OnLine Application Processing (OLAP) : Software Design Document

Verified:			
Name	Signature	Date	Rev.nr.

Accepted:		
Work Package Manager	System Engineering Manager	Program Manager
John W. Romein ..... *date:*	André W. Gunst ..... *date:*	Jan Reitsma ..... *date:*

Distribution list:

Group:	For Information:
Review committee	**LOFAR Project**
E. van den Heuvel (chair, University of Amsterdam)	Chris Broekema
A. Berg (SARA)	Martin Gels
W. Brouw (University of Groningen)	André Gunst
R. Schilizzi (SKA-ISPO/U. Leiden)	Auke Latour
C.H. Slump (University of Twente)	Ronald Nijboer
A.B. Smolders (NXP Semiconductors)	Ruud Overeem
E. Stolp (Thales Naval Nederland)	Jan Reitsma
	Michael Wise

Document revision:

Revision	Date	Section	Page(s)	Modification
0.0	2005-Aug-04	-	-	Creation
0.1	2005-Sep-09	-	-	First draft
0.2	2005-Sep-23	-	-	Ready for final review
1.0	2005-Oct-03	-	-	Final
2.0	2007-Mar-30	-	-	Rewrite for April 2007 CDR

List of Acronyms:

ACC	Application Configuration and Control
ADD	Architectural Design Document
AIPS++	Astronomical Information Processing System
BG/L	Blue Gene/L
CDR	Critical Design Review
CEP	Central Processing
DMA	Direct Memory Access
EoR	Epoch of Reionization
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
FPU	Floating-Point Unit
GbE	Gigabit (per second) Ethernet
GFLOPS	Giga (10^{12}) Floating-Point Operations per Second
IP	Internet Protocol
ITRF	International Terrestial Reference Frame
LOFAR	Low Frequency Array
MAC	Monitoring and Control
MPI	Message Passing Interface
MS	Measurement Set
OLAP	OnLine Application Processing
PPF	PolyPhase Filter
RFI	Radio Frequency Interference
RSP	Remote Station Processing board
SAS	Scheduling and Specification
TCP	Transmission Control Protocol
TFLOPS	Tera (10^{15}) Floating-Point Operations per Second
UDP	User Datagram Protocol

Abstract

This document describes the design and current implementation of the OnLine Application Processing (OLAP) subsystem, and provides input for the LOFAR System Critical Design Review of April 2007.

1 Introduction

OLAP (OnLine Application Processing) performs all central, real-time processing for LOFAR, and runs on the Central Processor (CEP) in Groningen. Its main tasks are to receive station data, apply delay corrections, split subbands into channels, correlate, and store the results on disk for subsequent (offline) processing. Traditionally, custom hardware is used for these computationally intensive tasks, but OLAP performs all processing in *software*, increasing the flexibility and reconfigurability of the system. A 12,288-core IBM Blue Gene/L supercomputer, embedded in conventional Linux clusters, provides the required processing power.

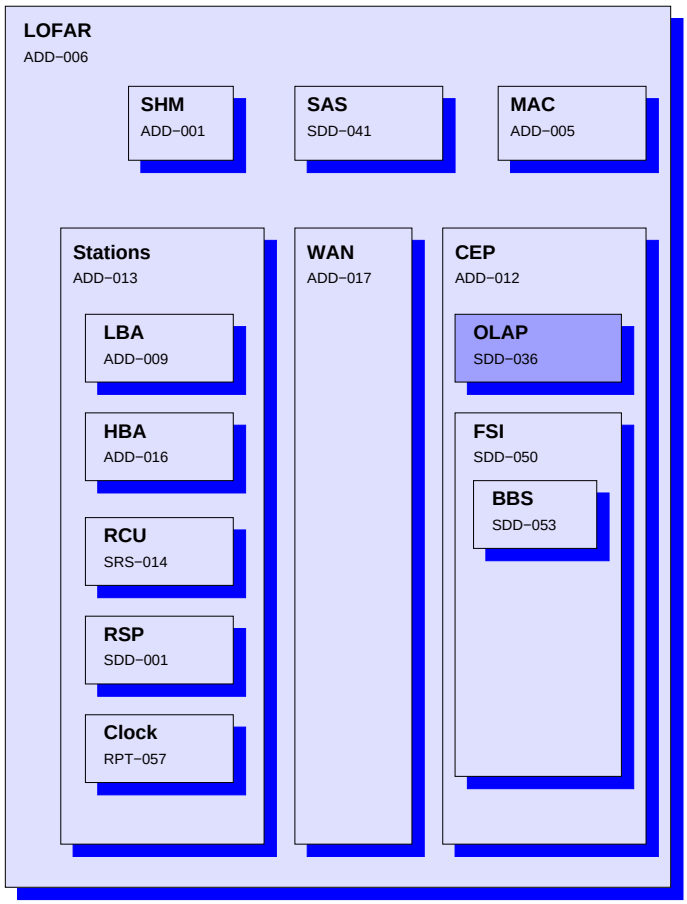

Figure 1: Context of this document.

This document is a rewrite of the OLAP Software Design Document that was used for the CEP Critical Design Review (CDR) held in 2005. It goes beyond a *Software* Design Document, because it describes the hardware architecture as well, and beyond a Software *Design* Document, since it also describes the current implementation. We provide new insights and report on achieved goals, current developments, and actual issues. Parts of this document are copied from a conference paper on the Blue Gene/L processing [4]. The document's context is depicted in Figure 1: OLAP is part of the CEP, for which there is a Architecture Design Document (ADD) [5], and CEP is part of LOFAR, for which there is a separate ADD [1].

At the time of writing, LOFAR CS1 is being built, which is a small subset of the eventual LOFAR system. CS1 comprises four partial stations (the final LOFAR will have 77 or more stations) and a reduced number of subbands. We will refer to CS1 at several places in the remainder of the document.

1.1 Description of different sections

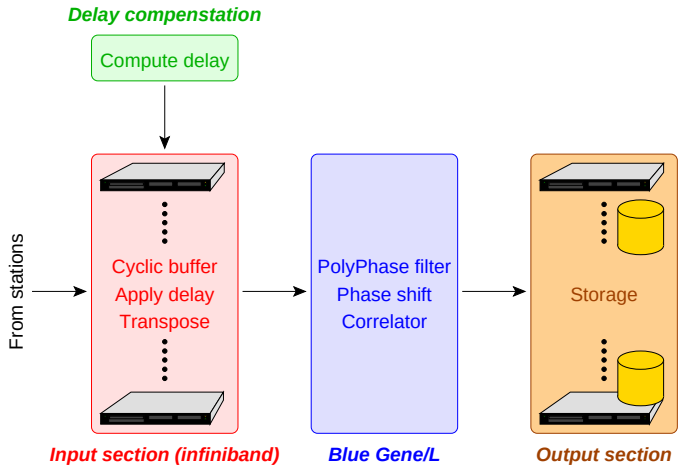

Figure 2: The four OLAP sections.

OLAP consists of the following four sections (see also Figure 2):

- *Delay Compensation*
  The Delay Compensation unit calculates the difference in traveling time that geographically distributed stations experience when receiving a wave from the observation direction. The Input Section and Blue Gene/L correct the received signals for these delays, so that they can be correlated coherently.

- *Input Section*
  The Input Section receives subband data from the stations. It provides a small (some tens of seconds) data buffer to synchronize the station data and to handle temporary delays in the processing pipeline. It applies part of the delays as computed by the Delay Compensation section. Additionally, it reorders the data so that

the data can be processed more easily by the next section. It runs on a cluster of off-the-shelf server-class PCs, connected by a high-speed Infiniband network.

- *Blue Gene/L Processing*
  This section splits the subband data into frequency channels using a PolyPhase filter, and applies the remainder of the computed delays to the data. Subsequently, the channel data are correlated. Due to the high computational requirements, it runs on a Blue Gene/L supercomputer.

- *Output Section*
  The Output Section receives the visibilities from the correlator and writes them to disk as AIPS++ Measurement Set. It runs on commodity hardware.

1.2 Interfaces

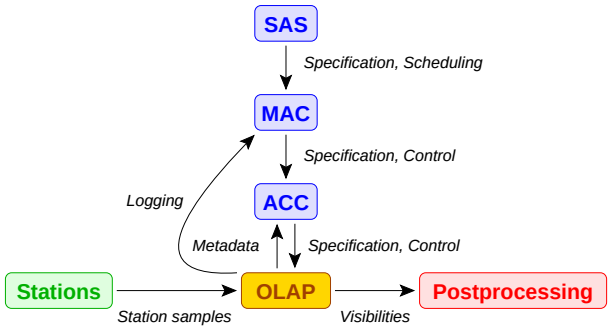

Figure 3: Position of OLAP within other subsystems.

The OLAP applications interface with several other subsystems, as illustrated by Figure 3 and explained in the following sections.

1.2.1 Interfacing with the stations

Each station locally combines the signals from its antennas and sends samples via a dedicated wide-area network to OLAP. 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.

The stations support sampling frequencies of 200Mhz and 160MHz. By filtering, each station divides the spectrum into 195 KHz resp. 156 KHz-wide independent subbands, resulting in 195312.5 or 156250 samples per subband per polarization per second per station. Each station will send up to 160 resp. 200 selected subbands, possibly from multiple observations. Thus, a (noncontiguous) band of up to 32 MHz wide can be monitored during an observation. Currently, the stations can successfully send 39 resp. 49 subbands over the wide-area link to the Input Section, pushing the Gigabit-Ethernet interfaces of the Input Section nodes to 988 Mbit/s.

1.2.2 Postprocessing

In the days after an observation, the data will be postprocessed by several applications, typically the Flagger, Selfcalibration, and Imager. Although this is not a real-time process, there is storage space available for only a few days, so the offline processing must keep pace with the online processing since the online processing continues generating data from subsequent observations. In the full LOFAR system, the available storage will be in the order of a Petabyte, but since a 77-station, 5-hour observation already yields 75 Terabytes of data, the data cannot be retained for a long period. As the postprocessing applications currently use AIPS++, the visibilities are stored on disk in the AIPS++ Measurement Set format. In the future, a data format tailored to large data volumes may become necessary.

1.2.3 System Control

The OLAP applications will be controlled by SAS (Scheduling and Specification) and MAC (Monitoring And Control). SAS schedules an observation and schedules the resources needed for the observation. SAS informs MAC, which starts and stops the OLAP applications at the right moment through ACC (Application Configuration and Control). ACC provides the applications with information about the observation (e.g., the beam direction and the subband numbers) through a parameter file. The application can send some monitoring information directly to MAC using the so-called property interface, which writes the monitoring values directly into the MAC real-time database. We are currently integrating the OLAP applications with SAS and MAC.

1.3 Delay Compensation

In the following four sections, we describe each of the OLAP sections in more detail. We start with Delay Compensation.

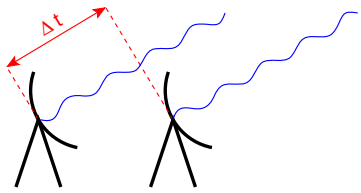

Figure 4: The left antenna receives the signal later.

Since light travels at a finite speed, the signal from the observed source does not reach the different stations at the same time, as shown by Figure 4. To coherently correlate two signals (see Section 3.2.3), the signal from one of the receivers must be delayed to compensate for the difference in travel time. The delay depends on the positions of the receivers and on the observation direction. 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 Section 2.2). This amount represents a multiple of 1/195312.5 or 1/156250 second (depending on the sample rate). The second stage is performed later and compensates for the fraction of time that remains, by shifting the phase of the signal, also known as fringe stopping. Note that delays can be negative (when the station receives a signal later than the reference point), "speeding up" the signal of a station, which is possible by using appropriate buffering techniques. An observation with multiple beams uses different delays for each beam.

The station positions and observation direction are converted to ITRF. This has the advantage that we can now easily calculate the path length difference for the different stations by computing the inner product of the stations position vector and the source direction. The path length difference divided by the speed of light in vacuum yields the time difference.[1]

The delays are not computed by the Input Section but by a separate application, since we use AIPS++ tools for the conversions of different coordinate systems and we do not want to use the AIPS++ libraries everywhere in the OLAP applications (for the Blue Gene/L, that is not even possible). The Delay Compensation program uses the observation direction, observation time, and the station positions to generate a list of time delays, one for each station. Each list entry contains a time delay (in seconds) that must be applied at a specific moment. The Input Section applies the entire-sample delay; the remainder of the delay is performed by the Blue Gene/L, right after the PolyPhase filter. The Delay Compensation application pre-calculates delays for a longer period, hence reducing the number of times that the delays are sent to the other OLAP applications.

The stations use the same mechanism to form beams from the geographically spread dipoles. The signal of each dipole is delayed or sped up with respect to the geographical center of the station (but could have been chosen anywhere), which we call the *phase center*. Thus, the dipole signal is delayed (or sped up) as if it were received at the phase center. OLAP corrects the delays with respect to the phase centers of the different stations. It does so by taking an arbitrary station as reference point, and delaying the signals from the other stations.

CS1 defines *microstations* (partitioned stations) as a temporal solution to create more baselines. There was confusion about the definition of the phase centers and physical positions of the microstations. We now define that all microstations within a station share the same phase center (this is a constraint enforced by the stations), but that their physical locations are used to determine the right UVW coordinates, which are written in the Measurement Sets. OLAP does not treat microstations and stations differently; microstations are handled as different stations that happen to have the same phase center and different geographic positions.

[1] We do not have to compensate for the fact that the speed of light is different in air than in vacuum. Although the sources apparent direction will change due to refraction, the time delay will be the same as the effect of the decreased path length will be canceled by the decreased speed of light. The stations beam steering algorithm, however, does have to take this effect into account.

2 The Input Section

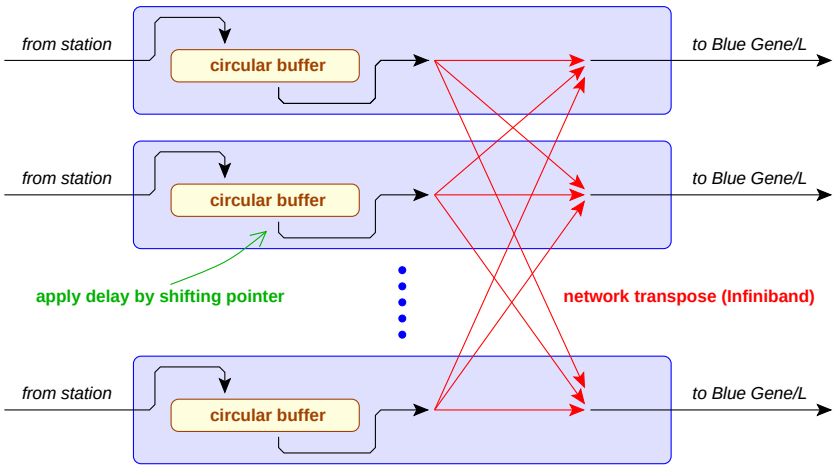

Figure 5: Structure of the Input Section.

The Input Section performs the following major tasks:

- It receives the data from the stations.
- It checks the validity of the data packets, and administrates bad or missing data to inform subsequent processing steps.
- It synchronizes the different input streams.
- It performs coarse-grained delay compensation.
- It reorders the data for subsequent processing.
- It sends data to the Blue Gene/L.

The structure of the Input Section is depicted in Figure 5. Below, the tasks are described in more detail.

2.1 The station interface

The station firmware supports sampling frequencies of 200Mhz and 160MHz (but not at the same time). This yields 195312.5 (or 156250) samples per subband per polarization per second. The stations currently send up to 39 (49) selected subbands to the Input Section. The station sends data using UDP/IP, an unreliable datagram protocol. Enforcing reliability requires too much computational, memory, and bandwidth resources, while in practice hardly any data are lost. Previously, the stations sent raw ethernet instead of UDP packets, but raw ethernet is hard to

route and requires a complex packet filter at the Input Section nodes, while the overhead of UDP is small. When remote LOFAR stations will be built that route data over public networks, we will even be forced to use UDP. The format of the UDP packets are described in the “RSP - CEP Beamlet Data Interface” document [2].

Each Input Section node receives all subbands from one or two stations. For a full LOFAR, multiple Gigabit-Ethernet interfaces per station are needed to handle 160-200 subbands.

2.2 Buffering and data check

To handle temporary stalls in the processing pipeline, the Input Section uses a cyclic buffer to store subband samples from each station. Some tens of seconds can be buffered, after which the old data are overwritten by new samples. One thread receives UDP packets from the stations and stores the samples in the cyclic buffer. This thread handles missing, duplicated, and out-of-order UDP packets. Missing data appears to the remainder of the processing pipeline as flagged dummies, so that they are not correlated and eventually appear as flagged visibilities in the measurement set. The Input Section uses an efficient algorithm to administrate missing data. Since flagged data are typically clustered, we use a *sparse set* datastructure. It maintains which ranges of data are flagged using a list of tuples that mark the beginning and end of each flagged range.

Another thread waits until the next second of data has been collected in the circular buffer, before reading the data out of this buffer. If data is requested that is not anymore in the cyclic buffer (due to very long stalls in the processing pipeline), flagged dummies are returned. This thread also applies the coarse-grained delay compensation by shifting the data an entire number of samples (although for CS1, the stations are so close to each other that delays do not result in shifted samples).

2.3 Reordering the data

The thread that reads the data from the circular buffer, reorders the data for further processing, collectively with the other Input Section nodes. Before the reordering, each data stream contains *all subbands* from *one station*, but the correlator needs *one subband* from *all stations*. This transpose step relies on the high speed of the Infiniband network.

Recently, we changed the Input Section to reorder the data using a collective `MPI_Alltoallv` operation, rather than using point-to-point CEPframe [6] streams. The advantages are that the network transpose is twice as fast (since each MPI vendor heavily optimizes `MPI_Alltoall` collectives for the network architecture it runs on, in this case Infiniband), that we use fewer processors, and that the program code has become simpler.

3 Processing on the Blue Gene/L

A six-rack IBM Blue Gene/L system provides the bulk of the computing power at the Central Processing site. This supercomputer, consisting of 6,144 custom dual-core processors, has a floating-point peak performance of 34.4 TFLOPS. For normal observations, up to four of the racks will be used for the computationally most-intensive tasks (EoR observations require all racks).

We first describe the Blue Gene/L system, followed by the three tasks we run on it: the polyphase channel filter, fringe stopping, and the correlator. We also show some test results and performance measurements.

3.1 Blue Gene/L system description

The Blue Gene/L is a massively-parallel supercomputer based on System-on-a-Chip components. Each Blue Gene/L compute node consists of two PowerPC 440 cores running at 700 MHz, keeping the power dissipation moderate and allowing very dense packing. Each of these cores is extended by two 64-bit floating-point units (FPUs). Each FPU can sustain one fused multiply-add instruction per cycle, giving the core a theoretical peak performance of 2.8 GFLOPS. The FPUs can read each other's registers and have instructions that operate on complex numbers, which is a big advantage for signal processing tasks. Each core has a 32-KB (noncoherent) L1 cache and a (coherent) L2 prefetch buffer. Both cores share a 4-MB L3 cache and 512 MB DRAM. Our system consists of 12,288 compute cores, providing 34.4 TFLOPS peak performance.

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, internal communication. Note that external communication is synchronous in either mode. We prefer virtual node mode, because it doubles the floating point performance.

3.1.1 External I/O on the Blue Gene/L

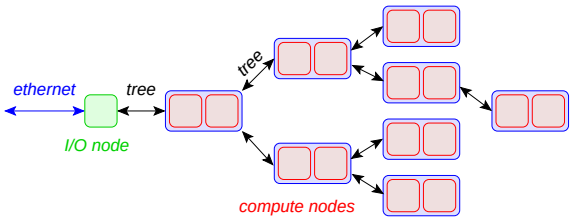

Figure 6: A *Pset*: eight compute nodes behind one ethernet interface.

The Blue Gene/L is equipped with several types of networks. It is connected to the outside world using conventional Gigabit-Ethernet (GbE) technology. Compute nodes communicate externally via an I/O node, that trans-

parently bridges between the internal tree network and external GbE interface. An I/O node consists of the same hardware as a compute node, but has its ethernet interface enabled, runs another operating system, and does not use the second core, due to the incoherent L1 caches. The Blue Gene/L was not specifically designed for streaming-data communication, but the atypically 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 GbE interfaces: each group of 8 compute nodes (16 cores) is connected to one I/O node, as shown in Figure 6. 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*. The system has 768 GbE interfaces in total. Currently, only a small fraction of the Blue Gene/L has been connected in this I/O-rich configuration; the remainder will be connected when the bandwidth is required, waiting for GbE switch prices to drop.

The compute-node kernel implements a very basic TCP/IP stack, and transparently forwards all I/O-related system calls to its I/O node. The compute cores can only create client sockets, not server sockets. For our purposes, this is hardly a limitation. Socket options cannot be set: `setsockopt()` is not implemented, and asynchronous communication is not available. This implies that communication and computations cannot be overlapped, but the lack of DMA hardware makes this infeasible anyway.

The 3D-torus is another internal network, that connects all compute nodes, but not the I/O nodes. Each core is connected by 6 bi-directional, 1.4 Gbit/s links, making it a fast network. Many MPI calls use the torus network, but some collective operations use the tree network, and the barrier operation uses a special barrier network. We currently do not use the torus, but may do so in the future.

There are many papers that provide more information on the Blue Gene/L; a special issue of IBM's Journal of Research and Development [3] is an excellent starting point.

3.2 Blue Gene/L processing

The application on the Blue Gene/L consists of three functional units: a polyphase filterbank, a phase shifter on behalf of delay compensation, and the central correlator. The dataflow from a single subband is shown by Figure 7. The application receives time-series data from each station via the Input Section. The polyphase filterbank splits these subbands into 256 frequency channels. Subsequently, the phases of the channel samples are corrected. Finally the data are correlated, and sent to the Storage Section. Nearly all processing is done using floating point arithmetic, because the floating-point performance of the BG/L is significantly higher than the integer performance. Coincidentally, it leads to more accurate results. Unfortunately, the data is transposed in memory between nearly all functional units (note that the FFT is orthogonal to the data stream), which complicates an efficient implementation due to undesirable cache behavior.

The most time-critical parts of the application have been written in assembly. This was necessary to achieve adequate performance; C++ code turned out to be 4 to 12 times slower. For portability and testing purposes, we maintain equivalent C++ code. Unlike thought previously, we are quite positive that the application is fast enough to process the computationally most intensive (EoR) observation mode, provided that we succeed in increasing the

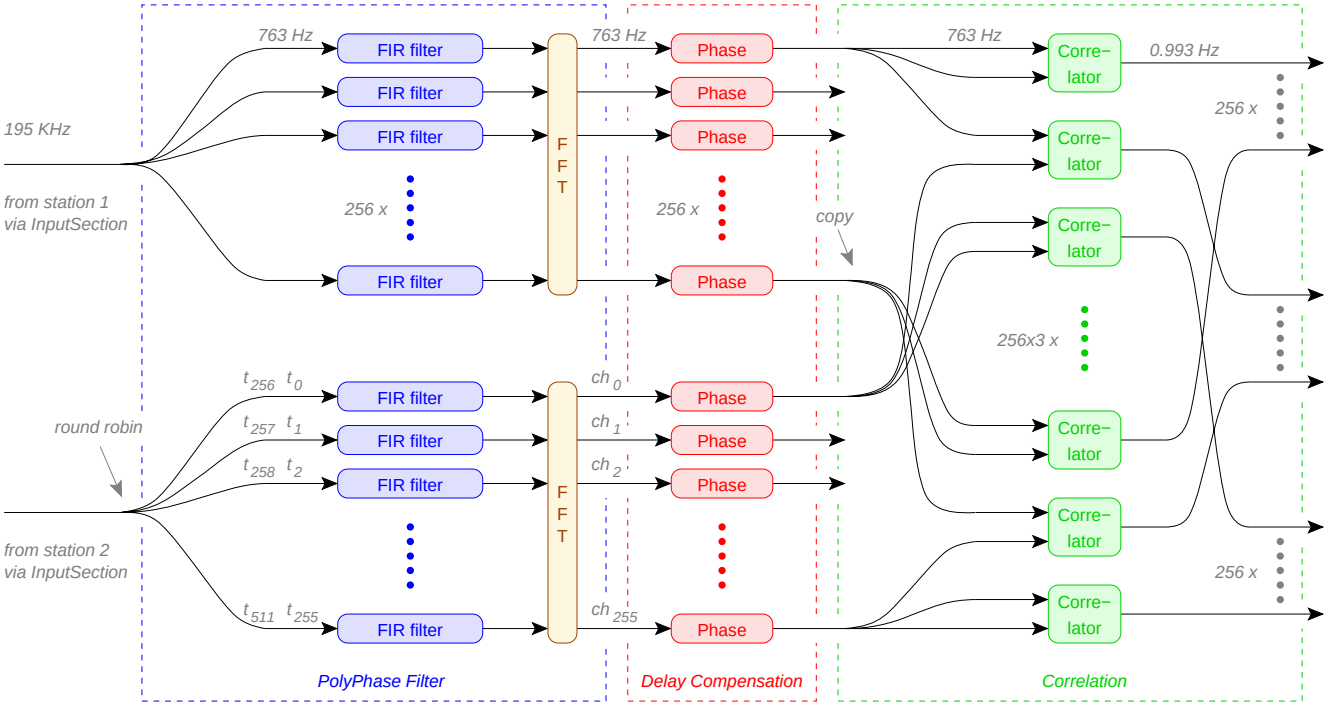

Figure 7: Data flow through the Blue Gene/L.

network bandwidth into the Blue Gene/L (see Section 5.1).

The processing on the BG/L has been described in more detail in a conference paper[4], which includes an in-depth analysis of the performance measurements.

3.2.1 The PolyPhase Filter

The first processing block on Blue Gene/L splits a 195 KHz (or 156 KHz) subband data stream into 256 frequency channels of 763 Hz (610 Hz) each. Splitting the data into narrow frequency channels allows the offline processing to flag narrow-band RFI, so that unaffected channels remain usable. The PolyPhase Filter (PPF) itself consists of 256 Finite Impulse Response (FIR) filters, the outputs of which are Fourier transformed (see the respective boxes in Figure 7), as explained below. The (coarse-grain delayed) stream of station samples is round-robin distributed over the FIR filters, and a FFT over the FIR filter outputs yields 256 frequency channels.

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 8). Each clock tick, a sample goes in, shifting history data to the right. A weighted sum of the 16 taps goes out. Each of the 256 FIR filters has its own weight vector, that together determine the PolyPhase Filter characteristics. The FIR filters significantly suppress a signal leaking into the wrong frequency channels.

The FIR filter converts the 16-bit, little-endian integer input to big-endian floating point (in software, since no

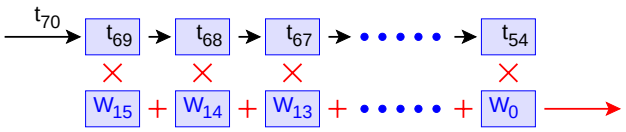

Figure 8: A 16-tap FIR filter.

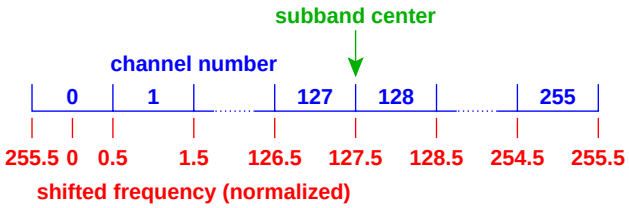

Figure 9: Half-channel frequency shift.

hardware integer-to-floating-point conversion instruction is available). To ensure optimal performance, the FIR filter and 256-point FFT have been written in assembler. This allows for the most direct and efficient control over the execution flow inside the Blue Gene/L processors, at the cost of significantly increased complexity of the code.

The FFT in the polyphase filter effectively shifts the frequency of the entire subband by the width of half a channel, as illustrated by Figure 9. This has two consequences. First, the rightmost half-channel in the subband is folded into the first channel, that also contains the leftmost half-channel. Thus, the first channel contains signals from distant frequencies, and becomes invalid. This is not a real problem, since the PolyPhase subband filter at the stations already significantly attenuates the signals at the edges of the subband (to prevent leaking into other subbands), making them unusable anyway. OLAP always flags channel 0 of each subband. The second consequence is that the frequency at the center of the subband changes. Since we always use the center of a subband (or channel) to define its frequency, OLAP subtracts half a channel width. For example, the 60,000,000 Hz subband from the stations, becomes the 59,999,694.82 Hz subband after filtering (assuming a 160 Mhz clock at the stations). The frequency shift is also visible in the Measurement Set produced by the Storage Section.

3.2.2 Fringe stopping

As soon as the subband is split into frequency channels, the fractional delay $\Delta\tau$ as a result of delay compensation is applied to the data by multiplying the phase of the signal by $e^{-i2\pi f\Delta\tau}$, where f is the channel frequency. The phase shift depends on time and frequency. The Delay Compensation section computes the fractional delays for the first and last sample of each integration period (one second) and sends these delays via the Input Section to the Blue Gene/L section (note that this is an insignificant amount of data). The Blue Gene/L section interpolates the delay for each sample, computes the channel-dependent correction factor, and applies this to the sample. Thus, the phase of each sample is multiplied by an unique correction factor. Note that the phase correction does not take additional run time, as all computations are hidden by the latency of the memory transpose between the FFT and the correlator.

3.2.3 The Correlator

The most important component of the OLAP applications is the correlator. The correlator calculates the auto and cross correlations between all pairs of stations, for each channel and for each of the XX, XY, YX, and YY

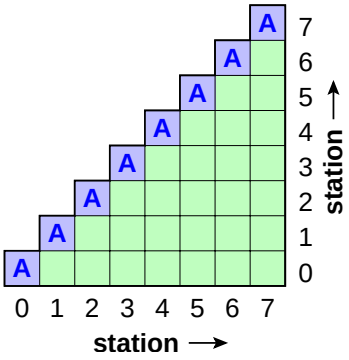

Figure 10: A correlation triangle.

polarizations. A correlation is the complex product of a sample of one station and the complex conjugate of a sample of the other station. The results are integrated (accumulated) over one second of data. Since the correlation of station S_1 and S_2 is the conjugate of the correlation of station S_2 and S_1, we only compute the correlations for $S_1 \leq S_2$. The final product consists of a visibility triangle for each of the channels and polarizations (see Figure 10). The entries marked A are auto correlations; the others are cross correlations. The output data rate of the correlator is significantly lower than the input data rate.

To achieve optimal performance, the correlator consists of a mix of both C++ and assembler, with the critical inner loops written entirely in assembler. The assembly 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. The paper on the BG/L processing [4] provides details on the implemented optimizations needed to obtain this performance.

3.3 Work distribution

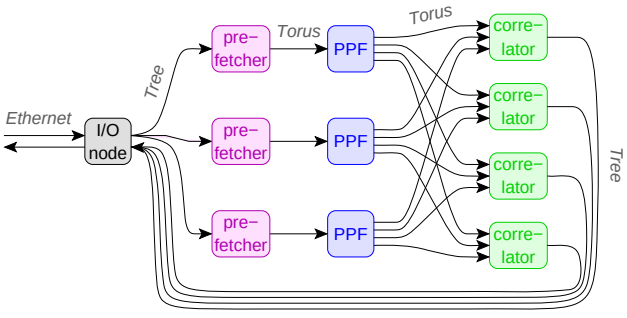

Figure 11: TeraFlop Correlator work distribution.

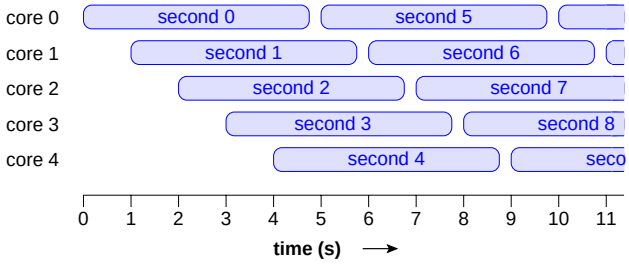

Figure 12: Round-robin work distribution.

The new Blue Gene/L application distributes the work quite differently from the old TeraFlop Correlator, which was described at the previous CDR. The TeraFlop Correlator used separate compute cores for the PolyPhase Filter and the correlator (see Figure 11). This resulted in a huge amount of communication between the compute cores over the 3D-torus network inside the BG/L. Despite the high speed of this network, this scheme has serious scaling

problems, and is not fast enough to process EoR observations.

In the new application, each core is involved in *all* kinds of processing. A core receives one second (the integration time) of subband samples, filters, phase-corrects, and correlates the data, and sends the visibilities to the Storage Section. Since processing the data generally requires more time than is available in real time (e.g., processing one second of subband data from 32 stations requires approximately two seconds of processor time, plus time for communication), multiple cores are used to process the data from one subband. The first second of data is processed by the first core, the second second by the second core, etc., and wraps round after the last core, as illustrated by Figure 12. Usually, we apply this form of scheduling on a per-Pset base, and distribute the data round robin over the cores within a Pset.

The subbands that need to be processed are evenly divided over the available number of Psets. The application supports complex configurations. For example, if 24 subbands must be processed using 8 Psets and 16 cores per Pset, each Pset processes 3 subbands. Each second, the three next cores within a Pset are selected and start processing one subband each. The 16 cores are thus jointly responsible for processing three subbands. In the future, when large amounts of stations need to be processed, the situation will be reversed: multiple Psets will be required to process one subband. This is necessary because a single GbE interface will then not provide enough bandwidth to communicate all data.

The advantage of the round-robin scheme is that it is much more efficient than the TeraFlop correlator, because no communication between the compute cores is necessary. Moreover, it resulted in simpler program code, despite the complex scheduling. Since each core receives its input directly from an Input Section node and sends it output directly to an Output Section node, the 3D-torus network is not used. A small disadvantage is that each second, the FIR filter history buffers must be filled with the last 15 samples from the previous second, but the increase in the amount of external communication (2%) does not compare to the 100% decrease of internal communication. In Section 5.5, we discuss an issue concerning the latency with which the correlator generates output.

Careful node allocation is necessary to schedule work on the right core in the right Pset. We prevent that many compute nodes want to communicate data over the same ethernet device concurrently, as they would have to share the bandwidth, making hardly any progress. We also make sure that outside the Blue Gene/L, data travels over well-defined paths. For example, data going from the Input Section to the Blue Gene/L passes through only one switch chip within only one GbE switch, avoiding contention within or between GbE switches (we had to open a GbE switch to figure out its internal structure). SAS knows these paths and will use this knowledge to schedule the hardware for an observation. Note that, to some extent, it is possible to use different routes, so that it is easy to use a spare machine to replace a broken computer, but we avoid excessive packet switching within or between ethernet switches.

3.4 Test 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

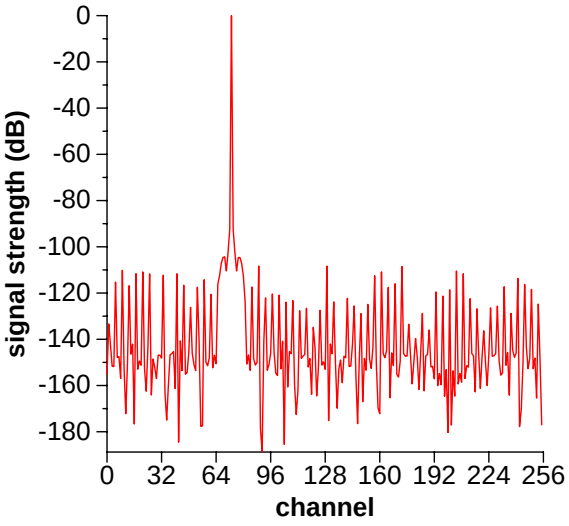

Figure 13: Correlated signal strength of a monochrome signal.

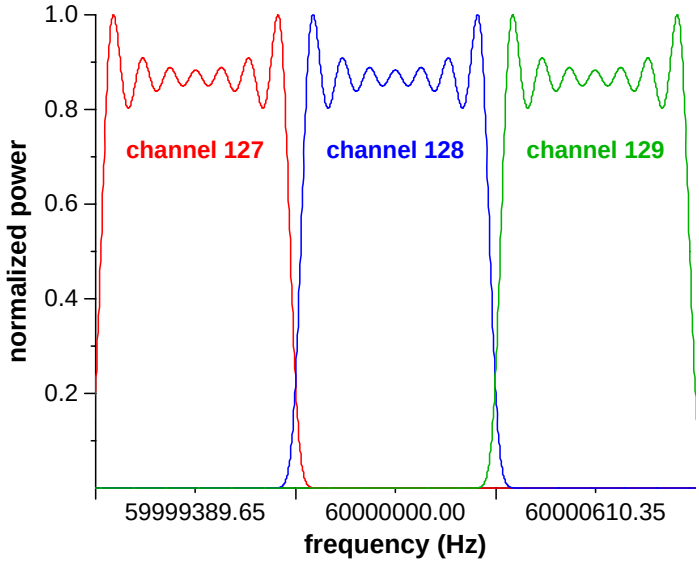

Figure 14: PolyPhase Filter response.

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 13 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.

The filter is not equally sensitive to all frequencies in a channel. Figure 14 shows autocorrelations as function of frequency of a monochrome signal for three different channels (the channel width in this figure is 610.35 Hz). A clear ripple is visible on top of each channel. Leaking into neighboring channels is limited, at the cost of a high insensitivity at the borders of a channel. A filter with different characteristics is straightforward to implement, as long as there are 256×16 FIR-filter constants.

3.4.1 Performance

We measured the computational performance of the PPF and the correlator. The correlator is extremely efficient: it achieves 98.1% of the theoretical peak performance of 4 floating point operations per cycle. Figure 15 shows execution times on a single compute core to process one second of 195 KHz real-time data, for up to 77 stations. The total height of each bar reflects the total execution time, excluding communication. The “miscellaneous” area includes two memory 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.

Early on we discovered that the performance of the external I/O needs careful consideration. Fortunately, stability and bandwidth have improved in the more recent kernels. Figure 16 shows throughputs as function of message size, obtained with a simple benchmark program, using 16-byte aligned data. The graph shows an unusual peak at 64K, indicating that it is better to receive a large message in chunks of 64K. Unfortunately, we have not been able

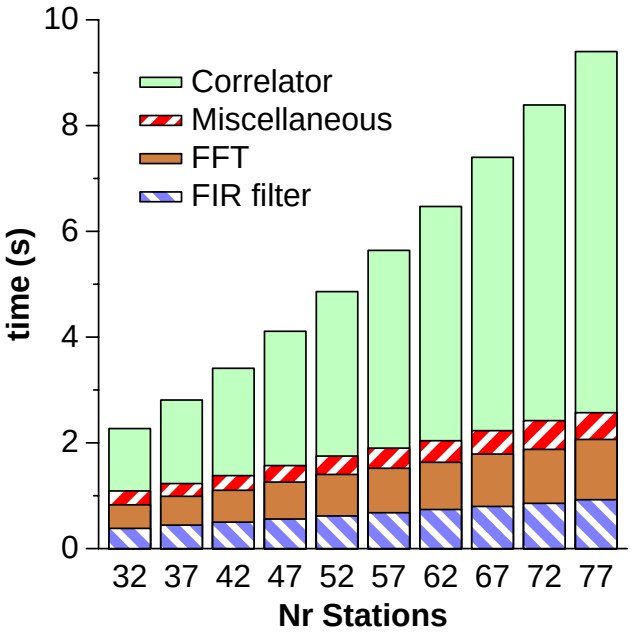

Figure 15: Execution times for several program parts.

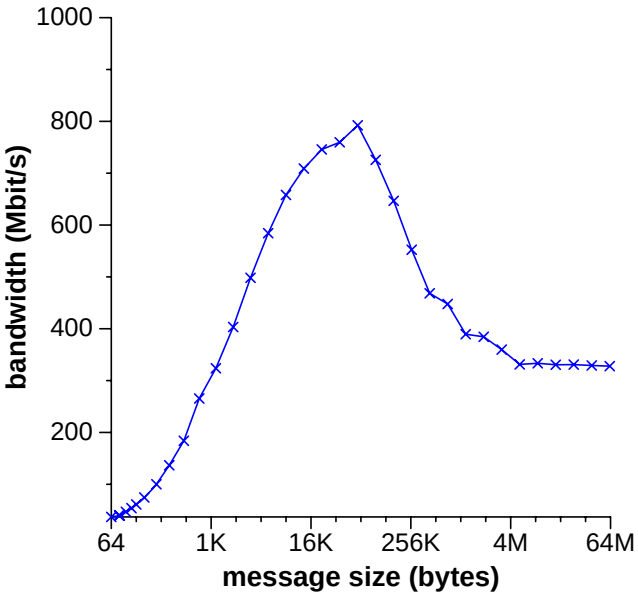

Figure 16: Obtained bandwidth as function of message size.

to achieve these bandwidths in the real application, which exhibits more complex communication patterns. We changed our application so that the kernel can transfer the data more efficiently (e.g., receive all data in chunks of 64K and align the data on 16-byte boundaries), but performance problems arise if we try to sustain bandwidths of more than 500 Mbit/s per GbE interface. Since most, if not all observation modes are bandwidth bound, the amount of required resources (Psets) is fully determined by the obtained bandwidth, while the allocated compute cores are mostly idle. EoR observations require higher bandwidths as well, although this mode has a better balance between computations and communication. In Section 5.1, we elaborate on a new approach to address the bandwidth problems.

4 The Output Section

The data from the correlators are sent to the output section, and written into AIPS++ Measurement Sets (MS). This allows postprocessing using standard tools. For current CS1 observations, we integrate 60 seconds of data in the Output Section, to reduce the MS sizes and to increase the signal-to-noise ratio (the small amount of installed antennas do not provide enough sensitivity to make smaller integration times useful).

The Output Section runs on several nodes. Each node combines multiple data streams from the Blue Gene/L and writes a MS that can contain multiple subbands.

Each row of visibilities is associated with a weight that reflects the number of used station samples over which is integrated. Data flagged by the Input Section can lower the number of samples and thus the weight. The weight

provides a measure for the quality of the visibilities, which can be used for calibration.

In the future, the output size will considerably grow (due to an increased number of stations, more subbands, and shorter integration times), and another file format may be necessary.

5 Discussion and future work

This section describes future plans for the OLAP applications and issues that deserve special attention.

5.1 Blue Gene/L external communication

The external bandwidth to and from the Blue Gene/L is still not sufficient for full LOFAR operation, and is the limiting factor in the entire processing pipeline. Together with Argonne National Laboratory (ANL), we are developing a new network protocol that runs on the I/O nodes and compute nodes, and speeds up external communication. This is possible since we are now able to run programs on the I/O nodes and we now understand the (undocumented) tree network protocol that internally connects all BG/L node (this was not possible or foreseen when the BG/L was acquired). A current prototype shows already a modest performance improvement over the IBM socket layer.

Performance measurements suggest that the 700 MHz PowerPC 440 core of the I/O node is fast enough to sustain 1 Gb/s over the GbE device or over the internal tree network, but not both at the same time. Thus, more computational power in needed to make a more significant bandwidth improvement. We plan to adapt the Linux kernel to enable the second, unused processor core on the I/O node, doubling the available processing power. This is quite complicated, since the L1 caches of both cores are not coherent. The second core will be used to offload functions that read and write from the tree network. The work will also be done in cooperation with ANL.

We also plan to use the I/O nodes for simple operations on the data (such as adding visibilities computed by different compute nodes, which is necessary for 10-second integration times of the EoR), to reduce the amount of external communication.

5.2 Reducing the Input Section size

The collective reordering step in the Input Section is a challenge for the full LOFAR system, since it involves switching much data. We seriously consider moving the reordering step to the Blue Gene/L for EoR observations, since this would reduce the size of the Infiniband switch of the Input Section by a factor of three (compared to normal observations), saving in the order of 100,000 Euros. Experiments with the 3D-torus network of the BG/L showed that it is sufficiently fast to reorder the data. For normal observations, we may want to keep the transpose on the Input Section, for flexibility reasons. However, this flexibility is not needed for EoR observations, since

there will be no processing power or bandwidth available to support other, concurrent observation modes anyway. Note also that we consider other high-speed, moderate-cost interconnects like Myrinet for the cluster machines that have to be bought.

5.3 4-Bit samples and RFI

In the original LOFAR design, the PolyPhase channel filter was planned to run on FPGAs at the stations. When it became clear that the Blue Gene/L correlator ran so efficiently that there was spare capacity left for additional computing, a significant cost reduction could be achieved by moving the channel filter to the Blue Gene/L. As a result, the stations send subband data, rather than channel data.

For EoR observations, the stations will send 4-bit samples (4 bit real + 4 bit imaginary) instead of 16-bit data, otherwise more data would be sent than could be received by the Blue Gene/L. However, with 4-bit *subband* data, this poses a problem for subbands that contain narrow-band RFI: the four bits will encode RFI only, leaving no space to encode signals for the unaffected channels. Consequently, the entire subband is lost.[2]

Either a smarter encoding algorithm has to be developed (if such an algorithm exists), or the channel filter has to be implemented at the stations, or we have to accept that we cannot observe subbands with RFI, even if the RFI is narrow band. Note that this discussion involves the core stations only (plus possibly a few remote stations): there is no need to implement channel filters at the remote stations, since they will not be used for 4-bit (EoR) observations.

5.4 Other Key Science Projects

Apart from the standard observing mode, LOFAR defines four Key Science Projects that require special processing pipelines for OLAP (see the LOFAR CEP Architecture Description [5]). In the months to come, we will define these processing pipelines in greater detail. We plan to implement these pipelines in the next few years.

5.5 Feedback loops

Figure 15 showed that the time to correlate a second of data from 77 stations takes almost 10 seconds, plus the communication time. This means that the correlator outputs data with a large latency, which should be taken into account if, in the future, feedback loops will be implemented that originate from behind the correlator output and go back to before the correlator input (online calibration, for example). There are several solutions to handle this (e.g., halving the latency by sending the first half of a second of data to one compute core and the other half to another core, and adding the visibilities afterward). However, all of these options require additional processing.

[2] Another consequence of having the channel filter in the BG/L is that a dropped UDP packet from the stations invalidates all FIR filter banks (one cannot take a FFT over partially missing data), but this consequence is not as severe as the RFI problem.

Since the EoR observation mode is computationally so expensive that we cannot afford extra processing, we need the computationally least expensive method anyway. For standard observations, we will use the same, efficient scheme, and decide upon appropriate measures if the latency turns out to be prohibitively large.

5.6 CEPframe issues

We currently use the CEPframe library [6] as a toolkit to build and connect distributed OLAP applications. Although there are many things that work well (e.g., transparent byte swapping in the communication between architectures of different endianness), there are some deficiencies in both the design and the implementation of the library, that cannot be easily fixed or circumvented. One major shortcoming is the inability to stop the distributed OLAP applications in a proper way. Another problem is that CEPframe data structures are too static: data must always be sent in the same format. This makes it, for example, impossible to avoid sending 60 MB of flagged zeros, which is important to recover from a stall in the processing pipeline. Moreover, the Application Programming Interface is not intuitive, which makes it hard to develop programs with this library. Clean solutions for these issues will require significant effort.

5.7 Fault tolerance

The OLAP applications should be made more resilient to failing hardware. With the status information that MAC will collect, it has the ability to stop and quickly restart OLAP avoiding the offending machine, using some hot-spare hardware. We consider stopping and restarting the applications as a more feasible solution than writing software that continuously runs trying to avoid broken hardware, as there are too many types of possible failures, each requiring a different solution. Moreover, software libraries like MPI do not tolerate crashed members.

6 Summary

This document describes the four sections in the LOFAR OnLine Application Processing subsystem: the Input Section, Blue Gene/L processing, the Output Section, and Delay Compensation. The major tasks of the OLAP applications are to receive, synchronize, buffer, transpose, channel filter, correlate, and store the data, and compensate for the different receive times at the stations.

We discuss design choices and changes with respect to earlier designs. Finally, we discuss a list of issues that need to be addressed. Future work includes work on higher network bandwidth into the Blue Gene/L, additional functionality for the key science projects, attempts to reduce the costs of new hardware that will surround the Blue Gene/L, and software quality. The issue with the 4-bit polyphase filter requires discussion in a broader community.

Acknowledgments

Chris Broekema, Ger van Diepen, Martin Gels, Edwin de Lang, Marcel Loose, Ellen van Meijeren, Ruud Overeem, Kjeld van der Schaaf, and Walther Zwart contributed to the OLAP applications.

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.

References

[1] André W. Gunst. LOFAR Architectural Design Document of the Astronomical Applications. Technical Report LOFAR-ASTRON-ADD-006 version 5.0, March 2007.

[2] W. Lubberhuizen and E. Kooistra. RSP - CEP Beamlet Data Interface. Technical Report LOFAR-ASTRON-SDD-009, March 2007.

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

[4] John W. Romein, P. Chris Broekema, Ellen van Meijeren, Kjeld van der Schaaf, and Walther H. Zwart. Astronomical Real-Time Streaing Signal Processing on a Blue Gene/L Supercomputer. In *ACM Symposium on Parallel Algorithms and Architectures (SPAA’06)*, pages 59–66, Cambridge, MA, July 2006.

[5] Kjeld van der Schaaf and Auke Latour. LOFAR Central Processing Facility Architecture Description. Technical Report LOFAR-ASTRON-ADD-012 version 3.0, March 2007.

[6] Ellen van Meijeren. The CEPFrame Libraries Family. Technical Report LOFAR-ASTRON-SDD-042, August 2005.