

THE UNICORE GRID AND ITS OPTIONS FOR PERFORMANCE ANALYSIS

Sven Haubold, Hartmut Mix, and Wolfgang E. Nagel

Center for High Performance Computing (ZHR), TU Dresden

Dresden, Germany

{haubold,mix,nagel}@zhr.tu-dresden.de

Mathilde Romberg

Central Institute for Applied Mathematics (ZAM), Research Center Juelich

Juelich, Germany

m.romberg@fz-juelich.de

Abstract UNICORE (Uniform Interface to Computer Resources) is a software infrastructure to support seamless and secure access to distributed resources. It has been developed by the projects UNICORE and UNICORE Plus in 1997 - 2002 (funded by the German Ministry of Education and Research) and is going to be enhanced in the EU-funded projects EUROGRID and GRIP. The UNICORE system allows uniform access to different hardware and software platforms as well as different organizational environments. The core part is the abstract job model. The abstract job specification is translated into a concrete batch job for the target system. Besides others, application specific support is a major feature of the system. By exploiting the plugin mechanism, support for performance analysis of Grid applications can be added. Here, as an example support for Vampirtrace has been integrated. The UNICORE user interface then gives the option to add a task using Vampirtrace with runtime configuration support into a UNICORE job and retrieve the generated trace files for local visualization. Together with the support for compilation and linkage and for metacomputing, the plugin mechanism may be used to integrate other performance analysis tools in future.

Keywords: Grid computing, UNICORE, Vampir, performance analysis, application specific interface, Java

1. Introduction

The Grid [2] has become the major research topic for all kinds of scientists. Computer scientists, physicists, biologist, engineers and other application people try to get this new computational infrastructure to real production mode. It is still a very complex environment, and the only way to make significant steps is collaboration across all disciplines. Actually, in many projects the solution of a special application problem takes the major amount of work, others focus on the definition of new functionality or features which are needed to provide a flexible environment. The process of standardization - beyond de-facto standards - has fruitfully started, and now the aspects of performance get into focus.

Performance analysis in a Grid environment has several aspects: Performance of distributed applications, performance of Grid components, and performance of the overall system. Projects like for example crossgrid ([14]) work on verification and performance prediction tools as well as detection of performance bottlenecks in applications in Grid environments. Crossgrid will propose a set of performance metrics to describe concisely the performance capacity of Grid configurations and application performance, and it will develop and implement benchmarks that are representative of typical Grid workloads. Based on that, it will provide on-line tools which allow application developers to measure, evaluate, and visualize the performance of Grid applications with respect to data transfer, synchronization and I/O delay as well as CPU, network and storage utilization.

The Global Grid Forum ([15]) works on the standardization in the field of monitoring and performance analysis. Working Groups on Discovery and Monitoring Event Description (DAMED-WG) and Network Measurement (NM-WG) together with the Research Group Grid Benchmarking (GB-RG) deal with the definition of a basic set of monitoring event descriptions, the definition of a hierarchy of network measurements for Grid applications and services, and the definition of metrics to measure performance of Grid applications and architectures and rate functionality and efficiency of Grid architectures.

This article will focus on the flexible support for application performance analysis in the UNICORE framework. UNICORE is a major initiative to provide a stable Grid production framework for HPC applications running on different sites in Germany and beyond. It provides an interface which allows to transparently use computer resources in different Grid centers. We will describe the mechanisms which have been developed to support an easy, flexible, and transparent performance analysis process for applications running on different sites. As the basis, we have used Vampir as the well established performance analysis tool for parallel environments.

The remainder of this article gives first a general introduction into the UNICORE Grid infrastructure, the user functions it offers, and relevant interfaces.

A description of Vampir and its capabilities follows in section 3 while section 4 explains the integration of Vampir into the UNICORE framework. The outlook summarizes the exploited features and describes options for further performance analysis.

2. UNICORE

The idea to build a *uniform interface to computing resources* goes back to mid 1996 when the management of the supercomputer centers in Germany were looking for improved end user access and closer cooperation: Already at that time the vision was that the users should be able to use the most appropriate computer system for their application without taking care about system and site specific commands, file spaces, security mechanisms, and such. This initiative has been supported and funded in part by the BMBF (German Ministry for Education and Research) in two projects, UNICORE (1997-1999, [16]) and UNICORE Plus (2000-2002, [17]). The project goal has been to provide a seamless, secure, and intuitive interface to distributed heterogeneous computer resources. Partners in these projects are from universities, research laboratories and software companies¹. This group has developed the Grid system UNICORE described in more detail in the next subsections.

2.1 The Grid Infrastructure UNICORE

The UNICORE three tier architecture developed consists of user, server, and target system tier as shown in Figure 1 (see also [10]). The user tier comprises the graphical user interface. It offers the functions to prepare and control UNICORE jobs and to set up and maintain the user's security environment. The security architecture is based on the Secure Socket Layer (SSL) protocol. SSL is used for all communication over public networks. Communication over the intra-net at a site is done unencrypted by default but SSL is a configurable option. From the user input, the user interface generates an Abstract Job Object (AJO) which is sent via SSL to the Gateway. The AJO is a key component in the architecture. It comprises the UNICORE protocol between user interface and the Network Job Supervisor together with the abstract job specification generated from the user input (references [1] and [9] explain the AJO in more detail).

¹Partners are: Forschungszentrum Juelich GmbH, Deutscher Wetterdienst (DWD), Pallas GmbH, Rechenzentrum der Universitaet Stuttgart (RUS), Konrad Zuse Zentrum fuer Informationstechnik (ZIB), Leibniz Rechenzentrum der Bayrischen Akademie der Wissenschaften (LRZ), Paderborn Center for Parallel Computing (PC²), Rechenzentrum der Universitaet Karlsruhe (RUKA), Fujitsu Laboratories of Europe (former fecit), Genias Software GmbH (UNICORE project only), and Zentrum fuer Hochleistungsrechnen an der TU Dresden (ZHR, UNICORE Plus project only).

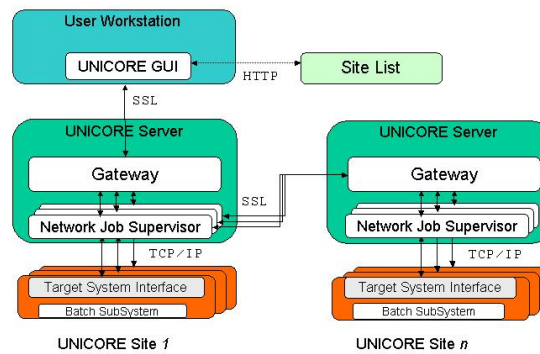


Figure 1. Architecture Overview

The Gateway is the entrance to a site. It takes care of user authentication, secure communication between client and server, and provides the client with information about the available resources at the site. It also talks to the Network Job Supervisor (NJS) servers at its site to send jobs and data, status requests, and control commands for further processing and to receive data to make it available to the user.

Each Network Job Supervisor controls one Vsite (Virtual Site) which is a single system or one cluster of systems governed by one resource management system and sharing the same userids and file space. It fulfills the following tasks:

- analyze the AJO (representing the UNICORE Job) to extract local and remote jobs and tasks;
- map the UNICORE userid to the local userid for the target system (Vsite);
- translate the local tasks contained in the AJO into real batch jobs for the target system;
- send sub-jobs to be executed at other UNICORE sites to the corresponding gateway;
- provide local resource information to the gateway;
- take care of the necessary file transfer;
- schedule sub-jobs and tasks according to the specified work-flow;

- provide job status information and job output.

The NJS assigns a UNICORE job directory to each UNICORE job which serves as a UNIX working space for the job. It is a temporary directory only existing during lifetime of the job at the site. All data needed for job execution has to be imported from permanent file space to the job directory. All data which is needed after the job has finished has to be saved to permanent file space. The user specifies the data to be imported and those to be exported when constructing the job. The transfers are done by UNICORE transparently to the user.

The Target System Interface (TSI) implements the interface to the local operating and resource management system.

2.2 Application Specific Interfaces

UNICORE offers a flexible mechanism to easily integrate existing applications: The plugin interface. It consists of two parts: the first is the interface in the graphical user interface and the second is the corresponding definitions for the software resource and its execution on the target system in the Network Job Supervisor's Incarnation Data Base. These interfaces have been initially exploited for the Car Parrinello Molecular Dynamics application (see Figure 2, and [5] for details).

The plugin mechanism allows for adding new job task elements into the job preparation part as well as components for the output interpretation (i.e. visualization) into the job monitoring part. The core classes and their methods from the basic user client can be exploited by the application specific interface to access user data, model internal dependencies, build the corresponding AJO, retrieve output and alike. Hereby a powerful tool is provided to integrate any further application into the UNICORE framework.

3. Vampir

Performance optimization remains one of the key issues in parallel computing. With the emergence of Grid applications running on more than one system, the task of analyzing and tuning scientific applications actually becomes harder. Tools need to be extended to enable performance analysis also for this new application class in a global Grid environment.

Up to now, the predominant parallel architecture classes have been the classic shared-memory systems with limited scalability and the scalable distributed-memory MPP systems. For each class, performance analysis methodologies and tools (AIMS[12], PABLO [11], PARADYN[7], Paragraph[4], Paraver[6], Vampir [8], et. al. [13]) have been developed and many significant scientific and engineering codes have been ported and optimized. Even for clustered SMP platforms, this picture has changed. To achieve a good price / performance ratio, these systems do not provide an efficient virtual-shared-memory sys-

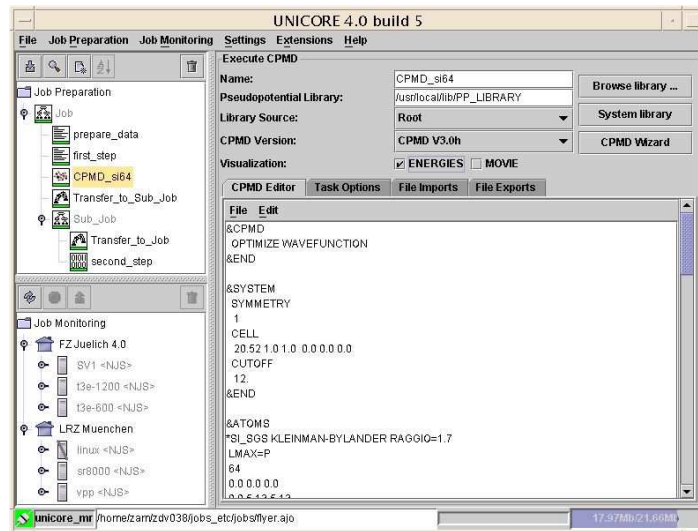


Figure 2. Plugin Interface for CPMD

tem, but expose their structure to the applications. An application programmer aiming at scalability beyond one SMP node has to make a choice between the message-passing programming model, or a combination of message-passing between SMP nodes, most probably MPI and a shared-memory model within one node, most probably OpenMP. In both cases, performance analysis faces new challenges, and there has been already a strong need to extend the current MPP performance-analysis tools by

- support for multi-threading,
- analysis of memory and CPU statistics,
- display of scheduler events and interactions.

The main challenge is an appropriate organization of performance data, both internally to the tools and more important to the end-user: the potentially enormous amount of performance information (in particular if event tracing is used) has to be processed and displayed in a way that an ordinary user can understand.

Vampir [8] is a performance analysis tool which has addressed these kinds of topics for many years. It supports the performance analysis process and makes it easy for the programmer to get insight into the parallel execution of a program on any homogeneous parallel system. Vampir converts trace information into a variety of graphical views, e.g. timeline displays (Figure 3)



Figure 3. Vampir Timeline with Zooming Capabilities

showing state changes and communication, profiling statistics displaying the execution times of routines, communication statistics indicating data volumes and transmission rates, and more. The displays can be related to the source code, and Vampir's advanced navigation functions allow to easily zoom into arbitrary time intervals. The profiling and communication statistics help in identifying performance bottlenecks.

The Vampirtrace profiling tool for MPI applications produces tracefiles that can be analyzed with the Vampir performance analysis tool. It records all calls to the MPI library and all transmitted messages, and allows to define and record arbitrary user defined events.

The performance analysis process gets even more difficult if the program is executed in the Grid where accessibility, connectivity, and security aspects make the situation much more complex.

4. Vampir Integration in UNICORE

To achieve a user-friendly interface which helps the user to generate tracefiles during his application's run, an application specific plugin for Vampirtrace has been integrated in UNICORE.

This interface assists the user in choosing the right configuration settings and preparing the necessary operating system environment for the instrumented program run. Normally, in many environments supporting commercial tools and programs it is not sufficient to have an executable program which was

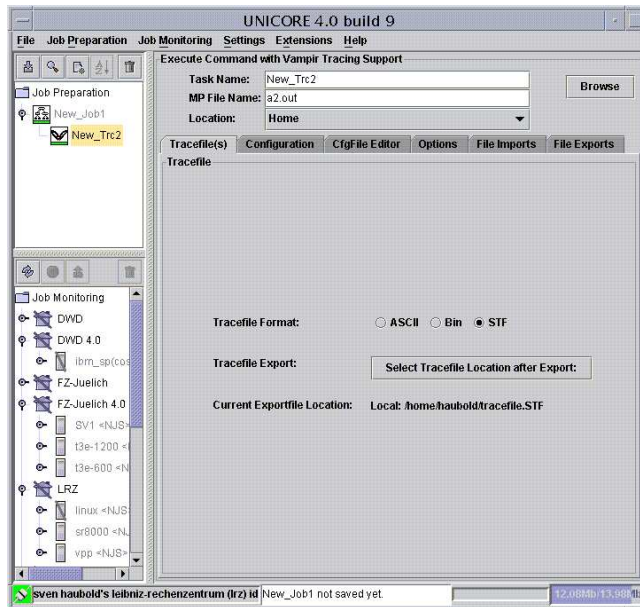


Figure 4. Application Specific Interface for Vampir

compiled and linked using the appropriate profiling libraries. For example, in order to use Vampirtrace on a system, you must have access to a license key. The license keys are stored in a plain ASCII file. Before the program can be started, the user has to specify where the program can find the necessary license file. This information is given to the application transparently by some environmental variables. Besides the user has the possibility to specify the extend of information which is written to the tracefiles by some configuration directives placed in a configuration file. With this configuration file, the user can customize various aspects of Vampirtrace's operation and define trace data filters.

The Vampirtrace support plugin has a graphical user interface (Figure 4) which is called from the Job Preparation Menu. In the same way as in the normal Command Task Panel the name and the location of the executable has to be chosen here. Additionally, this panel allows the user to specify the type of the tracefile which has to be generated. The user must specify the location to which the produced tracefile has to be exported after program termination. The Vampirtrace Plugin has a second panel which works as a assistant or wizard (Figure 5).

This panel provides some often used configuration settings. These settings can be used also by non-specialists by simple selections. Therefore, it is not

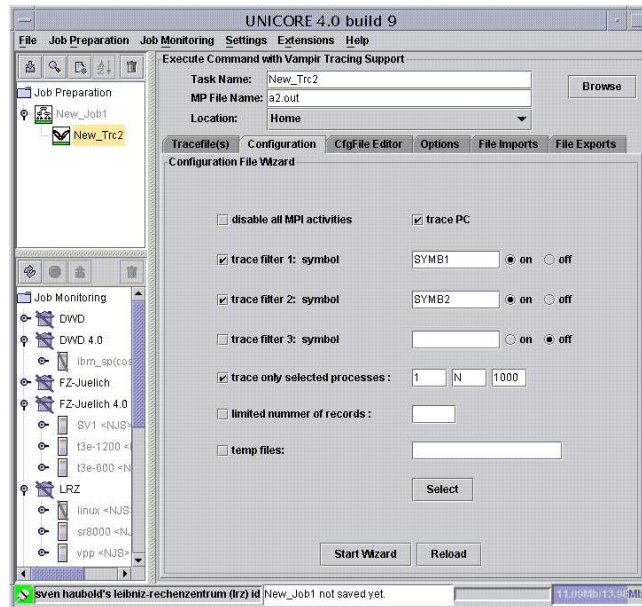


Figure 5. Vampirtrace Wizard

necessary that the user knows the exact syntax of the configuration file. Otherwise, for more experienced users it is also possible to specify all available configuration directives and their parameters in a Configuration File Editor Window (Figure 6).

Besides the specified parameters, the plugin will add the name and type directives for the tracefile automatically to this configuration file. After confirmation of the input, the plugin mechanism will check the correctness of all inputs and produce the necessary configuration file.

As an important second part of the plugin mechanism, all target system specific information like the installed software version or the location of the license file are provided by the system administrator of the Vsite. This information is stored in the NJS Incarnation Data Base and will be added to the AJO during job generation.

So the user can prepare his application to run with simultaneous instrumentation without detailed knowledge of the site or system specifics. The generated tracefile automatically will be transferred to the specified location after task completion and can then be analyzed locally with the Vampir performance analysis tool.

Figure 7 shows a test run of an application running on two different SGI Origin 3800 machines (romulus and remus). Within Vampir, we have added

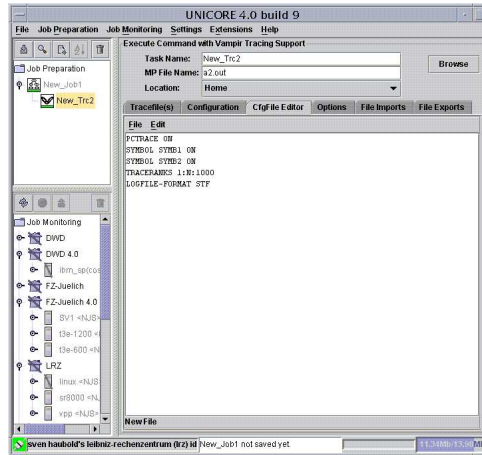


Figure 6. Vampirtrace Configuration File Editor

a grouping concept supporting clusters of machines. This is now part of the whole internal infrastructure within Vampir and makes it easier to analyze communication behavior: fast communication within the machines (diagonal in the *Message Statistics Display*: more than 100 MB/s) and slow communication rates between the systems (less than 10 MB/s). Moreover, the concepts allow to use all well-known features naturally also for Grid applications running distributed between different sites.

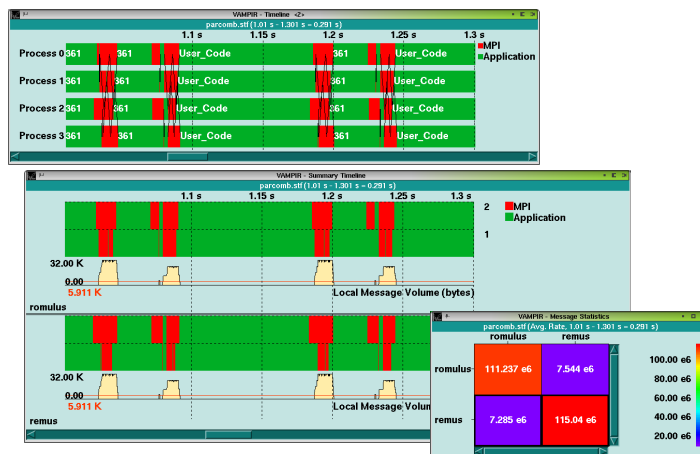


Figure 7. Vampir Displays Showing an Example Run on two Different Machines

5. Outlook

Application execution on the Grid is still a challenge, and many scientist are working on a worldwide level to provide functionality and standards. We have shown in which way the UNICORE Grid framework supports performance analysis for complex applications running distributed between different sites. By extending the new grouping concept in the Vampir infrastructure, the full set of Vampir features can now be used also for Grid applications. The focus of the project was the transparent access to the basic trace information via plugins which now allows to ease performance analysis even on a homogeneous system. So far, the UNICORE framework has been proven to support effective performance analysis for distributed Grid applications.

In the near future, the aspects of performance analysis - and the resulting optimization steps - will become one of the major issues addressed by many projects. We will see how general solutions will be to use them also in different environments.

Acknowledgments

Part of the developments described in this paper have been funded by the German Ministry of Education and Research (bmb+f) in the projects UNICORE and UNICORE Plus under Grant 01 IR 703 and 01 IR 001. We also want to thank H.Ch. Hoppe and his team from Pallas GmbH for their support with Vampirtrace and M. Resch and his team from HLRS Stuttgart for the support with PACX and the application PCM.

References

- [1] D.W. Erwin and D.F. Snelling. UNICORE: A Grid Computing Environment. in *Proceedings of Euro-Par 2001*, Springer LNCS 2150, August 2001, pp.825-834
- [2] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufman Publishers, 1999
- [3] I. Foster, C. Kesselman, J.M. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. <http://www.globus.org/research/papers/ogsa.pdf>, June 2002
- [4] M. T. Heath, A. D. Malony, and D. T. Rover. Visualization for parallel performance evaluation and optimization. In I. J. Stasko, J. Domingue, M. H. Brown, and B. A. Price, editors, *Software Visualization*, pages 347–365. MIT Press, Cambridge, 1998.
- [5] V. Huber. Supporting Car-Parrinello Molecular Dynamics Application with UNICORE. in *Proceedings of the Computational Science - ICCS 2001 International Conference*, San Francisco, May 2001, Part I, pp.560-566
- [6] J. Labarta, S. Girona, V. Pillet, T. Cortés, and L. Gregoris. DiP: A Parallel Program Development Environment. In *2nd International EuroPar Conference (EuroPar 96)*, Lyon,

France, August 1996.

<http://www.cepba.upc.es/paraver>.

- [7] B. P. Miller, M. D. Callaghan, J. M. Cargille, J. K. Hollingsworth, R. B. Irvin, K. L. Karavanic, K. Kunchithapadam, and T. Newhall. The Paradyr Parallel Performance Measurement Tools. *IEEE Computer*, 28(11):37–46, November 1995.
<http://www.cs.wisc.edu/~paradyn>.
- [8] W. E. Nagel, A. Arnold, M. Weber, H.-C. Hoppe, and K. Solchenbach. VAMPIR: Visualization and Analysis of MPI Resources. *Supercomputer 63*, XII(1):69–80, January 1996.
<http://www.pallas.de/pages/vampir.htm>.
- [9] M. Romberg, The UNICORE Architecture: Seamless Access to Distributed Resources. in *Proceedings of the eighth IEEE International Symposium on High Performance Distributed Computing*, August 1999, pp.287-293
- [10] M. Romberg. The UNICORE Grid Infrastructure. in *Scientific Programming Special Issue on Grid Computing* IOS Press, Volume 10, Number 2, 2002
- [11] L. DeRose and D. A. Reed. SvPablo: A Multi-Language Architecture-Independent Performance Analysis System. In *Proceedings of the International Conference on Parallel Processing (ICPP'99)*, Fukushima, Japan, September 1999.
- [12] J. C. Yan. Performance Tuning with AIMS – An Automated Instrumentation and Monitoring System for Multicomputers. In *Proceedings of the 27th Hawaii International Conference on System Sciences*, volume II, pages 625–633, Wailea, Hawaii, January 1994.
<http://www.nas.nasa.gov/Groups/Tools/Projects/AIMS>.
- [13] Pointers to tools, modules, APIs and documents related to parallel performance analysis.
<http://www.fz-juelich.de/apart/wp3/modmain.html>.
- [14] The crossgrid project "<http://www.crossgrid.org>"
- [15] Global Grid Forum "<http://www.globalgridforum.org>"
- [16] The UNICORE project. "<http://www.fz-juelich.de/unicore>"
- [17] The UNICORE Plus project. "<http://www.fz-juelich.de/unicoreplus>"
- [18] UNICORE Forum e.V. "<http://www.unicore.org>"
- [19] The EUROGRID project. "<http://www.eurogrid.org>", funded by the EU, grant period 1.11.2000 - 31.10.2003, grant id IST-1999-20247
- [20] The GRIP project. "<http://www.grid-interoperability.org>", funded by the EU, grant period 1.1.2002 - 31.12.2003, grant id IST-2001-32257