GraphProgramming: a tool for grid and parallel programming

Felipe Severino
Universidade de Passo Fundo
Instituto de Ciências Exatas e Geociências
Passo Fundo, RS, Brasil
felipelseverino@gmail.com

Marcelo Rebonatto
Universidade de Passo Fundo
Instituto de Ciências Exatas e Geociências
Passo Fundo, RS, Brasil
rebonatto@upf.br

Abstract

This paper describes a project of a tool to assist the development of parallel and distributed applications with emphasis in grid environments. A graph structure will be used by the tool to represent the processes and the communication among them, and also to generate the application source code. In addition, this work will present the characteristics of grid computing and Globus Toolkit, which is a standard in terms of middleware for grid computing.

1. Introduction

The grid computing allows the user to execute their application in a high performance structure using resources widely distributed. Even that the grid computing be similar to conventional parallel and distributed environments, some characteristics differ the grid from these environments, like geographically distributed resources and multiple administrative domains.

To create this complex environment, a specific kind of software, called middleware, is needed. One of the most used middlewares for grid computing is the Globus Toolkit, a set of components that works together to deal with issues like heterogeneity, security and execution of the applications. To develop applications for this environment, is common to use a solution used by conventional parallel and distributed environment, like MPI or Web Services, with few alterations.

Tools for visual programming, that represent graphically the concurrence of the processes, become a solution for the development of complex applications. Those tools help the programmer to develop the application without regard about the technology used. That way, a tool that allows the user to define graphically the execution flow of the applications to be executed in the grid environments can help the development of applications. In addition, the tool can generate the communication code (among the processes) of the application, abstracting from user the technology used.

This paper describes a project of a tool for help the development of applications for grids using the MPI standard. The tool should use visual resources, like graphs, to represent the communication and processes of the application, and generate the source code of the developed applications. The section 2 describes the main characteristics of grid computing and Globus Toolkit. Section 3 brings some requirements for grid programming. In the section 4 the GraphProgramming tool project is presented. The section 5 conclude this paper.

2. Grid Computing

Grid computing is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [5].

Even that grid computing looks similar to conventional parallel and distributed environments, some characteristics differ from that environments [14], like scale and selection of resources, heterogeneity, dynamic and unpredictable structure and multiple administrative domains [4]. A grid environment can have thousands of computational resources, widely distributed, each one with distinct access politics, authorization and authentication mechanisms or availability. The application that executes in the grid should select the resources available that matches with the application needs (the availability of resources changes frequently, common causes for the lack of resources may include the network and problems with the resource).

One way to build a grid infrastructure is use a specific kind of software, called middlewares. The middleware is used to change information between different programs, hiding from the programmer the differences of communication protocols and operational system dependences [2]. One option of middleware widely used is the Globus Toolkit (GT), developed by Globus Alliance, considered a standard in terms of middleware for grid computing [7].

Maintaining the same configuration of older versions of the toolkit, the Globus Toolkit 4 (GT4) is compounded by a group of components that implement the basic services of a grid environment, like security, resource management and communication [6]. Each component of the toolkit can be used separately in a specific application, or all components can be used together to build a complete grid structure. Examples of services are the Globus Resource Allocation Manager (GRAM) Globus Metacomputing Directory Service (MDS) and Globus Security Infrastructure (GSI).

The GT4 has, basically, two models for applications communications: Web Services (WS) and Message Passing Interface (MPI). The tool described in this paper will focus in the implementation of the MPI standard, letting to the future works the implementation of others communication models.

To programming for GT, the most common implementation of the MPI standard is MPICH-G2 library, a grid-enable implementation of the library MPICH. The MPICH-G2 library uses services from GT, like job startup, security, and so on, to execute the applications in the grid [10]. To use the MPICH-G2 a device (globus2) should be choose in the installation of the MPICH library, and the applications will run over this device. More information about Globus Toolkit installation and MPICH-G2 can be found in [7] and [13], respectively.

3. Grid Programming

The grid computing is a recent technology and, as every new computational structure, takes a while to develop a specific programming model to this environment. Currently, the development of applications for grids uses the same solutions used in parallel and distributed environments, because of similarity of the environments.

Considering the technologies used in the environment, the requirements of a tool for grid programming should follow the same requirements of a tool for parallel and distributed programming. Some of the requirements are portability, essential in a heterogeneous environment as the grid, and programming abstraction, that allow the programmer to read their application in a high level abstraction, focusing in the solution of the problem [12]. Other important requirement for a development tool is the representation of the concurrence among the processes, which helps in both development and depuration of the application. This concurrence can be represented by a graphic interface, using a graph structure, which is considered more appropriated for this programming paradigm [12].

In terms of tools for application development, some related works can be provided to establish a comparison with the tool described in this paper. The first tool is the P-GRADE [15], which have very similar proposes, but isnt free and cant be expanded with collaboration of other developers. Other tool is called DOBuilder [3], which propose to help the development of applications that uses distributed objects. Unfortunately, as the development of the tool described in this paper is not concluded, a complete comparison of the tools cannot be done yet, that should be done after the conclusion of the development of the tool.

4. The GraphProgramming Tool

In this section the project of a tool to help the development of applications for computing grids is presented.

4.1. General Characteristics

To help the development of applications for complex environments, such as grid, some visual resources may help the user to comprehend the structure of the program [11]. In the proposed tool, the structure of the application will be represented as a graph, which will have process elements represented as nodes, and the communication elements as arcs.

The main goal of the tool will be help the development of applications for grid computing, using C/C++ language with MPI library. To achieve this goal, the tool will use some resources to assist the user in some steps of the development, which are: define the execution flow; choose the type of communication; insert the processing code in each node of the application. Those steps dont have to occur in order. The user can insert some nodes and choose the communication types and data transferred in those nodes, then insert other nodes, and so on. When the entire flow were defined (the tool will have marks for begin and end of the flow), the tool will be able to generate de source code of the application, based on the informations inserted previously.

For the first version of the program, the MPI standard was chosen as communication model. The reason for that are many, which include the high number of users around the world, the documentation available and the similarity with the conventional parallel and distributed programming. Thus, routines will be available for point-to-point communication (MPI Send, MPI Recv, so on) and collective communication (MPI_Bcast, MPI_Reduce, so on). The types of data transferred by those communications will be scalar and vector.

Another functionality of the tool will be the support to generate applications MPI using the library C-XSC. C-XSC is a library for development of numerical algorithms with high accuracy and automatic verified results [9]. The use of C-XSC with MPI is a recent union, and its not knew a visual tool with similar functionality [8].

4.2. Processing Elements

The processing elements will be represented like nodes in the graph that represent the developed application structure's. Each node of the structure should contain a complete set of informations needed for the code generation. Some of these informations are the source code of the process represented by the node and the data (with their respectively types) to be communicated by the process.

There is three types of nodes that can be used to represent groups of processes: the P nodes, that represent a single process; the N nodes, that represent all nodes of the application; and the N-1 nodes, that represent all the nodes except the root (usually the process 0). This nodes described above should be used with the communication elements to create the structure of the developed application.

4.3. Communication Elements

Graphically represented by the arcs in the graph, the communication elements will get the information in the nodes to create the communication almong the processes. The MPI standard define dozens of functions, but only the following functions will be supported by the tool.

4.3.1. MPI_Init,MPI_Comm_rank, MPI_Comm_size and MPI_Finalize These functions are, usually, called only once and with predefined arguments. Thus, the tool should call these functions automatically, defining the application's limits, and these limits should be represented graphically.

4.3.2. MPI_Send and MPI_Recv These functions compound the point-to-point group of functions supported by the tool. The uses of these functions can be mapped with the nodes described in the section 4.2, producing the results showed in the Figure 1.

In the Figure 1, the communication with a symbol (represented as a vector) at his side means that the communication send (or receive) more than one data in the message. As can be noted in the figure, the possibilities of use of the send and receive can replace the use of other functions supported by the tool. The user should choose the comunication that fits better with the developed application.

4.3.3. MPI_Bcast, MPI_Reduce, MPI_Scatter and MPI_Gather These collective communication functions are supported by the tool. As the point-to-point group, the collective communications can be mapped with the nodes described above. The result of this mapping is show in the Figure 2.

4.3.4. MPI_Barrier The barrier function cause to all processes to wait to the last one reach the call of the function, synchronizing all the processes. Their representation should differ from the others, simbolizing the call of the function for all the processes.

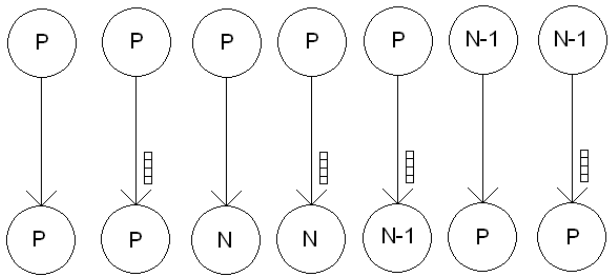

Figure 1. Mapping of the send and receive functions

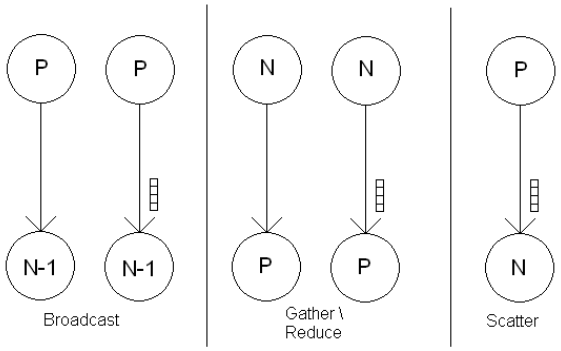

Figure 2. Mapping of the collective communication functions

4.4. Code insertion

To help the programmer in the development of the application, the tool must have two different interfaces: graphic and textual. As the graphic interface, the textual interface must use some resources to help the development of the application. Some of those resources are indentation and highlights [1].

5. Conclusions

Both programming and utilization of the grid are, in most cases, done thought a textual interface, as the shell in Linux

operating systems. This interface can inhibit the use of the system for inexperienced or new users, as occurs in users of other areas of research, like basic science. This is one reason for the tool use resources to help the development, like visual resources, that was described in this paper. Others possible users for this tool can be the students of parallel and distributed programming that, can use the graphic representation to better understand the paradigm.

The next step of this work include the implementation of the GraphProgramming tool. The language for the implementation will be Java, because some characteristics of the language are important for the tool, as portability (allowing execution in different systems, architectures, etc.) and flexibility (allowing the reuse of the code used in the LAM/MPI Designer tool [1], that implements some resources for textual programming). The tool should have opensource, enabling the development of improvements for the tool.

References

[1] M. P. Bastos. Uma ferramenta para facilitar a escrita de aplicações paralelas com mpi. Trabalho de conclusão de curso (Graduação em Ciência da Computação). Universidade de Passo Fundo, 2006.

[2] G. F. Coulouris, J. Dollimore, and T. Kindberg. *Distributed systems: concepts and design*. Addison-Wesley, Harlow, 4 edition, 2005.

[3] DOBuilder. Dobuilder. http://www.inf.ufrgs.br/procpar/hetnos/DOBuilder, august 2008.

[4] I. Foster and C. Kesselman. Globus: a metacomputing infrastructure toolkit. *International Journal of High Performance Computing Applications*, 11(2):115–118, 1997.

[5] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*, chapter 2, pages 15–51. Morgan Kesselman Publishers, Inc., San Francisco, 1999.

[6] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*, chapter 11, pages 259–309. Morgan Kesselman Publishers, Inc., San Francisco, 1999.

[7] Globus. About the globus toolkit. http://www.globus.org, may 2008.

[8] C. A. Holbig, M. T. Rebonatto, and M. J. Brusso. Resolução numérica de aplicações com alta exatidão em ambientes de alto desempenho. In *Escola Regional de Alto Desempenho*, pages 5–41, 2008.

[9] Karlsruhe. C-xsc. http://www.rz.uni-karlsruhe.de/ĩam/html/language/cxsc/cxsc.html, july 2008.

[10] C. Lee and D. Talia. *Grid Computing: make the global infrastructure a reality*, chapter 21, pages 555–578. Wiley, West Sussex, 2003.

[11] J. Malacarne. Ambiente visual de programação distribuída em java. Dissertação (Mestrado em Ciência da Computação). Universidade Federal do Rio Grande do Sul., 1999.

[12] J. Malacarne. Ambientes de programação visual paralela e distribuída. Trabalho Individual (Curso de pós-graduação em Ciência da Computação). Universidade Federal do Rio Grande do Sul., 1999.

[13] MPICH-G2. Mpich-g2. http://www.hpclab.niu.edu/mpi, march 2008.

[14] Z. Németh and V. Sunderam. A formal framework for defining grid systems. In *IEEE/ACM International Symposium on Cluster Computing and the grid (CCGrid'02)*, pages 202–211, 2002.

[15] P-GRADE. P-grade. http://www.lpds.sztaki.hu/pgrade/, august 2008.