Plugging Computer Labs to the Grid

Pedro Ribeiro, Pedro Pereira, Luís Lopes, Fernando Silva

DCC-FCUP & LIACC, Faculdade de Ciências, Universidade do Porto, Portugal
{pribeiro,pdr,lblopes,fds}@dcc.fc.up.pt

Abstract. We present an architecture that allows the seamless configuration of computer labs to work as dedicated computing clusters during periods of user inactivity. The operation of the cluster is fully automated by making use of differentiated network booting and a job management system. We have prepared it to be plugged to a larger computational grid. We provide some preliminary performance results obtained.

1 Introduction

The need for high-performance computing platforms to solve challenging problems has been the main force behind the development of new hardware and software frameworks to aggregate computational power. Cluster computing (developed during the late 80s) was a relatively low-cost and potentially highly scalable solution for this problem. The late 90s introduced a very high level abstraction for high-performance computing resources: Grid Computing. A multi-layered software package manages the resources in the computational grid and schedules the requests of the clients in a transparent way. Several of these frameworks have been implemented, and most follow the original specification of the Globus Toolkit [6].

In this paper we describe a case study experiment in which we aggregate the computational resources of our Department's computer laboratories during their idle hours. Despite the multitude of software frameworks available to aggregate and explore these resources, the task of dynamically building such a cluster is often hindered by the amount of work required for configuring the entire system to automatically switch behaviors and start processing jobs. One of the contributions of this paper is therefore an architecture and configuration system that allows interested parties to setup a cluster and make it work in dual mode in a fully automatic way. After that, we can plug this resource to a wider computational grid in a seamless way.

2 Architecture

Our framework has a very concrete target: computer laboratories (with normal desktop computers) that are used daily by students, either during classes or free time, for a multitude of goals. Internally, the computers are connected with basic Ethernet networks.

One possible approach to aggregate the computational power of such a resource would be to use a system such as Condor [8] to harvest idle cycles on the

computers, while maintaining the original operating systems running, but this would present several disadvantages (no dedicated resources, higher volatility and a need of a more close coordination with the system administrators).

In this context, we decided to transform the laboratories into a dedicated cluster for some hours each day, and design a solution that makes the change in context from the normal daily use of the computers to the cluster, and vice-versa, seamless. In this way, we can have the dedicated computers we need (although only during the time the labs are closed) and we manage to separate the cluster management system from the normal operating system images. This is very important as we want to make the infra-structure independent from the type of system the labs are running and as much as possible from the system administration. This option also guarantees less failure points.

To achieve this goal, we decided to use a master-slave architecture with a job-scheduler serving as control point, responsible for launching and executing unattended background programs. For the purpose of our work, we wanted an experimented and flexible job-scheduler. We analyzed in detail four fully working schedulers (SGE [3], Condor [8], OpenPBS [2] and Torque [4]) and decided to use SGE. Condor was a close call, but, besides the problems described above, its more complex structure (it was originally built for cycle scavenging and gradually upgraded) and lack of available source code (although available by special request) made us look elsewhere. SGE allows us to use message parsing interfaces parallel environments, and a more detailed description of it can be seen on [3]. Figure 1 illustrates the architecture of our Grid environment.

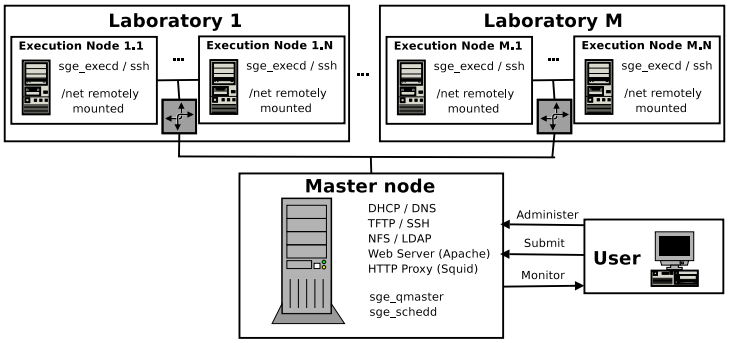

Fig. 1. Proposed architecture for the computer labs based Grid environment.

The Master is responsible for everything related to the cluster. It isolates as much as possible the cluster administration, making it independent from local services. It should be a secure server, with no open ports except the strictly needed ones. To achieve this goal, we used the Ubuntu Server distribution, although other similar Linux distributions could be used without affecting the global architecture. The master node in SGE provides a multitude of services, as seen on figure 1. Given the central and vital role of the master, the SGE natively and transparently supports a shadow master that can automatically substitute and assume the master role in case of failure. The master is also the

only administration and submit host in the SGE context, providing access to the command line and graphical user interface. Finally, the master has a local package repository, for serving the necessary software for the execution nodes.

All computational nodes are, of course, executing hosts in the SGE environment. Besides that, they have all necessary software for running specific problems. As in the master host case, we have chosen Ubuntu Server as the distribution to use, providing security and homogeneity.

Several problems arise when the machines to be used in the cluster are not cluster-dedicated. To make things worse, these machines are heavily used in classes, which means that they are frequently abused and mistreated leading to a higher failure rate. This means that the machines are frequently repaired and their software reinstalled. Therefore, a system for a fully automatic install was needed. The choice of Ubuntu was not innocent. Being a derivative of Debian, Ubuntu provides the same process of install automatization akin to Red Hat's Kickstart. This process known as preseeding is fairly flexible for simple setups but lacks options for more advanced ones. Those problems were solved by patching the install `initrd`. We then used PXELINUX to make a network automated install on all computational nodes.

Preseeding does not solve all the problems, though. Since the labs are internally interconnected using Fast Ethernet (different labs are interconnected using Gigabit Ethernet) maintaining the highest network capacity for running jobs is problematic. To relieve the pressure on the network all software except SGE is locally installed and not network mounted as it is usual. We manage this by creating a meta-package that "depends" on all packages that should be installed in the cluster nodes. Adding a new package is done by simply rebuilding the meta-package and performing an upgrade on each node.

One nuclear part of our infrastructure is the transition between normal lab mode and cluster mode. In our case, several computer labs are physically closed for about seven hours, each day, after midnight thus allowing us to make the transitions remotely. The only task our cluster requires from the existing systems administration is to automatically restart the machines at the right time. From that point, we take control over the boot process by providing a netboot server.The netboot process (using PXELINUX) loads a grub image to chainload to the cluster partition. The transition back to normal lab mode is done using a cron entry in every node and by changing the PXELINUX and DHCP configuration in the master.

We plan to use Globus Toolkit [6] to integrate our cluster in a grid environment. Globus can be directly deployed to use SGE as the job manager. In particular, we plan to use Globus GRAM web-service [5] for remote job submission and control and GridWay [7] for meta-scheduling, execution management and resource brokering.

3 Validation and Benchmarks

In order to validate our results we measured the computational power of our infra-structure. All laboratory machines are equipped with one gigabyte of mem-

ory and 100Mbps network cards (connections between labs uses Gigabit Ethernet). Our choice for benchmarking was HPL (High Performance Linpack) [1], which is almost a "de facto" standard for high performance computing. For our initial performance experiment, we selected 30 machines from a total of 34 that are available in two labs (17 were "Athlon XP 2600+" and 13 were "Athlon XP 2500+").

Analyzing all the preliminary tests we did, the best performance we got (our `Rmax`) was 36.31 GFLOPs. On a single machine, LINPACK [1] attained the maximum value of 1.9 GFLOPs ("Athlon XP 2600+") which gives theoretically a maximum of 57 GFLOPs with thirty machines, assuming a perfect network connection. This would mean that our cluster has an efficiency of 63%. However, the real theoretical peak (`Rpeak`) of the whole thirty computers is 171 GFLOPs.

4 Conclusions and Future Work

In this paper we described an architecture that allows computational clusters to be built seamlessly from computer labs. The cluster works in a dual mode between computing periods and normal computer lab usage. The management of the cluster is fully automatic and the context switch is performed using a scheme of differentiated network boot. Our solution also has the advantage of being orthogonal to the software installed in the labs and almost completely independently administered. The current implementation of the architecture is running and is stable. The preliminary results obtained with our four computer labs and a basic assessment of the computational topology for the LINPACK benchmark are encouraging.

We plan to extend the cluster to all laboratories (around 100 computers) and acquire a better understanding of the network performance. We also plan to plug this resource to the University of Porto Grid Computing platform, using Globus. The deployment of a web-service for job submission, resource management and visualization is also in our plans.

References

1. HPL – a portable implementation of the high-performance linpack benchmark for distributed-memory computers. http://www.netlib.org/benchmark/hpl/.
2. PBS – portable batch system. http://www.openpbs.org/.
3. Sun grid engine. http://gridengine.sunsource.net/.
4. TORQUE resource manager. http://www.clusterresources.com/pages/products/torque-resource-manager.php.
5. Martin Feller, Ian Foster, and Stuart Martin. GT4 GRAM: A functionality and performance study, 2007.
6. Ian Foster and Carl Kesselman. Globus: a Metacomputing Infrastructure Toolkit. *Int. Journal of High Performance Computing Applications*, 11(2):115–128, 1997.
7. Eduardo Hudo, Ruben S. Montero, and Ignacio M. Llorente. The GridWay framework for adaptive scheduling and execution on grids. *Scalable Computing: Practice and Experience*, 6:1–8, 2005.
8. W.J. Litzkow, M. Livny, and M.W. Mutka. Condor-a hunter of idle workstations. In *8th International Conference on Distributed Computing Systems*, pages 104–111, June 1988.