MyGrid – A complete solution for running Bag-of-Tasks Applications

Lauro Beltrão Costa, Loreno Feitosa, Eliane Araújo, Gustavo Mendes, Roberta Coelho, Walfredo Cirne, Daniel Fireman

Departamento de Sistemas e Computação - Universidade Federal de Campina Grande (UFCG) Caixa Postal 10.106 – 59.109-970 – Campina Grande – PB – Brasil

{lauro,loreno,eliane,gustavo,roberta,walfredo,fireman}@dsc.ufcg.edu.br

Abstract. *MyGrid is a complete grid solution for running Bag-of-Tasks applications (i.e. parallel applications whose tasks are independent) over whatever resources are available to the user. MyGrid middleware empowers users to interoperate with heterogeneous computational resources across geographic and administrative boundaries. Due to MyGrid's flexible architecture, it is easy to add a new kind of machine in the grid through an abstraction called grid machine. The grid machines implementations currently available in MyGrid are user agent, grid script and globus grid machine. This work shows how MyGrid works and its main concepts. We present a parallel application, MyPhotoGrid, in order to demonstrate a practical use of MyGrid.*

1. Introduction

Grid computing is becoming a buzzword ubiquitously present not only in academia but also in industrial area. What is the aim of grid computing and why more and more people are interested in it? Grid computing promises the virtualization of computing, making it possible to think about computing on demand. Congregating machines from different networks and using its idle cycles would provide computational power similar to those offered by supercomputers.

In fact, the grid infrastructure to support this vision is not yet available. Nevertheless, high performance computing and scientific academic applications are starting to take advantage of grid computing. MyGrid intends to be a complete solution for grid computing [6]. It offers not only the middleware but also the abstractions and services needed for creation and usage of a grid. In order to achieve its goals, MyGrid forfeits supporting arbitrary application for focusing on Bag-of-Tasks (BoT) applications.

BoT applications are those parallel applications whose tasks are independent of each other. Despite their simplicity, BoT applications are used in a variety of scenarios, including data mining, massive searches (such as key breaking), parameter sweeps [1], simulations, fractal calculations, computational biology [2], and computer imaging [3][4].

MyGrid's design goals are to create a system that is *simple*, *complete* and *encompassing*. By simple, we imply that MyGrid should be as close as possible to an out-of-the-box solution. Users want to run their applications. The least they get involved into grid details, the better. Towards this goal, MyGrid team is working on minimizing the installation effort. This is important because if the user had to manually install MyGrid on the numerous machines that form the grid, the simplicity of the solution would suffer.

Complete means that MyGrid must cover the whole production cycle, from development to execution, passing through deployment and manipulation of input and output. This goal is the key for MyGrid to be useful in practice. In order to support all activities within the production cycle of a BoT application, MyGrid provides the notion of working environment, which is a small set of abstractions that enables users to manipulate their files on the Grid.

Owing to their loosely coupled nature, BoT applications can potentially use a very large number of processors. Therefore, we do not want MyGrid to preclude the user from using a given processor. We want MyGrid to be encompassing, in the sense that all machines the users have access to can be utilized to run their BoT applications

The remaining of this paper is structured as follows: Section 2 presents the key concepts needed to understand MyGrid. Section 3 describes the MyGrid's architecture and details the main components of its architecture. Section 4 shows MyPhotoGrid, a sample parallel application that runs over MyGrid. Finally, Section 5 finishes up the paper with our final remarks.

2. MyGrid Main Concepts

In order to understand how MyGrid works it is necessary to get familiarized with some concepts. A BoT application (and thus a MyGrid job) is composed by a set of independent tasks. The user must have a machine to coordinates the execution of BoT applications through MyGrid. This machine is called *home machine*. The user submits the tasks that compose the application to the home machine, which is responsible for farming out the tasks in the user's grid. The home machine will oftentimes be the user's desktop. In this sense, the user has good access to the machine and may set up a working environment on it.

The home machine schedules tasks to run on *grid machines*. In contrast to the home machine, grid machines may not have been customized by the user to create a familiar working environment. Moreover, grid machines do not necessarily share file systems with the home machine, nor do they have the same software installed on them (even the operating system can differ). Ideally, the user does not want to treat grid machines individually.

Each MyGrid *task* is formed by *initial*, *grid*, and *final* subtasks, which are executed sequentially in this order. Subtasks are external commands invoked by MyGrid. The initial and final subtasks are executed on the home machine. The initial subtask is meant to set up the task's environment by, for instance, transferring the input data to the grid machine. The final subtask is typically used to collect the task's results back to the home machine. The grid subtask runs on a grid machine and performs the computation per se.

3. MyGrid Architecture

The main goal of the MyGrid is to provide a global execution environment composed of all machines that user has access to. In order to achieve this goal, MyGrid strives to isolate two points of heterogeneity: (i) the usage of a resource (ii) and the way to gain its access. This is done by defining two interfaces: the *GridMachine* (GuM) interface and the *GridMachineProvider* (GuMP). The MyGrid scheduler (the module responsible for starting and monitoring the user's tasks) uses GuM as an abstraction for a machine and GuMP as an abstraction for a set of machines.

GuM encapsulates the minimal set of services that must be available in a machine, allowing its use as part of a grid. These services are:

- Task start-up on a grid machine (remote execution);
- Cancellation of a running task;
- File transfer from the grid machine to the home machine;
- File transfer from the home machine to the grid machine;
- Connectivity check (ping);

This virtualization enables a simple working of the MyGrid since it does not need to know details about grid machines. Note that this also eases the addition of new resources to MyGrid, since implementing the interface is all that is needed.

GuMP make it possible for MyGrid to use resources controlled by third entities. This is key for MyGrid to be able to use parallel supercomputers. Parallel supercomputers are important elements in the Grid [5], because of its great computational power. Since MyGrid aims to use every resource that users have access to, it is natural to provide a way to use supercomputers resources, increasing the size of the users' grid. However, their way of use is different from other computational resources. Supercomputers are *space-shared,* what means that many users can use the resource but not at same time, differently from most machines, which are *time-shared* (more common). The space sharing is enforced by the supercomputer resource manager (or scheduler).

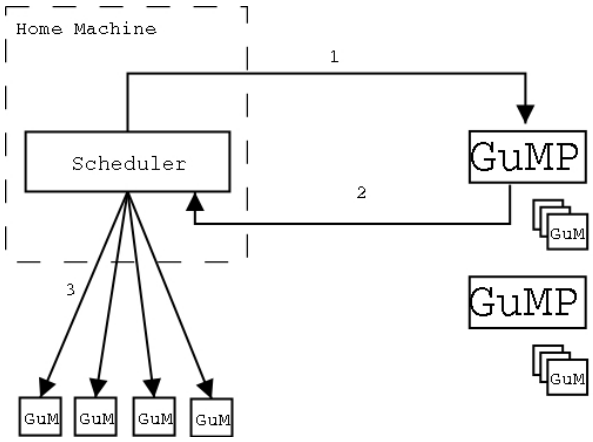

Figure 1 – MyGrid architecture

Our approach to fulfill this necessity is the GridMachineProvider. The *GridMachineProvider* (GuMP) abstracts how the user must gain access to resources

(i.e. the job submissions details that are particular of each resource manager). Each GuMP obtains and controls the access to a set of resources, managing their availability and access gaining method. Thus, it provides MyGrid with GuMs that can be used to access transparently these resources, without knowing the underlying means used by the implementation of the GridMachine interface to use the resources.

The Figure 1 shows how the architecture works. The MyGrid Scheduler asks for GuMs (1), the GuMProvider gains access to requested resources and delivers the GuMs (2); the Scheduler assigns tasks to received GuMs (3). This approach keeps the MyGrid Scheduler simple since a GuMP "translates" space shared resources into time-shared ones.

3.1. GridMachine Implementations

As shown in section 2, the GuM interface provides methods that enable MyGrid to communicate with grid machines. MyGrid has three native implementations of GuM: (i) UserAgent, (ii) GridScript, and (iii) GlobusGridMachine. Figure 2 shows the GridMachine interface and its implementations, which will be described below.

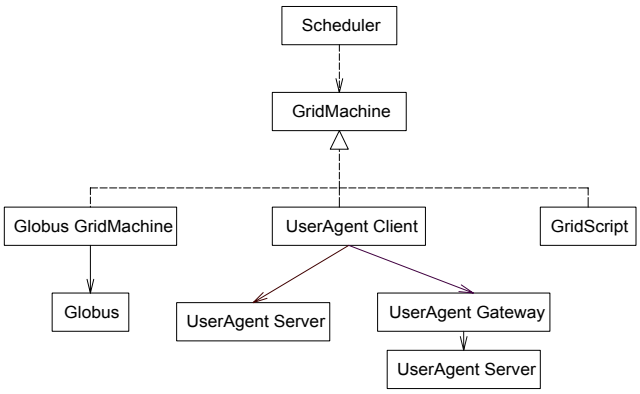

Figure 2 - Grid Machine Interface and its implementations

UserAgent

UserAgent (UA) is simple Java-based implementation of GuM designed for the situation on which it is easy for the user to install software on the grid machines. The communication between MyGrid and UA is done via *Remote Method Invocation* – RMI [9].

Since UserAgent is a component of MyGrid, it was extended to implement other functionalities like instrumentation and fault handling. Another important feature implemented in the UserAgent is a relay that allows communication with machines within a private network. Once such machines are identified by a private IP addresses, they could not be accessed directly by the *home machine*, only via a host that has access to *home* and *grid machines*. To solve this problem, we developed a user-level relay called UserAgent Gateway.

GridScript

The Grid Script (GS) provides the same functionalities of User Agent using scripts instead of running a Java process. The script can be used when the user has access to the GridMachine through widely deployed software, such as: ssh, rsh and ftp. Users can take advantage of GS to access a GridMachine through a firewall since firewalls typically leave common service ports open. Moreover, users can furnish GS with whatever access scripts they have. This enables MyGrid to fulfill its promises of using whatever resources a user can access (provided that the user has a command-line way to do remote execution and file transfer to the resource).

Globus

In order to interoperate with Globus 3.0 [8], maybe the most popular grid middleware and the first complete implementation of OGSI [10] [11], we developed a GuM implementation named GlobusGridMachine. This integration allows redirecting requests received from MyGrid to Globus machines. File transfer and tasks executions are performed by Globus using its grid services. These requests are done in a secure way, using the Globus Security Infrastructure – GSI [8].

Globus uses two services to perform remote execution and file transfer, respectively: Master Fork Managed Job Factory Service - MFMJFS and Grid File Transfer Protocol – GridFTP. The first one is a grid service that redirects MyGrid requests' to Grid Resource Allocator Manager – GRAM, responsible to execute tasks. The second one is a service that should be installed in *home machine* and *grid machines* to allow transfer files.

3.2. GuMP Implementations

In the current version, two GuMPs are available in MyGrid: LocalGuMP that manages machines which user has direct access to and CRONOGuMP that communicates with CRONO Scheduler. CRONO [13] is a supercomputer resource manager developed by PUC-RS. Its main purpose is to reduce the complexity of the management, avoiding generic resources, keeping it simple to install and use.

The jobs submitted to CRONO by CRONOGuMP are not the tasks of MyGrid's Job themselves but UserAgents. In this way, when a UserAgent is available it informs to CRONOGuMP. This is a technique very similar to shown in [3].

This approach has several advantages comparing with sending the tasks of MyGrid's job: (i) the functioning of MyGrid Scheduler is the same, (ii) we can use a processor many times before losing it and (iii) since we implement UserAgent is easier to adapt it and causing it able to monitor and inform some events to CRONOGuMP and consequently to MyGrid.

In future versions MyGrid will provide others GuMPs to communicate with others Supercomputers' schedulers (such as PBS and Maui).

4. MyPhotoGrid

While MyGrid was developed to support scientific applications, we selected for the demo, an application that could appeal to wider audience. The introduction of high-resolution digital cameras placed regular users as the producers of enormous amounts of

data. For example, one of authors of this paper produced 18 GB of digital photos in the last 6 months. An interesting alternative would be to use the grid to create digital albums from these photos. It is supercomputing reaching the masses! That is exactly what MyPhotoGrid does.

MyPhotoGrid creates a web album from large set of digital images. A possible kind of digital albums is composed by three images (a thumbnail, a regular size image and a big one) generated from the original picture. The album index contains a list of thumbnails that are hyperlinks to the regular image, which links to the big image.

The process of album creation is time consuming because processing thousands of images can take a lot of time. Since photo editing is independent from each other what makes it a BoT application. The idea is to use MyGrid to make photo editing in a parallel way. When all photos processing results were available, the index will be built locally. MyPhotoGrid is an application available in MyGrid distribution that performs digital photo album generation.

In order to use MyGrid it is necessary to create a grid description file and a job description file. The grid description file contains needed information for MyGrid to communicate with each grid machine. A sample of grid description file generated by MyPhotoGrid can be seen in Figure 3.

```
proc:
name = jedi.lsd.ufcg.edu.br
type = user_agent
attributes = linux
copy_to = scp %localdir/%file %machine:%remotedir/
copy_from = scp %machine:%remotedir/%file %localdir/
rem_exec = ssh %machine %command

proc:
name = yoda.lsd.ufcg.edu.br
type = user_agent
attributes = linux
copy_to = scp %localdir/%file %machine:%remotedir/
copy_from = scp %machine:%remotedir/%file %localdir/
rem_exec = ssh %machine %command
```

Figure 3 - Grid description file

It is also necessary to describe the job. Such description specifies, for each task, how to execute the task, which files must be copied in order to the application works and which files must be collected. A sample job description file generated automatically by MyPhotoGrid is shown in Figure 4. For further information about configuration files, see [11].

```
JobAttribExpression = linux

task:
init = scripts/init.myphotogrid images/TimesSquare.tif
remote = java ImageProcessor images/TimesSquare.tif
final = scripts/collect.myphotogrid TimesSquare.tif
task:
init = scripts/init.myphotogrid images/NewYork.tif
remote = java ImageProcessor images/NewYork.tif
final = scripts/collect.myphotogrid NewYork.tif
```

Figure 4 - Job description file

MyPhotoGrid works as shown in Figure 5. It generates the job description file from the set of photos to be processed (1). The job and grid description are passed to MyGrid that uses them to schedule the tasks, processing images in grid machines (2). After all photos are generated and their results are available in the home machine (3), the album creation is performed (4).

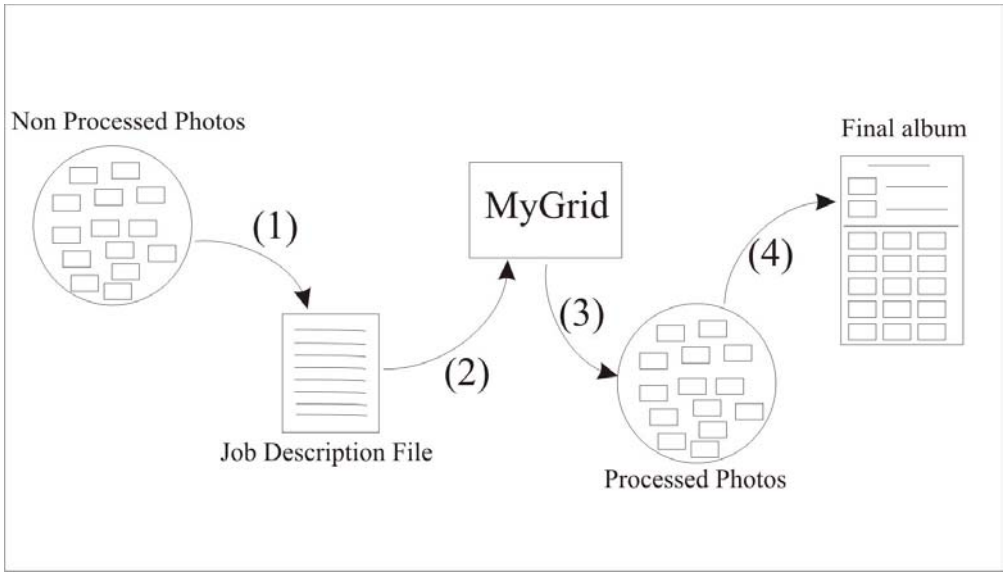

Figure 5 - MyPhotoGrid lifecycle

We have performed a set of experiments to generate an album to 89 photos with a total input size of 740MB. And 13 similar non-dedicated machines of our lab composed the grid description file.

The experiments show that MyPhotoGrid has attained a performance that is 4.2 times faster than its correspondent standalone version. It is worth to point out that the standalone version executed in a dedicated machine with similar configuration and local data access instead of data transfer across LAN like performed in MyPhotoGrid.

To obtain good results, the *init* and *final* sub-tasks could not consume a lot of time in comparing with *remote* sub-task. This occurs because the grid machines must wait the data transfer to begin execution of a task, i.e. data transfer is overhead. Further discussion about efficiency of grid for BoT applications can be found in [6][13] .

5. Final Remarks

This work presented a practical view of MyGrid along with its abstractions and architecture. MyGrid is a system designed to make grid computing real. Or, at least, real for the users of Bag-of-Tasks applications (those parallel applications whose tasks are independent). The rationale behind MyGrid is to focus on a narrower problem than the general grid problem, but solve it thoroughly, in a way that real users can benefit from .

While most Bag-of-Tasks applications are very technical, we purposely selected an application that could interest everybody. We intend to present MyPhotoGrid, a high-performance MyGrid-based solution to process and organize large amounts of digital photos. Not only MyPhotoGrid is appealing to a wider audience, it is also a good

example of the grid's promises: Widely available large compute power to change the way we do computing.

References

[1] D. Abramson, J. Giddy and L. Kotler. *High Performance Parametric Modeling with Nimrod/G:Killer Application for Global Grid*, IPDPS'2000, pp. 520-528, Cancun Mexico, IEEE CS Press, USA, 2000.

[2] J. R. Stiles, T. M. Bartol, E. E. Salpeter, and M. M. Salpeter. *Monte Carlo Simulation of Neuromuscular Transmitter Release Using MCell, a General Simulator of Cellular Physiological Processes*. Computational Neuroscience, pages 279-284, 1998.

[3] S. Smallen, W. Cirne and J. Frey et al. *Combining Workstations and Supercomputers to support Grid Applications: The Parallel Tomography Experience*. Proceedings of the HCW'2000 - Heterogeneous Computing Workshop. 2000. http://walfredo.dsc.ufcg.edu.br/resume.html#publications

[4] S. Smallen, H. Casanova, and F. Berman. *Applying Scheduling and Tuning to On-line Parallel Tomography*. Proceedings of Supercomputing 01, Denver, Colorado, USA, November 2001.

[5] I. Foster, C. Kesselman, and S. Tuecke. *The Anatomy of the Grid: Enabling scalable virtual organizations*. International Journal of Supercomputer Applications, 2001.

[6] W. Cirne, D. Paranhos, L. Costa, E. Santos-Neto, F. Brasileiro, J. Sauvé, F. Alves, C. Osthoff, C. Silveira, *Running Bag-of-Tasks Applications on Computational Grids: The MyGrid Approach,* Proceedings of the ICCP'2003 - International Conference on Parallel Processing, 2003.

[7] Globus Project Web Site, http://www.globus.org. (January, 2004)

[8] Remote Method Invocation, http://java.sun.com/products/jdk/rmi/

[9] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, D. Snelling, Open Grid Services Infrastructure (OGSI), Global Grid Forum

[10] I. Foster, D. Gannon,The Open Grid Services Architecture Platform, 2003. http://www-unix.gridforum.org/mail_archive/ogsa-wg/doc00016.doc (January, 2004)

[11] MyGrid http://www.ourgrid.org/mygrid (January, 2004)

[12] M.A.S. Netto, C.A.F. de Rose, *Crono: A configurable and easy to maintain resource manager optimized for small and mid-size GNU/Linux Clusters,* Proceedings of the ICCP'2003 - International Conference on Parallel Processing, 2003.

[13] W. Cirne, F. Brasileiro, J. Sauvé, N. Andrade, D. Paranhos, E. Santos-Neto, R. Medeiros. *Grid Computing for Bag of Tasks Applications.* Proceedings of the Third IFIP Conference on E-Commerce, E-Business and E-Goverment. September 2003