9 shows how we correlate multiple stations at the same time. Each square represents the XX, XY, YX, and YY correlations of the stations as indicated by row and column number. The figure is triangular, because we only compute the correlation of each pair of stations. The squares labeled "A" are autocorrelations, that are treated specially since they require less computations. The triangle is divided into as many 2×3 tiles as possible, since this offers the highest level of reuse, while still fitting in the register file. These 2×3 tiles are correlated in the same iteration. For example, the lower right-hand-side rectangle correlates stations 9 and 10 with stations 0, 1, and 2. Each of the 6 tiles requires 4 complex registers to accumulate the correlations. With 32 complex register available, there are 8 left to load the X and Y samples from the stations. The correlation of multiple stations in the same iteration also helps to hide the 5-cycle instruction latencies of the fused multiply-add instructions, since the correlations are independently computed.

5. RESULTS

We tested the correctness of the software by inserting a simulated, time-delayed, monochrome signal, which is (under)sampled by the PolyPhase Filter, delay-compensated, and correlated. A 49,665,069.58 Hz complex signal is sampled at a 195,312.5 Hz rate in the band starting at 49,609,375 Hz, resulting in 256 channels of 762.94 Hz wide. Delay compensation is tested by virtually placing two stations at different locations. Figure 10 shows correlations for the two stations. There is a strong peak in the channel where we expect it; its peak is over 90 dB above the digital noise level.

5.1 Computational performance

We measured the computational performance of the PPF and the correlator. Figure 11 shows execution times on a single compute core to sample 1 second of real-time data, for up to 77 stations. The total height of each bar reflects the total execution time. The "miscellaneous" area includes two transposes and delay compensation. Clearly visible is the $O(n^2)$ complexity of the correlator, while the other components scale linearly. For 37 stations, 2.81 seconds are required; for 77 stations, 9.45 seconds.

The correlator is extremely efficient: it achieves 98.1% of the theoretical peak performance of 4 floating point operations per cycle. The FIR filter is at 78% of the peak performance, and is somewhat limited by the number of concurrent L3 cache line reads that can be kept in flight. The time required to perform the phase shift on behalf of the delay compensation is negligible, since it only requires two complex multiplications per sample, which are done in the latency of a memory transpose.

5.2 Communication performance

Since communication performance is of great importance for the streaming-data volumes that we need to process, we measured the I/O performance between the compute nodes and external systems. Remember that the compute nodes are connected to an I/O node via the tree, and that the I/O node bridges between the tree and an external Ethernet interface. We were unpleasantly surprised to find a severe bottleneck between the I/O node and the compute nodes. Despite the high-speed (2.8 Gbit/s) tree links, a single compute node

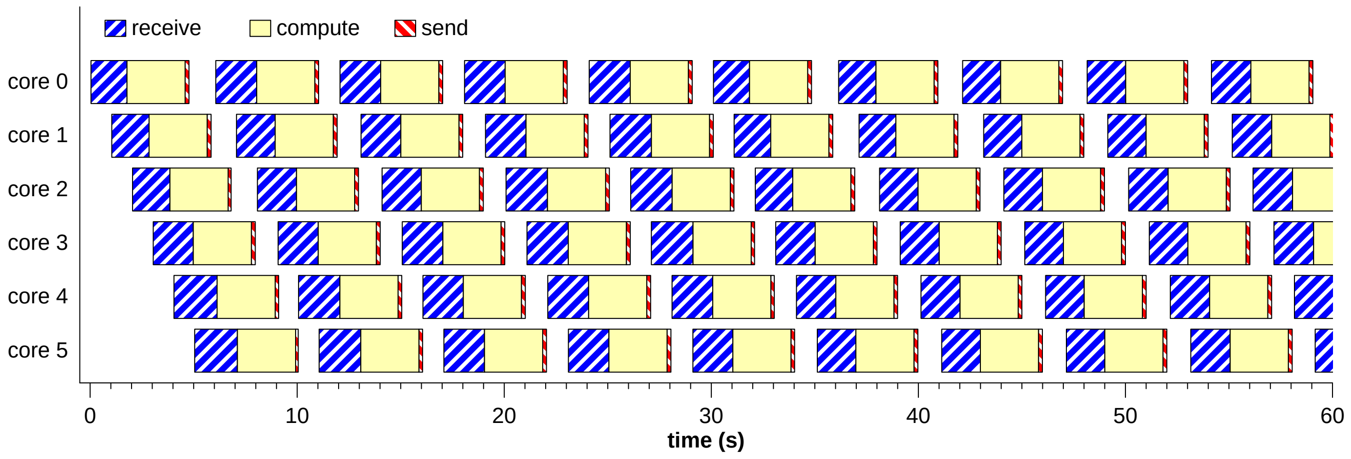

Figure 13: Processing of one subband from 37 stations on 6 cores.

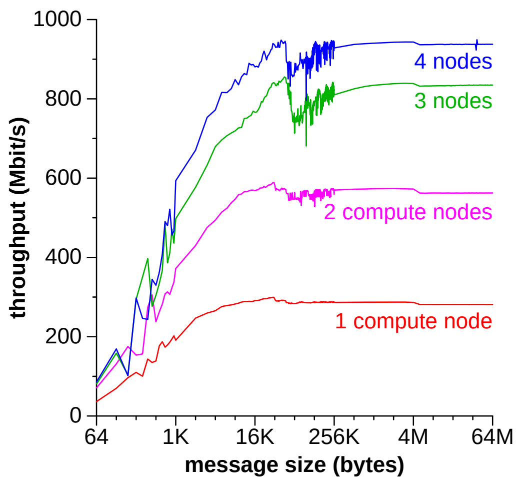

Figure 12: Bandwidth from I/O node to compute nodes.

does not achieve more than 290 Mbit/s, apparently due to an inefficient implementation of the tree API. However, higher bandwidths can be obtained by overlapping communication from one I/O node to multiple compute cores. Figure 12 shows the obtained bandwidth, for up to four concurrently-communicating compute nodes and as function of message size. Note that only the first compute core in each compute node is used; the second core communicates even two to three times slower (this is not shown in the figure).

To process one subband with 37 stations, we need to receive 60 MB/s from the input section and send 6 MB/s to the output section over one gigabit-Ethernet interface. This means that a single compute core cannot meet the real-time communication requirements, and that multiple cores must communicate concurrently. Fortunately, the round-robin scheme automatically overlaps multiple communications, when compute cores communicate slowly.

5.3 Combined performance

Since the stations have not yet been installed, the input section generates simulated data for our performance measurements. We processed one subband with 37 stations, and measured the times that the compute cores started receiving, computing, sending, and idling (see Figure 13). We use 6 (out of 16) compute cores from different compute nodes; the remaining cores are not used in this measurement. Each second, new work with one second of station samples is sent from the input section to one of the cores. A core needs about two seconds to receive the samples from the input section, 2.81 seconds to process it, and about 0.2 second to send the correlations to the output section. The core idles during the remainder of the 6 seconds, waiting for the input section to provide new data. There are always two or three cores communicating concurrently. The round-robin scheme does not need prefetcher cores, unlike the scheme that uses specialized workers. Nevertheless, much time is wasted, since each core needs more than two seconds to communicate data that in theory could have been transmitted in about 0.6 second over the Ethernet interface.

6. DISCUSSION

The Blue Gene/L system is an excellent choice to do the signal processing computations, but the operating system is not optimized for high-bandwidth, streaming computing. We discuss the strong and weak points below.

We found the FPU extensions of the PowerPC 440 core very useful. The variety of parallel fused multiply-add instruction variants (including the ability to swap real and imaginary parts of a complex value) perfectly match the PPF's and correlator's requirements. We also thank the good performance results to the large FPU register files. Our application does not really require the 64-bit precision of the FPUs, but the quad 32-bit-precision AltiVec operations found in other PowerPC architectures are not available on the Blue Gene/L. A weak point is the absence of integer-to-floating-point conversion hardware; it was quite hard to implement an efficient software solution.

The PowerPC 440's L1 data cache uses a round-robin replacement strategy. We found that this is particularly bad in a streaming environment: each incoming sample indiscriminately evicts an entry from the cache. It is therefore not possible to keep a small table in the L1 cache.[3] A strong point is the existence of the `dcbz` instruction, that can be used to avoid reading a cache line from memory when it will be entirely written.

Computationally, the machine performed better than we expected beforehand, but the external connections are not optimized for our purposes. The fact that sufficient bandwidth can only be obtained

[3]The PowerPC 440 has hardware support to lock a small table in the L1 cache, but the operating system does not provide support for this.

by three simultaneously communicating cores instead of one core, effectively wastes two out of six cores. Nevertheless, for the standard observation mode, we have enough cores available to compensate for the slow communication, and still have many cores left for future, additional processing.

The round-robin work distribution scheme is the most efficient scheme, but the latency with which the correlations appear, grows rapidly with the number of stations. For 37 stations, the latency (5.0 seconds) is acceptable, but when LOFAR eventually grows to 77 or more stations, the latency may be too high to react to galactic events. The upper bound on an acceptable latency is, however, currently unknown and depends on future developments in the area of transients detection. The latency can be halved by using the processor in a SMP way. If this is not sufficient, it will be necessary to use another distribution scheme, at the price of higher complexity and less efficiency.

The round-robin scheme automatically recovers from incidental, temporary stalls, for example, when a disk hiccup in the output section temporarily freezes the machine. Flow control causes the processing on the Blue Gene/L to be behind schedule, but the input section can tolerate a ten-second delay without dropping data. Normally, the compute cores are somewhat less than 100% of the time busy; the remainder of the time, and some spare network bandwidth are used to make up arrears. Nevertheless, we make sure that no more than three compute cores receive data over the Ethernet interface; if too many cores communicate, the available bandwidth per core drops below 290 Mbit, the compute cores need more time to communicate, and the problem aggravates.

We successfully demonstrated the processing of a single subband for the most common observation mode. As far as the processing on the Blue Gene/L is concerned, scaling to the full range of 160 subbands is trivial, since all subbands are processed independently. However, the input section needs to distribute all subbands from a station, requiring a transpose over infiniband in the order of ten gigabytes per second. Scaling to more than about 45 stations requires the use of an additional Ethernet interface (and hence a second Pset) to process one subband, since a single gigabit Ethernet interface cannot sustain the increased amounts of data in real time. These interfaces are available.

However, it is not the scaling to many stations that is the most challenging, but the scaling to the EoR observation mode. This mode will observe the sky in 24 directions using the 32 stations from the central core, possibly extended by the first five remote stations. The huge amounts of sampled data require the full Ethernet bandwidth and computational performance of the entire Blue Gene/L machine. Although EoR observations are not planned before late 2007, we have already partially implemented this mode. Initial performance measurements show that we need higher per-core communication bandwidth to process these data in real time. However, developers from IBM have been very helpful in analyzing the communication performance and are working on a new release of the Blue Gene/L software that will show significantly higher per-core bandwidths.

Future developments will focus on new functionality, such as support for new observation modes, and on-the-fly calibration and imaging to detect galactic events.

7. CONCLUSIONS

In this paper, we described a new approach to perform real-time, streaming signal processing for a software telescope on a Blue Gene/L supercomputer. The approach forms an integral part of LOFAR. LOFAR is the first of a new generation of telescopes that does not use large, expensive dishes, but is built as a distributed sensor network of simple antenna receivers. Also, much of the processing is done in software, where dedicated hardware was used traditionally. The use of software increases flexibility and reconfigurability, e.g., the possibility to perform multiple concurrent observations and to change the observation direction instantaneously.

We presented a highly-optimized implementation for the normal observation mode, that involves PolyPhase Filtering, phase shifting, and correlation. The computational performance is excellent: e.g., the correlator achieves 98% of the floating-point peak performance of a Blue Gene/L core. This was not possible without writing the time-intensive program parts in assembly. Since Blue Gene/L was not designed for high-bandwidth, streaming-data processing, obtaining high external communication bandwidth with the current operating system is not trivial and requires multiple simultaneously-communicating cores, which may pose a problem for one of the future (EoR) observation modes. However, normal observations can be adequately handled, and allows LOFAR to grow to 77 stations and beyond.

Acknowledgments

LOFAR is being funded by the Dutch government in the BSIK programme for interdisciplinary research for improvements of the knowledge infrastructure. Additional funding is being provided by the European Union, European Regional Development Fund (EFRO) and by the "Samenwerkingsverband Noord-Nederland", EZ/KOMPAS.

8. REFERENCES

[1] H.R. Butcher. LOFAR: First of a New Generation of Radio Telescopes. *Proceedings of the SPIE*, 5489:537–544, October 2004.

[2] A.G. de Bruyn, R.P. Fender, J.M.E. Kuijpers, G.K. Miley, R. Ramachandran, H.J.A. Röttgering, B.W. Stappers, M.A.M. van de Weygaert, and M.P. van Haarlem. Exploring the Universe with the Low Frequency Array, A Scientific Case, September 2002. http://www.lofar.org/PDF/NL-CASE-1.0.pdf.

[3] J. Lorenz, S. Kral, F. Franchetti, and C.W. Ueberhuber. Vectorization Techniques for the Blue Gene/L Double FPU. *IBM Journal of Research and Development*, 49(2/3):437–446, March 2005.

[4] C. Phillips, T. Tzioumis, A. Deller, S. Tingay, C. Harris, and K. Haines. Electronic Transmission and Computation of Very Long Baseline Interferometry and Its Application to Next Generation Radio Telescopes. Poster in the first IEEE International Conference on e-Science and Grid Computing, December 2005.

[5] J.J. Ritsko, I. Ames, S.I. Raider, and J.H. Robinson, editors. *Blue Gene*, volume 49, number 2/3 of *IBM Journal of Research and Development*. IBM Corporation, March/May 2005.

[6] K. van der Schaaf, C. Broekema, G. van Diepen, and E. van Meijeren. The LOFAR Central Processing Facility Architecture. *Experimental Astronomy*, 17:43–58, 2005.

[7] C.M. de Vos, K. van der Schaaf, and J.D. Bregman. Cluster Computers and Grid Processing in the First Radio-Telescope of a New Generation. In *IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 156–160, May 2001.

[8] Private discussions with people from the Joint Institute for VLBI in Europe, March 2006.