A Non-intrusive and Incremental Approach to Enabling Direct Communications in RPC-Based Grid Programming Systems

Alexey Lastovetsky, Xin Zuo, and Peng Zhao

School of Computer Science and Informatics
University College Dublin
Belfield, Dublin 4, Ireland
{Alexey.Lastovetsky, Xin.Zuo, Peng.Zhao}@ucd.ie

Abstract. This paper advocates a non-intrusive and incremental approach to enabling existing Grid programming systems with new features. In particular, it presents a software component enabling NetSolve applications with direct communications between remote tasks. The software component is a supplementary one working on the top of the basic NetSolve system. Its design also allows remote tasks to be freely mixed in a single application, independent on whether each particular task is enabled for direct communications or not. Experiments with this software are also presented.

1 Introduction

High performance Grid programming systems have reached a certain level of maturity. Two examples are NetSolve [1-3] and Ninf [4] that allow scientific programmers to develop reliable Grid applications. On the other hand, the constantly growing number of users and applications results in the need of further development of such systems in terms of functionality and quality.

Traditionally, addition of a new feature to a Grid programming system is achieved by changing the code of the system to produce its new version. This approach to the evolution of Grid programming systems has two serious disadvantages. First of all, the change of the system's code may introduce bugs and result in some applications not running properly anymore or even crashing. Secondly, the new version of the system has to replace the old version on all computers of the Grid in order to support the development and execution of applications enabled with the new feature. Such simultaneous and total replacement can have very high organizational overhead and sometimes be simply unrealistic as different computers on the Grid are managed and administered by independent and, very often, loosely connected users.

The goal of our research is to investigate if an existing Grid programming system can be enabled with new features in a non-intrusive and incremental way. *Non-intrusiveness* means that the original system does not change and the new features are provided by a supplementary software component working on the top of the system. *Increment* means that the supplementary software component does not have to be

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 1008 – 1011, 2006.

installed on all computers to enable applications with the new features. It can be done step by step and the new features will be enabled in part, with the completeness dependent on how many nodes participating in the execution of the application have been upgraded with the supplementary software component. In this paper, we use NetSolve and one particular feature that is direct communication between remote tasks, to demonstrate the feasibility of the non-intrusive and incremental evolution of Grid programming systems.

The rest of the paper is structured as follows. Section 2 describes in detail the design and implementation of a supplementary software component enabling NetSolve applications with direct communications between remote tasks in a non-intrusive and incremental way. Section 3 presents some experiments with this software. Section 4 outlines related work and concludes the paper.

2 Enabling Direct Communications in NetSolve

NetSolve is positioned as a programming system for high performance distributed computing on global networks based on GridRPC [5]. In NetSolve, output data of remote tasks are typically sent back to the client upon completion of each remote task even if the data are only needed as input for some other remote tasks, resulting in so-called bridge communications which increase the execution time of applications. In this paper, we propose a lightweight supplementary software component that enables direct communication between remote tasks in NetSolve in a non-intrusive and incremental way, without recompilation or reinstallation of the original NetSolve programming system. We start presenting the software component by a short description of its use. The only thing for **client programmers** to do is to install the wrapper API and Job Name Service on the client side, then compile the client program with the wrapper library. The **procedure developers** should do nothing to enable direct communications and develop their own procedures as usual. To enable direct communications on server side, **the server administrator** needs to register the software component as a new problem file to NetSolve. No re-installation and re-compilation for the system. The proposed software component consists of three parts: Client API & Argument Parser, Server Connector and Job Name Service (JNS).

Client API provides a uniform interface for the client to make remote procedure calls. Despite the modification on the remote side, the wrapper API allows the calls to be made in the same manner. The only difference is in the arguments that can be not only variables storing real data but also handlers, which is a variable storing real data, the local IP address and the port number are used as such communication info. If input argument is a handler, then a request is sent to the JNS to get the IP address and the port number of the remote resource and this information is used as communication info for this handler. If this output argument is a handler, the returned result information from computational servers is sent to JNS and registered there. In this sense, upon making a call to NetSolve, only a handler array that is transferred to the remote server. The Server Connector manages all the other I/O data transaction.

Server Connector is on the server side, which is a proxy program responsible for interacting with clients and other Server Connectors to enable direct communications. When all necessary data have been acquired by Server Connector, it re-submit to the

local host to perform computations that the user exactly requested for. There is no difference in the way the client and computational servers download the result of the computations. The Server Connector firstly returns the result's communication information to the client. Then it sets up a socket waiting for the client or the server to connect in to download the result of computations.

Job Name Service (JNS) is responsible for registration of procedure upon its invocation during RPC call. Other procedures may send requests to the JNS to search for registered procedure. During the execution of the application, it contains all information about every handler. Only client has the permission to register or access a handler on the JNS. There is no communication and interaction between JNS and computational servers. Because JNS is designed as a system-independent system on the client side, it can be applied to different RPC-based systems and not influenced by any fault or crash on the server side.

3 Implementation and Experiments

For our experiments we choose the same remote computational task that has been used in experiments with REDGRID presented in [6], namely, matrix multiplication. Experiments in [6] used 2 remote servers to perform 3 matrix multiplications, and the client, agent and servers all were in the same Ethernet segment. In our experiments, we used 8 remote servers to perform 8 matrix multiplications. The interconnecting network is based on 100 Mbit Ethernet with a switch enabling parallel communications between computers.

Based on our experiments results we can make conclusion that communication cost is visibly reduced by using direct communications, where seven communication bridges were eliminated among twenty four communications. So, the theoretical speedup is 7/24 = 29.2%. The obtained experimental speedup ranges from 24% to 27%, which is close to the theoretical value. We can also see that the experimental results are similar to the REDGRID ones, which are ranging from 18% to 28%. The speedup depends on the ratio of the number of eliminated bridge communications and the total number of communications. If communication links connecting remote computers are much faster than communication links connecting the remote computers and the client computer, the speedup will be much higher. Another experimental results show that speedup is around 54% while bridge communications is performed at the rate of 10 Mbit/sec and the direct communications between remote servers is performed at the rate of 100 Mbit/sec.

4 Related Works and Conclusion

To enable direct communications, NetSolve introduces an original mechanism called Request Sequencing [7]. The most restrictive of which is that all the tasks have to be performed on the same computing node. Another effort to reduce the overhead of bridge communications in NetSolve is the Logistical Computing and Internetworking (LoCI) [8]. The mechanism is mainly aimed at replicating data in order to keep them even in the case of crash of some of the computers. Although it is sufficient for

enabling direct communications, the goal of building a complete network storage system makes LoCI over-heavy for enabling just this particular feature. The REDGRID project [6] is closest to our approach sharing the similar idea behind its design, which uses an intrusive and non-incremental approach and requires re-compilation and re-installation of the modified NetSolve on all involved computing nodes. The main difference is that REDGRID is built into NetSolve and difficult to be migrated to other GridRPC-based systems. Another related project is SmartNetSolve [9], an extension of NetSolve aimed at higher performance of Grid applications, which also enables direct communications in an intrusive and non-incremental way.

In this paper, we have presented an approach to reducing unnecessary bridge communications in RPC-based Grid programming systems. The main advantage of the approach is that it is non-intrusive, requiring no changes in the enabled programming system. It does *NOT* need recompilation or reinstallation of the Grid programming system. The approach is incremental by nature allowing remote tasks both enabled for direct communication and not, to be freely mixed in a single application. It can be applied to different RPC-based Grid programming systems. Finally the experimental results have shown that the performance of Grid applications can be significantly improved by using our supplementary software component.

This work was supported by the Science Foundation Ireland.

References

1. http://icl.cs.utkedu/netsolve/
2. Casanova H., Dongarra J.: NetSolve: A Network Server for Solving Computational Science Problems. The International Journal of Supercomputer Applications and High Performance Computing, Vol. 11, No. 3, pp. 212--223, 1997
3. Arnold D., Casanova H., Dongarra J.: Innovation of the NetSolve Grid Computing System. Concurrency: Practice and Experience, Vol. 14, No. 13-15, pp. 1457-1479, 2002
4. Tanaka, Y., Nakada, H., Sekiguchi, S., Suzumura, T., Matsuoka, S.: Ninf-G: A reference implementation of RPC-based programming middleware for Grid computing. Journal of Grid Computing, Vol.1, No.1, pp. 41--51, 2003
5. Seymour, K., Nakada, H., Matsuoka, S., Dongarra, J., Lee, C., Casanova, H.: Overview of GridRPC: A Remote Procedure Call API for Grid Computing. In: Proceedings of the Third International Workshop on Grid Computing, pp. 274--278, Springer-Verlag, 2002
6. Desprez, F., Jeannot, E.: Improving the gridrpc model with data persistence and redistribution. In: Proceedings of ISPDC 2004 / HeteroPar'04, pp. 193--200, IEEE Computer Society, 2004
7. Arnold, D., Agrawal, S., Blackford, S., Dongarra, J., Miller, M., Seymour, K., Sagi, K., Shi, Z., Vadhiyar, S.: Users' Guide to NetSolve V1.4.1. Technical Report ICL-UT-02-05, University of Tennessee, Knoxville, TN, 2002
8. Beck, M., Arnold, D., Bassi, A., Berman, F., Casanova, H., Dongarra, J., Moore, T., Obertelli, G., Plank, J., Swany, M., Vadhiyar, S., Wolski, R.: Middleware for the use of storage in communication. Parallel Computing, Vol. 28, No. 12, 2002
9. Brady T., Konstantinov E., .Lastovetsky A.: SmartNetSolve: High Level Programming System for High Performance Grid Computing. In: Proceedings of IPDPS 2006, IEEE Computer Society, 2006