CONCURRENCY—PRACTICE AND EXPERIENCE
Concurrency: Pract. Exper. 2000; **00**:1–7 *Prepared using* cpeauth.cls *[Version: 2001/03/05 v2.01]*

An Integrated Software Development Environment for Grid-Computing

M. Müller*, E. Gabriel and M. Resch

HLRS - High Performance Computing Center Stuttgart, Allmandring 30, 70550 Stuttgart, Germany

SUMMARY

Grid-Computing has become a popular concept in the last years. While in the beginning the driving force was metacomputing, the focus has now shifted towards resource management issues and concepts like ubiquitous computing. For the High Performance Computing Center Stuttgart (HLRS) the key challenges of Grid-Computing were coming from its users and customers demands. With high speed networks in place, programmers expect to be able to exploit the overall performance of several instruments and high speed systems for their applications. In order to meet these demands, HLRS has set out a research effort to provide these users with the necessary tools to develop and run their codes on clusters of supercomputers.

KEY WORDS: MPI; Grid; Software Development

Overview

During the last 5 years the High Performance Computing Center Stuttgart (HLRS) has set up a framework of collaborations and research projects in the field of Grid-Computing. All these efforts are circled around the central goal of enabling high performance simulations on distributed supercomputers. With international high speed networks in place, HLRS very early on tried to make use of such distributed resources and make them available to its users in industry and research. In 1996, HLRS together with the Pittsburgh Supercomputing Center (PSC) was the first to connect two supercomputers in the US and Europe making use of transatlantic network connections and the US access point of STAR-TAP. In 1997, the first real simulations were done on this same configuration. Building on co-operations in Asia, the US and Europe, HLRS in 1999 was honored for its efforts with the NSF Award for distributed high performance computing during Supercomputing 1999.

*Correspondence to: mueller@hlrs.de

 Revised

With the focus changing from resource management to programming, the main issues for a developer of simulation codes are the following:

Programming Model The obvious choice for Grid-computing is the message passing paradigm. Shared memory concepts may be implemented and may ease the programming. However, given the still low performance of wide area networks compared to internal networks of supercomputers, it seems sensible to let the programmer explicitly define and manage the exchange of data.

Since MPI [20] has become the standard in message passing programming, almost all users at HLRS have given up on PVM or proprietary models. What is therefore required for Grid-Computing is a grid-aware implementation of MPI.

Debugging With so many processors being used and executables being distributed across several systems, debugging becomes a major issue for a code developer. It is assumed that codes are tested on several parallel systems. In this way most bugs should be found in a homogeneous and more easy to handle environment. However, experience shows that the more complex environment of Grid-computing frequently reveals bugs that only occur when communication constraints are tougher and heterogeneity imposes its penalties. Debugging of codes on Grid-computers is therefore an issue, even if it can not go to the full extent of a standard parallel debugger.

Performance Analysis Technical simulation on Grid-computers is appealing to users because of its potential to solve bigger problems by accumulating performance. The potential reasons for performance problems are the heterogeneity of the Grid-computer and the bad performance characteristics of the connecting networks. In order to be able to improve the performance of a code these problems have to be made visible. The user should be able to examine the details of the performance of a code to find out where the problems are.

The goal of HLRS is to find tools that help to solve these problems, to implement parts that are missing and to integrate them with visualization environments to allow interactive work for engineers and scientists. Several projects have been setup to solve the problems mentioned, especially the European projects METODIS [8], DAMIEN [1] and the german UNICORE [23] project are developing key-components for realizing the Grid-components needed for an HPC-Center.

In the following we will explain the concept of the overall architecture giving an overview of the role of each building block in the following chapter. Relevant implementation issues for components and for their integration are described afterwards. Finally a summary and discussion of future requirements and steps are given.

Architecture

The framework of the environment was developed in the Esprit project METODIS, and is currently further developed in the IST-project DAMIEN. Thererfore, we would like to present first the structure of DAMIEN, and show afterwards how in other projects different components are currently developed to complete the Grid-environment.

The overall objective of DAMIEN is to create a toolbox, which enables the development and adaptation of industrial applications to computational grids, to ease and to enable the access

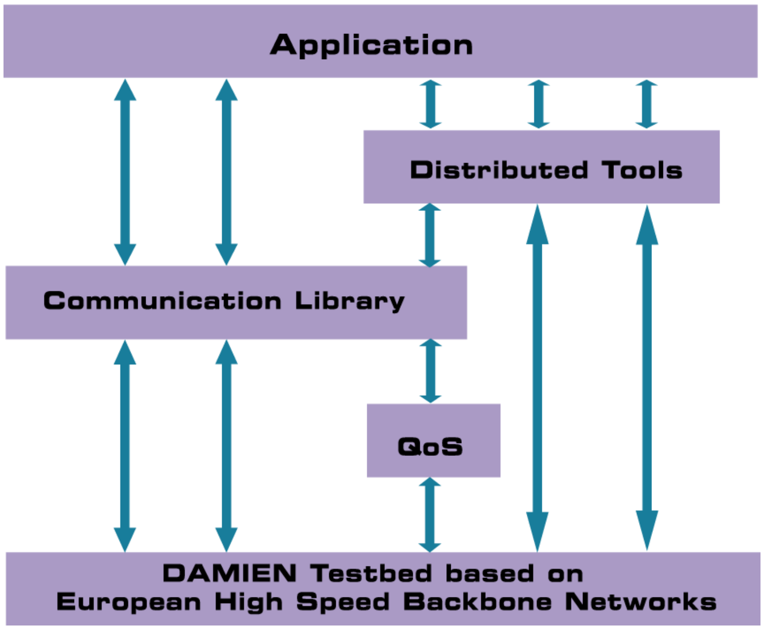

Figure 1. Architecture of DAMIEN

and the usage of distributed resources. Central part of the architecture are therefore standards used by industry, mainly the Message-Passing Standard MPI [20]. Additionally, the developers of MPI-applications are used to tools, which ease the development and analysis of MPI applications on High-Performance Computing Systems. When moving from Supercomputers to Computational Grids, the end-user faces the problem, that these tools are currently not available.

The overall architecture of DAMIEN as presented in figure 1 is based on the requirements of end-user applications. To make the application run in a distributed environment, several tools and libraries are required:

- Communication library: to enable the usage of several computing resources for a single simulation, the MPI-library has to support the coupling of distributed resources in an efficient manner.
- QoS Module: the performance of distributed application is dominated by the network used to link the computing resources. Therefore a Quality-of-Service Manager is required to dynamically handle the bandwidth requirements of the application.
- Performance Analysis: applications in grid-environments often face performance problems, like on any other system. To enable and improve an in-depth analysis of

the application, performance analysis tools have to be aware of the aspects of clustered supercomputers.

- Performance Prediction: a different aspect of performance analysis is the prediction, how changes in the source-code will affect the performance of the application within a given environment. Based on the known performance on a homogeneous parallel platform it is also possible to predict the performance of an application on the Grid.
- Code-Coupling-Interface: one of the most promising approaches for applications in grid-environments are coupled application. Based on already existing codes, which are well established, the user couples several codes to solve multi-disciplinary phenomena. Giving each single part of the code the possibility to run on the platform, on which it is best adapted to, is a natural extension.

The toolbox consisting of these components already creates an environment, which gives application-developers access to the tools, which he is familiar with. Nevertheless, one component is still missing, which is a debugger for heterogenous, distributed environments. Despite the popularity of MPI and Grid programming there is still a lack of tools that support debugging on the Grid [22] or detect the complex problems of a parallel programs in semi-automatich way [25].

Implementation

Integration

The integration of the tools presented in the previous section to a single, user-friendly toolbox is one of the key issues in DAMIEN and other projects. Since all tools except the QoS-Module used in the project are based on the Message-Passing Standard MPI, the interfaces between the tools are based mainly on this specification. These are:

- *PACX-MPI* [7] is an implementation of MPI optimized for Grid-environments. It provides to the application a seamless access to distributed environments.
- *Vampir* and *Vampirtrace* [16] are widely accepted tools for performance analysis of MPI applications. The interface between MPI and the library generating the tracefiles, which are used later for the analysis of the application is based on the Profiling Interface of MPI. This is a powerful mechanism defined in MPI-1 to give the user the possibility to replace MPI-calls by calls with a different functionality, and providing a second, name-shifted version of all MPI-routines.
- *Dimemas* [10] is a performance analysis and prediction tool, which gives the user an impression, how changes in the source code or parameter variations will affect the performance of the application in the Grid-environment.
- *MpCCI* [12] is a Code-Coupling Interface used for the coupling of applications on a numerical level. The library implements data-exchange on a higher abstraction level by providing all required functionality (e.g. interpolation between different meshes). The basic communication is again based on MPI.

- *MARMOT* is a MPI debug and verification tool. It tries to increase the portability of programms by testing the standard conformance of MPI usage. In this way it detects possible problems before the program is ported to the Grid. It also debugs the running application by detecting deadlocks and race conditions.

In the following, two of the tools, which are developed at the HLRS, will be described in more detail. These are the Communication Library PACX-MPI and the MPI-Debugger MARMOT.

Communication Library: PACX-MPI

During the last couple of years, several projects have been working on the problem of making *one* MPI application run on a cluster of (heterogeneous) supercomputers [3, 6, 8, 14, 18, 19]. This problem arises, because the implementations of the MPI libraries of different supercomputer vendors are not interoperable. IMPI [15] is a protocol which aims to enable interoperability between different MPI implementations, but currently it faces two problems: First, there is currently no vendor-MPI library implementing this protocol, which the authors of this paper are aware of. Second, the current version of IMPI does not cover the whole MPI-1 standard, but just a subset. Thus, MPI applications using routines, which are not specified in IMPI, would require additional work to make them run with IMPI-compliant libraries.

PACX-MPI is an implementation of the message-passing standard MPI which aims to support the coupling of high performance computing systems distributed in a grid. The characteristics of such clustered systems show two different levels of quality in the communication. For communication between MPI processes on the same host, typical communication latencies are in the range of microseconds and bandwidth in the range of several hundred Megabytes/second, whereas one has to deal with high communication latencies (in the range of tens of milliseconds) and small bandwidth (ranging from a few Kilobytes/second to a few Megabytes/second) for communication between MPI processes on different hosts. Communication between processes on the same host will be refered to as *internal communication*, while communication between processes on different hosts will be called *external communication* throughout the rest of this paper.

Taking the characteristics of clustered system into account, PACX-MPI relies on three main concepts:

1. Two level hierarchy: in clustered systems, a message-passing library has to deal with two different levels of quality of communication. Therefore the library uses two independent layers, one to handle internal operations, and one to handle external ones.
2. Usage of the optimized vendor-MPI library: Internal operations are handled using the vendor-MPI environment on each system. This allows to fully exploit the capacity of the underlying communication hardware in a portable manner.
3. Usage of communication daemons: on each system of the metacomputer two daemons take care of communication between systems. This allows to bundle communication and to avoid to have thousands of open connections between processes. In addition, it allows to handle security issues centralized. The daemon nodes are implemented as additional, local MPI processes. Therefore, no additional TCP-communication between

the application nodes and the daemons is necessary, which would needlessly increase the communication latency.

The current functionality of PACX-MPI includes the full MPI-1 standard as well as several parts of the MPI-2 document. Additionally, several features make PACX-MPI well suited for heterogeneous, clustered systems, including:

- Optimized collective operations for hierarchically clustered systems [8], which minimize the necessary amount of communication between the different hosts
- Optimized handling of derived datatypes [9]
- Optional data compression for the communication between different machines to reduce the size of the transferred data and to increase the effective bandwidth.
- Optional secure communication based on the SSL protocol, in case the user wants to have a secure communication setup and to encrypt messages.

All these features have allowed PACX-MPI to be used successfully in a number of metacomputing projects linking resources in Japan, the US, and Europe [4, 17, 21].

MPI Debugging and Verification Tool: MARMOT

Due to the complexity of parallel programming there is a clear need for debugging of MPI programs. Running a parallel program with a Grid-enabled MPI library significantly increases the possible problems. According to our experiences made in different testbeds, there are several reasons for this: first, the MPI standard allows many implementation defined behavior, e.g. whether or not a communication is blocking. Because most Grid-enabled MPI libraries make use of the native MPI library for local message delivery, the application has to run smoothly on several, different implementations at the same time. Second, the heterogeneous network with its high latencies and low bandwidth for external messages will not only change the performance but also alter the overall way of execution. E.g. the amount of unexpected messages and necessary buffering might increase significantly. Last but not least the number of processors are typically larger than on any single computer where the application has been developed or tested.

Debugging MPI programs has been addressed in two different ways: Classical Debuggers have been extended to address MPI programs. This is done by attaching the debugger to all processes of the MPI program. This concept can also be extended for distributed MPI programs running on the Grid [13, 22]. approach is to provide a debug version of the MPI library (e.g. mpich). This version is not only used to catch internal errors in the MPI library, but it also detects some incorrect usage of MPI by the user, e.g. a type mismatch of sending and receiving messages [11]. Neither of these approaches address portability or reproducibility, two of the major problems when using MPI on the Grid. The idea of MARMOT is to verify the standard conformance of an MPI program and help to debug the program in case of problems. The design goals are:

- Portability: by verifying that the program adheres to the standard it enables the program to run on any platform on the Grid in a smooth and seamless way.

- Scalability: the use of automatic techniques that do not need user intervention allows to debug programs running on hundreds or thousands of processors.
- Reproducibility: the tool contains mechanism to detect possible race conditions. It will also automatically detect deadlocks and notify the user where and why it has occurred.

MARMOT uses the MPI profiling interface to intercept the messages and analyze them. Like PACX-MPI it adds an additional MPI process for all tasks that cannot be handled within the context of a single MPI process, like dead-lock detection. Information between the MPI processes and this additional debug process are transferred using MPI. Another possible approach is to use a thread instead of an MPI process and use shared memory communication instead of MPI [25]. The advantage of the approach taken here is that the MPI library does not need to be thread safe. Without the limitation to shared memory systems the tool can also be used on a wider range of platforms. Running on top of PACX-MPI it can be used to debug an application running distributed on the Grid.

Integration with Grid-Services

The framework as described here focuses on the development of applications for distributed computing in Grid-environments. In order to ease the handling of such applications, the framework presented here has to be integrated into software environments. It is then the role of these environments to take care of resource management and security.

The following issues have to be addressed

- Security: Users have to be able to submit jobs to systems on which they are allowed to work. The framework has to be compatible with a wide range of security policies.
- Resource Management: For distributed runs, suitable co-allocation or advanced reservation mechanisms have to be implemented.
- Program startup: Users have to be able to start their jobs without having to log into all computers they would like to use. The framework has to support inter-operability with a wide range of existing mechanisms in a seamless way for the user.
- Data Management: It has to be ensured that data are where they are required.

In order to provide these feature, HLRS has started to work with developers of software environments. During a joint co-operation with Argonne National Laboratories, PACX-MPI has been recently extended to support the GLOBUS [5] startup-mechanisms. Thus, the user is using the security features integrated in the GLOBUS-environment in a transparent manner. PACX-MPI is currently also integrated in the German Grid-environment UNICORE [2, 23]. Again, the purpose is to make use of process startup features for distributed applications. In the future PACX-MPI will be integrated with the Task Mapping Editor developed by the Japan Atomic Energy Research Institute (JAERI) [24].

In the field of data management for an application HLRS has started to work on the problem of MPI-IO in distributed environments. Since File-I/O is a key-issue regarding the portability and overall-performance of distributed applications, it is of major importance for an HPC-Center providing a Grid-environment to offer its users a solution for data management. Thus, the implementation of the MPI-I/O interface is currently proceeding in a co-operation with the University of Tennessee.

Project Status and Future Plans

The achievements of this framework made it possible to run a large number of research and industrial applications on the Grid. They cover a wide range of different areas like particle simulations, computational fluid dynamics, electronic structure simulation and fluid-structure interactions. To faciliate the every-day use the ongoing integration into Globus, TME and UNCIORE is one important point. In close co-operation with our end-users we will not only improve the tools presented here, but will also develop new solutions for their problems.

ACKNOWLEDGEMENTS

This work was supported by:

- Gigabit Testbed South Project / DFN TK602-NT107
- METODIS, EU-Project Esprit 29909
- DAMIEN, EU-Project IST-2000-25406
- UNICORE Plus 01-IR-001
- DAAD Project Nr. 9922935

The authors would like to thank HLRS, PSC, SNL, TACC, MCC, NIC, MPG, DFN, German Telekom, STAR-TAP, Abilene, vBNS for support in doing experiments.

REFERENCES

1. WWW, July 2001. http://www.hlrs.de/organization/pds/projects/damien/.
2. D. Erwin, (Ed.) *UNICORE - Gemeinsamer Abschlußbericht des BMBF-Verbundprojekts* UNICORE Forum e.V. 2000, http://www.unicore.org
3. G. E. Fagg, K. S. London, and J. 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.
4. *Metacomputing im Gigabit-Testbed Süd und Berlin* http://webdoc.sub.gwdg.de/ebook/ah/dfn/GTBSB-Metacomputing.pdf ZIB, Berlin; Max-Planck-Institute: MFK-Stuttgart, MPA-Garching, MPIP-Mainz; Rechenzentren der Uni Stuttgart (RUS) und der MPG (RZG)
5. I. Foster, C. Kesselman *Globus: A Metacomputing Infrastructure Toolkit* International Journal of Supercomputer Applications, 11, pages 115-128, 1997.
6. I. Foster and N. T. Karonis. A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems. In *Proceedings of SC 98*. IEEE, Nov. 1999. http://www.supercomp.org/sc98.
7. E. Gabriel, M. Resch, T. Beisel, and R. Keller. Distributed Computing in a Heterogenous Computing Environment. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Lecture Notes in Computer Science. Springer, 1998.
8. E. Gabriel, M. Resch, and R. Rühle. 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.

9. E. Gabriel, M. Resch, and R. Rühle. Implementing and Benchmarking Derived Datatypes in Metacomputing. in B. Hertzberger, A. Hoekstra, R. Williams (Eds.) *High-Performance Computing and Networking*, pages 493-502, Springer, 2001.
10. S. Girona, J. Labarta, and R. M. Badia. Validation of Dimemas communication model for MPI collective communications. In *7th EuroPVM/MPI 2000*, Balatonfüred, Lake Balaton, Hungary, September 2000.
11. W. D. Gropp. Runtime checking of datatype signatures in MPI. In Jack Dongarra, Peter Kacsuk, and Norbert Podhorszki, editors, *Recent Advances In Parallel Virtual Machine And Message Passing Interface*, pages 160–167. Springer, 2000.
12. M. G. Hackenberg, R. Redler P. Post, and B. Steckel. MpCCi, multidisciplinary applications and multigrid. In *Proceedings ECCOMAS 2000*, CIMNE, Barcelona, September 2000.
13. R. Hood. Debugging computational grid programs with the portable parallel/distributed debugger (p2d2). In *The NASA HPCC Annual Report for 1999*. NASA, 1999. `http://hpcc.arc.nasa.gov:80/reports/report99/99index.htm`.
14. T. Imamura, Y. Tsujita, H. Koide, and H. Takemiya. An Architecture of Stampi: MPI Library on a Cluster of Parallel Computers. In J. Dongarra, P. Kacsuk, and N. Podhorszki, Editors, *Recent Advances in Parallel Virutal 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.
15. IMPI Steering Committee. IMPI – Interoperable Message-Passing Interface. World Wide Web. `http://impi.nist.gov/`.
16. Pallas GmbH. *Vampir 2.5 User's Guide*, 1999.
17. S.M. Pickles, J.M. Brooke, F.C Costen, E. Gabriel, M. Müller, M. Resch, and S.M. Ord. Metacomputing across Intercontinental Networks. *Future Generation Computer Systems*, 17(8):911–918, June 2001.
18. N. Karonis and B. Toonen. MPICH-G2. World Wide Web. `http://www.niu.edu/mpi`.
19. T. Kielmann, R. F. H. Hofman, H. E. Bal, A. Plaat, and R. A. F. Bhoedjang. MagPIe: MPI's Collective Communication Operations for Clustered Wide Area Systems. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (Ppopp'99)*, Pages 131–140. ACM, May 1999.
20. Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard (version 1.1).* Technical report, 1995. http://www.mpi-forum.org
21. M. Resch, D. Rantzau and R. Stoy, *Metacomputing Experience in a Transatlantic Wide Area Application Testbed*, Future Generation Computer Systems, **(15)**5-6 (1999), pp. 807-816.
22. S. Reynolds. System software makes it easy. *Insights Magazine*, 2000. NASA, `http://hpcc.arc.nasa.gov:80/insights/vol12/`.
23. M. Romberg. UNICORE: Beyond web-based job-submission. In *Proceedings of the 42nd Cray User Group Conference*, 2000. available at http://www.fz-juelich.de/unicoreplus/index.html.
24. H. Takemiya, T.Imamura and H. Koide *TME A Visual Programming And Execution Environment For A Meta-Application* Jaeri Internal Report, 2000.
25. J.S. Vetter and B.R. de Supinski. Dynamic Software Testing of MPI Applications with Umpire. In *SC2000: High Performance Networking and Computing Conf.* ACM/IEEE, 2000.