# Software Development in the Grid: The DAMIEN Tool-Set

Edgar Gabriel[1,2], Rainer Keller[1], Peggy Lindner[1], Matthias S. Müller[1], and Michael M. Resch[1]

[1] High Performance Computing Center Stuttgart,
Allmandring 30, D-70550 Stuttgart, Germany
[2] Innovative Computing Laboratories,
Computer Science Department,
University of Tennessee, Knoxville, TN, USA
{gabriel, keller, lindner, mueller, resch}@hlrs.de

**Abstract.** The development of applications for Grid-environments is currently lacking the support of tools, which end-users are familiar with from their regular working environment. This paper analyzes the requirements for developing, porting and optimizing scientific applications for Grid-environments. A toolbox designed and implemented in the frame of the DAMIEN project which closes some of the gaps and supports the end-user during the development of the application and its day-to-day usage in Grid-environments is then presented.

## 1 Introduction

The Grid is generally seen as a concept for 'coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organization' [9]. 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 [5], the idea of distributing 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 [2,11], increased throughput on the machines and reduced turnaround times.

Distributing a parallel 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 in several senses. First problems arise with different data representations on different machines, which requires data conversion either in the communication layer or in the application. Various processor speeds, differences in the available memory and the usage of shared resources require the application to have a smart initial load distribution as well as dynamic load balancing. The differences in the communication characteristics for communication between processes located on potentially different sites require distinct programming techniques for hiding the wide area latency and dealing

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2657, pp. 235–244, 2003.

with the low bandwidth. Another level of heterogeneity is introduced by the different methods of access to the different machines in a Grid, e. g. ssh [21], UNICORE [1] or globus GSI [9].

This paper focuses on application development for the Grid, specifically scientific applications coming from the area of high performance computing. The application usually has to pass several stages until it can be used for production runs. These steps will be analyzed and we will show how the tools developed in the frame of a European Grid-computing project called DAMIEN supports end-users during these stages.

The structure of the paper is as follows: section 2 analyzes the software lifecylce of an application in the Grid, presenting its different stages and steps, through which it passes during its development and usage. The tools developed in the DAMIEN project supporting the development and usage of applications for Grid environments are shown in section 3. The DAMIEN testbed and usage scenarios shown during the Supercomputing Conference 2002 in Baltimore, USA, are shown finally in section 4.

## 2 Software Life-Cycle of an Application in the Grid

A scientific application has to pass two different stages during its lifetime:

- The development phase: in this phase the application is modified to enable the solution of certain problems. At the end of this phase, the application is stable and ready for production runs.
- The production phase: the code can now be used for solving the problems which it was originally designed for.

The boundaries between these two stages are not well defined. It is quite common to modify programs which have already been used for production runs for new problems. Therefore, during its lifetime, the code will pass both phases several times. However, since these two stages require quite different handling from an end-user's point of view, we would like to keep the distinction and adapt this scheme to the development of scientific applications in Grid-environments.

The development phase as depicted in figure 1, is described in the following section. Starting point for a Grid-enabled application is either a sequential code or a parallel message-passing code, which has already been used on regular high performance computing platforms. Although rarely done, a program may also be developed from scratch targeting from the very beginning for distributed Grid-environments. Furthermore, a new class of applications, which combine several applications from different scientific areas, are getting more and more popular among scientists and in industry. These applications, known as multi-disciplinary or coupled applications, are a promising approach for Grid-environments, since with careful handling by the end-user, the communication pattern of the application may reflect quite well the communication characteristics of the Grid-computer.

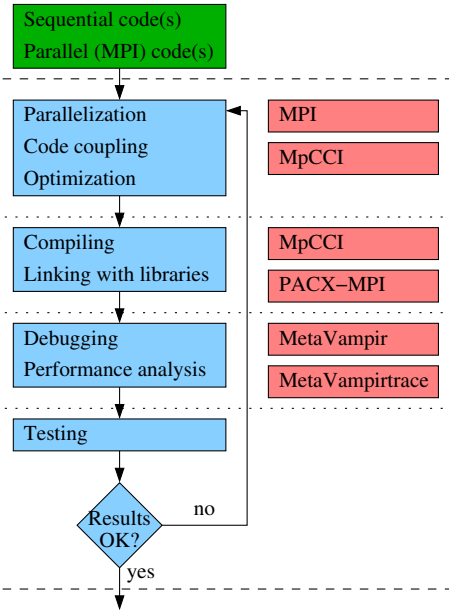


**Fig. 1.** Different steps of an application during the development phase and the DAMIEN tools supporting this stage.

Depending on the starting point provided by the application, the code has to be parallelized in the case of a sequential program, or the codes have to be coupled in some way for multi-disciplinary applications. For coupling applications, some coupling-libraries may be used, which provide a high-level communication interface between the applications.

When having a first version of the 'new' application, the user compiles the code and links the application with the necessary libraries. These usually consist of the communication library, nowadays usually an implementation of the Message Passing Interface (MPI) [19], maybe the coupling-library and some libraries containing numerical routines, e. g. the BLAS-libraries [7].

First tests with some smaller test-cases may reveal problems and bugs. To remove these bugs, special tools for debugging will be used. A special case of a bug is, if the application produces correct results, but has a much lower performance than originally expected. In this case a performance analyzer will be used, which gives hints about the communication pattern (and bottlenecks) and processor-performance. Depending on the problem found, the user has to go back to the parallelization/optimization step, or maybe even have a look at the algorithms used.

Finally, after a certain number of iterations, the user will have a working code, and some experience with it for small or medium size problems. Together with the given problem, he's seeking to solve, these are the input parameters

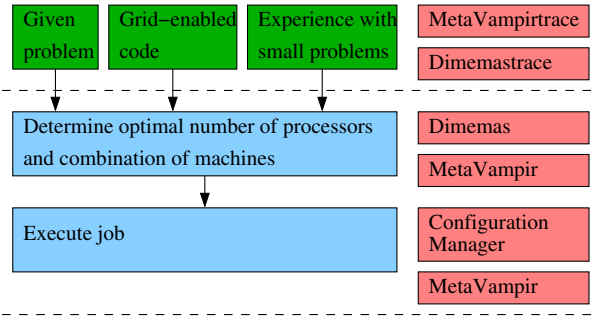


**Fig. 2.** General description of the necessary steps for a production run and the DAMIEN tools supporting these steps.

for the second stage of the application, the production phase (see figure 2). The problems of the end-user are now slightly different. He has to figure out, how many processes and which machine(s) he should use for optimal performance. This decision is usually driven by the need for minimizing the turn-around time of his simulation. This, on one hand is depending on the optimal performance of the application on the used machines and networks, and on the other hand by the current load of the machines.

After determining these parameters, he can finally start the job on the cluster of machines he decided to use. The execution of the application has again different problems, like co-scheduling of all used resources, including the network, and authentication and authorization of the user on all resources.

## 3 The DAMIEN Project

The DAMIEN project (Distributed Applications and Middleware for Industrial Use of European Networks) started early 2001 as one out of originally three Grid-projects funded by the European Commission. The consortium consists of five partners: the European Center for Parallelism of Barcelona (CEPBA), the Regional Computing Center of the Haute Normandie (CRIHAN), the High Performance Computing Center of Stuttgart (HLRS), the software company Pallas GmbH and the European Aeronautic Defense and Space company (EADS). The goal of the project is to develop a tool-set for Grid-computing, based partially on existing and widely accepted tools from the area of high performance computing, and partially by developing new tools. All tools together form an environment, which supports application developers and end-users in a completely new way in Grid-environments.

### 3.1 Tools for the Development Phase

The tools supporting the end-user during the development phase of an application are shown on the right side of figure 1. The parallelization paradigm supported in the DAMIEN project is based Message Passing Standard MPI [19, 20]. The communication library used in the frame of the project is called PACX-MPI[10] and is an optimized implementation of the MPI-standard for Meta- and Grid-computing environments. Its current functionality includes the whole MPI-1 standard as well as three parts of the MPI-2 specification, which were considered to be of interest for Grid-Computing (extended collective operations, language interoperability functions and the canonical pack/unpack functions). In the frame of the DAMIEN project, PACX-MPI is extended to support the handling of several network connections simultaneously between each pair of hosts. This is important for optimal usage of the available bandwidth on nowadays Gigabit high speed backbone networks.

For coupling of several applications, the Code Coupling Interface MpCCI [14] is extended to support Grid-environments. MpCCI provides the end-user data exchange functions on a higher level by providing all required functionality (e. g. interpolation between different meshes). Since the communication inside MpCCI is based again on MPI, the library is ported to work on top of PACX-MPI.

For debugging and performance analysis of a Grid-enabled scientific application, MetaVampir [4] and its tracing library MetaVampirtrace are included in the tool-set. The tracing library is again ported to PACX-MPI and can deal with the hierarchical communication subsystem given in Grid-environments. The interface between PACX-MPI and MetaVampirtrace is based on the profiling interface of MPI, which gives the user the possibility to replace MPI-calls by routines with different functionality, and providing a second, name-shifted version of all MPI-routines. MetaVampir can then visualize the data produced by MetaVampirtrace, allowing some additional analysis based on the machines, and not just on the level of processes. To gather a more detailed view of the application and of the used tools, both MetaVampir and MetaVampirtrace are extended to enable an application level view of the tracefile or a more low level view. As an example, the user can decide, whether he would like to see the communication daemons of PACX-MPI or not. This gives him the flexibility to see internal communication routines of PACX-MPI, if required, and get more detailed information about a message.

### 3.2 Tools for Production Phase

For the production phase, things are more complicated. Assuming that the user has finally a working version of the code, we need to reflect somehow the *experience with the code for small and medium size problems* mentioned in section 2. This is done by requiring a tracefile of the application for Dimemas [12]. This can either be done by using the Dimemastrace library directly, or using MetaVampirtrace with a special option. Dimemas is a tool for performance prediction, which can estimate the behavior of an application based on an initial tracefile.

The working approach for Dimemas is as follows. Based on the initial tracefile, the user can modify certain parameters of the run, e. g. the quality of the network between the machines, or the distribution of the processes on the machines. Using a simulator, Dimemas estimates the behavior of the application with the modified parameters. For checking the result, the user can than write a tracefile, which can be analyzed with MetaVampir, or, in case he trusts the results (because he analyzed this application several times), he can just use the resulting simulation time. Several configurations may thus be tested. Finally, the user will determine which configuration from the tested ones is best for the given problem. Ideally, this loop, which is also represented in figure 3, would be executed by some tool automatically by taking into account a certain machine-pool, the number of available nodes on each machine, the communication characteristics of each machine and between each pair of machines, and finally telling the user the best solution. However, this goal is out of scope for this project, but is a target for future research between the partners.

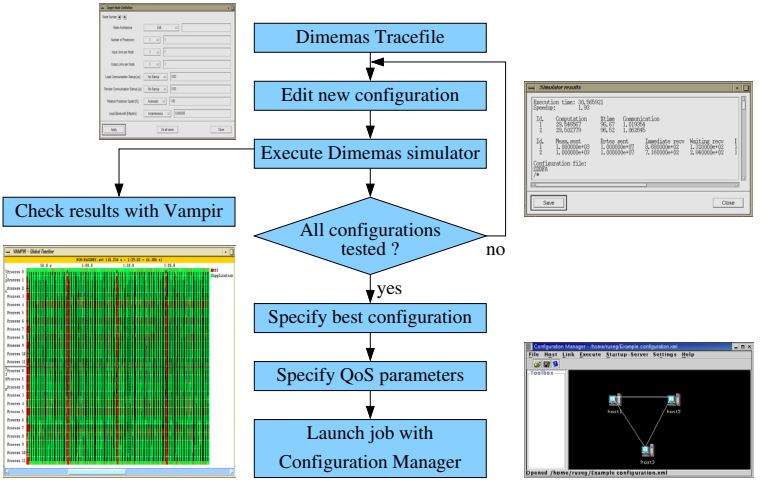


**Fig. 3.** Handling of the DAMIEN tools in the production phase.

After determining which combination of machines he would like to use, the user may start the application on these machines. To support this step, a Configuration Manager [18] for PACX-MPI is developed in the frame of the project. This Configuration Manager has foremost three goals:

- Create the configuration files for PACX-MPI based on the specification of the user. To ease the specification of the resources, a GUI is provided to the end-user.
- Distribute the configuration files onto all participating machines.
- Start the application on all specified machines.

For the second and third step, a direct interaction with the target machines is required. Since we strongly believe that in the near future we will have a heterogenous Grid-infrastructure regarding the methods how to access a machine (e. g. ssh [21], UNICORE [1], globus [9]), another goal of the Configuration Manager is to hide this complexity from the end-user by supporting several of these mechanisms.

The final step – and maybe most critical point for achieving the required performance for a production run – is to ensure a certain level of quality of the network between the machines. Therefore, a Quality of Service Manager (QoS Manager) is currently developed in the frame of the DAMIEN project, having the goal to set certain QoS parameters whenever required. The QoS Manager will be integrated into the communication library PACX-MPI. The specification of the required parameters will be supported by the Configuration Manager. However, this point is critical, since the QoS Manager has to rely on certain functionality provided by the network. While this is ensured by the new high speed backbone networks, like e. g. Géant, it might be a problem when using several network providers, which offer networks of poorer quality than desired. This however, is currently necessary to reach Géant.

### 3.3 The DAMIEN Testbed

To test the tools developed in the DAMIEN project, a testbed between the academic partners has been set-up. The DAMIEN testbed consist of several computers installed at the various sites of the project partners, namely

- a 4 processor SGI Origin 2000 at CRIHAN,
- a 64 processor SGI Origin 2000 as well as
- a $4 \times 16$ processor IBM RS6000 SP at CEPBA and
- a $16 \times 8$ processor Hitachi SR8000 as well as
- a 20 processor SGI Origin 2000 installed at HLRS.

From a top-down view, the interconnecting network consists of the European high speed backbone network Géant, connecting the French Renater 2, the Spanish Rediris, and the local provider BelWü in Germany.

Another testbed for the industrial partners consisting of several PC-clusters on different locations of EADS is currently under development, and expected to work early 2003.

Several applications are used and tested on the testbed. The application EADS wants to test within the project is a multi-disciplinary vibro-acoustic simulation, which can be used for solving several problems (e. g. noise reduction simulations in airplanes). Since the EADS application is currently still in the development phase, the application being run over the testbed consisted of an application for the simulation of non-equilibrium flow around space vehicles reentering the atmosphere, called URANUS [3].

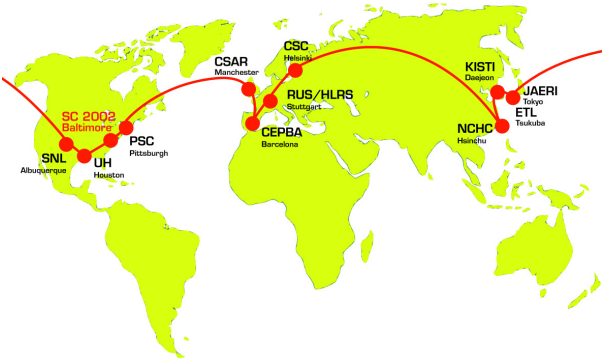


**Fig. 4.** Global Wide Area Testbed presented at Supercomputing 2002 in Baltimore

## 4 Demonstrations at Supercomputing

At the HPC-Challenge at SuperComputing 2002, we demonstrated the usage of a Metacomputer coupled with PACX-MPI and running a bioinformatics simulation of the folding of single-stranded mRNA. This code was developed at the University of Vienna by Ivo Hofacker[15] and improved by Sandia National Labs and HLRS within the frame of this work. Specifically, the energy parameters were updated, the application was integrated into a visualization environment for virtual reality and the communication pattern improved to be efficient for Metacomputing. The computational complexity of this code is in the order of $O(n^3)$, while the communication is in the order of $O(n^2)$, with $n$ being the length of the mRNA-sequence. Among the 22 machines used at the partners sites (see Fig. 4) were:

1. Cray T3e/512 (HLRS, Germany; MCC, UK; PSC, US; CSC Finland)
2. SGI Origin 2000 (AIST, Japan; CEPBA, Spain)
3. SGI Origin 3800 (MCC, UK; NCHC, Taiwan)
4. IBM p690 (KISTI, Korea; NCHC, Taiwan)
5. Compaq Terascale Computing System (PSC, US)
6. Compaq Alpha Cluster (JAERI, Japan)
7. Hitachi SR8000 (HLRS, Germany)
8. NEC SX5 (HLRS, Germany)
9. IBM RS6000 SP (CEPBA, Spain)
10. Fujitsu PrimePower VPP (JAERI, Japan)

The heterogenous Metacomputer consisted of up to six computers, distributed over the three continents. To facilitate the starting of this Metacomputer, the Configuration Manager was used.

This heterogeneous set of computer worked on jobs submitted by the user to the computers with the GUI from the VR-environment at the booth of HLRS at SC02. The partners were then able to join the demonstration live via AccessGrid. Integration of the application into this framework enabled the user to:

- Visualize and collaboratively work on the computed results.
- Start new simulations on the connected computers (Metacomputer and single hosts) interactively.
- Monitor progress of the computation on the host.

## 5 Conclusion

This paper analyzed the different steps an application has to pass, before it can be used for a production run. It has been shown, that each of these steps require several tools, which support the end-user during each step. The DAMIEN project develops a new tool-set which supports end-user and application developers in the complex task of porting applications to heterogeneous Grid environments as well as guiding him through the difficulties of using the Grid for every day production runs. Some tools developed in the project (PACX-MPI, Configuration Manager, QoS Manager) are freely available, while Vampir, Dimemas and MpCCI will remain available as commercial ones. The interaction of the tools as well as the scenarios supported by the project are unique among the currently running Grid projects, by not even claiming to solve all problems, but by focusing on a very special task. The tools have proved their usefulness with several applications in a European testbed, during international demonstrations at SC2002 in Baltimore, and are currently tested under industrial conditions.

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

## References

1. J. Almond, D. Snelling: *UNICORE: Secure and Uniform Access to Distributed Resources via the World Wide Web.* A White Paper, October 1998.
2. G. Allen, T. Dramlitsch, I. Foster, N. T. Karonis, M. Ripeanu, E. Seidel, and B. Toonen: *Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus.* Supercomputing 2001, Denver, 2001.
3. T. B. Bönisch, R. Rühle: *Efficient Flow Simulation with Structured Multiblock Meshes on Current Supercomputers* in ERCOFTAC Bulletin No. 50: Parallel Computing in CFD, 2001.
4. H. Brunst, M. Winkler, W. E. Nagel, and H.-C. Hoppe: *Performance optimization for large scale computing: The scalable vampir approach* in V. N. Alexandrov, J. J. Dongarra, B. A. Juliano, R. S. Renner, and C. K. Tan, editors, Computational Science – ICCS 2001, Part II, number 2074 in LNCS, pages 751–760, San Francisco, CA, USA, May 2001. Springer.
5. C. Catlett and L. Smarr: *Metacomputing* in Communications of the ACM, 35(6):44–52, 1992.
6. *DAMIEN – Distributed Application and Middleware for Industrial Use of European Networks.* World Wide Web: `http://www.hlrs.de/organization/pds/projects/damien`.

7. J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson: *A set of Level 3 Basic Linear Algebra Subprograms.* ACM Trans. Math. Soft., 16 (1990), pp. 1–17.
8. 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.
9. I. Foster, C. Kesselmann, S. Tuecke: *The Anatomy of the Grid: Enabling Scalable Virtual Organizations* in International Journal of Supercomputing Applications, 15(3), 2001.
10. E. Gabriel, M. Resch, T. Beisel, and R. Keller: *Distributed Computing in a Heterogeneous Computing Environment* in Vassil Alexandrov, Jack Dongarra (Eds.) 'Recent Advances in Parallel Virtual Machine and Message Passing Interface', pp. 180–188, Springer, 1998.
11. E. Gabriel, M. Lange, and R. Rühle: *Direct Numerical Simulation of Turbulent Reactive Flows in a Metacomputing Environment* in Proceedings of the 2001 ICPP Workshops, pp. 237–244, 2001.
12. 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.
13. W. Gropp and E. Lusk: User's Guide for mpich, a Portable Implementation of MPI.
14. M. G. Hackenberg, R. Redler, P. Post, and B. Steckel: *MpCCI, multidisciplinary applications and multigrid* in Proceedings ECCOMAS 2000, CIMNE, Barcelona, September 2000.
15. I. L. Hofacker, W. Fontana, L. S. Bonhoeffer, M. Tacker, P. Schuster: *Vienna RNA Package*, World Wide Web: `http://www.tbi.univie.ac.at/~ivo/RNA`, October 2002.
16. 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 (eds.), Recent Advances in Parallel Virutal Machine and Message Passing Interface, number 1908 in Springer Lecture Notes in Computer Science, pages 200–207, September 2000. 7th European PVM/MPI Users' Group Meeting.
17. N. Karonis and B. Toonen: *MPICH-G2.* World Wide Web: `http://www.niu.edu/mpi`.
18. P. Lindner, N. Currle-Linde, M. M. Resch, E. Gabriel: *Distributed Application Management in Heterogeneous Grids* in Proceedings of the Euroweb Conference, Oxford, GB, pp. 145–154, December 17–18, 2002.
19. MPI Forum: *MPI: A Message-Passing Interface Standard.* Document for a Standard Message-Passing Interface, University of Tennessee, 1995.
20. MPI Forum: *MPI2: Extensions to the Message-Passing Interface Standard.* Document for a Standard Message-Passing Interface, University of Tennessee, 1997.
21. *Mindterm Secure Shell.* World Wide Web: `http://www.mindbright.se`.
22. M. Müller, E. Gabriel and M. Resch: *A Software Development Environment for Grid-Computing* in Concurrency and Computation – Practice and Experience, Vol. 14:1543–1551, 2002.
23. S. M. Pickles, J. M. Brooke, F. C. Costen, E. Gabriel, M. Müller, M. Resch and S. M. Ord: *Metacomputing Across Intercontinental Networks* in Future Generation Computer Systems (17) 2001, pp.911–918, Elsevier Science.