A PC CLUSTER FOR THE MICROTOMOGRAPHY

A. Bertrand, R. Krempaská, M. Stampanoni

A cluster of Intel computers has been set up, managed by an in-house developed software, in order to process in an efficient way the large amount of data produced by a microtomography experiment

TOMOGRAPHY AND DATA CRUNCHING

The tomography technique requires a high number of high-resolution images to be acquired, which are then processed in order to obtain cross sections (slices) of the investigated sample.

Before starting this project, each final image needed 3-4 minutes in order to be processed. This time has been significantly reduced with the cluster to 16 images per minutes.

ONLINE SINOGRAM CREATION

Before raw data can be reconstructed, an intermediate processing is needed which produces images taking the same line on all the raw images. Those intermediate images are called sinograms. In order to speed up the data elaboration process, we implemented this software in such a way that they are produced automatically during the microtomography scan. The synchronization of this application with the CCD camera is done by monitoring an EPICS [1] variable.

HARDWARE INVOLVED

We chose to build the cluster based on Intel P4 computers. By cluster we mean simply a stack of 8 rack-mounted computers. We didn't use multi-processor computers, as they are more expensive than the same number of single CPU boxes.

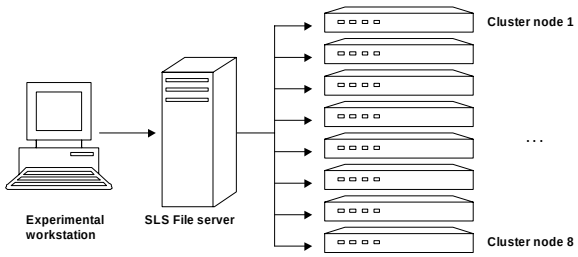

Fig. 1: Hardware configuration

All data are taken on a workstation running at the beamline, which stores the images on a SLS file server via a 100 Mbit network connection. Cluster nodes and file server are linked via a gigabit network.

SOFTWARE AND PROTOCOL

Instead of following the approach of complex clusters like commercial software or Beowulf [2] solutions, we decided to implement the control software of the cluster by using Python [3] scripts and C/Fortran codes for the reconstruction software.

Because of its good performance and easy implementation, we chose the HTTP standard. This choice enabled us also to debug our system, during the development, with any web browser.

DATAFLOW

The implementation chosen allows for queuing commands in a sequential order. The server will then dispatch single reconstruction jobs across available nodes.

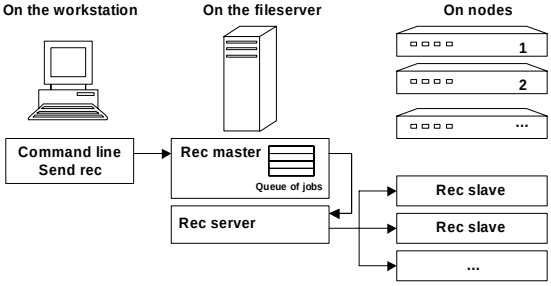

Fig. 2: Processes and dataflow

As the time needed to transfer our images is not negligible we implemented the main control and nodes control in a multi threaded way in order to transfer data in parallel to the reconstruction of another image. In this way the CPU is used all the time for the reconstruction and transfer back and forth (data and results) is done in the background.

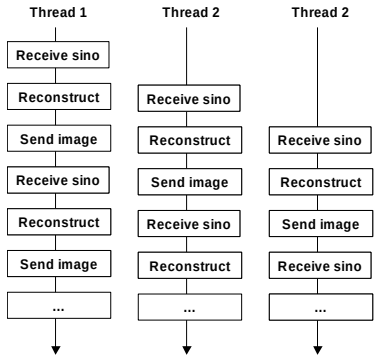

Fig. 3: Multithreading on the node

SOFTWARE REUSABILITY

As the software functionalities are to transfer data and in parallel launch some command line software, the same concept can be reused on any other application where a long processing time is needed without interaction between processes.

REFERENCES

[1] http://www.aps.anl.gov/epics

[2] http://www.beowulf.org/

[3] http://www.python.org/