ALMA Correlator Real-Time Data Processor

J. Pisano, R. Amestica, J. Perez
NRAO, Charlottesville, VA USA
jpisano@nrao.edu

Abstract

The design of a real-time Linux application utilizing Real-Time Application Interface (RTAI) to process real-time data from the radio astronomy correlator for the Atacama Large Millimeter Array (ALMA) is described. The correlator is a custom-built digital signal processor which computes the cross-correlation function of two digitized signal streams. ALMA will have 64 antennas with 2080 signal streams each with a sample rate of 4 giga-samples per second. The correlator's aggregate data output will be 1 gigabyte per second. The software is defined by hard deadlines with high input and processing data rates, while requiring interfaces to non real-time external computers.

The designed computer system – the Correlator Data Processor or CDP, consists of a cluster of 17 SMP computers, 16 of which are compute nodes plus a master controller node all running real-time Linux kernels. Each compute node uses an RTAI kernel module to interface to a 32-bit parallel interface which accepts raw data at 64 megabytes per second in 1 megabyte chunks every 16 milliseconds. These data are transferred to tasks running on multiple CPUs in hard real-time using RTAI's LXRT facility to perform quantization corrections, data windowing, FFTs, and phase corrections for a processing rate of approximately 1 GFLOPS.

Highly accurate timing signals are distributed to all seventeen computer nodes in order to synchronize them to other time-dependent devices in the observatory array. RTAI kernel tasks interface to the timing signals providing sub-millisecond timing resolution.

The CDP interfaces, via the master node, to other computer systems on an external intra-net for command and control, data storage, and further data (image) processing. The master node accesses these external systems utilizing ALMA Common Software (ACS), a CORBA-based client-server software infrastructure providing logging, monitoring, data delivery, and intra-computer function invocation.

The software is being developed in tandem with the correlator hardware which presents software engineering challenges as the hardware evolves. The current status of this project and future goals are also presented.

Introduction

The Atacama Large Millimeter Array will be the largest millimeter wavelength astronomical observatory in the world (www.alma.nrao.edu). Located in the Atacama desert of northern Chile at an elevation of 5000 meters, the interferometer will consist of 64 12-meter fully-steerable antennas providing a total collecting area ~7240 m^2, with a resolution of 10 milliarcseconds in the wavelength region 10 mm to 350 microns (30 – 950 GHz). ALMA will provide dramatic inroads in the understanding of cosmology, galaxy formation, stellar evolution, proto-planetary systems, and the creation of elements from supernova ejecta. The antennas themselves will be movable to provide different configurations and spacing ranging from a compact configuration of 150 meters to 12 kilometers in diameter.

An important aspect of this project is its international scope. Effort and costs are equally divided among North America, Europe, and Japan. Initial commissioning is planned for 2008 with full science operations in 2012.

Each antenna has four separate intermediate frequency (IF) pairs which are digitized at 3 bits and 4 giga-samples per second. Each signal cable in an IF pair provides orthogonal polarizations for a given frequency band. These digitized streams are fed to a custom-built digital signal processing engine (correlator) which computes the cross-correlation function for each unique pair of antennas (baselines). There are 2080 baselines with 64 antennas leading to an input data rate to the correlator of ~750 gigabytes per second and a processing rate of approximately $2x10^{16}$ integer calculations per second. These input data are time averaged within the correlator to provide an aggregate output stream (lags) of 1 GB/second.

The correlator output data are sent to the Correlator Data Processor (CDP), a cluster of sixteen rack-mounted, diskless PCs via high-speed 32-bit data ports. These CDP compute nodes process the raw lag data into spectral results which are then transferred via a *master node* to external computers for archival and further processing into astronomical images – the end product of the observation. Each CDP node performs the same types of calculations, but on separate data. Thus the CDP is not a true Beowulf cluster, but it has many parallel processing attributes similar to a Beowulf cluster. ALMA is expected to initially produce ~500 gigabytes of data per day (~180 terabytes of data per year), with future enhancements increasing this amount.

One can compare ALMA to existing radio astronomy interferometers to better appreciate the complexity and scope of ALMA. For example, the Very Large Array (VLA) in New Mexico has 27 25-meter antennas and a resolving power of 100 milliarcseconds. Its correlator performs 5.8x10^5calculations per second with an output data rate of ~10 KB/second and an annual rate of ~500 gigabytes.

Software Description

Software Overview

To better understand the purpose of the CDP, one must look not only at its relationship to the correlator hardware, but also to other computer systems within ALMA. Figure 1 provides a logical view of the external computer systems to which the correlator software system interfaces. These include *Control*, *Telescope Calibration*, *Quick Look Pipeline*, *Archive*, and *Executive*. Note that there are other software subsystems which are note shown and do not directly interface to the Correlator system including experiment proposal preparation, observation scheduling, and final data processing.

The correlator hardware is a 'configure and run' system. It operates continually, changing modes depending on the observation, but it never stops. The correlator stops only for scheduled maintenance intervals. Recall that a radio astronomy observatory can observe day and night.

The Control system is the main interface to the Correlator system. It sends control and configuration commands, some of which are passed on to the correlator hardware. The Control system also supplies data streams to the Correlator system that allow for real-time adjustments reflecting the changes occurring during an observation. These are changes due to the rotation of the earth, antenna hardware problems, and phase changes due to fluctuations in the atmosphere.

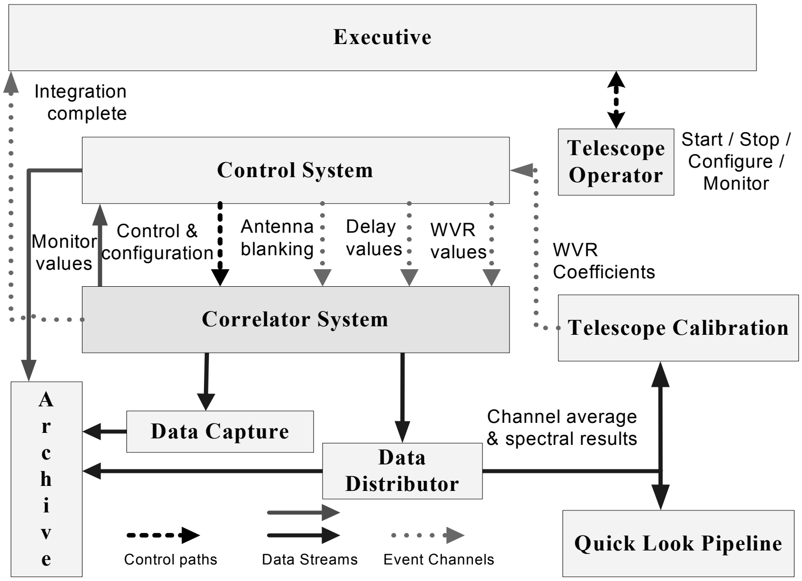


Figure 1 Logical Architecture

The Telescope Calibration system watches the Correlator system's spectral output ensuring data quality consistency and issues commands to the Control system to make fine adjustments to various hardware and software systems. The Quick Look Pipeline provides near real-time imaging of the spectral output enabling the operator or observer to view images as data are collected – another important quality assurance process.

The Archive stores the correlator spectral results plus observation description information via the Data Capture component for delivery to the experiment's scientists for further analysis. Also the archive stores *monitor* data – data describing the state of all of the ALMA hardware during an observation. The Data Distributor is a component of the archive which multiplexes the Correlator system output to the Telescope Calibration system, Quick Look Pipeline system and to the physical data storage component of the Archive.

Lastly, the Executive system is a high-level software system which provides operator interfaces to control and monitor all of the ALMA components, tracks administrative functions, and provides a security model. The Correlator system transmits events to the Executive system to track an observation's progress.

All computer subsystems rely on the ALMA Common Software (ACS) interconnection infrastructure [1]. ACS is a CORBA-based system which defines an object model of distributed control system components and provides a common set of APIs for remote object method invocation, information logging services, error tracking and reporting, notification and event channels, data base access, and (soft) real-time control and monitoring of hardware.

The hard real-time constraints for the CDP are the incoming lag data and a timing bus whose signals are used to synchronize distributed hardware within the array. The Real Time Application Interface (RTAI) [2] real-time Linux software system has been utilized for these time-critical components to meet the hard deadlines imposed by the external hardware interfaces. The use of RTAI has played an important role in clarifying the division between hard and soft real-time demands of the CDP.

Real-Time Components

Three time critical components of the CDP exist.

1. Raw data transfer

Each of the sixteen CDP compute nodes is divided into four groups of four computers. Each subgroup is associated to a correlator quadrant which processes data from all 64 antennas for a given IF band. Thus data from all four IF bands for all 64 antennas are processed simultaneously.

Each compute node is physically connected to a 32-bit binary interface using a PCI64-HPDI32AL board from General Standards Corp. (www.generalstandards.com). Every 16 milliseconds, up to 1024 sets of 256 32-bit lags (1 MB of data) are dumped to each compute node. The HPDI32 performs a linked-list DMA transfer to a shared memory buffer in the PC. Once the transfer is complete, the data in the shared memory buffer are pulled into a data-processing pipeline whose details are described later.

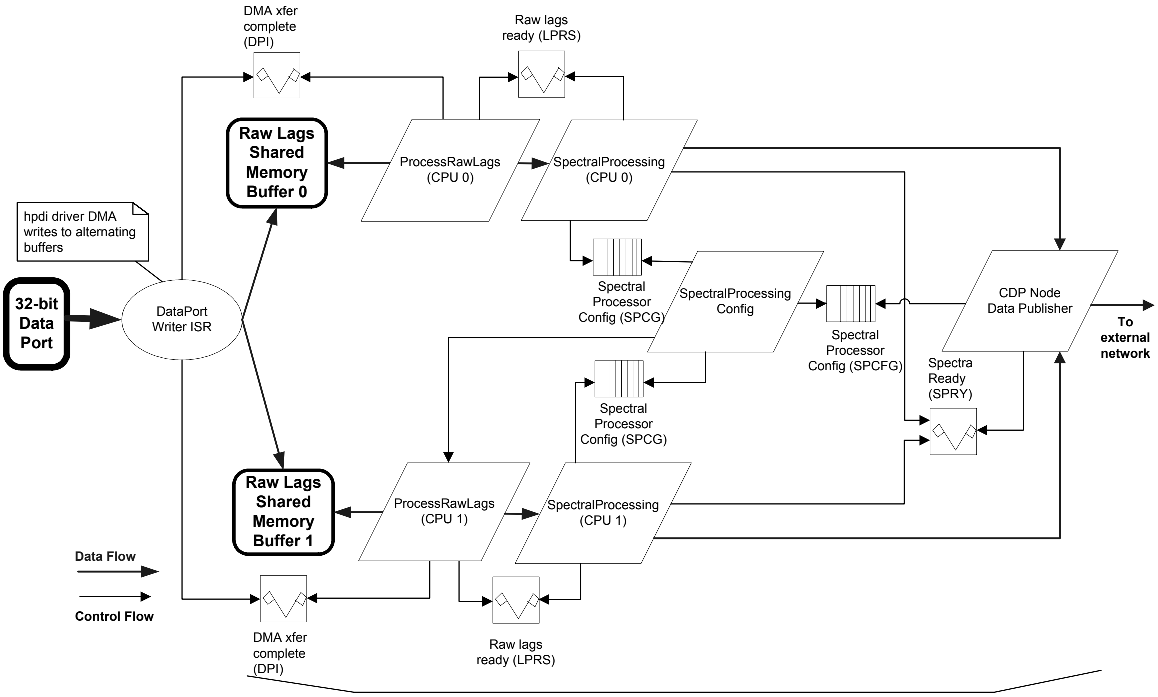
The deadline here obviously to pull the all the lag data from the shared memory buffer and process it before the next lag data sets arrive in 16 milliseconds. To accomplish this feat, various hardware and software decisions have been made.

The compute nodes are dual CPU Opteron-based motherboards with 66 MHz and 33 MHz PCI busses. The HPDI32-64PCI card uses 66 MHz/64 bit bus providing high-throughput DMA transfers. Dual CPUs enable us to utilize a double-buffering scheme where at each 16 millisecond cycle, the DMA transfers switch between two shared 1 MB memory regions. Then each CPU has almost 32 milliseconds to process the lag results (2 times the dump period minus the DMA transfer time). This double-buffering approach is shown schematically in Figure 2.

32 milliseconds to DMA transfer 1 MB of data is a trivial deadline. The real difficulty in meeting this deadline is to process those lags into spectra. Estimates [3] show that each node must perform ~5x10^4 floating point operations within the 28 millisecond window leading to a processing rate of roughly 1.6 GFLOPS.

The DMA transfers to shared memory are set up in kernel modules utilizing the RTAI API. A binary semaphore communicates between the data transfer kernel module and data processing routines running under RTAI's LXRT framework that utilize a prioritized real-time scheduler minimizing delays when data are ready to be processed.

Further complicating the architecture is that there can be multiple sub-groupings of antenna inputs (subarrays) to the correlator which must be processed independently. Thus there can be multiple ProcesssRawLags and SpectralProcessing tasks, one for each of 4 subarrays, although there is still one shared memory buffer per CPU.


DMA xfer complete (DPI)
Raw lags ready (LPRS)
Raw Lags Shared Memory Buffer 0
ProcessRawLags (CPU 0)
SpectralProcessing (CPU 0)
hpdi driver DMA writes to alternating buffers
32-bit Data Port
DataPort Writer ISR
Spectral Processor Config (SPCG)
SpectralProcessing Config
Spectral Processor Config (SPCFG)
CDP Node Data Publisher
To external network
Spectra Ready (SPRY)
Spectral Processor Config (SPCG)
Raw Lags Shared Memory Buffer 1
ProcessRawLags (CPU 1)
SpectralProcessing (CPU 1)
Data Flow
Control Flow
DMA xfer complete (DPI)
Raw lags ready (LPRS)

Each task and semaphore duplicated for each of 4 subarrays

Figure 2 – Double-buffering scheme

2. Data Processing

Special attention must be paid to floating point processing routines due to the high input data rates. There are many steps in processing raw lag data from the correlator into final output for the Archive:

- Discard bad lag data sets due to intermittent hardware issues, e.g. antenna blown off source, local oscillator fails to lock, delays or lost data packets, etc.
- Normalize lag data to remove data biases introduced by the correlator hardware.
- Perform a quantization correction which minimizes the effects of digitizing an analog signal. There are two digitizing steps in the signal chain which require two corrections.
- A windowing function which deals with edge effects of the finite data sets.
- A fast Fourier transform to convert time domain data to the frequency domain producing spectra (complex visibilities vs. frequency). The *FFTW* package from MIT is used (www.fftw.org).
- A bandpass calibration to correct for digital filter characteristics.
- Time averaging of spectral data.
- Data averaging which sums all spectral data over a given spectral range into a single value at specific time intervals.
- Corrections for signal delays due to the geometry of the antennas, correlator and astronomical source.
- Atmospheric phase correction which removes short term (½ second time frame) phase fluctuations due to different density ‘bubbles’ in the atmosphere.

Each of these processing steps must be carefully analyzed to optimize performance. Pre-calculation of constant values, algorithm research, profile analysis tools, optimizing compilers, and testing all play a role in fine tuning data processing performance.

3. The timing bus

Every 48 milliseconds an RS-485 compatible pulse is transmitted to various devices up to 12 kilometers in distance. These highly accurate timing pulses (timing events) are phased to account for propagation delays and are used to synchronize electronic switching among hardware and computer devices. A *master clock* computer aligns external UTC time to the timing events resulting in *array time* which computer systems use to synchronize events, e.g., the start of an observation. The master clock sets the CDP nodes' local time when they request it at startup.

Once confirmed that the time is correctly synchronized to the master clock, each CDP node (including the master node) uses the timing events to set its own internal clock via calls to *do_gettimeofday()/do_settimeofday()*. Consequently, array time can be maintained to 100 microseconds within the CDP nodes for time stamping of data. If a timing event is missed, a watchdog timer fires a process to resynchronize the local clock with the master clock.

Tasks running in kernel or user space (under LXRT) can be scheduled to run relative to timing events providing task synchronization. Tasks request to be woken up 0 - 47 milliseconds relative to a timing event. Using the RTAI mailbox facility, the timing event kernel module sends mailbox events to the scheduled tasks at the appropriate time offset allowing them to run periodically. The scheduled tasks then execute as needed and can count the number of timing events if they wish to execute some functionality at intervals longer than 48 milliseconds.

Logical Time Components

Interfaces to other computer systems fall into the *soft* real-time category. This is mainly due to the distributed nature of ALMA computers and the use of giga-bit Ethernet as an interconnection medium. The CDP master node is physically connected to the ALMA network with some computers in the Control subsystem at the high site, called the Array Operations Site or AOS, and the remaining computers at the Operations Support Facility (OSF) about 50 km away at an elevation of 2900 meters.

Since Ethernet is non-deterministic, array time is used to synchronize events. Time-tagged control commands are sent in advance and queued at the CDP in order to overcome any network latency and set-up time needed to construct software entities for new observation configurations. A scheduler in each CDP node based on array time schedules commands to be executed at a specific time.

ACS provides wrappers for CORBA simplifying its use. All computer systems in ALMA present an interface defining public function calls and data using CORBA IDL. Developers are then free to utilize programming languages with CORBA bindings appropriate to their applications. The CDP operates in C++ with test applications written in Python while other ALMA systems make extensive use of Java. The ORBs used are TAO for C++ (www.cs. wustl.edu/~schmidt /TAO.html), JacORB for Java (www.jacorb.org), and OmniORBpy for Python (omniorb.sourceforge.net). All three of these ORBs are open source, an important consideration for the ALMA software.

CORBA notification channels are employed to transmit and receive observation-time event data. For example, every 48 milliseconds each antenna publishes data valid events on a notification channel. Each CDP compute node subscribes to this channel discarding lag sets from the correlator which the antennas marked as invalid. In the atmospheric phase correction process, water vapor receiver event data are published from each antenna to which each CDP node subscribes and uses this data stream in its data processing.

Although these notification channels operate on a non-deterministic Ethernet network and with some non-real-time computers, their data are involved in hard real-time processes. Any potential delays are compensated by the use of time stamping data events and buffering in the CDP nodes. Notification and correlator data are buffered for approximately one second to ensure that all events arrive before they are used. Thus, an inherent latency of the data pipeline of one second is introduced. Also the processing software is written such that any missed events do not introduce corrupted data into the processing pipeline.

Hardware Description

Figure 3 shows a block diagram of the correlator hardware and associated computers. The computers shown on the right side of the diagram – ACC, Science & Monitor Data Archive, Telescope Calibration, and Quick Look Pipeline, are all at the OSF and connected via a 10 giga-bit Ethernet connection to the AOS. Computers at each antenna and correlator computers share a 10 giga-bit bridged Ethernet network at the AOS. The CDP compute nodes share a private network with the master node, the latter being connected to the external ALMA network.

The timing events are distributed to both the correlator quadrants and computers allowing for the synchronization discussed previously. A rack-mounted PC issues commands to the correlator hardware via multiple Controller Area Network (CAN) busses. All computers are identical dual-CPU Opterons for uniformity purposes.

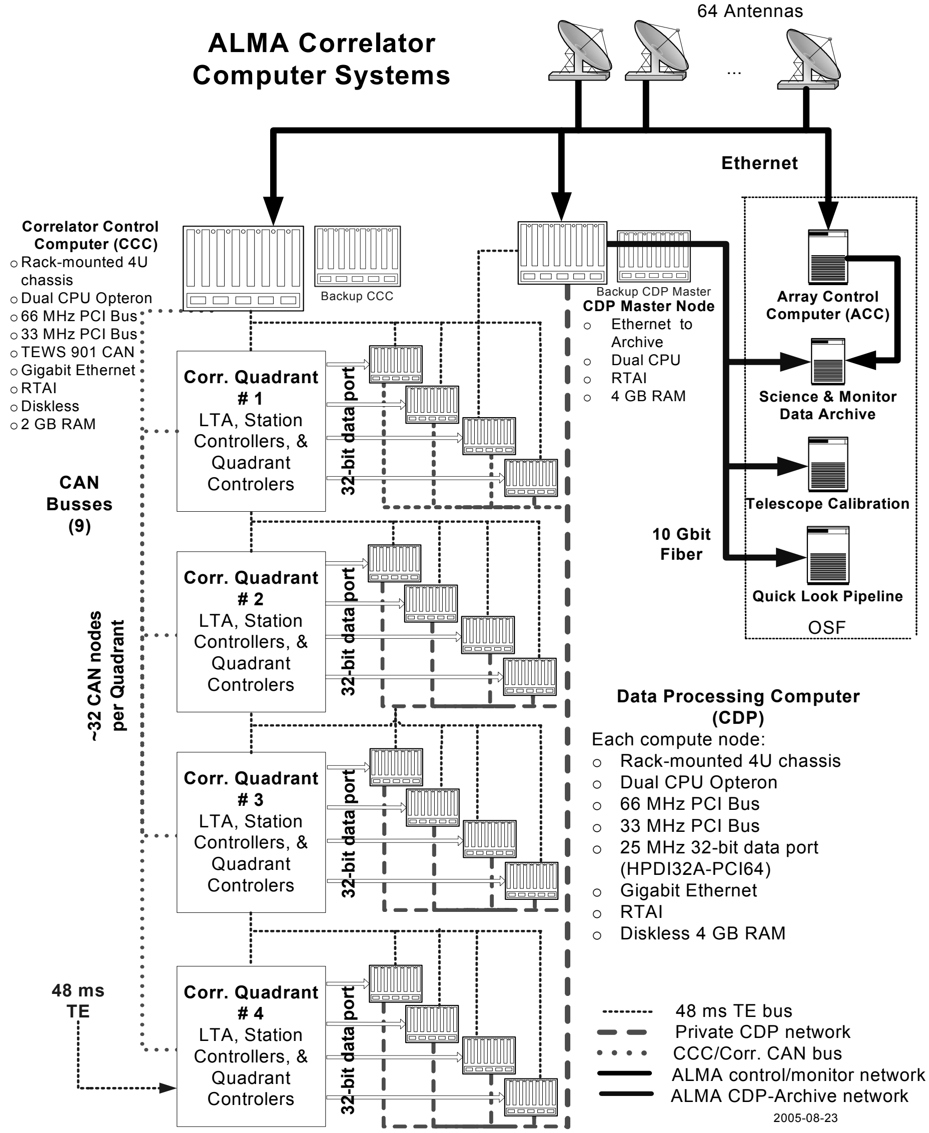


Figure 3 Hardware View

Project Status

A prototype version of the CDP is currently under development. This small two-computer version interfaces to a small prototype correlator which supports two antennas. These prototypes will work with other prototype hardware including antennas, receivers, digitizers, and signal transmission equipment. Although small, all of the fundamental components and capabilities can be tested including the data and processing rates. These prototypes are currently being assembled at the VLA site in New Mexico where hardware and software will go through an exhaustive testing and evaluation process. First versions of non-prototype hardware will begin to arrive in Chile in 2007 with total deployment completed by 2012.

References

[1] Chiozzi, G., Gustafsson, B., Jeram, B., 2003, *ALMA Common Software Architecture*, http://www.eso.org/~gchiozzi/AlmaAcs/OnlineDocs/ACSArchitecture-4.1.pdf.

[2] Mantegazza, P., *Real Time Application Interface*, http://wwww.rtai.org.

[3] Pisano, J., 1999, *MMA Correlator Output Data Rates*
http://www.mma.nrao.edu/development/computing/docs/memos/memo008/alma-sw-008.pdf