MP-Cluma - A CORBA Based Cluster Management Tool

Silke Schuch, Martin Pöppe
Chair for Operating Systems,
RWTH Aachen University,
Kopernikusstr. 16, 52056 Aachen, Germany
E-mail: {silke, martin}@lfbs.rwth-aachen.de
Tel: +49 241 80 27699 - Fax: +49 241 80 22339

Abstract

There are many underlying hardware architectures for execution of parallel applications. One of these is the Network of Workstations (NOW) which is an interesting solution for users who want to reuse existing hardware for parallel processing. The use of this architecture leads to special demands concerning the generation and start of MPI-applications. The MP-MPICH project of our institute provides the user with libraries for MPI-applications on different platforms with different operating systems and interconnects.

To enable an uniform and comfortable startup of the resulting MPI-applications, a special tool is needed. This tool named MP-Cluma is described in this paper. The heterogeneous hardware and software structure of the NOW leads to special implementation requirements, which will be specified. Furthermore, MP-Cluma illustrates the network assembly of the available nodes by mapping this structure to a hierarchical cluster structure. We explain why we considered CORBA to be the best communication middleware to fit these needs.

Keywords: *Heterogeneous Network of Computers (HNOC), Grid Computing, Cluster Management, Network of Workstations (NOW), CORBA*

1. Introduction

The decentralization of computing resources from the computing centers to the desktop in the past 20 years has led to the fact that most processors have very low average CPU load because they are used for word processing, web browsing etc. Many efforts have been made to use these idle resources for parallel and especially distributed applications. One prominent example is the SETI at home project [15], which allows the owner of a personal computer to donate CPU time for this special project. Other projects in the field of Grid Computing aim at using distributed resources for any application, which usually avoid communication at runtime because the networking performance is unknown and normally low. These applications have a more distributed than parallel characteristic.

In contrast to this, the MP-Cluma project is focused on a different kind of hardware configuration. We found that the use of networks of workstations (NOW) is still interesting nowadays for a class of applications which needs no fast networking interconnect. For users who can not afford a dedicated cluster, this is a way to re-use existing hardware for their parallel computing purposes. Prerequisite in most cases is that they have to accept the heterogeneity of hard- and software.

To support clustering of such heterogeneous conglomerations of PCs and workstations, we started the multi-platform cluster managing project MP-Cluma, which supports a multiplicity of hardware architectures and the most common operating systems.

There already exist many tools to support computational grids, e.g. Globus [1] and Legion [6]. These tools are mainly dependent on special operating systems or special hardware. SGE [16] for example only supports Unix operating systems, Windows XP seems to be no target system for most of these tools. The Legion and Globus toolkits support a wide variety of features, applications and programming models. Many of the potentialities of grid-software are dispensable, if one solely wants to start a parallel application.

We do not want to compete with these grid tools. We focus on the special needs of smaller hardware installations which we call *office grid* combined with small clusters. The main goal is to support MP-MPICH users in general which need a tool to organize their NOW and start MPI-applications on the nodes. In particular we want to support the easy startup of MPI [10] applications with MetaMPICH [7], [8] on Meta-clusters. In addition we wanted to evaluate if CORBA is appropriate to be used as the middleware for a cluster management tool.

International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2004), Las Vegas, USA, June 2004

This article is structured as follows: In section 2, we give a short description of our MP-MPICH project with special focus on the existing tools and MetaMPICH needs. We explain our techniques to create the new tool MP-Cluma in section 3, where the framework and the implementation is presented. In section 4 we explain the usage and the advanced features of MP-Cluma. The future work which has still to be done is described in the last section in addition to a short summary of this paper.

2. MP-MPICH

The MP-MPICH project [2] consists of a variety of MPI-libraries and tools. Our institute provides source code and libraries for Unix/Linux and Windows for different interconnects. NT-MPICH is the Windows part of MP-MPICH, SCI-MPICH uses SCI as the interconnect on Unix/Linux and Windows nodes. MetaMPICH is an extension to support metacomputing environments and inhomogeneous coupled clusters respectively.

MP-MPICH already delivers some remote execution tools which are described in the following subsection. Until now there was no tool to facilitate remote startup for the whole variety of MP-MPICH applications. Especially the start mechanism of applications on metacomputers requires special considerations; therefore we describe MetaMPICH in detail in subsection 2.2.

The demands of MP-MPICH applications lead to the requirements MP-Cluma has to comply. This is illustrated in subsection 2.3.

2.1. Remote Execution Tools

MP-MPICH provides startup tools for homogeneous Windows- and Linux/Unix-clusters, but these tools are not yet suited for process creation in inhomogeneous clusters. For the initiation of MPI-processes on a homogeneous cluster, the command-line tools `mpirun` and `mpiexec` respectively can be used.

The usage of `mpirun` is similar on Unix/Linux and Windows-systems, however the effect of the call is different on the varying platforms. When calling `mpirun` on a Unix node rsh is used to start remote processes, while the remote processes in a Windows environment are started by using a special RPC-service. NT-MPICH offers the Windows-service *rcluma* which uses Windows-RPC and provides several features for remote execution.

As a workaround for remote execution on heterogeneous clusters, it is possible to call `mpirun` for each operating system of a cluster on an appropriate node. Thus, manual analysis of the cluster structure and comparison with the arrangement of operating systems on the nodes is necessary.

It is also possible to use Windows-rsh to start remote processes on Windows and Unix-Nodes with the same tool. But simply using rsh as the underlying protocol does not turn the heterogeneous cluster into a homogeneous one.

Depending on the operating systems the executable files and commandline parameters differ on the nodes of a heterogeneous cluster. Moreover, the remote processes are to be started in the context of a dedicated user. The user profiles on different operating systems differ which means that one user can have different logins, passwords and access rights.

To fulfill these needs it is required to know the type of operating system and underlying hardware on the remote nodes. Furthermore, for comfortable cluster management, the exclusive use of `mpirun` is not adequate, therefore a graphical front-end for management of heterogeneous clusters is needed.

On Windows-Clusters with *rcluma*, the start of remote processes is not only available by using `mpirun` but also by using a graphical front-end called *RexecShell*. With this front-end, it is possible to gain system information about the remote hosts. For each host the type of operating system, number and type of processors and network devices, as well as the name of the user logged in and the processes running can be displayed.

As these useful features are very helpful to choose the nodes for the remote processes, they should also be included in a graphical front-end for heterogeneous clusters. Since the implementation of these features is using unique parts of the Windows-API, it is not possible to port these features to a heterogeneous environment.

The development of a compatible graphical front-end for Unix is possible but not reasonable, because this would mean extra effort when changes or extensions are necessary. The better alternative is to create a new graphical front-end whose source code is the same on the different operating systems.

2.2. MetaMPICH

As clusters of PC mostly contain general purpose hardware, it is not advisable and often not possible to extend an existing cluster with the same kind of hardware after a specific period of time. Therefore, multiple clusters with different fast interconnects often exist side by side. The MetaMPICH project focuses on coupling existing clusters by using conventional network hardware like fast ethernet. One cluster is referred to as a metahost, the coupled clusters are named metacomputer.

To connect different metahosts, MetaMPICH distinguishes between working processes and router processes. The router processes have to be located on the nodes with the adapters for cluster-interconnect. The type and number of working processes depend on the type and speed of nodes and interconnect of the individual clusters.

To support process creation on these coupled clusters,

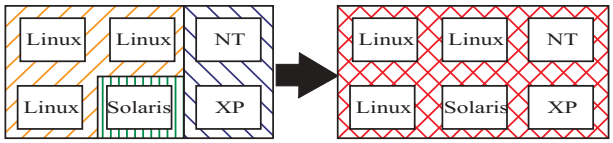

Figure 1. Merging Different Nodes to one Heterogeneous Clusters

detailed information about each node and the overall network structure is required. Detailed information of the nodes has to be provided by each node. The network structure can be represented by a mapping to a hierarchical node structure which assigns a single node to its appropriate cluster.

2.3. MP-Cluma Requirements

The new cluster-management tool should give a user the possibility to start one parallel application on a heterogeneous cluster. The user shall perceive the heterogeneous cluster as one entity as shown in figure 1. As differences between the nodes effectually exist, the tool must give the user the potentiality to distinguish between different types of nodes.

Moreover, this tool must also support the visualization of metacomputers consisting of metahosts.

At first sight, the simultaneous representation as a homogeneous and heterogeneous cluster is contradictory. But with the introduction of a hierarchical representation of clusters and nodes, MP-Cluma fulfills both needs.

Groups of nodes form separate clustrs, but all these clusters can be combined to one meta-computer.

One problem is the presence of different user profiles on different operating systems. To prevent the user from retyping account information for the varying operation systems, a special cluster account is needed. The user has to create an account containing his user information for the different operating systems. This account is used to start the remote processes. This demand leads to the implementation of a management of cluster-users.

Another requirement is the existence of different executables and libraries for different operating systems. The user must therefore have the ability to specify a parallel application as a set of hardware and operating system dependent executables and specifications.

The graphical front-end has to be executable on different operating systems. Furthermore, the user management has to be uncoupled from the graphical front-end. These considerations lead to requirements of the general structure of the MP-Cluma framework as described in section 3.

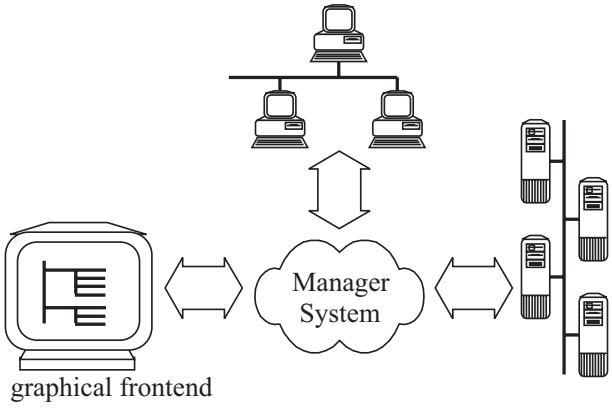

Figure 2. Scheme of MP-Cluma

3. Structure of the MP-Cluma Framework

The implementation requirements lead to the consideration of implementing MP-Cluma as a combination of three different parts. First of all, there has to be one process running on each remote node. This process is responsible for execution of the specified executable. Further on, this process delivers information about the node. Particulars about the type of hardware and operating system is obtained and made available to the user.

Second, there has to be a manager process. This process is responsible for the user management and the coordination of the application startup.

The third part is the graphical front-end. It is used to provide the user with information about the computing nodes. The user is able to specify the parts of the parallel application and the nodes to execute on. Then the user can start the application.

Figure 2 shows that MP-Cluma uses a special managing system as an interface between the graphical front-end and the computing nodes.

3.1. Implementation Issues

To implement a graphical front-end for different operating systems it is self-evident to choose Java as the programming language.

The managing system should also be executable on all supported operating systems. For the managing system does not need to access system information directly, it is possible to implement it in Java.

Another important part to implement are server processes on the different nodes. These processes also have to be implemented for different platforms and operating systems. But, in opposite to the graphical front-end, these processes depend on the different operating systems. It is not possible to start a linux-process in the same manner as a

windows-process. Since Java runs on its virtual machine, there is no Java-interface to the operating system. Because of this limitation, it is not wise to choose Java to implement the server processes on the computing nodes. We selected C++ as the programming language for the server processes.

The next issue was the type of communication model between the three parts of MP-Cluma. Because Java was not chosen for the server processes it was not possible to choose Java-RMI. We selected CORBA [13] as the communication middleware, as explained in the next subsection.

3.2. CORBA as Chosen Middleware

There are different charge-free CORBA-distributions for Java and C++. MICO [9] for instance is a freely available open source CORBA implementation which allows the usage of CORBA for C++. There are already many operating systems supported by MICO, e.g. Linux, Windows, Solaris, Free BSD and so on. In addition to that not only x86 is supported, but also SUN UltraSparc, DEC Alpha and others.

There are more implementations like JacORB [3] which can be used to develop Java programs using CORBA. There are also implementations which support high speed interconnects, among others ATM is supported by TAO [4, 17] and SCI by ROFES [14], [5]. As CORBA is a standard, all implementations supporting compatible versions can be combined.

It encapsulates the underlying platform characteristics and is therefore predestined to be used as a middleware on heterogeneous conglomerates of computing nodes.

In order to support the CORBA-needs, there has to be at least one naming service[12] and one event service[11] available. In the future we will refer to the combination of the CORBA-services and the manager process as the manager system.

The usage of a naming service supports the requirement of a hierarchical representation of the computing nodes. Each object, e.g. the server object on the remote node, is identified by a unique name in the service. Within the naming service one can create so called naming contexts. These naming contexts build a hierarchical naming tree, where the objects can be located on different ranks inside this structure. This is comparable to a hierarchical file system with files.

The structure of the NOW or Cluster can be mapped directly to the naming contexts. The computing nodes are then represented by objects using a path name.

As an example, all nodes of our Pentium 4-Cluster are represented by objects with the denotations DE.Country/LfBS.Organization/P4.Cluster/P4-01.HOST .. P4-08.HOST. To separate the cluster from the workstations, they can be found in the subdirectory DE.Country/-LfBS.Organization/WS.Cluster.

To support extensibility and multi-user needs, it it possible to use more than one manager system by one graphical front-end and also to use many graphical front-ends with one manager system at the same time.

The CORBA event service is necessary to provide MP-Cluma with the possibility to send messages between the different parts of the application. When a Server is started or stopped, an event is created and sent to the managing system. The appropriate host can then be added or removed from the list of available nodes.

3.3. Representation of the Computing Nodes

In order to choose the appropriate nodes for the parallel application the user must be provided with information about the computing nodes. The information is delivered by the server processes, but it is not directly received by the graphical front-end. The manager system retrieves the system information of the computing nodes and buffers it. The information is then on request delivered to the graphical front-end.

Using this technique, it will be possible in coming versions of MP-Cluma to use user dependent filter rules when providing node information. By now, it speeds up the information retrieval.

Although the system information is different according to the type of the node, a unique data format has to be chosen. The best choice for this purpose is the use of XML [18]. Another advantage of the use of XML is the simultaneous use of different versions of server and manager processes. If a newer version of a server provides further information (e.g. processor load), an old manager just ignores the unknown information. If a new manager requests information from an old server, the missing information is simply not displayed.

In figure 3, a sample representation of a windows node is shown. The server process delivers information about the hardware, the operating system, and the currently logged in user.

4. Usage of MP-Cluma

To start a parallel application, the user first has to specify the processes and parameters for the different platforms. The remote nodes are displayed as leaves in the tree representation of the NOW. The leaves can be displayed on each level, the maximum depth is four. To choose the remote computers it is possible to choose single leaves in the tree structure, each representing a remote node. It is also possible to choose a node in the tree representing a set of remote nodes, i.e. a cluster.

The chosen nodes are displayed in a list at the right side of the windows. In order to run more than one process per

```
<Machine_Info>
     <CPU_Name>X86</CPU_Name>
     <CPU_Count>4</CPU_Count>
     <Mem>1023</Mem>
     <Freq>2395</Freq>
     <Host>P4-01</Host>
     <OS>Windows XP (NT 5.1)</OS>
     <S_Pack>Service Pack 1</S_Pack>
     <Build_Nr>2600</Build_Nr>
     <User>System</User>
</Machine_Info>
```

Figure 3. XML-representation of a windows node

node, the tree node or leaf can be added more than once. The number of processes which will be started on a particular node is indicated by the number displayed after the node name. If a set of computing nodes consists of computers with different operating systems, then special buttons at the bottom of the list can be used to exclude nodes with specific operating systems.

After choosing the appropriate nodes one just presses one button to start the application. It is also possible to save and load configurations for an application.

A feature of MP-Cluma is the support of output-redirection. If the parallel application generates output, it is usually printed to a file. This is also supported by MP-Cluma. Alternatively, the output can be displayed in a window in the graphical front-end which is designed in MDI-style (Multiple Document Interface). This feature can be switched on or off for each node separately.

5. Conclusions and Outlook

In this paper, we presented a solution to use networks of workstations as a heterogeneous cluster. The tool MP-Cluma can be used to receive an overview of the structure of a NOW and to start MPI-applications. Thus, MP-Cluma fulfills the main demands we had towards a cluster management tool for NOW.

Additionally, we demonstrated that CORBA can be used as an adequate middleware for small cluster management systems. Although CORBA is based on direct p2p communication between single objects, it can also be used in a one-to-many communication structure.

The present implementation is just the skeletal structure of a powerful management tool for NOWs. There are several additions and features which will be implemented in the future. First of all, a command line based program like mpirun to start remote processes will be added. Furthermore, a mechanism for node reservation has to be implemented. This feature is necessary to run benchmarks unhindered. Another useful property is remote reboot including the selection of the operating system to boot next.

To support enhanced user management, an additional administrator tool is required. A desirable add-on is a scheduling system using automatic resource allocation with respect to hardware and operating system needs.

References

[1] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. 11(2):115–128, 1997.

[2] J. Worringen and K. Scholtyssik. *MP-MPICH, user documentation & technical notes*. Lehrstuhl für Betriebssysteme, RWTH Aachen, 2002.

[3] JacORB. http://www.jacorb.org.

[4] R. Klefstad, D. C. Schmidt, and C. O'Ryan. Towards Highly Configurable Real-Time Object Request Broker. In *5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2002)*, Washington DC, USA, April 2002.

[5] S. Lankes, M. Pfeiffer, and T. Bemmerl. Design and Implementation of a SCI-based Real-Time CORBA. In *4th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2001)*, Magdeburg, Germany, May 2001.

[6] Legion. http://legion.virginia.edu.

[7] M. Pöppe and J. Worringen. *Meta-MPICH, user documentation & technical notes*. Lehrstuhl für Betriebssysteme, RWTH Aachen, 2002.

[8] M. Pöppe and S. Schuch and T. Bemmerl. A Message Passing Interface Library for Inhomogeneous Coupled Clusters. In *Proceedings of ACM/IEEE International Parallel and Distributed Processing Symposium (IPDPS 2003)*, Nice, France, April 2003.

[9] MICO. http://www.mico.org.

[10] MPI Forum. MPI: A message-passing interface standard. *International Journal of Supercomputing Applications*, 1994.

[11] OMG Technical Document formal/01-03-01. *Event Service Specification*, 1.1 edition, 2001.

[12] OMG Technical Document formal/02-09-02. *Naming Service Specification*, 1.2 edition, 2002.

[13] OMG Technical Document formal/98-07-01. *The Common Object Request Broker – Architecture and Specification*, 2.2 edition, 1998.

[14] ROFES. http://www.rofes.de.

[15] SETI@home. http://setiathome.ssl.berkeley.edu/index.html.

[16] Sun Microsystems. Sun Grid Engine (SGE). http://wwws.sun.com/software/gridware.

[17] TAO. http://www.cs.wustl.edu/~schmidt/TAO.html.

[18] W3C Recommendation 6 October 2000. *Extensible Markup Language (XML) 1.0*, second edition, 2000.