Benchmarking of Integrated OGSA-BES with the Grid Middleware

Fredrik Hedman[1], Morris Riedel[2], Phillip Mucci[1], Gilbert Netzer[1], Ali Gholami[1], M. Shahbaz Memon[2], A. Shiraz Memon[2], and Zeeshan A. Shah[1]

[1] Center for Parallel Computers (PDC), Kungliga Tekniska Högskolan (KTH), SE-100 44 Stockholm, Sweden

[2] Jülich Supercomputing Centre, Forschungszentrum Jülich (FZJ), Leo-Brandt-Str. 1, Jülich, 52425, Germany

Abstract. This paper evaluates the performance of the emerging OGF standard OGSA - Basic Execution Service (BES) on three fundamentally different Grid middleware platforms: UNICORE 5/6, Globus Toolkit 4 and gLite. The particular focus within this paper is on the OGSA-BES implementation of UNICORE 6. A comparison is made with baseline measurements, for UNICORE 6 and Globus Toolkit 4, using the legacy job submission interfaces. Our results show that the BES components are comparable in performance to existing legacy interfaces. We also have a strong indication that other factors, attributable to the supporting infrastructure, have a bigger impact on performance than BES components.

Keywords: Performance analysis, Grid middleware, UNICORE, gLite, Globus Toolkit, OGSA-BES.

1 Introduction

Today's large-scale scientific research is supported by Grid and e-science infrastructures. Grids are composed of a set of heterogeneous resources that are managed by several interacting software components accessible as Grid services [1]. These infrastructures are increasingly often accessed via different flavors of Grid middleware technologies. Several different Grid infrastructures exist today. Some are primarily focused on maximized throughput, like EGEE, OSG and NGS; others are primarily driven by high-performance computing (HPC) needs, like DEISA and TeraGrid. With emerging Grid infrastructure interoperability, it is now becoming possible and realistic to combine these different types of resources for major scientific research areas that demand both high throughput and HPC to make progress [2]. In these scenarios, Grid middleware performance becomes important to measure and understand.

Recently, many Grid middleware technologies have been augmented with implementations of proposed recommendations from the Open Grid Forum (OGF). An example is the OGSA Basic Execution Service (BES) [3] for job management and submission. Adding BES to a middleware is motivated by the gain in interoperability between different Grids [4], and by the increase in performance and handling of jobs between these middleware technologies.

Performance analysis and benchmarking of Grid technologies in general, and Grid middleware in particular, is still an emerging area. Some results covering different approaches and tools can be found in [5,6,7].

In contrast to these approaches, we presented in [8] a "black-box" approach for analyzing Grid middleware, using a straightforward and non-invasive platform independent method. In this paper, we build on our previous work to assess the performance of the recently finalized BES implementations for three[1] different Grid middleware stacks: gLite, Globus Toolkit 4 and UNICORE 6. Performance analysis of the UNICORE implementation [9] becomes specifically important since it has been deployed on several DEISA sites for evaluations.

This paper is structured as follows. Section 2 gives a background and illustrates the design of our approach for BES. Section 3 provides implementation details of the benchmarked BES implementations. We present results from our performance measurements and provide insights by evaluating them with respect to the core technologies and job management handling within the different Grid middlewares in Section 4. Finally a conclusion is presented in Section 5.

2 Black-Box Benchmarks of Grid Middleware

From a user perspective Grid middleware adds a certain overhead compared to local execution. Independent of the size of the submitted job, fixed costs contribute to the overhead of the Grid system. It is important to be able to diagnose and address this overhead early in the design cycle of a new component. Grid developers can use this information for evaluating design and implementation trade-offs while Grid users can try to amortize the overhead by submitting fewer longer running jobs.

To analyze the performance of Grid middleware rather than just measuring the time an application spends on a resource, we have demonstrated a method to measure the time spent on the "grid work" per job in [8]. In this paper we build on our previous work and investigate the performance of recently developed alternative job submission components in different Grid middlewares. We assume that a component can be treated as a black box into which jobs are submitted and from which results are returned. The results obtained can be compared with our baseline measurements for consistency and rough estimates of the overhead introduced by the new components.

Features and capabilities provided by the investigated job submission interfaces vary. Thus we implemented three different variants of our basic benchmark. For the WS-GRAM baseline measurements we used the `globusrun-ws` command line client. It can submit and monitor one job at a time. Our implementation uses 10 parallel threads that first submit the required number of jobs and poll for their completion in a round-robin fashion. CondorG and CLIQ offer bulk submission capability: a single job submission followed by local polling is used. Finally

[1] Our work was done in the context of the OMII-Europe project which focused on these three Grid middleware platforms.

for the new BES and UNICORE 6 mechanisms we use a serial implementation, submitting and monitoring one job at a time.

To provide a basis for reproducible and repeatable results, it is necessary to create identical conditions when performing comparisons of Grid middleware. Our experience is that this can be very timeconsuming and in some cases not feasible. We claim a best effort approach can still give some comparable performance measurements across different Grid middleware.

3 OGSA-BES Implementations

BES provides easy, intuitive and standardized access to computational resources. The OMII-Europe project provided BES implementations for the three middleware stacks which we benchmarked.

The BES specification [3] defines two mandatory (`BESFactory` and `BESManagement`) and one optional (`BESActivity`) interface. The management interface allows to control the service itself. The factory interface provides job submission and bulk monitoring capabilities while the activity interface allows monitoring of a single job. Jobs submitted to the BES interface have to be described in the Job Submission and Description Language (JSDL) [10].

3.1 OGSA-BES Implementation of UNICORE 6

At the time of writing, UNICORE 6 offers two WS-based interfaces for job submission and management: BES and UNICORE Atomic Service (UAS) [9]. Both interfaces allow for job management and control functions using WS messages. Both interfaces are adopted within the middleware and deployed on top of resource management systems (RMS) (e.g. Torque, LoadLeveler) that in turn deal with job scheduling on computational resources.

Since BES allows for a flexible implementation strategy, the UNICORE developers have been able to directly use the interfaces of the execution back-end of UNICORE 6, which is the enhanced Network Job Supervisor (XNJS) [11]. The implementation consists of the mandatory `BESFactory` and `BESManagement` interfaces as well as the optional `BESActivity`. In comparison to BES, the UAS is a suite that additionally provides storage management and file transfer mechanisms that have been leveraged by the BES implementation. UAS also accepts jobs described in JSDL.

3.2 OGSA-BES Implementation of gLite

The integration of the BES interface into the gLite middleware was accomplished by extending the CREAM computing element which provides both the back-end core and the legacy interface. The new interface is implemented as a separate plugin on top of CREAM core. Both interfaces can share the same core so that jobs can be submitted to the same resource via both interfaces.

3.3 OGSA-BES Implementation of Globus

The BES service for the Globus toolkit is implemented as a thin wrapper layer in front of the WS-GRAM job submission service. The legacy WS-GRAM interface is also available to support the existing Globus infrastructure.

The BES service itself is stateless with the exception of a single flag that allows to stop job submission via the management interface. Each incoming request is translated into corresponding WS-GRAM requests, which involves a translation of the jobs description from JSDL to RSL needed by the WS-GRAM. This adds some overhead to each BES call but allows for a transparent deployment of BES interfaces into existing Globus infrastructures, since the BES service simply behaves like a WS-GRAM client and can even be in a completely different organizational domain.

4 Benchmark Results

Using the "black-box" approach described in [8] we have collected baseline performance data for a number of Grid middlewares. We use the baseline data to assess the performance of the new BES components.

4.1 Baseline Results

The computer platform used consisted of nodes with a 2.8 GHz Intel Pentium D CPU and 2 GB RAM connected through gigabit Ethernet and running Scientific GNU/Linux version 4 operating system. Even though the test-bed was a controlled environment with only local network traffic and consistent operating system and middleware installations, big variations between different runs were detected. This underlines the importance of a carefully controlled environment to arrive at repeatable and comparable experiments for quantitative analysis.

Figures 1 and 2 show a selection of our results from a number of benchmark runs for Globus Toolkit 4 and UNICORE 5/6 in our test-bed. Keeping the large variation in mind a number of qualitative conclusions can be drawn. For Globus Toolkit middleware we conclude:

- Submission via the CondorG interface yields best results for the workload of the benchmark. This is also the mechanism that is recommended by the Globus Alliance for the case of submitting a large number of jobs.
- Data staging has by far the biggest influence on job submission performance. The fastest runs occur for all tested submission mechanisms when staging of input and output is not used.
- The results for the WS-GRAM submission tests show that considerable stress is put on both the middleware and the submission script. For instance, the two runs with the longest run time in Fig. 1 show that about half of the jobs finished almost simultaneously which is most likely an artifact of the polling method used by the benchmark driver to detect job completion. The polling attempts load both the middleware and the benchmark driver and this load is highest in the beginning of the run when there are many unfinished jobs that need to be checked.

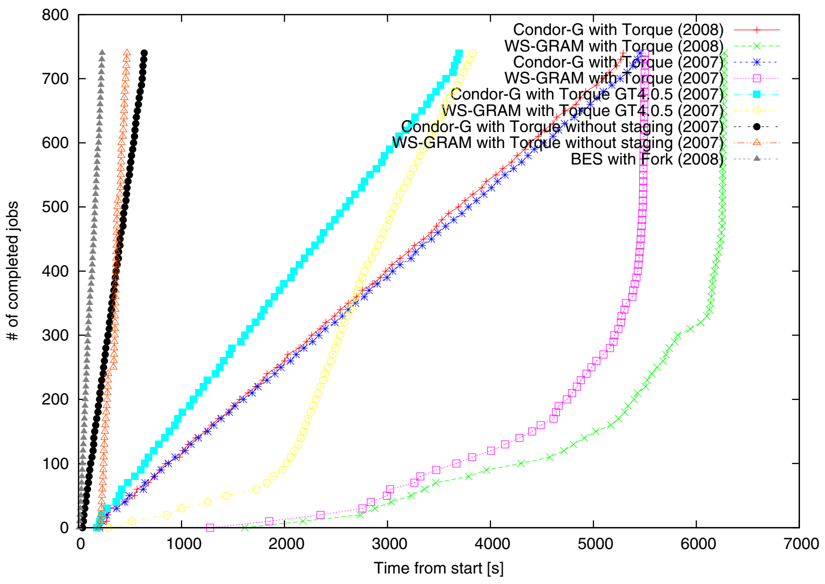

Fig. 1. Selected system level performance runs for the Globus job submission components. The data sets show that different job submission mechanisms (WS-GRAM, CondorG, BES) show different characteristics, but also that staging effort dominates over all other issues. BES does not do any staging and is shown for reference.

Fig. 2 shows the results from the evaluation of the system level performance for the UNICORE 6 middleware stack, including comparisons with UNICORE 5:

– The UNICORE 5 results show consistent behavior for both Torque and Fork local resource management. In this cases we used the batch submission capability provided by the CLIQ client.
– The results show also that the scheduling policy of the local resource manager (batch system) can have an impact on the total performance. This is exemplified by the results obtained when using the MAUI scheduler for the Torque batch system.
– The UNICORE 6 system level performance shows considerable spread in contrast to the UNICORE 5 results. Part of this could be attributed to the fact that different versions of the middleware were used to make the measurements. However the large spread between the Fork and Torque performance points more to the sensitivity of the UNICORE 6 benchmarks towards small changes in the benchmarking environment. This is probably caused by the fact that the current benchmark tools are restricted to serial submission when targeting UNICORE 6 which puts more emphasis on the latency of a single submission.
– In contrast to the Globus Toolkit, no surge of jobs completing can be seen. This leads to the conclusion that the selected benchmark method is well

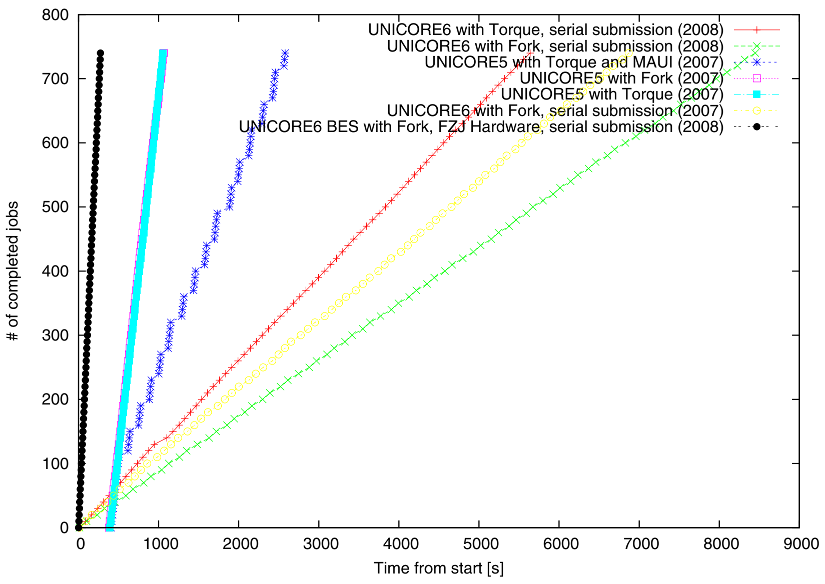

Fig. 2. System level performances for the UNICORE middleware stack. The data-sets shown are both for version 5 and 6 of UNICORE. As a reference the results for the BES submission are also given.

adopted towards the UNICORE middleware stack thus not exhibiting the polling problems of the Globus tools.

4.2 BES Components Performance

With the availability of the BES enhanced job submission components from the OMII-Europe [12] project, we have performed a first evaluation of the performance of these components. Since these components are handling job submission we compare them against the system level performance evaluations. It is however essential to bear in mind that no firm comparisons can be made because of differences in the platforms and setups that were used. A number of adoptions had to be made in order to conduct the performance evaluation at this early stage of deployment of the BES components. In particular the following differences to the system level evaluation are present:

- Because of the limited time we used the test services provided by developers of the respective BES components. This means that the hardware and software setup differs significantly. This is most apparent in the case of the gLite test service which uses a complete batch system and computing element (the CREAM-CE) for its backend compared to the simple process forking backends used by the Globus Toolkit and UNICORE services. The specification of hardware and software for BES endpoints used are:

CREAM/BES. The `omiivm03.cnaf.infn.it` host is deployed as a virtual machine on the omii005 host. This machine has 2 Intel(R) Xeon(R) 2.00GHz CPUs and 4 GB RAM. The machine hosts four virtual machines each with an equal share of the available resources of which `omiivm03` is one.

GT/BES. The GT/BES server is hosted on `romana.pdc.kth.se` and has a Intel Dual Core 2.13 GHz CPU with 1 GB memory running Gentoo GNU/Linux 2.6.

UNICORE/BES. The BES server hosted on `zam461.zam.kfa-juelich.de` has a Intel Dual Xeon 3 GHz CPU with 2 GB memory running SuSE GNU/Linux 9.3.

Benchmark client. An Intel Dual Core CPU 2.13 GHz running Gentoo 2.6 machine targeting different endpoint. The cost of a ping to the endpoints were approximately 30 ms.

– No attempts at handling input or output data (data staging) for the job were made.
– The current BES benchmark does not submit jobs in parallel but processes them in a serial fashion.

In total, the BES benchmark is considerably simpler in nature than the earlier system level benchmarks. Despite this fact, we found it helpful in uncovering a few interesting facts about the behavior of the OMII-Europe BES implementations:

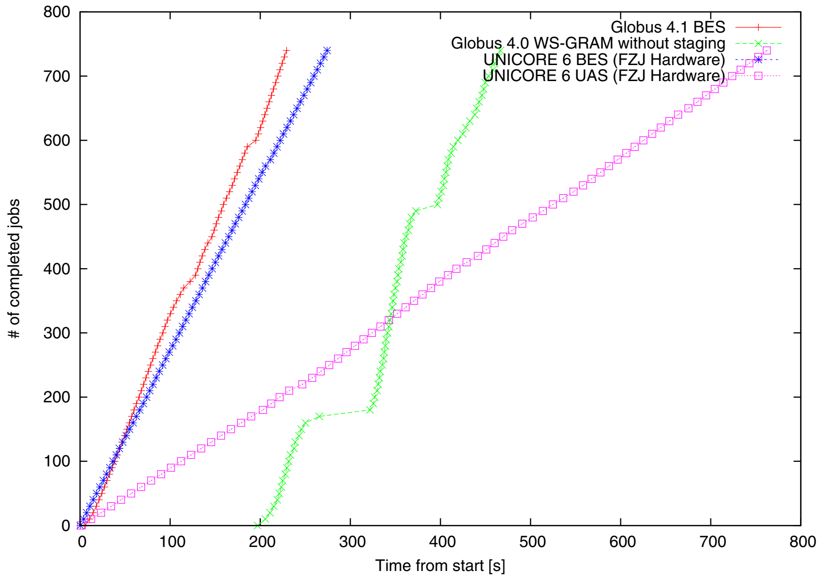

Fig. 3. Performance of Globus and UNICORE 6 BES implementations compared to the UNICORE 6 component performance for the legacy interface and the system level performance of Globus legacy WS-GRAM without any staging. The figure shows the number of jobs that have been completed versus the time from the start of the experiment.

- We were able to make a close comparison between the UNICORE 6 BES adoption and the legacy interface (i.e. UAS) using the same method. The legacy system seemed to exhibit 3 times higher latencies than the BES system (Fig. 3). The benchmark that we used however emphasizes the latency component because of the serial nature of the submission. Preliminary investigations show that this was caused by a polling rate limitation in the UAS client.
- Despite the differences in hardware and web services implementation, the performance of the UNICORE and Globus Toolkit implementations is about the same and comparable to the performance of the legacy Globus WS-GRAM when no data staging is done. This can be seen in Fig 3. The increased delay until completion of the first job in the system level script is caused by the concurrent submission of all 750 jobs in the beginning of the benchmark run.
- The impact of staging and logging is far bigger than any overhead introduced by the BES services. This becomes apparent in Fig 4 where the gLite BES implementation is compared to the legacy Globus WS-GRAM performance with and without staging. The inclusion of input/output staging into the WS-GRAM submission increased total execution times by a factor of roughly 6. Also the job completion rate in the early stages of the WS-GRAM experiment match the rate from the gLite BES. This is a strong indication that the performance of the gLite service is comparable to the other BES implementations with a more realistic backend and job setup where input and output

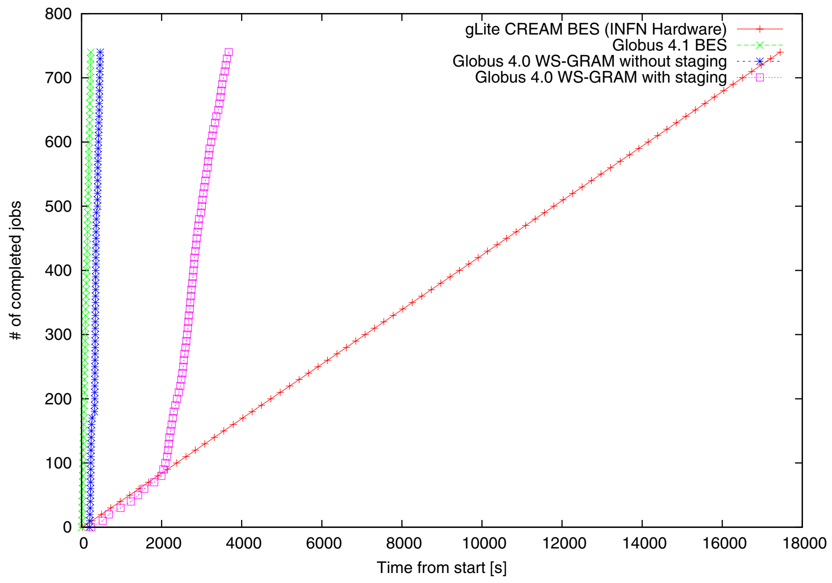

Fig. 4. Performance of the gLite BES implementation compared to the Globus BES implementation and Globus legacy WS-GRAM system level performance with and without staging. It is important to notice that the gLite CREAM BES uses a full batch system backend.

would be transferred to the user and/or permanent storage. Also, the lack of concurrency may slow down the gLite BES since logging phases cannot overlap with execution of the next job.

To sum up, the simple evaluation of the performance of the BES services does show that the performance of the BES services should be acceptable compared to the performance of the legacy job submission mechanisms. However, further targeted investigation in carefully controlled environments would be necessary to conclusively asses the performance of these new mechanisms.

5 Conclusion

A necessary prerequisite to benchmark a piece of software is the ability to execute this software under controlled and well known conditions. In fact, the BES components are dependent on local resource management systems to perform the actual job execution. As the presented "black-box" results suggest, configuration and services, for example handling of input and output data or logging, carried out by these "back-end" infrastructure have a large impact on the measured performance of the component. We tried to resolve this issue by estimating the effect of the back-end system onto the component performance by using alternate legacy components that utilized the same back-end infrastructure and use them as a baseline for relative comparison. We used this approach for the BES job submission benchmark provided in this paper.

In summary, our results show that the performance of the BES components that were evaluated are comparable to existing legacy solutions. Different security setups of the components may also lead to different performance, but in this paper we clearly consider them out-of-scope. The "black-box" experiments strongly indicate that other factors attributable to the supporting infrastructure have a bigger impact on performance than the use of BES components.

Acknowledgment

This work is supported by the European Commission through the OMII-Europe project-INFSO-RI-031844. Further information see [12].

References

1. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Elsevier, Amsterdam (2004)
2. Riedel, M., et al.: Improving e-Science with Interoperability of the e-Infrastructure EGEE and DEISA. In: 31st International Convention MIPRO, Conference on Grid and Visualization Systems (GVS), Opatija, Croatia (May 2008) (accepted)
3. Foster, I., Grimshaw, A., Lane, P., Lee, W., Morgan, M., Newhouse, S., Pickles, S., Pulsipher, D., Smith, C., Theimer, M.: OGSA Basic Execution Service Version 1.0. Technical report, Open Grid Forum (2007), `http://www.ogf.org/documents/GFD.108.pdf`

4. Marzolla, M., Andreetto, P., Venturi, V., Ferraro, A., Memon, A., Memon, M., Twedell, B., Riedel, M., Mallmann, D., Streit, A., van de Berghe, S., Li, V., Snelling, D., Stamou, K., Shah, Z., Hedman, F.: Open Standards-based Interoperability of Job Submission and Management Interfaces across the Grid Middleware Platforms gLite and UNICORE. In: Proceedings of International Interoperability and Interoperation Workshop (IGIIW) 2007 at 3rd IEEE International Conference on e-Science and Grid Computing, Bangalore, India, pp. 592–599. IEEE Computer Society, Los Alamitos (2007)
5. Dikaiakos, M.: Grid benchmarking: Vision, challenges, and current status. Concurrency and Computation: Practice and Experience 19(1), 89–105 (2007)
6. Nemeth, Z., Gombas, G., Balaton, Z.: Performance evaluation on grids: Directions, issues and open problems. In: 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2004), p. 290 (2004)
7. Snavely, A., Chun, G., Casanova, H., der Wijngaart, R.F.V., Frumkin, M.A.: Benchmarks for grid computing: a review of ongoing efforts and future directions. SIGMETRICS Perform. Eval. Rev. 30(4), 27–32 (2003)
8. Alexius, P., Elahi, B.M., Hedman, F., Mucci, P., Netzer, G., Shah, Z.A.: A Black-Box Approach to Performance Analysis of Grid Middleware. In: Bougé, L., Forsell, M., Träff, J.L., Streit, A., Ziegler, W., Alexander, M., Childs, S. (eds.) Euro-Par Workshops 2007. LNCS, vol. 4854, pp. 62–71. Springer, Heidelberg (2008), `http://dx.doi.org/10.1007/978-3-540-78474-6_10`
9. Riedel, M., Schuller, B., Mallmann, D., Menday, R., Streit, A., Tweddell, B., Memon, M., Memon, A., Demuth, B., Lippert, T., Snelling, D., van den Berghe, S., Li, V., Drescher, M., Geiger, A., Ohme, G., Benedyczak, K., Bala, P., Ratering, R., Lukichev, A.: Web Services Interfaces and Open Standards Integration into the European UNICORE 6 Grid Middleware. In: Proceedings of 2007 Middleware for Web Services (MWS 2007) Workshop at 11th International IEEE EDOC Conference The Enterprise Computing Conference. IEEE Computer Society, Los Alamitos (2007)
10. Anjomshoaa, A., et al.: Job Submission Description Language (JSDL) - Specification Version 1.0. Technical report, Open Grid Forum (2005), `http://www.ogf.org/documents/GFD.56.pdf`
11. Schuller, B., Menday, R., Streit, A.: A versatile execution management system for next-generation UNICORE grids. In: Lehner, W., Meyer, N., Streit, A., Stewart, C. (eds.) Euro-Par Workshops 2006. LNCS, vol. 4375, pp. 195–204. Springer, Heidelberg (2007)
12. Open Middleware Infrastructure Institute for Europe. Project no: RI031844–OMII-Europe, `http://omii-europe.org`