Influences between Performance Based Scheduling and Service Level Agreements

Antonella Galizia[1], Alfonso Quarati[1], Michael Schiffers[2,4], and Mark Yampolskiy[3,4]

[1] Institute for Applied Mathematics and Information Technologies, National Research Council of Italy, Genoa, Italy
{antonella.galizia,alfonso.quarati}@ge.imati.cnr.it
[2] Ludwig-Maximilians-Universität München, Germany
schiffer@nm.ifi.lmu.de
[3] Leibniz Supercomputing Centre, Garching, Germany
Mark.Yampolskiy@lrz.de
[4] Munich Network Management (MNM) Team

Abstract. The allocation of resources to jobs running on e-Science infrastructures is a key issue for scientific communities. In order to provide a better efficiency of computational jobs we propose an SLA-aware architecture. The core of this architecture is a scheduler relying on resource performance information. For performance characterization we propose a two-level benchmark that includes tests corresponding to specific e-Science applications. In order to evaluate the proposal we present simulation results for the proposed architecture.

Keywords: resource allocation, benchmarks, scheduling, SLA.

1 Introduction

A proper resource-to-job matching is of paramount importance for a better exploitation of e-Science environments where heterogeneous resources are shared for coordinated problem solving in multi-institutional virtual organizations [1]. In addition, specific requirements are often associated with compute intensive scientific jobs, e.g., weather prediction WRF[1], or molecular dynamics GROMACS[2], which may lead to further efficiency issues. In such computation intensive applications, a better resources-to-job matching can lead to significant improvements in the computation speed [2]. A performance aware job execution can be realized if there is adequate information available regarding the resource capabilities and the qualities of the services provided over the resources. A generally accepted method to evaluate and compare the performance of computer platforms is through benchmarking and benchmarks based metrics [3] [4].

[1] http://www.wrf-model.org/
[2] http://www.gromacs.org/

It is common practice to express service quality expectations in Service Level Agreements (SLA). SLAs are negotiated between customers of a service and service providers. This practice has proven to be an effective means not only for enforcing providers to the desired quality but also to reorganize the complete service provisioning in order to use available resources more efficiently. In this context is the optimal exploitation and semantics definition of supported quality ranks, e.g., gold, silver, bronze, still an unsolved problem.

We focus here on performance as a single quality parameter. In our research we consider SLAs as a description of performance objectives to be achieved and maintained during the job execution. The main idea is to apply the *congruent policy,* where resources are characterized by considering several performance ranks and jobs are allocated to the most suitable resource according to the performance rank specified for in the their submission. To enable the description of both jobs and resources, a proposal for Grid environments has been presented [5].

In this paper, we abstain from discussions about SLA negotiation and how parameters can be specified in an SLA or a Service Level Specification (SLS). Instead, in Section 2 we propose an SLA-aware architecture incorporating a novel scheduling mechanism which takes into account fine grained knowledge about resource capabilities, information about job preferences, knowledge about the load of involved resources, and requirements specified in the SLA. In Section 3 we present the benchmarks used to rank resources with respect to specific metrics. In Section 4 we simulate the behavior of the proposed job allocation policy based on performance aware SLAs. In Section 5 we conclude the paper and discuss future plans.

2 An SLA Aware Job Allocation Architecture

A Service Level Agreement (SLA) is a contract between customers of a service and its provider. This contract specifies all service related commitments, i.e., with which quality the particular service will be provided to the customer and how this quality can be measured in order to verify the fulfillment of the contract. In some cases SLAs also specify penalties which will be due in case the committed service quality cannot be achieved. Further, since the quality parameters committed to the customer cannot always be measured directly on the infrastructure, the provider usually associates an SLS with an SLA. The purpose of an SLS is to specify how the provider's infrastructure is monitored and how the monitored parameters are used in order to calculate quality parameters committed to the customer.

In this paper we do not discuss the SLA negotiation process and issues related to the specification of parameters SLAs or SLSs. Instead, we are interested in architectural considerations necessary for predicting a job's quality and for scheduling of jobs to resources the performance of which is sufficient for the fulfillment of commitments. In this work, we consider SLAs as a source for the end users specific requirements which should be fulfilled. For instance, a user could specify in SLA that his submitted application should be scheduled to be executed in the next half hour and the job processing should not take longer than two hours.

Figure 1 shows the general principle of job submissions in the context addressed here (see also [1]). The job submitted by a customer/user is placed in the queue of a global scheduler. The main goal of the global scheduler is to decide on which infrastructure component this job should be computed. As of now, this is often done taking into account only the current filling state of local queues of all available resources and the very coarse grained classification of these resources, e.g., CPU- or GPU-based computation unit. After the decision is taken, the job is moved from the global queue to the local queue of the selected computation unit.

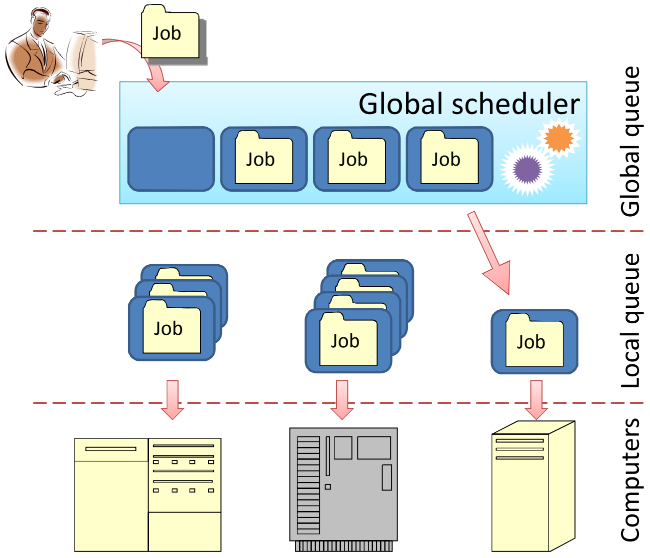

Fig. 1. Two-layer job scheduling

In order to better support performance-aware SLA requirements, we see the necessity to extend this model significantly. This is in particularly useful in Grid environments where most of the existing meta-schedulers, as Maui/Moab scheduling suite [6], Condor-G [7], and GridWay [8], mainly focus on resource requirements, queue policies, and average load. By the way, we argue that for this purpose the global scheduler should incorporate two complex components: 1) a fine grained analysis of the performance of the available resources based on an evaluation of different (artificial) computational tasks; and 2) a scheduling mechanism which takes into account fine grained knowledge about resource capabilities, information about job preferences, knowledge about the load of involved resources, and requirements specified in SLAs.

We propose using benchmarks as an approved and broadly accepted technique for such a fine grained assessment of resource qualities. Figure 2.a) outlines this strategy. A set of well-prepared benchmarks can be defined in advance and stored as a part of this unit. Generally, two benchmark scheduling strategies can be used. First, benchmarks can be scheduled event-based, e.g., if some hardware/software change events were encountered. However, this will require either a notification system or the benchmarks must be started manually. An alternative strategy is to start the benchmarks periodically. This eliminates the necessity of an event messaging system, but it bears the risk of possible interferences with productive jobs. Therefore, this strategy is often combined with additionally defined policies, e.g., to schedule benchmarks only in the case of empty local queues. For our work, both approaches could be adopted and we abstain from recommendations and further discussions of this topic.

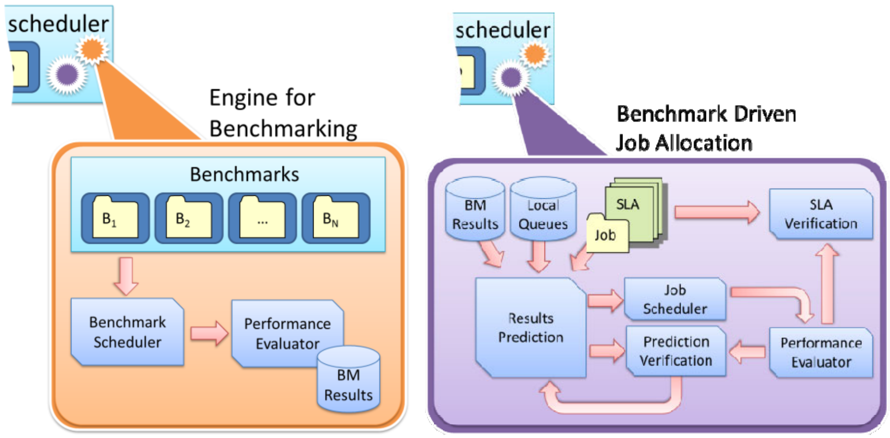

Fig. 2. – a) Fine grained resources evaluation – b) Benchmark driven job allocation

The extended scheduling engine is outlined in Figure 2.b). The result prediction component is the core of the engine. In the first place, it takes into account the information about fine grained resource performance, the states of the local queues, and the job description. During submission phase, job requests have to specify; the job description should include a specification to which class of computations this particular job belongs. This information is needed in order to perform a better match with the benchmark tests used for the resource ranking. Based on the information and scheduling policies the device for executing the job is selected. After the job is scheduled the performance evaluator component is in charge of qualitatively monitoring the job execution. This information can be used for the verification of performance goals as stated in SLA. Further, the evaluation of the job execution performance – together with the previous predictions – should be used in the prediction verification component. The purpose of this component is to determine the deviation of the results from their predictions. The deviation in turn can be used in the result prediction component to reduce the prediction error before signing any SLA.

Therefore, in order to fulfill the end-user requirements specified in SLA, it is necessary to take into account two main information: estimated execution time at different available resources and the estimated waiting time of the related queues. For both estimations we consider the results provided by the prediction component, which in turn is based on use of benchmarks.

The remainder of this paper focuses on the benchmark part of the proposed architecture, the core components of the proposed architecture as depicted in Figure 2b. In order to explain the principles of the component we abstain from a discussion of the job allocation in its full extend. Instead, we simulate the benchmark driven job allocation without the feedback loop including prediction and verification components.

3 Benchmarks Characterizing Resource Performance

The rank of resources on a performance basis may be obtained by expanding the description of computational resources with some indicator that characterizes their reaction under different workloads, [5].

To this aim, we integrate two complementary approaches: 1) the use of micro-benchmarks, to supply basic information derived from low-level performance metrics; 2) the exploitation of application-driven benchmarks, to get a closer insight into the behaviour of resources for a class of applications under more realistic conditions. In particular, we considered the following tools for micro-benchmarks: I) Flops [9] returns *Million of Floating-point Operations Per Second* (MFLOPS) to measure CPU performance, II) STREAM [10] and CacheBench [11] measure the bandwidth required for writing and reading operations, expressed as *Bytes per second,* to evaluate respectively main memory and cache, III) MPPTest [12] measures the *Latency and Bandwidth* to evaluate machine's interconnection, and IV) b_eff_io [13] returns *Bandwidth* to estimate I/O systems. These metrics are well established and generally used to evaluate resource performance capacities; moreover we use freely available tools that could be widely deployed and run [14]. Application-driven benchmarks are more suitable to mimic the real job workload because of their proximity with the application at hand. In the following we consider, as case studies, two applications of our interest, i.e., linear algebra and isosurface extraction. For the first class of applications, we selected the well-known High Performance Linpack (HPL) benchmark [15]. For the second, we realised a lightweight version of the application [16], characterized by a reduced computational cost, but still capable to maintain a representative run of the real application (ISO). A deep discussion about the definition and effectiveness of a two-level benchmark methodology has been presented in [17].

4 Evaluation of a Performance-Based Job Allocation

To evaluate the effectiveness of our architecture we simulated the job allocation policy based on performance SLAs and supported by benchmark results. We considered different application scheduling scenarios to appreciate the actual impact on SLA commitments. In particular, we compared the performance-based SLAs, i.e., taking into account the congruent policy, with a general global scheduler, depicted in

Figure 1. It is reasonable to base the job allocation strategy on the classical round-robin procedure. We further considered the rank of the resources based on an established application benchmark, i.e., ISO and HPL ranks.

To test the two components added to the global scheduler, we collected performance values of five resources under our domain/access, considering both level of benchmarks. To simulate the chosen scenarios and to compare the scheduling strategies we employed the Java Modelling Tools [18], an open source tool for performance evaluation and workload characterization of computer and communication systems based on queuing networks. In the reminder of this section, we present the resources and experimental results. Please note that in order to focus on the evaluation of the overall concept we simplify the job allocation component via removing the feedback loop consisting of prediction and verification components.

4.1 Characterizing the Test Bed

We collected the performance information of five resources under our domain/access. The aim is to consider different architectures to test the effectiveness of the first component added to the global scheduler, i.e., the fine grained analysis of the performance, and the improvement we achieved because of the second component, i.e., the benchmark driven job allocation. Resources are described in Table 1; it actually highlights the architectural heterogeneity of our test bed, especially regarding the computing power (number of CPUs), the type of interconnection and the memory size.

Table 1. Test bed infrastructure

	Proc. Type	N° Core	Network	RAM
Ibm	2 Quad Core Xeon 2.5 GHz	32	Infiniband	64 GB
michelangelo	2 AMD Opteron 275 2,2GHz dual core	64	Gigabit Ethernet	424 GB
SC1458	Proprietary	372	proprietary	1.9 TB
Paperoga	dual 3 GHz Intel Xeon	8	Gigabit Ethernet	16 GB
Cluster1	2.66 GHz Pentium IV	16	Gigabit Ethernet	16 GB

The double-level benchmark was run to gain a precise description of the actual performance offered by the computational systems along different metrics axes. Figures 3 and 4 depict the performance values of the respective micro and application benchmarks, we briefly discuss them in the following.

As Figure 3 outlines, the resources provide different performances with respect to the considered benchmarks. For example, *SC1458* achieves almost the best ranks for the aggregated values and interconnection performance but performs poorly considering the ranks of the single cores. For the benchmarks *michelangelo* and *ibm* performs better.

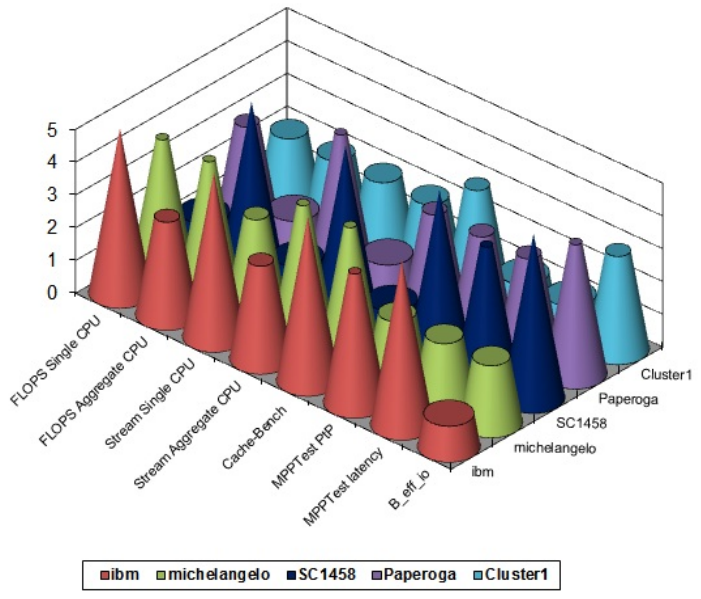

Fig. 3. Ranking of resources based on micro-benchmarks

Figure 4 reports the relative performance of ISO and HPL, each resource is tagged with a value in the range [1,…,5], where greater values correspond to worse performance (e.g., *ibm* and *SC1458* rank first according to ISO and HPL respectively). The ranking was based on the execution Wall Clock Time (WCT).

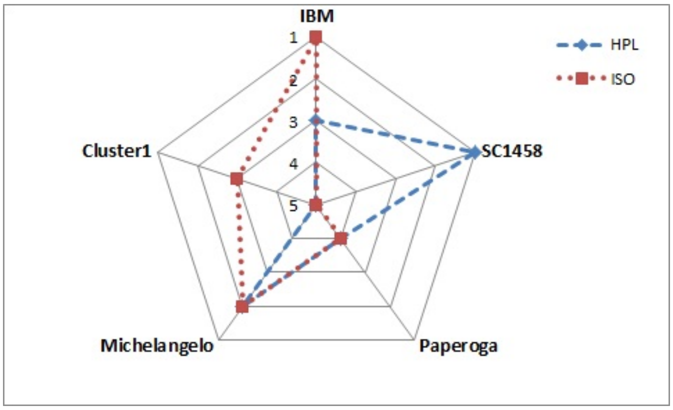

Fig. 4. Test bed ranking according to HPL and ISO benchmarks

Figures 3 and 4 show that, as expected, none of the resources is the best in all cases, therefore the importance of an accurately designed performance-aware scheduling of the jobs is essential for fulfilling the SLA.

4.2 Simulating the Architecture

In order to the compare the performance of a fine grained description of available resources regarding different computation tasks, and information about job preferences, we model our systems as a queuing network composed of 5 nodes, corresponding to our heterogeneous test bed, plus a scheduler which dispatches arriving jobs to the resources. In the global scheduler depicted in Figure 1, different scheduling strategies can be used, e.g., a round robin job allocation. However, for the performance-based SLA architecture we favor the usage of the Congruent Policy job allocation, which takes into account the appropriate resource properties. Moreover, we considered two more job allocation strategies based on information derived using the established ISO and HPL benchmarks respectively. Our objective is to minimize the Response Time of the system, that takes into account the time that a job takes to be executed (service time) plus the time spent in queue (waiting to be executed).

In the simulation we considered a workload composed of two parallel applications (linear algebra and isosurface extraction) that have been modelled as two open classes with exponentially distributed inter-arrival and service times [19]. Service times are obtained through a real experimentation on the base of the benchmark values as reported in Table 2. They can be considered as the results of the prediction component.

Table 2. Mean service times of each application class (in parentheses the number of processors spawned for each resource)

	IBM (32)	Michelangelo (32)	SC1458 (128)	Paperoga (8)	Cluster1 (16)
ISO	2.4	3	35	13	7
HPL	33	25	4.5	55	62

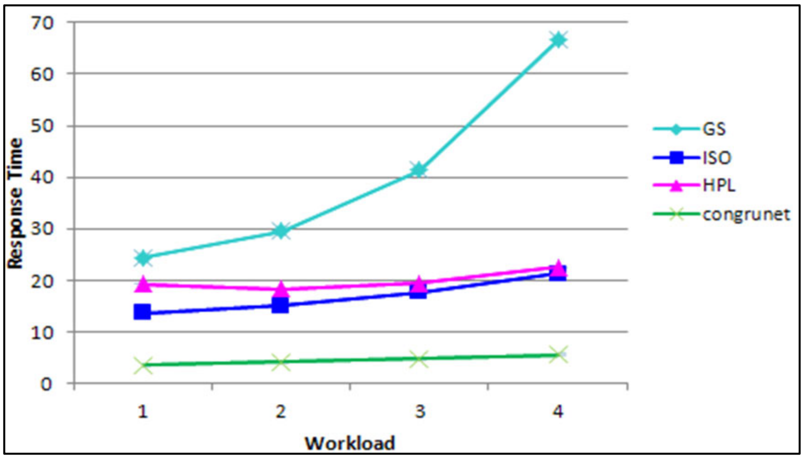

Fig. 5. Response times according to different scheduling strategies at increased workload

In Figure 5 the response times of each strategy at increasing workloads are shown. It is immediately clear that the proposed performance-based SLA outperforms the other schedulers. This is not surprising since each resource is exploited as its best respect to the incoming workloads, i.e. each application is allocated to the resources that execute the code in the most efficient way, in our analysis with minor execution time. It leads to faster execution and lower waiting time. Both parameters impact (in this case positively) on the response time. An increase of computation intensive workloads also influences our scheduling mechanism, however the growth of response time is moderate compared with other tested strategies.

5 Conclusion

In this paper we proposed a performance-based SLA-aware architecture. The main idea is to characterize resources on the base of specific benchmarks in order to allow suitable job allocations. We have demonstrated simulation results which show clear benefits and which give an indication of what can be expected if our proposed architecture will be implemented for the job scheduling.

In particular, we have analyzed and tested just a first part of the proposed architectural concept. We plan to spend further efforts in the elaboration and analysis of the performance prediction and evaluation components. This will include an evaluation of different methods for the prediction of expected job execution performance as well as for the correction based on the deviation between expected and measured results.

Acknowledgements. The authors would like to thank the members of the Munich Network Management (MNM) Team for their support and many useful discussions. As a group of researchers from the Ludwig-Maximilians-Universität München, the Technische Universität München, the University of the German Federal Armed Forces, and the Leibniz Supercomputing Centre of the Bavarian Academy of Science and Humanities, the MNM Team focuses on computer networks, IT management, High Performance Computing, and inter-organizational distributed systems. The team is directed by Prof. Dr. Dieter Kranzlmüller and Prof. Dr. Heinz-Gerd Hegering. For more information please visit http://www.mnm-team.org.

This work has partially been funded by the Seventh Framework Program of the European Commission (Grants 246703 (DRIHMS) and 261507 (MAPPER)), and by the project REsource brokering for HIgh performance, Networked and Knowledge based applications (RE-THINK), P.O.R. Liguria FESR 2007-2013.

References

1. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure, 2nd edn. Elsevier (2004)
2. Distributed European Infrastructure for Supercomputing Applications (May 10, 2011), `http://www.deisa.eu/science/benchmarking`

3. Hockney, R.W.: The science of computer benchmarking. Software, environments, tools. SIAM, Philadelphia (1996)
4. Simmhan, Y., Ramakrishnan, L.: Comparison of Resource Platform Selection Approaches for Scientific Workflows. In: 19th ACM International Symposium on High Performance Distributed Computing, HPDC 2010, pp. 445–450 (2010), doi:10.1145/1851476.1851541
5. Clematis, A., Corana, A., D'Agostino, D., Galizia, A., Quarati, A.: Job–resource matchmaking on Grid through two-level benchmarking. Future Generation Computer Systems 26(8), 1165–1179 (2010)
6. Bode, B., et al.: The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters. In: 4th Annual Linux Showcase and Conference, Atlanta, USA (2000)
7. Frey, J., Tannenbaum, T., Livny, M., Foster, I., Tuecke, S.: Condor-G: A Computation Management Agent for Multi-Institutional Grids. Cluster Computing 5(3), 237–246 (2002)
8. Huedo, E., Montero, R., Llorente, I.: A framework for adaptive execution in grids. Software Practice and Experience 34(7), 631–651 (2004)
9. Flops Benchmark (May 10, 2011), `http://home.iae.nl/users/mhx/flops.html`
10. McCalpin, J.D.: Memory Bandwidth and Machine Balance in Current High Performance Computers. In: IEEE Technical Committee on Computer Architecture (TCCA) Newsletter (1995)
11. Mucci, P.J., London, K., Thurman, J.: The CacheBench Report, University of Tennessee (Cachebench Home Page) (May 10, 2011), `http://icl.cs.utk.edu/projects/llcbench/cachebench.html`
12. Gropp, W., Lusk, E.: Reproducible Measurements of MPI Performance Characteristics. In: Margalef, T., Dongarra, J., Luque, E. (eds.) PVM/MPI 1999. LNCS, vol. 1697, pp. 11–18. Springer, Heidelberg (1999), `http://www-unix.mcs.anl.gov/mpi/mpptest/`
13. Rabenseifner, R., Koniges, A.E.: Effective File-I/O Bandwidth Benchmark. In: Bode, A., Ludwig, T., Karl, W.C., Wismüller, R. (eds.) Euro-Par 2000. LNCS, vol. 1900, pp. 1273–1283. Springer, Heidelberg (2000)
14. Tsouloupas, G., Dikaiakos, M.: GridBench: A Tool for the Interactive Performance Exploration of Grid Infrastructures. Journal of Parallel and Distributed Computing 67, 1029–1045 (2007)
15. The High Performance LINPACK Benchmark (May 10, 2011) `http://www.netlib.org/benchmark/hpl/`
16. D'Agostino, D., Clematis, A., Gianuzzi, V.: Parallel Isosurface Extraction for 3D Data Analysis Workflows. Distributed Environments, Concurrency and Computation: Practice and Experience (2011), doi:10.1002/cpe.1710
17. Clematis, A., D'Agostino, D., Galizia, A., Quarati, A.: Profiling e-Science Infrastructures with Kernel and Application Benchmarks. Submitted for the publication in Journal of Computer Systems Science and Engineering
18. Casale, G., Serazzi, G.: Quantitative System Evaluation with Java Modeling Tools. In: ICPE 2011, Karlsruhe, Germany, March 14-16 (2011)
19. Lazowska, E.D., Zahorjan, J., Scott Graham, G., Sevcik, K.C.: Quantitative System Performance - Computer System Analysis Using Queuing Network Models. Prentice-Hall, Inc. (1984)