Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

Efficient usage of HPC horsepower for the LOFAR telescope

Kjeld v.d. Schaaf

ASTRON, Postbus 2, 7990 AA Dwingeloo, The Netherlands

Abstract. The operation of novel phased array based radio telescopes requires vast amount of computational power for the processing of observed data. In the LOFAR telescope, the central on-line and off-line systems will use HPC resources provided by a 1000+ nodes cluster computer. In order to profit from the enormous computational power of this machine an application development framework is built. This framework support the transformation of the current single-node data reduction applications to parallel equivalents that can execute on large clusters. In the online systems, real-time signal processing pipelines can be executed. Furthermore, the complexity of controlling many concurrently running observations and analyses on an inhomogeneous hardware set is tackled.

1. The LOFAR Central Processor Facility

The central processor of LOFAR should provide the necessary processing and storage capacity for the applications running on it. Although multiple types of sensors are attached to the LOFAR infrastructure that provide data for multiple scientific fields, we focus here on the radio astronomical data. For these applications, we can identify three key processing types:

- For sky imaging applications, the central processor is used for correlation, beamforming, integration and, after short term storage, calibration and image creation.
- For pulsar observations, an "extended core" of up to 50% of the stations produce data for a central beamformer. A major processing task is the de-dispersion measuring.
- For transient detection and analysis of time scales of a second or more, series of snapshot images can be analysed using Fourier transforms. For shorter time scales, detection algorithms operate directly on the station beams or antenna data.

The aim of the central processor architecture is to provide first of all enough processing power, data transport bandwidth and storage capacity to allow for continuous operation of the observation facility. Secondly, the central processor will be built as a signal-processing cluster aimed at multiple types of data processing.

The hardware architecture of the central processor is based on cluster computer techniques with hybrid cluster nodes and a high bandwidth switch fabric interconnect system. The different types of cluster nodes represent the required flavours of processing capacity. We have for instance nodes based on a normal workstation to which digital signal processing boards are added, containing system on chip processors such as a FPGA with a DSP IP core. Another flavour of nodes is for example the storage node, which could be a workstation, connected to a RAID disk array.

The central processor facility is constructed as a series of planes with high bandwidth point-to-point connections between the planes, thus forming pipelines. Orthogonal to the pipeline direction, an all-to-all network topology is present which is used to bring data streams to the correct pipeline. Especially in the first planes, such routing capacity is needed for re-ordering the input data. For example in the case of imaging applications for astronomy, a complete matrix transpose of all input data is required, which translates to an all-to-all data transport tasks on the first one or two planes in the cluster.

The correlator architecture of LOFAR requires a transpose operation on all input signals at the central site. With 100 stations each providing 32000 spectral channels, this is a gigantic routing task. In the central processor design this transpose function is executed in two steps. In the first step, packages of multiple frequency channels are routed in the first two planes of the cluster. The fine-grained transpose operation that remains to be performed can efficiently be executed by the node processors in cache memory. The transpose operation requires a bandwidth of 1Gbps between all nodes in a plane. Other applications may use the in-plane routing capacity to distribute large processing tasks over multiple pipelines.

The combination of pipeline connections and fully routed planes is designed as one large switch fabric, although different technologies might be used for the planes and pipeline connections.

The combination of fully routed planes and pipelines being fed by those planes offers a completely scalable hardware structure. The available processing power can easily be distributed over multiple applications running in parallel. More pipelines can be added to the switch fabric operating as data farmer.

In the processing pipeline a few planes are organised as storage system. This section of the cluster is responsible for the storage of complete measurements until the observation data is fully processed and the final data products are produced. This provides the transformation of streaming data into a coherent dataset. Since the data analysis applications are off-line processes, storage facilities for several days are supplied. The required storage capacity of order 500 TByte can be implemented in two planes containing storage servers, each equipped with RAID disk arrays.

The central processor design is based on abstractions from available techniques to avoid dependency on actual available implementations. This is very important to be able to fully profit from Moore's law.

2. Programming Model

A programming model has been designed in which the applications can be expressed and mapped onto the available hardware. The programming model is provided through an application development library layer built on the middleware layer. The application is modelled in pipes and filter style using a hierarchy of processing steps that implement functional behaviour. The processing steps are connected to each other using an abstraction of the message passing paradigm, thus embedding parallel programming in the framework.

The scientific applications are expressed in functional steps in the process step hierarchy. This whole hierarchy, along with the functional connections is programmed into a single set of source files. The source files can then be mapped onto the available hardware in multiple ways. During the mapping, the exact processing hardware for the processing steps are selected and the connection implementations are chosen. The application source code can be mapped onto one ore more interacting processes. Each such application has its own virtual machine responsible for the correct sequential execution of the application steps. Since the virtual machines in all applications are generated from the same source file, the synchronisation between the applications is guaranteed.

The processing model supports multiple message passing implementations. Currently, implementations have been tested for MPICH, ScaMPI, ShMem, Memcpy and Corba.

The Corba implementation uses remote invocations for the actual data transport. Cyclic input and output buffers are used together with event handlers in separate threads in order to get the required blocking behaviour for the programming model. Performance tests have shown similar bandwidth characteristics for the MPICH and Corba implementations.

The signal processing boards added to part of the cluster nodes are controlled by a middleware layer that is currently being developed by the project members. This middleware layer provides the appropriate hardware abstractions required by the application development framework. This also allows for adoption of the new hardware items into the monitoring and control system.

Note that the programming model, and therefore the application code, is independent on the data transport implementation used since only a small subset from the message passing paradigm is needed. Other middleware implementations can be used for the connections. Also direct device-to-device connections can be controlled from the connection abstraction in the model.

The multiple message passing implementations supported in the programming model can also be mixed. A demonstrator has been built in which a ScaMPI application performs data routing tasks and sends its data to multiple processing pipelines implemented with MPICH communication. The data transport between the ScaMPI and MPICH programs is made by a Corba connection.

The application development platform is optimised to minimise overhead. The data transport mechanisms in the framework are based on existing middleware implementations were feasible. Dedicated transport mechanisms are implemented were needed. There the data transport bandwidth for data transport between processes on a SMP node is measured.

The mapping of the application's processing steps on to the hardware is made independent of the processing step's implementation. Therefore the same application can be mapped onto the hardware in multiple ways. This is used for performance optimisation by mapping of the processing blocks on the appropriate "flavour" of cluster nodes. The strategy pattern is used to select the correct executable for the target processor.

The switching fabric can be operated by a data farming process that farms out blocks of data to pipelined processes. This data farming process is used for both load balancing and fault tolerance. The fault tolerant operation is demonstrated using the Corba connector in combination with Corba control methods added to the processing steps. The pipeline applications are now both controlled and fed with data through Corba objects.

References

Achterop, S., 2001, Proceedings of ProRISC

Tessier, R., & Burleson, W., 2001, Journal of VLSI Signal Processing, vol. 28 , 7 , 27

Vos, C.M. de, Schaaf, K. v.d. & Bregman, J.D., 2001 , Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid, Brisbane, 156, 160