Distributed Application Management in Heterogeneous GRIDS

P. Lindner, N. Curre-Linde, M. M. Resch
High Performance Computing Center Stuttgart (HLRS), Stuttgart, Germany

E. Gabriel
High Performance Computing Center Stuttgart (HLRS), Stuttgart, Germany
and
Innovative Computing Laboratory, Computer Science Department
University of Tennessee, Knoxville, TN, USA

Abstract

Distributing an application on several machines is one of the key aspects of Grid-computing. In the last few years several groups have developed solutions for the occurring communication problems. However, users are still left on their own when it comes to the handling of a Grid-computer, as soon as they are facing a mix of several Grid software environments on target machines. This paper presents a tool, which eases the handling of a Grid-computer, by hiding some of the complexity of heterogeneous Grid-environments from the user.

1. INTRODUCTION

The Grid is generally seen as a concept for 'coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organization' [5]. The original idea came from scientists, who were mainly interested in the scientific solution of their problem. Their jobs can be executed on any machine, respectively on any set of machines, to which they have access. Like in Metacomputing, which was a popular concept in the Mid 90s [4], the distribution of jobs onto several machines is an important part of the Grid concept. The need for doing so mainly comes from applications, which have a high demand on computational resources [10] [2] or for increasing throughput on the machines and reducing turnaround times.

Distributing an MPI job onto several machines imposes however many problems to the user and the application. Most of the problems stem from the fact, that the Grid is heterogeneous from several aspects. This starts with different data representations on different machines, which requires data conversion either in the MPI-library or in the application. Various processor speeds and differences in the available memory require the application to have a smart initial load distribution and/or dynamic load balancing. The differences on the communication characteristics for communication between processes located on potentially different sites require distinct programming techniques for hiding the wide area latency and dealing with the low bandwidth. The latter aspects have been handled in the last couple of years by several projects, which focused on the development of MPI libraries for Grid environments, like MPICH-G2 [12], STAMPI [11], MPI_Connect [6] and PACX-MPI [9].

Another aspect of heterogeneity in the Grid has been up to now not really handled. The user has to deal also with several mechanisms of how to access a machine and how to start a job on it. While some computing centers run ssh [17], rsh or similar protocols, Grid software environments like Globus [5], Legion [16] or UNICORE [1] are getting more common. There is a high probability that the user has to deal with a mix of these access mechanisms when distributing an application onto several hosts. In the following this is the aspect of heterogeneity which we mean, if we refer to the Grid as being heterogeneous.

Beside troubling the user, this aspect of the Grid affects also the startup procedure of MPI libraries. Library developers tend to integrate their communication library and the startup procedure into a specific Grid software environment. For example, MPICH-G2 is part of the the Globus software toolkit, and uses the Globus

mechanisms for authentication, authorization and communication. However, this implies, that it is hard to execute an MPICH-G2 job on a machine, where Globus is not installed.

The default startup procedure of PACX-MPI requires just a login on each machine. In addition, further methods for starting PACX-MPI jobs are provided, which interface directly with several Grid computing environments. Together with the Globus group, an interface has been defined and implemented, which allows a PACX-MPI job to use all Globus features mentioned in the previous section [22]. Furthermore, a plug-in for the UNICORE Grid-computing environment for PACX-MPI is currently under development.

However, none of these solutions support the user, if he has to deal with a heterogeneous Grid-computing environment. If Globus is installed on the first machine, UNICORE on the second one, and ssh is the only way to access the third machine, the user currently cannot use a unique startup procedure with any of the libraries mentioned above. He might run a PACX-MPI job on all machines, but in this case, he is not taking advantage of any of the features of the Grid software environments.

This situation is not satisfactory. Our experience in the last years doing distributed simulations all over the world (see e.g. [13]) showed furthermore, that starting with a certain number of machines, it is not easy to handle a terminal for each machine. Especially, PACX-MPI requires some configuration files to be available on all hosts. When modifying the configuration of the metacomputer, the user has to update the configuration files of PACX-MPI on all hosts. While this is not a problem for two or three machines, it is an error-prone procedure for more.

To solve the problems described, a tool is currently under development in the frame of the DAMIEN project [19] sponsored by the European Commission called the 'Configuration Manager', and it is designed to handle distributed MPI-applications in heterogeneous Grid-environments. The primary goal of the Configuration Manager is to ease the handling of resources and applications for PACX-MPI jobs.

The structure of this paper is as follows: in the next section we describe PACX-MPI and the required configuration files for the library. Then, we present the goals, concept and the implementation of the Configuration Manager. The following section gives some usage examples of the Configuration Manager. Finally, we summarize the paper and give a brief overview about the ongoing work in this area.

2. PACX-MPI

This section describes the communication library PACX-MPI [9] . The focus is on the startup procedure and the configuration files of PACX-MPI, since these aspects are necessary for understanding the concept of the Configuration Managers.

PACX-MPI is the implementation of the message passing standard MPI that is optimized for Meta- and Grid-computing environments. Key aspect of these environments is the difference in the characteristics of the communication layers regarding latency and bandwidth. If communication occurs between two processes within the same machine, the latency is usually in the range of microseconds, and the bandwidth in the range of tens to hundreds of MBytes/s. However, if the communication partners are distributed on different machines, latencies tend to be in the range of tens of milliseconds, and bandwidth drops to some MBits/s. PACX-MPI takes these characteristics into account by using two independent layers for handling communication between processes on the same machine (*internal communication*) and for handling communication between processes on different machines (*external communication*). Internal communications are handled by using the vendor MPI-library on each machine, since this is a highly efficient and portable communication method. For external communications two communication daemons have been introduced, which translate the MPI messages into TCP/IP messages or vice versa. Furthermore, PACX-MPI has optimized the high level functions of the MPI-standard, e.g. the collective operations [7] or the derived datatypes [8] , to these environments. For more details, see e.g. [9].

When starting an MPI-application on several hosts simultaneously, PACX-MPI reads two configuration files, which answer the following questions:

- Which machines are participating in the current run ?

- Which is the rank of this machine in the given configuration ?
- Which network protocol is a machine using for communicating to each other host ?

The first question is important for determining which machines have to be contacted at all. The second question is dealing with the problem of how to assign a global, unique rank to each process in the global configuration. This problem is solved in PACX-MPI by using for each participating machine a unique host-rank and the local MPI-rank of each process. Finally, the third question clarifies the communication protocol between each pair of hosts and the attributes of the protocol. This information consists of a list of protocols supported by PACX-MPI (currently TCP/IP, a native ATM protocol and a secure protocol based on SSL), and some protocol dependent attributes, e.g. the port number under which a machine can be reached for the TCP/IP protocol.

This information is split into two configuration files: the hostfile, which contains the information regarding the participating machines, and the netfile, which deals with the network parameters of the current run. While the hostfile is mandatory for each run, the second configuration file is optional. In case the library does not find this file, it assumes, that all machines are using the default networking protocol (TCP/IP) with default port-numbers.

In the following, we would like to describe the configuration files using a simple example of three machines, called host1, host2 and host3, with 16 processors on the first two machines, and 24 processors on the last one. While the first two machines are within the same institution and protected by a firewall, the third machine is located at a different site. Therefore, one can use standard TCP/IP between host1 and host2, while a secure protocol is required to the third machine. This configuration is also shown in figure[1].

In the most simple case, the according hostfile looks like the following:

```
host1 16
host2 16
host3 24
```

FIGURE 1: Example Configuration consisting of three hosts

The hostfile consists of a list of the machines and the number of processes used for the application on each machine. The format is straight forward and based on the well-known machinefile of MPICH [15]. This information however is not enough for cluster-like machines (e.g. IBM RS6000SP, PC-Clusters), where each processor has its own IP-address, and therefore the machine can not be identified by a single name or IP-

address. In this case, the format of the hostfile and the whole startup procedure is different: it contains the keyword `Server`, the name or the IP-address of the Startup-Server and the rank of the machine in the global configuration.

```
Server <Startup-Server > <host-rank>
```

The Startup-Server is an independent process. Its purpose is to collect relevant startup-information, e.g. the IP-addresses of all PACX-MPI daemons, and redistribute this in a second step to all hosts.

For the example described above the second configuration file, the netfile looks like the following:

```
BEGIN 1;
HOST=3, PROTOCOL=SSL;
END;
BEGIN host2;
HOST=3, PROTOCOL=SSL;
END;
BEGIN 3;
HOST=1, PROTOCOL=SSL;
HOST=2, PROTOCOL=SSL;
END;
```

The file contains one section for each participating machine. A section starts with a `BEGIN` statement and either the rank of the machine in the global configuration or the name of the host and ends with an `END` statement. Inside each section, the network connections of this host to all other machines are described. To simplify the usage of this file, just those connections between hosts have to be described, which are not using default parameters. Therefore, the connection between `host1` and `host2` need not be described.

3. GOALS, CONCEPT AND IMPLEMENTATION OF THE CONFIGURATION MANAGER

To automatize the handling of resources for a PACX-MPI job, the Configuration Manager has to provide the following basic functionality:

- Create the configuration files for PACX-MPI: Based on a GUI, the user describes the required resources for the current run.
- Distribute the configuration files onto all machines: Since the configuration files of PACX-MPI have to be available on all machines, the Configuration Manager has to transfer the most recent version of these files to all machines.
- Start the application on all hosts: Among the information, which the user has to provide when specifying a machine in the Configuration Manager, is a command which will be used to start the application on the given host.

As described in the introduction, the heterogeneity of the Grid will not only consist of different types of machines with different data representations, but also the method how the user can access the machines. Grid-software like Globus, Legion and UNICORE are meanwhile well established. On the other hand, some machines will probably continue to be accessible just by `ssh`, `rsh` or similar methods. Therefore, another goal of the Configuration Manager is to hide the complexity of how to access each individual machine for the every-day use.

To hide the complexity of heterogeneous Grids from the user, the Configuration Manager requires the following

functionality for each access-mechanism:

- Authentication and Authorization: This can be either done by a combination of username and password for ssh, or a certificate for Globus and UNICORE.
- File-transfer: This is required for sending the current versions of the PACX-MPI hostfile and netfile to all hosts. Furthermore, file transfer is necessary for executable staging or transferring input files to the according machines respectively moving output files back onto local machines.
- Job submission: This is used to start each part of the distributed job on all machines.

The job submission again can be detailed to the following items:

- Name and path of the application: This has to be handled separately for each machine, since the requirement of having the same name and path for the application on all systems can often not be fulfilled (e.g. the name is <myappl>.<architecture> .
- Number of processors: Again this is a machine and run-specific information and has to be stored for each host separately.
- Generic MPI command: Specifying the command how to start an MPI application on a certain machine. In fact, the user often has to face the problem, that several MPI libraries are installed on a machine with different MPI start commands (e.g. mpirun, mpiexec, poe).

In the concept of the Configuration Manager foreseen is the possibility to store the settings for a certain machine. Typically, an end-user will have a limited number of machines which he uses for his distributed runs. This feature allows him also to store the same machine using different MPI implementation (e.g. <machinename>.<vendormpi> and <machinename>.<mpich1.2.3>).

3.1 Implementation Issues

The Configuration Manager is implemented in Java, following Model-View-Controller (MVC) principles. This concept separates input, processing and output in graphically oriented software. Therefore, the programmer using the MVC principles has to deal with three different components:

- The model, which represents the application object
- The view, which realizes the representation of the model on the screen and
- The controller, which defines the reaction on input of the user-interfaces.

FIGURE 2: Screenshot of the Configuration Manager

The current implementation of the Configuration Manager supports two different methods to save and open a configuration. In case the same user wants to reuse the configuration, it is convenient for him not to enter the passwords or to specify the certificates for each machine every time. Therefore, the Configuration Manager encrypts all passwords and certificates when saving the configuration. Thus, the user has to enter a password when opening a configuration. The encryption functionality is handled using the Java Cryptography Extensions (JCE), using an implementation of Bouncy Castle [21].

However, it is sometimes desirable to pass a configuration to another person. In this case passwords and certificates should not be included in the configuration. The Configuration Manager supports this scenario by providing export/import functions.

For the integration of the startup-server mechanism of PACX-MPI for handling cluster-like machines, an implementation of the startup-server in Java has been developed. When using this startup method of PACX-MPI, the GUI automatically changes the content of the hostfile. The user can then launch the startup-server, which will be executed in a separate thread. Thus, the handling of different formats of the PACX-MPI configuration files is nearly seamless for the end-user, improving the usability of PACX-MPI.

3.2 An ssh-based Grid-terminal

The first access-mechanism implemented in the Configuration Manager was ssh. Due to its current wide usage, ssh is still considered to be the most widespread method to access a high-performance computer. The implementation in the Configuration Manager is based on the Java ssh-implementation of Mindbright [18]. Based on this library, a *grid-terminal* (gridterm) has been developed, which allows the user to execute a command simultaneously on one, a subset or all hosts in the current configuration (see fig.[3]). Stdout and stderr are merged for all hosts into a single window. This helps to avoid the situation mentioned in the introduction, where starting with a certain number of machines, the user has problems to manage regular terminals on all hosts. It is therefore a major improvement of a regular terminal for Grid-environments. At the current stage, the Configuration Manager supports password based ssh-access to the different machines.

FIGURE 3: Screenshot of the grid-terminal

3.3 Integration with UNICORE

Currently ongoing work is the implementation of UNICORE as a second access mechanism. UNICORE (Uniform Interface to Computer Resources) is a software infrastructure having the goal to provide the user an easy way to submit jobs or chains of jobs to supercomputers. Furthermore, UNICORE supports the idea of specifying a multi-site job, where the first part of an application is executed on the first machine, and the second part is executed on the second machine. In the frame of the UNICORE Plus project, the handling of Metacomputing jobs with PACX-MPI is already supported. To realize this, the plug-in mechanism of UNICORE is used for creating the PACX-MPI configuration files and for launching the jobs. However, as mentioned previously in the introduction, this mechanism requires UNICORE to be installed on all machines, which are part of the Metacomputer. Therefore, another solution for handling mixed environments is currently being developed for the Configuration Manager.

Like ssh, UNICORE supports file-transfer and the starting of a job on a machine. Two major differences to ssh are however important when integrating UNICORE into the Configuration Manager:

- UNICORE does currently not support interactive access to computing resources. While this is ongoing work in the frame of the Eurogrid project [14], this does mean, that all jobs (even a simple ls-command) will be executed as batch jobs. Since co-allocation of resources across several sites is still an unsolved problem, this does also impose currently, that it is up to the user to make sure, that all resources are available at the same time.
- The access to UNICORE resources is based on X509 certificates, and not on passwords. Since this is also part of the ssh2 protocol, the Configuration Manager is currently being extended to support the management of certificates as-well.

4. EXAMPLE OF USAGE SCENARIOS

The Configuration Manager has been used in several demonstrations and scenarios. It is currently tested under industrial conditions by the European Aeronautic Defense and Space Company (EADS) in the frame of the DAMIEN project. In this section, we will give brief descriptions of another scenario presented in the frame of the DAMIEN project, where computational resources in Stuttgart/Germany, Rouen/France and Barcelona/Spain were connected for a distributed simulation between these sites. The machines used were an SGI Origin 2000 in both Rouen and Stuttgart and an IBM RS6000SP in Barcelona.

During the preparation of the demonstration, the network between the sites was continuously modified in order to improve the connection of each site to the new European high-speed backbone network Geant. This time, the Configuration Manager turned out to be extremely useful, since the launching of the application could be done within seconds. Therefore, an immediate response to the networking experts about the quality of the network and the impact of their modifications could be provided.

The application used for the simulation was a flow-solver called URANUS [3]. This code simulates the flow around a space vehicle during the reentry phase. In the final run, a simulation of the MIRKA-vehicle has been successfully demonstrated for a medium size problem.

Of importance for the Configuration Manager is one technical issues of URANUS, its directory structure. During compile-time, a directory will be created for each architecture. Thus, the executables are not located in the same directory for the IBM RS6000SP and for the SGI Origin2000. While the whole run could also be done without the Configuration Manager, its usage eased the management of the application and provided an easy to use tool for Grid- and Metacomputing.

5. CONCLUSIONS AND FUTURE PLANS

In this paper we presented a tool which helps to ease the usage of distributed resources in Grid-environments. The Configuration Manager creates the configuration files for PACX-MPI, transfers them to the participating machines and starts the application on all hosts simultaneously. Furthermore, it hides parts of the complexity of the Grid from the user by supporting several mechanisms of how to reach different machines. Currently implemented are ssh and (partially) UNICORE. This tools has proved its usability in several demonstrations and projects, and is currently being tested in an industrial environment.

A further aspect of the Configuration Manager has proved its usefulness beyond the submission of PACX-MPI jobs. The grid-terminal (gridterm) is an extension of a regular terminal, which enables simultaneous execution of commands and applications on several hosts in a Grid environment.

Further plans for the tool involve the support for other Grid software environments such as Globus or Legion. Together with the already implemented mechanisms, we believe we will then cover the most relevant software environments for Grid-computing.

In the frame of the DAMIEN project, PACX-MPI is extended to handle Quality-of-Service parameters. The Configuration Manager will support the specification of these parameters as-well.

An omnipresent question currently is the convergence of all Grid-tools toward the Open Grid Software Architecture (OGSA) [23], since most of the Grid software environments, like e.g. Globus or UNICORE, have

committed to convert toward this new architecture. In this concept Grid-computing is constructed out of several Grid Services with defined interfaces and semantics. These services are accessible by several protocols e.g. the Simple Object Access Protocol (SOAP) [24]. The development and implementation of a tool like the Configuration Manager could potentially be eased by a common architecture, since all operations should be formulated as requests to an abstract system, independent of the implementation underneath. However, such a common architecture would not really affect the Configuration Manager for two reasons: first, none of the systems are currently exceeding the status of pure demonstrations, and second, the argument, that some systems will remain just accessible through protocols like ssh, remains still valid.

ACKNOWLEDGMENTS

This work was supported by the European Commission in the DAMIEN project under contract number IST-2000-25406.

References

- [1] Jim Almond, Dave Snelling, 'UNICORE: Secure and Uniform Access to Distributed Resources via the World Wide Web, A White Paper, October 1998.
- [2] Gabrielle Allen, Thomas Damlitsch, Ian Foster, Nicholas T. Karonis, Matei Ripeanu, Edward Seidel, Brian Toonen, 'Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus', Supercomputing 2001, Denver 2001
- [3] Thomas B. Boenisch, Roland Ruehle, 'Efficient Flow Simulation with Structured Multiblock Meshes on Current Supercomputers', ERCOFTAC Bulletin No. 50: Parallel Computing in CFD, 2001.
- [4] Charly Catlett and Larry Smarr, 'Metacomputing', Communications of the ACM, 35(6):44-52, 1992.
- [5] Ian Foster, Carl Kesselmann, Steven Tuecke, 'Anatomy of the Grid: Enabling Scalable Virtual Organizations', International Journal of Supercomputing Applications, 15(3), 2001.
- [6] Graham E. Fagg, Kevin S. London, and Jack J. Dongarra, 'MPI_Connect Managing Heterogeneous {MPI} Applications Interoperation and Process Control', In V. Alexandrov and J. Dongarra, editors, Recent advances in Parallel Virtual Machine and Message Passing Interface, volume 1497 of Lecture Notes in Computer Science, pages 93–96. Springer, 1998, 5th European PVM/MPI Users' Group Meeting.
- [7] Edgar Gabriel, Michael M. Resch, and Roland Ruehle, 'Implementing MPI with Optimized Algorithms for Metacomputing', In Y.-S.-D. Anthony-Skjellum, Purushotham V. Bangalore, editor, Proceedings of the Third MPI Developer's and User's Conference, 1999.
- [8] Edgar Gabriel, Michael M. Resch, and Roland Ruehle, 'Implementing and Benchmarking Derived Datatypes in Metacomputing', In B. Hertzberger, A. Hoekstra, R. Williams (Eds.) High Performance Computing and Networking, pp. 493-502, Springer, 2001.
- [9] Edgar Gabriel, Michael M. Resch, Thomas Beisel, Rainer Keller, 'Distributed Computing in a Heterogeneous Computing Environment', Vassil Alexandrov, Jack Dongarra (Eds.) Recent Advances in Parallel Virtual Machine and Message Passing Interface, pp. 180-188, Springer, 1998.
- [10] Edgar Gabriel, Marc Lange and Roland Ruehle, 'Direct Numerical Simulation of Turbulent Reactive Flows in a Metacomputing Environment', Proceedings of the 2001 ICPP Workshops, pp. 237-244, 2001.
- [11] Toshiyuki Imamura, Yuichi Tsujita, Hiroshi Koide, and Hiroshi Takemiya, 'An Architecture of Stampi: MPI Library on a Cluster of Parallel Computers', J. Dongarra, P. Kacsuk, and N. Podhorszki, editors, Recent Advances in Parallel Virtual Machine and Message Passing Interface, number 1908 in Springer Lecture Notes in Computer Science, pages 200–207, Sept. 2000. 7th European PVM/MPI Users' Group Meeting.
- [12] Nicholas T. Karonis and Brian Toonen, MPICH-G2 web page <http://www.globus.org/mmpi>
- [13] Steven M. Pickles, John M. Brooke, Fumie C. Costen, Edgar Gabriel, Matthias Mueller, Michael M. Resch, and S. M. Ord, 'Metacomputing Across Intercontinental Networks', Future Generation Computer Systems (17),

pp. 911-918, Elsevier Science.

[14]EUROGRID web page: <http://www.eurogrid.org>

[15] William Gropp and Ewing Lusk, 'User's Guide for mpich, a Portable Implementation of MPI'.

[16] Andrew S. Grimshaw, Wm. A. Wulf, and the Legion team, 'The Legion vision of a worldwide virtual computer', Communications of the ACM, 40(1):39-45, 1997.

[17] OpenSSH web page <http://www.openssh.org>

[18] Mindterm Secure Shell web page: <http://www.mindbright.se>

[19] Matthias Mueller, Edgar Gabriel and Michael Resch, 'A Software Development Environment for Grid-Computing', Concurrency and Computation - Practice and Experience, special issue on Grid computing environments, to be published.

[20] DAMIEN - Distributed Application and Middleware for Industrial Use of European Networks. Web page to be found at <http://www.hlr.de/organization/pds/projects/damien>

[21] Bouncy Castle Crypto package. Web page to be found at <http://www.bouncycastle.org/documentation>

[22] B. Toonen, D. Ashton, E. Lusk, I. Foster, W. Gropp, E. Gabriel, R. Butler, N. Karonis, 'Interfacing Parallel Jobs to Process Managers', Proc. of the Tenth IEEE International Symposium on High Performance Distributed Computing, pp. 431-432., August 7-9, 2001, San Francisco, USA.

[23] Ian Foster, Carls Kesselman, Jeffrey M. Nick, Steven Tuecke, 'The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration', June 22, 2002. Document available at <http://www.globus.org/research/papers.html>

[24] SOAP (<http://www.w3.org/2000/xp/Group/>)

© P. Lindner, N. Currie-Linde, M. M. Resch, E. Gabriel 2002