

# GMAP - Grid aware Monte-Carlo Array Processor

A. Moreton, G.D.Patel, T.J.V. Bowcock (University of Liverpool)  
E-mail: gdp@hep.ph.liv.ac.uk

## Abstract

The Monte-Carlo Array Processor (MAP) has been designed using commodity off the shelf (COTS) items to provide the CPU requirements of full event simulation for the LHC experiments. The solution is however completely general, so any CPU intensive application with limited input requirements can be run on the system.

Operating control software has been written to manage the data flow over the 100 BaseT ethernet connecting the 300 nodes (400 MHz PII's) to the 6 master control nodes (700 MHz PIII's each with 500Gb of disk). Upgrade to 1000 nodes is planned. Job control software that allows the user to run the same job on all nodes, whilst allowing for small differences in initialisation parameters between nodes has also been written.

GMAP is the GRID aware MAP control software. This allows remote job preparation and submission using globus toolkit for authentication and communication. The software will be available and opens the possibility for doing massive Monte Carlo production over several remote MAP sites simultaneously.

Keywords: GRID, PC Farm, Simulation, Distributed Computing, Network Control Software

## 1 MAP Hardware and Network Architecture

The MAP architecture [1] is based around the use of cheap commercial off the shelf (COTS) components. MAP consists of 300 identical custom built rack mounted boxes each containing a 400 MHz PII CPU with a 100 BaseT ethernet card and a 20 Gb hard disk. Each rack houses 30 nodes and two 16 port 100 BaseT ethernet switches to which the top 15 and bottom 15 MAP nodes are connected. There are a total of 10 racks and the 20 ethernet switches are connected to 6 COMPASS nodes, mounted in two additional racks, via another pair of 100 BaseT ethernet switches.

The COMPASS nodes were designed for I/O intensive operation and consist of a 700 MHz PIII CPU, plus 10 SCSI disks, each of 50 Gb. Thus each COMPASS node has 500 Gb of disk storage. The COMPASS nodes are equipped with two ethernet cards, one that connects to the private internal MAP network, and one for the internet. The individual MAP nodes are not visible on the internet and can only be accessed via the COMPASS nodes. The ethernet switches between the MAP and COMPASS nodes can be configured with MAP and COMPASS nodes partitioned into sub-groups on separate disconnected internal networks. The control software can deal with this hardware configuration as well multiple queues via software configuration. The network architecture is illustrated in Figure 1.

The 300 MAP nodes are kept as simple as possible and a stripped down version of RedHat 6.2, (about 200 Mb), is installed on them. There are no user application libraries on the MAP nodes. The six COMPASS nodes also run RedHat 6.2 and have the requisite user dependent application libraries. The Linux executables maybe created there for downloading to the MAP nodes. The MAP system has been successfully used for the following experiments, LHCb, ATLAS, H1, CDF DELPHI and also by the DESY machine group for beam simulations.

## 2 MAP Control and Configuration

Control of the system is achieved using a set of daemons, designed and written in Liverpool, running on the MAP and COMPASS nodes. There is a single MAP Queue Daemon (MAPQD) that has ultimate control over the MAP nodes and the COMPASS nodes, running on one of the COMPASS nodes. There is a MAP Local Control Daemon (MAPLCD) running on each

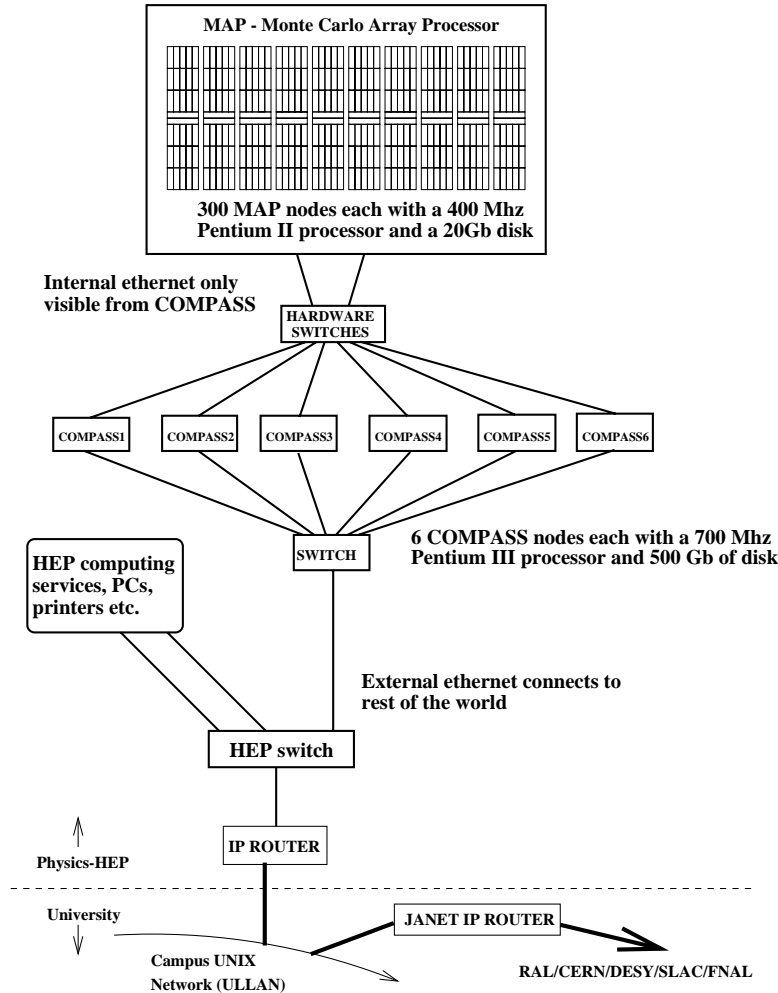


Figure 1: MAP Network Architecture

COMPASS node, which makes each COMPASS node capable of acting as the Local Master of the MAP array. There is a Slave Daemon (SLD) running on each MAP node, plus a supervisor daemon (RUNSLD) that ensures that SLD is always present. The MAP nodes can be partitioned by hardware and software configuration and the master MAP Queue Daemon (MAPQD) handles job submission and multiple queues for each partition. MAPQD assigns a particular MAPLCD to run the job when placing it on the queue.

There are three configuration files, the first, **map.conf**, maps the physical and logical structure of the nodes of a MAP site. It contains a unique seven letter site code that defines the location via the domain address. The name of the machine running the MAP Queue Daemon (MAPQD), which must be visible on the internet, is specified as the entry point to the site. The site can then have one or more independent sub-groups capable of running MAP jobs. Each sub-group is defined by a list of one or more COMPASS nodes running a MAP Local Control Daemon (MAPLCD) plus a list of slave MAP nodes associated with each queue on that sub-group. As an example the Liverpool MAP system, during a development phase, might be split into two sub-groups, 15 attached to a single development COMPASS node and 285

machines for production running attached to 5 COMPASS nodes and logically divided into two queues, a test queue with just 1 node and another queue with 284 nodes.

There are two configuration files that allow access control specification, the first, **clients**, contains a list of domains with associated seven letter site code from which external access requests are allowed. The second, **accounts**, allows for the creation of group accounts specified by an `account_name`. Each group account has an associated list of users specified as `user_id.site` allowed to use this account, a disk space allocation for this account and a list of queues to which the account can submit. Wild-carding is allowed, e.g. `*.lvpool` would be all users from the Liverpool site.

### 3 MAP User Interface

The user interacts with MAPQD, using job submission and control commands. For the next job to run MAPQD determines which COMPASS node the output is returned to and authorises the MAPLCD on that COMPASS node to take control. The MAPLCD spawns a MAPRUNJob process to submit the job to MAP and control the flow of data between itself and the nodes. The I/O flow control software is designed to overcome the problems of overloading of the ethernet hardware and drivers by large numbers of simultaneous packets and is robust against packet loss.

MAP operates in a single user mode. Currently the job is prepared on a COMPASS node in a directory that contains a shell script for running the job, an executable, all the files referenced by the job and a MAP Job Control File (JCF). When the job is submitted, the entire directory is tarred and transmitted to the COMPASS node that will run the job. When the job is ready to be run on MAP, the tar file is broadcast to all 300 MAP nodes simultaneously and the contents of the users original directory is recreated in `/mapuser` which then contains everything required to run the job on each individual MAP node. *There are therefore no limitations on the applications that can use MAP, provided they can package all they need in a local directory.* The JCF file specifies input files to the job that need to be different on each MAP node and the MAPRUNJob process transmits those files individually to the MAP nodes, modifying the files in a way specified by the user. For example for Monte Carlo generation, the user can ensure that each MAP node starts with a different random number seed and also specify different run/event numbers and any other parameters.

The JCF file specifies the name of the shell script that is to be run, and the names of the output files that are to be returned to the COMPASS node. The MAPRUNJob process will copy the required output files to the COMPASS node concurrently whilst the jobs are running on the MAP nodes. With Monte Carlo type jobs that are CPU intensive it is possible to handle all the output data with the available network. Thus the same job is run on all 300 nodes and output from each node is returned to a designated disk on the COMPASS node. (Maximum output from a single job is presently limited to 50Gb on a single disk.) It is upto the user to sort/catalogue/merge upto 300 output files of each type requested, since this is dependent upon the way that data is stored within the file and there is no general way to merge files. The user has the possibility to run an `end_of_job_script` (**ends**) on the COMPASS node on the final output by specification in the JCF file which could perform merging operations.

### 4 MAP Job Control File (JCF)

The commands listed in Table 1 have been implemented and allow the user to control the differentiation of his input files over the MAP nodes, and also to specify the output. The user creates a Job Control File, the obligatory entries are **jd**, **exec**, **acct**, **mapq**, **run** and **disp**. There maybe several output files created on each MAP node and one **retf** card is

required for each file to be returned. Where the user has input files that have to be different on each MAP node, he specifies these with a **edif** and within the file the character strings to be replaced are specified by the string **#!replace\_filename**. Each occurrence of a string **#!replace\_filename** will be replaced by the next available (white space delimited) string in the file specified by **replace\_filename** in directory **jdir**. The **kopt** instruction is useful, in that it terminates the job cleanly once a given percentage of the MAP nodes have completed and no more finish after a specified time. This allows for a clean end to the job, when some small fraction of nodes have failed and are not going to terminate.

command	arguments	Comments
<b>jdir</b>	<b>directory_name</b>	Name of directory containing all requisite files. [Default .] This directory is tarred and downloaded to the MAP nodes
<b>exec</b>	<b>executable_name</b>	Name of executable to be run on MAP, found in <b>jdir</b> . This can be a multi-step shell script.
<b>acct</b>	<b>account_name</b>	Used to allocate output disk at <b>odir</b> Output placed in a subdirectory called <b>job_name</b> .
<b>mapq</b>	<b>mapqueue_name</b>	Name of queue in which job should run
<b>runt</b>	<b>estimated_run_time</b>	Estimated run time. Real number in hours
<b>disp</b>	<b>estimated_disk_space</b>	Estimated disk space required per MAP node for <b>retf</b> files. Real number in Gbytes.
<b>retf</b>	<b>return_file_name</b>	Name of file to be returned as <b>return_file_name.N</b> [N=map node number) placed in <b>odir/job_name</b>
<b>edif</b>	<b>edit_file_name</b>	Name of file to be modified for each MAP node. Strings <b>#!replace_filename</b> in the file are replaced using strings from the file specified by <b>replace_filename</b> in <b>jdir</b> .
<b>excf</b>	<b>exclude_file_name</b>	Name of file in <b>jdir</b> , that need not be sent to MAP nodes. [e.g. .output files are excluded]
<b>kopt</b>	<b>%ncpu wait_time</b>	The job will be terminated after <b>%ncpu</b> percent of MAP nodes have finished and no further nodes complete in the next <b>wait_time</b> seconds.
<b>rnds</b>	<b>random_number_seed</b>	Seed to start the random number sequence
<b>ends</b>	<b>end_of_job_script</b>	Script run at end of job in output directory

Table 1: JCF commands

## 5 GMAP - Grid aware MAP Systems

The control software has been written with GRID awareness in mind. The Globus toolkit has been installed on the COMPASS nodes, and this development is sufficient to make MAP available on the GRID, with no necessity to install on all the individual MAP nodes. A job to run on MAP can be prepared on any machine with RedHat 6.2. An exportable module will make map commands available on machines at external sites and will allow remote job submission and return of output. The JCF return file (**retf**) directive will accept URI file specifications and the output can be written back to the remote machine. The return of the output will be staged via a temporary local copy so that the network load can be monitored.

## References

- [1] Themis Bowcock , “Performance of MAP at Liverpool”, CHEP’2000, Padova, Spring 2000.