Quality of Service on the Grid with User Level Scheduling

Jakub T. Mościcki[1,5], Marian Bubak[2,5], Hurng-Chun Lee[3], Adrian Muraru[4], Peter Sloot[5]

[1] CERN, IT Department, CH-1211 Geneva, Switzerland
[2] Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Krakow, Poland
[3] Academia Sinica, No. 128, Sec. 2, Academic Road, NanKang, Taipei 115, Taiwan
[4] University Politehnica of Bucharest, Faculty of Computer Science and Automatic Control, RO-060042, Bucharest, Romania
[5] Faculty of Sciences, Section of Computational Science, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Abstract

Currently the largest Grids lack an appropriate level of the Quality of Service (QoS) in two ways: the infrastructure and middleware is not enough reliable and a simple, batch-oriented processing model is suboptimal for a number of applications. User-level scheduling is a light software technique that enables new capabilities to be added and QoS characteristics and reliability to be improved, on top of the existing Grid middleware and infrastructure.

User-level scheduling techniques may be used to reduce the job turnaround time and to provide a more stable and predictable job output rate. Splitting the processing into many fine-grained tasks improves the load balancing and ensures that the workers are used efficiently.

The DIANE user-level scheduler developed at CERN allows users to create transient Master/Worker overlays above the regular Grid infrastructure. The customizable scheduling and synchronization mechanisms allow to support a large class of non-communication-intensive applications. Applications succesfully deployed with DIANE include High Energy Physics data analysis, Monte Carlo simulation, Bio-med applications and others. Distributed radio frequency analysis in the context of International Telecommunication Union (ITU) and autodoc-based drug discovery are examples of successful, large scale activities with an international impact.

We discuss the implications of this technique for the users, the application developers and the resource providers.

1 Introduction

Grids enable scientific computing at unprecedented scale in terms of computing, storage capacity, resource integration and scientific collaboration.

The world's largest production Grid infrastructure to date, EGEE Grid, is a multidisciplinary project, successful enabling diverse user communities such as

high energy physics, bio-informatics, medical physics, earth observation, telecommunications. The creation of the EGEE Grid has been driven by High Energy Physics (HEP) to support the needs of the LHC (Large Hadron Collider) experiments at CERN. Therefore the EGEE Grid must reach its full operational capacity by the startup of the LHC experiments at CERN (end of 2007). In contrast to many, small-scale experimental Grid installations which are research playgrounds for future usage, the EGEE Grid applications must be deployed in a robust, reliable and efficient production Grid of an unprecedented scale in the nearest future.

A large distributed system such as EGEE Grid is inherently dynamic: resources are constantly reconfigured, added and removed; the total number of failures is correlated with the size of the system; the user activities are not coordinated and the load may change rapidly. Independent observations and everyday user experience confirms the large variations in the performance and reliability of the EGEE Grid. Additionally, there is a common feeling in the user community that currently the Grid infrastructures do not provide the required level of service.

A User Level Scheduling technique has been successfully used in the EGEE Grid in a number of scientific activities to improve the Quality of Service (QoS) experienced by the users of the Grid and to provide a robust execution environment customized to the needs of applications.

Section 2 presents the monitoring data which exemplifies the insufficient quality of service in EGEE Grid and contains an analysis of the most important Quality of Service characteristics which need improvement.

Section 3 of provides a summary of techniques investigated in the context of Grid QoS research and explains what are the constraints in applying such techniques in a large scale production environment such as EGEE Grid.

Section 4 presents the User Level Scheduling technique using a DIANE project as an example. It also describes the experience in applying User Level Scheduling in EGEE Grid and its implications for users, application developers and resource providers. Section 5 present the outlook of the future work.

2 Motivation for Improving the QoS on the Grid

The large scale Grids to date do not provide the QoS guarantees. The best-effort Quality of Service is a function of the reliability and performance of the Grid which depends on the scale of the infrastructure and the architecture of Grid services. The EGEE Grid is the largest Grid infrastructure to date and, as of January 2007, it consists of around 200 sites and more than 32,000 worker nodes. Similarly to the classical batch systems, the EGEE Grid is designed for optimizing the throughput of long, non-interactive jobs. A job submission chain typically involves the workload management system (e.g. a resource broker), a computing element and finally a batch system. As shown in [1] the overhead of hierarchical Grid scheduling is not suitable for certain applications which require responsiveness and interactivity. It may also be unacceptable in high-granularity

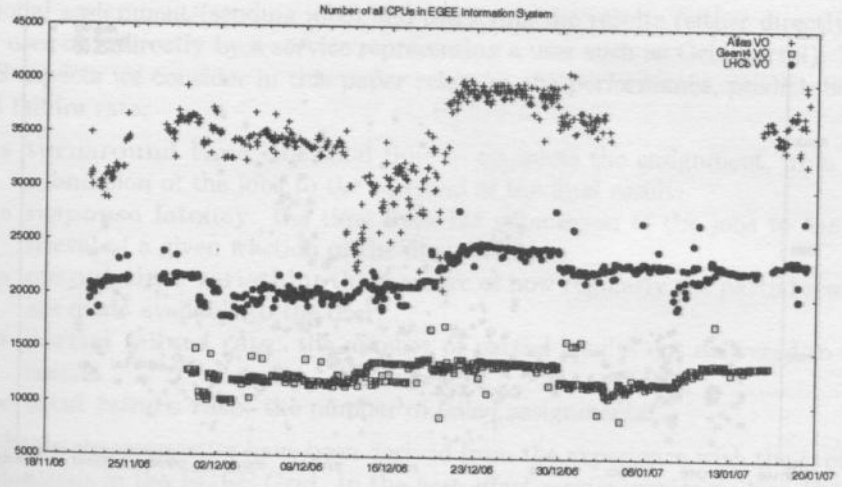

Fig. 1: The number of processors in the EGEE Information System. The data was collected using `lcg-infosites` tool.

job splitting scenarios i.e. many thousands of short jobs completing in a short time.

Additionally, the infrastructure exhibits large variations in stability. Fig. 1 show the number of available CPUs in the same period as visible in the Grid Information Service. Clearly such large variations indicate the limitations of the middleware and cannot be fully accounted for the Grid reconfiguration procedures (e.g. adding and removing CPUs). We do not analyze the reasons for the observed behavior here. From the point of view of a user however, the reliability and performance of job scheduling is affected because the Information Service is used as a source for job brokering decisions. This affects the job handling performance of the Grid as a whole, makes it harder to reliably estimate the job completion time and may increase the brokering failure rate e.g. when the resources compatible with job's requirements cannot be found. Fig. 2 shows the turnaround time of simple test jobs in three different virtual organizations in a period of 6 weeks.

Besides the system related problems, such as worker node crashes or network problems, the Grid users are exposed to numerous configuration/application errors, which in practice cannot be avoided because of the scale and heterogeneity of the Grid infrastructure. What to a user appears as a failed job, may be a software version mismatch, wrong environment, missing application files or out-of-date application software on the worker node. Table 3 presents examples of failures for a physics simulation program based on Geant-4 toolkit [3]. The simulation program is compiled on-the-fly on the worker node prior the execution in order to cope with different flavors of Linux operating system. Each of the selected sites exposes a different problem what clearly illustrates the scale of

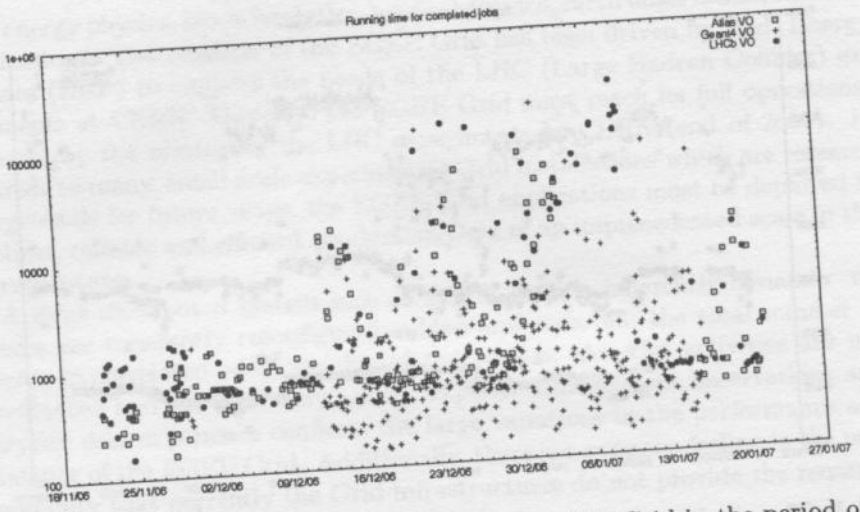

Fig. 2: The turnaround time of test jobs in the EGEE Grid in the period of 8 weeks. Only successfully completed jobs are shown. Jobs in Atlas and LHCb VO were submitted using gLite resource broker. Jobs in Geant4 VO were submitted using LCG resource borker.

the problems a user must cope with. In data-intensive applications the data management is an additional source of failures such as missing input files, bad file transfers etc.

Computing Element	Failure Reason
grid-ce0.desy.de	problem with /usr/include/python2.2
tbn20.nikhef.nl	Operating system version 'CentOS release 3.7' is not supported.
grid109.kfki.hu	g++: Internal error:Segmentation fault (program cc1plus)
ce102.cern.ch	GNUmakefile:28:binmake.gmk:No such file or directory
lcg-ce0.ifh.de	error while loading shared libraries: libstdc++-libc6.2-2.so.3

Fig. 3: Examples of failures of a Geant4 program at the compilation or running phase in the Grid worker nodes in various Computing Elements.

Efficient failure recovery must take into account specific application-related failures, taking into account how the application is deployed and configured. Generic middleware only provides very general, coarse mechanisms to handle errors such as the retry mechanism, but make no attempt to analyze the reason of failure and decide on an appropriate action. Such mechanisms may only be implemented by the application providers and adjusted by the application users.

Quantitative QoS characteristics, typically expressed in terms of latency, jitter, throughput and packet-loss, originated in the network applications [4]. The QoS characteristics of interest to a end-user are related to creating a compu-

tational assignment (sending jobs) and retrieving the results (either directly by the user or indirectly by a service representing a user such as Grid portal). The QoS aspects we consider in this paper relate to the performance, predictability and failure rate:

- **turnaround time**: the total time to complete the assignment, from the submission of the jobs to the retrieval of the final result;
- **response latency**: the time from the submission of the jobs to the retrieval of a given fraction of the final result;
- **output time variation**: the measure of how regularly the partial results are made available to the user;
- **partial failure rate**: the number of partial results not delivered to the user;
- **total failure rate**: the number of failed assignments.

These characteristics have been derived from the experience with the typical applications in the EGEE Grid. In the *best-effort service* service such as EGEE Grid we expect to be able to optimize these characteristics and and to improve their predictability.

3 QoS Improvement Techniques on the Grid

In general the QoS guarantees are not implemented in large-scale, productions Grids. Several techniques which could potentially be used to implement the QoS, such as the job preemption (suspension of lower priority jobs) and job migration (such as Condor [24] system based on process migration [25]) are not universally deployed and may not be applicable to all applications as they may require linking with checkpointing-aware I/O libraries and impose other limitations on a checkpointable process.

Existing mechanisms, such as the shallow and deep retry count in JDL (Job Description Language), provide a last-resort handling of failure because they are not "application-aware" and may not be customized with sufficient flexibility. Therefore to improve the failure rate user communities do the laborious testing and debugging of the application environment on the Grid sites. It must be repeated each time a new version of the application software is installed or the changes in the underlying resources are applied.

To improve performance the administrative actions such as reconfiguration of batch queue priorities in certain sites or dedication of resources cannot be considered as a scalable solution. Such actions typically require lengthy negotiation process and eventually create policy conflicts between various user groups, including the non-Grid users of Grid sites. Such an approach may only be an option for a very powerful user community but nonetheless it cannot handle middleware limitations.

On the other hand the approaches which modify the middleware itself, such as improved scheduling based the Service Level Agreements (SLA) [5], advanced reservations [6] or special Grid QoS Management Services [7] promise to provide a Quality of Service guarantees built into the system. Unfortunately in a large

production Grid the evolution of the middleware is measured in the cycles of several months if not years and it is a complex administrative process. Therefore it is not realistic to expect that the technologies which are currently under development and in the proof-of-concept phase, will provide the stable and mature QoS support in the middleware deployed in a large scale in the nearest future. Additionally, the users do not plan ahead the interactive works so the concept of advanced reservations may not apply in such user scenarios.

Therefore an approach which exploits the existing infrastructure without system or administrative changes is investigated in the next section.

4 User Level Scheduling

In a classic Grid (and batch) approach, a decision to send a particular job to a particular resource is done at submission time or shortly after either by a user or an appropriate service, typically a resource broker. Additionally, there is no distinction between the physical job (the "slot" to perform the computations on some resource) and a logical task (the application-specific definition of the computation). A single job executes a single task. This association is defined before the job is run.

A late binding technique (also known as place-holders or pilot agents) defers the decision to execute a task to the job run time. The Grid job runs a generic agent which acquires the task from a scheduler. A single agent (so a single Grid job) may execute one or more tasks. When the task is done the agent may terminate and the Grid job terminates, release the worker node.

The late binding approach has been exploited in the generic contexts e.g. Condor glide-ins [14]. Application-specific implementations also exist. For example the data production systems of several High Energy Physics experiments at CERN, such as Alien [9] and DIRAC [10] use scheduling services to create permanent overlays above the existing Grid infrastructure. Such an approach allows to control the scheduling in a flexible and application-oriented way without modifying existing middleware. This has been proven very successful over last years and enabled to make and efficient and fault-tolerant use of early, not mature Grid infrastructures [26]. Such approach is suitable for large Virtual Organizations and coordinated activities such as data production in High Energy Physics. However this approach has also several limitations:

- it requires the maintenance of central services to manage the task queue;
- it uses special services in the sites (so-called VO-boxes which are dedicated machines under the VO control);
- it compromises the Grid accounting and traceability because the real users are mapped to generic VO users, sharing the Grid credentials.

User Level Scheduling is based on late job binding of jobs but removes the drawbacks of the permanent overlays. A scheduler is a service which is run by the user. The agents and a scheduler form a transient overlay which is destroyed when the processing is terminated. The system runs entirely in the user space,

it does not require special services and it does not compromise the accounting and traceability in the Grid.

User-level schedulers which are embedded in the applications such as medical image analysis [12], earthquake source determination [8] or bio-informatics [13] have been succesfully used, increasing the performance and reliability of the generic Grid infrastructure. The schedulers are typically implemented with the TCP/IP sockets or MPI. However it is impractical to redevelop a scheduler for every application. Therefore the more structured approaches have been investigated. For example AppLeS/APST [11] provides a framework for the parameter sweep applications and adaptive scheduling. The Condor M/W [15] provides an framework for master/worker applications.

In the next section we present a User Level Scheduler which, has been succesfully used as add-on to the world's largest Grid infrastructure – the EGEE Grid.

5 DIANE User Level Scheduling and QoS

DIstributed ANalysis Environment (DIANE) [16], [1],[17] is a User Level Scheduler developed at CERN . The Master/Worker overlay supports a broad class of typical of not-communication-intensive applications on the Grid. The scheduler performs several functions: adaptive scheduling (assigning the tasks to agents), task synchronization (selecting the tasks for execution) and failure recovery (analyzing the reason for task failure and deciding on the action). DIANE provides the software framework to plug-in modules which modify the behavior of the scheduler. The framework provides the software containers for the application components for flexible application integration: from black-box executables (e.g. image processing [2], telecommunications [18], physics simulation regression testing[19] and data analysis [22]) to interfacing the applications at the source-code level (e.g. bio-informatics [21] and medical physics [20]).

A typical execution trace of a DIANE application is shown in Fig. 4. Each cross represents the start time of a worker agent and shows the overhead of the Grid job submission. Each worker agent executes a series of tasks represented as horizontal line segments. If a failure occurs the tasks are automatically reassigned to another worker. The worker agents execute on heterogenous Grid nodes and their computing power may differ. If the task processing time is not equal the scheduler selects the appropriate workers at runtime to balance the load and improve the *turnaround time*. For example, in May and June 2006, CERN successfully supported a series of large-scale data-processing activities carried out by the International International Telecommunications Union (ITU) as part of the ITU's Regional Radiocommunication Conference. Several sites of the EGEE infrastructure provided a computing Grid of more than 400 nodes to work on each analysis in parallel, and the processing was conducted using the user scheduling layer [18]. The system completed more than 200 000 very-short frequency analysis jobs (so called frequency requirements clustered in around 40 000 processing tasks) in around one hour, proving that on-demand comput-

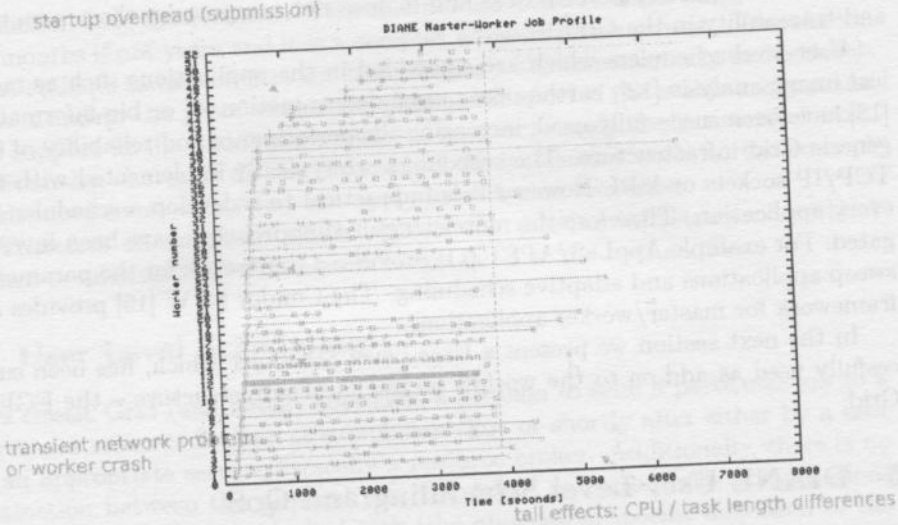

Fig. 4: Execution profile of a DIANE job illustrates the runtime behavior of individual worker agents and the progress of task processing.

ing with a short deadline is possible on the Grid (see Fig. 5). The maximum observed *partial failure rate* was not more than $3 * 10^{-4}$.

In 2006 DIANE was used to perform a sizeable fraction of an in silico drug discovery application using the EGEE and other Grid infrastructures [21]. The challenge was to analyze possible drug components against the avian flu virus H5N1. This activity showed that a user-level scheduler can improve the distribution efficiency on the Grid from below 40% to above 80% by optimizing the allocation of the fine-grained computing tasks. Efficient automatic-error recovery mechanisms proved to be efficient in extended periods of continuous work (30 days) with zero *total failure rate*.

[1] contains examples of other QoS characteristics, such as the *response latency* and *output time variation*.

6 Summary and Future Work

The DIANE User Level Scheduling system has been successfully applied in a number of applications and important activities. The system provided a very stable environment with low failure rates and considerable efficiency improvement. The lightwieght mode of operation in the user space is a key factor which enables quick porting of diverse applications to the Grid environment.

For more complete analysis of the Quality of Service we need to create a model to describe the interaction of the User Level Scheduler with the underlying Grid environment and to formally quantify the QoS metrics described in Section 2.

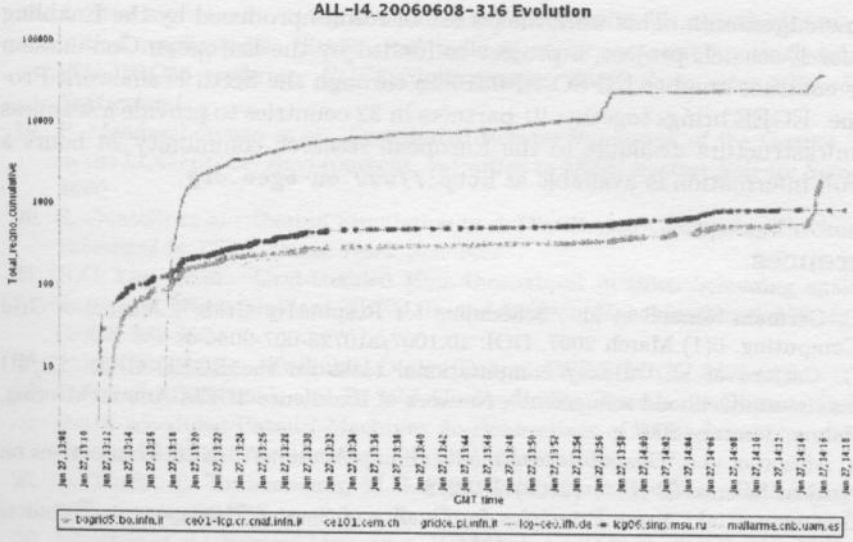

Fig. 5: The number of processed units of computation (frequency requirements) in the function of time in the ITU application. The plot shows the contribution of individual Grid sites.

The worker agent starting time is an important component of the overall efficiency (see Fig. 4). Ability to predict the queuing time of the Grid jobs would allow to give better predictions if the QoS requirements set by the user can be satisfied. Techniques which use order-based statistics (quantiles) rather than moment-based statistics (mean, standard deviation) such as Binomial Method Batch Predictor [23] have been used to predict the queuing time in the batch systems. A User Level Scheduler could take an advantage of such predictions, if they can be effective for the hierarchical scheduling in the Grid environment.

DIANE scheduler also makes it easy to introduce new use cases such as interactivity. Once the worker nodes are acquired a user may repeatedly give new assignments to the system with fast feedback.

In the current usage, the worker agents terminate as soon as the task queue in the master agent is empty. In an enhanced system, the workers could be shared between multiple masters which belong to the same user. It would allow to prioritise the user assignments and QoS control (within statistical bounds of the underlying Grid infrastructure).

Finally, to be largely accepted, the User Level Scheduling must be proven not to penalize ordinary Grid users by blocking the access to the resources. It also needs to be shown that mutual exclusion of large user level overlays may be efficiently avoided. An important challange is to do it in a way which does not compromize the lightwieght nature of User Level Scheduling, i.e. not relying on heavy, permanent services and infrastructure. It may require the negotiation algorithms between the user level overlays to be developed.

Acknowledgements. This work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless Grid infrastructure available to the European research community 24 hours a day. Full information is available at `http://www.eu-egee.org`.

References

1. C. Germain-Renaud et al: "Scheduling for Responsive Grids", Journal of Grid Computing, 6(1) March 2007, DOI: 10.1007/s10723-007-9086-4
2. G. Carrera et al: "Heavy computational tasks on the EGEE Grid: 2D/3D maximum-likelihood refinement", Network of Excellence 3DEM Annual Meeting, Palma, January 2007
3. J. Allison et al.: "Geant4 developments and applications", IEEE Transactions on Nuclear Science 53 No. 1 (2006) 270-278
4. Bochman et al.: Some Principles for Quality of Service Management, Technical Report, Universite de Montreal, 1996.
5. J. MacLaren et al.: "Towards service level agreement based scheduling on the Grid". In Procs 14th Int. Conf. on Automated Planning and Scheduling (ICAPS 04), 2004.
6. A. Roy et al.: GARA: A Uniform Quality of Service Architecture, International Series In Operations Research And Management Science, Issue 64, 2003
7. R.J. Al-Ali et al.: Analysis and Provision of QoS for Distributed Grid Applications, Journal of Grid Computing, Vol.2, 2004
8. D. Weissenbach et al.: Faster earthquake source mechanism determination with EGEE , 1st EGEE Conference, Geneva, 2006
9. AliEn–ALICE environment on the GRID, P. Saiz et al.: Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume 502, Issues 2-3, 21 April 2003, Pages 437-440
10. A. Tsaregorodtsev, V. Garonne, and I. Stokes-Rees. DIRAC: A Scalable Lightweight Architecture for High Throughput Computing. In *Procs 5th IEEE/ACM Int. Workshop on Grid Computing (GRID'04)*, 2004.
11. F. Berman et al.: Adaptive Computing on the Grid Using AppLeS, IEEE Transactions on Parallel and Distributed Systems, Vol. 14, No.4, April 2003
12. C. Germain-Renaud, R. Texier and A. Osorio. Interactive Reconstruction and Measurement on the Grid. *Methods of Information in Medecine*, 44(2):227- 232. 2005.
13. mpiBLAST: Open-Source Parallel BLAST, `http://www.mpiblast.org`
14. J. Frey, T. Tannenbaum, M. Livny, I. Foster and S. Tuecke: A Computation Management Agent for Multi-Institutional Grids, Cluster Computing Journal, Vol. 5, No 3, July, 2002, pages 237-246
15. G. Shao, R. Wolski, and F. Berman: Master/Slave Computing on the Grid; in Proceedings of the 9th Heterogeneous Computing Workshop, Cancun, Mexico, May 2000, pp. 3-16.
16. J.T. Mościcki: Distributed analysis environment for HEP and interdisciplinary applications; *Nuclear Instruments and Methods in Physics Research A 502 (2003) 426-429*
17. DIstributed ANalysis Environment, `http://cern.ch/diane`

18. A. Manara et al.: Integration of new communities in the Grid for mission critical applications: distrib uted radio-requency compatibility analysis for the ITU RRC06 conferece; *EGEE'06 Conference, 25-29 Septe mber 2006, Geneva, Switzerland*
19. P. Mendez-Lorenzo et al.: Distributed Release Validation of the Geant4 Toolkit in the LCG/EGEE Envi ronment; *submitted to IEEE Nuclear Science Symposium 2006*
20. S. Guatelli et al.: Geant4 Simulation in A Distributed Computing Environment; *submitted to IEEE Trans. Nucl. Sci. 2006*
21. H.C. Lee, et al.: Grid-enabled High-throughput in silico Screening against influenza A Neuraminidase, IEEE Transaction on Nanobioscience, Vol. 5, No. 4 (2006) 288-295
22. Atlas Computing – Technical Design Report *CERN-LHCC-2005-022.*
23. J. Brevik, D. Nurmi, and R. Wolski: Predicting Bounds on Queuing Delay for Batch-scheduled Parallel Machines. In Proceedings of ACM Principles and Practices of Parallel Programming (PPoPP), March 2006.
24. D. Thain, T. Tannenbaum, M. Livny: Condor and the Grid, Grid Computing, 2003, pages 299-335
25. Milojicic et al.: Process Migration, HP Laboratories Technical Report HPL, 1999, Part 21, ISSN 1368-6798
26. I. Stokes-Rees et al.: Developing LHCb Grid software: experiences and advances, Concurrency and Computation: Practice and Experience, 2007, Vol. 19, No. 2, pages 133-152