A Performance-Based Methodology To Improve Grid Exploitation

A. Clematis[1], A. Corana[2], D. D'Agostino[1], A. Galizia[1], A. Quarati[1]

[1] IMATI-CNR, Via De Marini 6, 16149 Genova, Italy,
{clematis, dagostino, galizia, quarati}@ge.imati.cnr.it
[2] IEIIT-CNR, Via De Marini 6, 16149 Genova, Italy
corana@ieiit.cnr.it

Abstract Due to their complexity, the exploitation of Grid environments is not a trivial activity for many users, and a key factor is to enable a simplified and transparent orchestration of resources and jobs. Particularly critical is the deployment of matching procedures capable to effectively meet user's requirements with resources offer. We introduce GREEN a management tool primarily devoted to the matchmaking process, based on a performance characterization of both resources and job requirements. Leveraging on a two-level benchmarking methodology, GREEN allows users to express performance preference through an appropriate extension to Grid submission and description languages such as JSDL and Glue. Operating at intermediate level between applications and Grid middleware, GREEN reduces the gap between users' needs and available resources thus enabling a seamless exploitation of the Grid.

Keywords: Grid management, Benchmark-driven matchmaking, Grid language extensions

1 Introduction

Grid environments are service-oriented infrastructures that facilitate the sharing of instruments, knowledge, data and computational resources managed by different organizations in widespread locations and supply their exploitation through the submission and the execution of users' jobs. Since their first appearance, Grids showed great potentialities for the scientific community as they allow the definition of virtual spaces providing huge computational power and collaboration tools to scientists [1]. Examples of Grid adoption in the scientific realm are found in projects such as CaBIG, Worldwide LHC Computing Grid, AstroGrid investigating respectively bioinformatics, high-energy physics and astronomy issues [2-4]. The Grid is also exploited to guide business experiments, for example, the Business Experiments in Grid project (BEinGrid) is aimed to highlight scenarios, solutions and results in 25 case studies [5].

To support experiments and investigations, distributed resources and jobs have to be orchestrated in such a way that user's objectives are addressed without requiring a deep and difficult interaction with the resources. Actually, the consumption of the shared resources in a Grid could be not trivial as they are heterogeneous and generally belong to different Physical Organizations (POs). POs are subject to a variety of configuration settings and are usually federated in Virtual Organizations (VOs). VOs group people with similar interests and aims, thus leading to the identification of sets of common (i.e. most used) applications, each owning specific requirements and execution modes [6].

Due to the organizational and technological complexity of these environments, practices and tools to manage resources and to model and maintain their consistent description are required. In particular, information about resources properties, their current state and user's specific requirements is essential to guarantee that a job submitted by a user will be forwarded to the most appropriate resource. Indeed, this supply-demand coupling process is a critical one, since it reflects on the effective execution of each distinct user' application, and significantly impacts on the overall performance of a VO as a whole [7]. The responsibility for performing this crucial activity is commonly left to the matchmaking component, whose main task is to grant the discovery of available resources and services on the base of the specific properties defined by users and expressed through pertinent requests [8]. To this end, the matchmaker may greatly benefit from a performance characterization of resources based on the employment of benchmarks [9].

Benchmarking represents a powerful mean to investigate, characterize and compare the performance of different computer systems in order to select the most suitable resource to execute a class of applications. Considering traditional microprocessors as well as High Performance Computing systems, it is possible to outline two categories: *Micro-benchmark* and *Application-specific benchmarks*. The former is apt to profile resources considering isolated low-level capabilities such as CPU, memory, and interconnection speed [10]. The latter is apt to stress simultaneously several aspects of the system, and corresponds to the computationally demanding part of real applications. Moving towards Grids, the characterization of computational resources through benchmarks is largely acknowledged together with its intrinsic criticality [11,12], mainly due to the multi-layered, dynamical, heterogeneous structure of the Grid, and often hindered by the specific procedures adopted by each VO in classifying and making resources accessible.

In this paper, we present GREEN, Grid Environment ENabler, a management tool designed to assist Grid administrators and users to set-up, administrate and exploit Grid infrastructures, with prior activity the matchmaking process. To fulfil this goal, GREEN relies on a two-level benchmark methodology, i.e. *Micro* and *Application-specific*, through which every resource of a PO is tagged with the performance results obtained under different workloads. Operating at intermediate level between applications and Grid middleware, GREEN focuses on the discovery of resources satisfying user requirements ordered by performance ranking, while the selection of any particular amongst them, is left to a scheduler, responsible to apply the proper policies. To sustain the matching operation, GREEN offers administrators and users, functionalities to store benchmarks results and to submit jobs, respectively. From the administrator's point of view, GREEN supports the creation and maintenance of the performance description, allowing to efficiently respond to user's requests of integrating new relevant application-driven benchmarks. From user's point of view, GREEN enables the declaration of a ranking preference for the resources during job submission, i.e. the selection of the benchmark to guide the matching process. GREEN receives requests of job submission initiated by users; it uniforms execution requests, expressed through different Job submission languages, thus addressing interoperability issues; and carries-out their subsequent submission to the underlying middleware.

The outline of the paper is as follow. Section 2 discusses some valuable contributions in the fields of matchmaking and benchmarking on Grid. In Section 3, we present our two-level benchmarking methodology along with some preliminary results highlighting its appropriateness in Grid scenarios. Section 4 introduces GREEN as a management tool for Grid environments, focusing on a technical overview. Section 5 gives some concluding remarks.

2 Related Works

The implementation of an efficient and automatic mechanism for the effective discovery of the resources that best fit the requirements of users' job is one of the major problems in present Grids. A possible way to improve the efficiency of this step is to drive the search towards resources that show good performance in the execution of jobs with similar or known behavior. This issue initially obtained little attention from the middleware designers and developers, thus several projects and tools proposed solutions to address the topic.

From the middleware point of view, the Globus toolkit did not provide, originally, a resource matchmaking/brokering as core service. However, since June 2007, the GridWay metascheduler [13] has been included in the Globus distribution as an optional high-level service. GridWay allows users to specify a fixed and limited set of resource requirements, mainly related to the queue policies of the underlying batch job systems. Benchmarks are not considered at all, and this choice limits the ranking of resources. On the contrary, gLite has a native matchmaking/brokering service that takes into account a richer set of requirements, including benchmark values. This service is based on a semi-centralized approach, and may result in long waiting time in the job execution. The set of benchmarks actually considered by gLite, i.e. the SPEC suite, mainly evaluates CPU performance [14]; thus, the description of system performance may result partial, hence not completely suitable to specific application requirements. A more accurate strategy should take into account some of the proper characteristics of the application at hand, as claimed in Section 3.

Due to the peculiar nature of the Grid, performance evaluation in a dynamical, heterogeneous context is more complex and less deterministic than in traditional scenarios. In fact, the Grid has a multi-layered structure, thus benchmarks investigating performance aspects of the different Grid layers should be considered in order to grasp a predictable behaviour of a real application run [11]. Actually, besides the set of interesting parameters to measure the single isolated resource, e.g. CPU speed, memory and interconnection bandwidth, different factors have to be taken into account when considering the execution of a benchmark (suite) on Grid. For example, the Grid Assessment Probes [15] has a means of attempting to provide an insight into the stability, robustness, and performance of the Grid. The probes are designed to serve as simple Grid application exemplars and diagnostic tools. They test and measure performance of basic Grid functions, including file transfers, remote execution, and Grid Information Services response. GridBench [9] is a complex and interesting tool that provides a graphical interface to define, execute and administrate benchmarks. It takes into account interconnection performance and evaluates resource workload, and can be used to rank Grid resources. The NAS Grid Benchmark (NGB) suite [16] is defined by NASA, and represents typical activity of Computational Fluid Dynamics applications. It provides a set of computationally intensive benchmarks representative of scientific, post-processing and visualization workloads, and tests the Grid capabilities to manage and execute distributed applications.

A brokering mechanism based on benchmarking of Grid resources is proposed by Elmroth and Tordsson [17]. However, the scope of the broker is focused on the ARC middleware and the NorduGrid and SweGrid production environments, and it adopts an extension of RSL (earlier Globus submission language) to submit user's jobs, conversely to our proposal aimed to follow a more interoperable approach.

3 A Two-Level Benchmarking Methodology

To describe Grid resources, we propose a two-level methodology aimed to give a useful enriched description of resources, and to facilitate the matchmaking process. Our methodology considers two approaches: I) the use of micro-benchmarks to supply a basic description of resource performance; II) the deployment of application-driven benchmarks to get a closer insight into the behavior of resources under more realistic conditions of a class of applications. Through application-driven benchmarks, it is possible to add an evaluation of the resources based on the system indicators that are more stressed by an application.

3.1 Micro-Benchmarks

In order to supply a basic resource characterization, mainly based on low-level performance capacity, we considered the use of traditional micro-benchmarks. To this aim, a reasonable assumption is that the performance of a machine mainly depends on the CPU, the memory and cache, and interconnection performance [10]; therefore, we individuated a concise number of parameters to evaluate in order to provide an easy-to-use description of the various nodes. We selected a set of five, largely widespread, benchmarks able to capture relevant metrics to characterize computational resources' performances. In particular, *Flops* provides an estimate of peak floating-point performance (MFLOPS) by making maximal use of register variables with minimal interaction with main memory [18]. *Stream* is the industrial de facto standard benchmark to measure sustained memory bandwidth [19]. *CacheBench* is designed to evaluate the performance of the memory hierarchy of computer systems, expressed by raw bandwidth in megabytes per second [20]. *Mpptest* measures the performance of some of the basic MPI message passing routines in a variety of situations [21]. *Bonnie* performs a series of tests on a file of known size. For each test, it reports the bytes processed per CPU second, and the percentage of CPU usage [22].

The micro-benchmarks used in this phase generally return more than a value. To obtain results easily usable in the matchmaking process, we considered for each benchmark synthetic parameters or the most significant value. These results are managed by GREEN to populate the benchmark description of resources.

3.2 Application-Specific Benchmarks

Micro benchmarks are a good solution in the case of applications stressing mainly one architecture aspects, e.g. CPU intensive, or not frequently executed. Indeed, usually the participants to a VO have similar aims, from which a set of the most used applications emerges. In these cases, a more suitable approach is to evaluate system performance through application-specific benchmarks that approximate at best the real application workload. This benchmarking level offers two procedural approaches a) the use of a “light” version of the application at hand, with a reasonable computational cost but still representative of the real behaviour; b) the use of well known application specific benchmarks largely employed in the scientific community.

As case studies, we considered some applications of our interest, i.e. image processing, isosurface extraction, and linear algebra. For the first two classes of applications, we adopted approach a) using a sequential code aimed to emphasize precise aspects of the considered metrics. With respect to image processing, we selected a compute intensive elaboration applied to a reference image of about 1 MB; in this way, CPU metrics are mainly stressed. Hereafter we refer to this code as Image Processing Benchmark (IPB). The isosurface extraction application provides a more exhaustive performance evaluation of the system, as it also heavily involves I/O operations. In this case, we considered the processing of a small 3D data set of 16 MB, producing a result of 67 MB. Following approach b) for the class of applications based on linear algebra, we selected the well-known High Perfomance Linpack (HPL) benchmark [23]. For application-driven benchmarks, the metric considered to characterize resources is wall clock time. Similarly, to the micro-benchmarks case, the results are stored in the internal data structure of GREEN.

3.3 Methodology Evaluation

To evaluate the effectiveness of our methodology, we experimented upon two specific resources: 1) a Beowulf Cluster made up of sixteen nodes interconnected by a Gigabit switched Ethernet. Each node is equipped with a 2.66 GHz Pentium processor, 1 GB of RAM and two EIDE disks interface in RAID 0 2) the SiCortex SC1458 system with 243 SiCortex node chips, each equipped with six cores; linked by a proprietary interconnection network supporting large message bandwidth of 4 GBytes/sec. This system pursues the Green Computing guidelines, through extremely low energy consumption. By a quick comparison, clearly emerges that the two resources vary greatly both in terms of the number of CPUs and in terms of individual CPU performance. In fact, SC1458 has a greater number of CPUs than the Beowulf Cluster, but the latter has faster CPUs and better memory bandwidth. Notwithstanding from these technical differences, one may infer consequent performance results, this expectation is contradicted by our experiments.

Starting from micro-benchmark results, the SC1458 achieves better performance in almost each case and parameters evaluated, when considering aggregate computing power. However, its single cores have relatively low performance compared with the single CPU of the Beowulf Cluster, and the actual power of the resource derives from the high number of provided cores and the native fast connection among processes. To outline CPU performance, we depicted in Figures 1 and 2 the results obtained with FLOPS and STREAM.

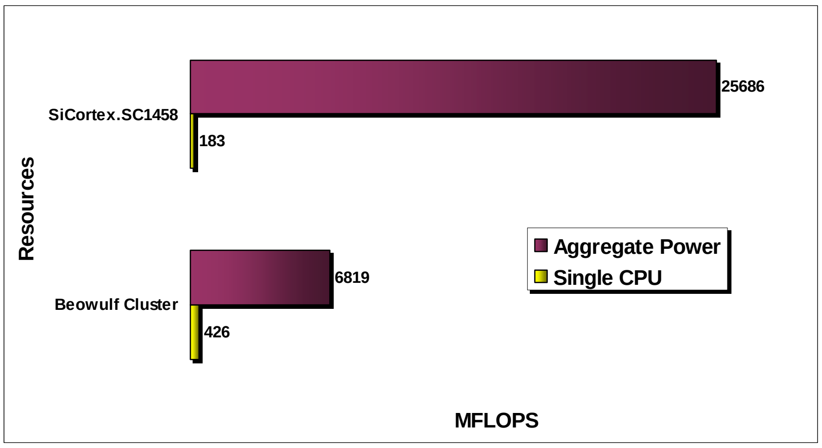

Figure 1 Comparison between resources according to FLOPS

Both benchmarks have been run on a CPU/core independently, and then the aggregated results are gathered to represent the performance of the whole parallel resources [9]. For each resource, we present the evaluation of the single CPU/core and the parallel resources.

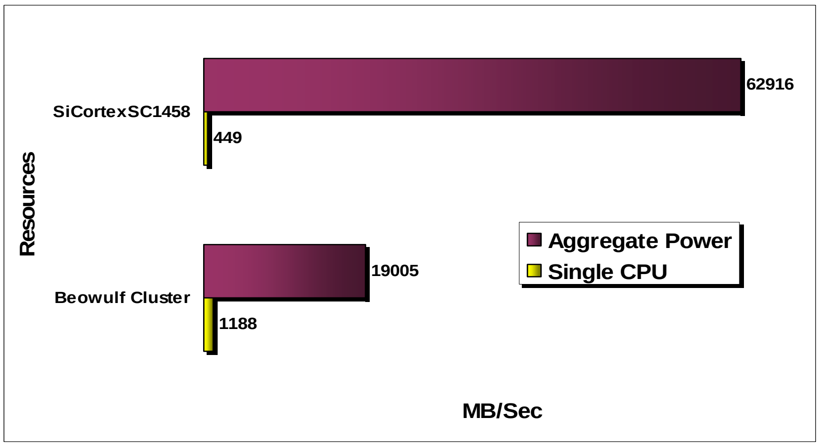

Figure 2 Comparison between resources according to STREAM

Also with respect to interconnection evaluation, the SC1458 achieved definitely better performance, as reported in Figure 3. We tested point-to-point communication performance, through the MPPTest benchmark; results are expressed in MB/Sec. As mentioned above, the Beowulf Cluster employs a Gigabit Ethernet, while SC1458 has a proprietary interconnection that performed significantly better.

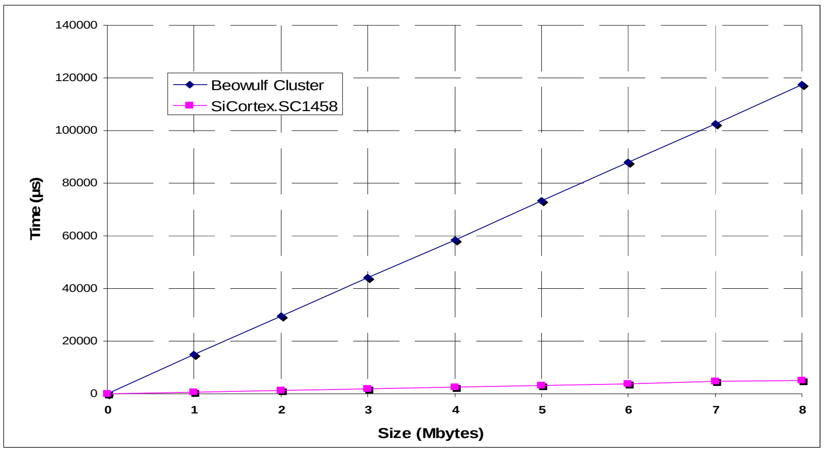

Figure 3 Comparision wrt MPPTest

Considering the second level of benchmark, the situation is quite different. In fact, depending on the application domain, better results were obtained alternatively by both resources. We conducted our tests by using IPB and HPL benchmark, and considering the execution times (Wall Clock Time) as metric to evaluate performance. The results are normalized according to a base value; to this end, we adopted the values returned from the Beowulf Cluster. Table 1 reports the values obtained for IPB and HPL benchmark. As already said, in the latter case, we considered all available processes for the Beowulf Cluster, i.e. 16 nodes, while for the SC1458 resource we examined separately the use of different number of processors (16, 64, 128).

Table 1 Comparison of executions performance, normalized wrt Beowulf Cluster

	Beowulf Cluster	**SC1458**	**SC1458 16 p**	**SC1458 64 p**	**SC1458 128 p**
IPB	1	6.1			
HPL benchmark	1		0.44	0.13	0.08

The first row of Table 1 shows that Beowulf Cluster performed significantly better considering the image processing application, but the situation is exactly the opposite for HPL benchmark as expressed in row 2, which highlights that SC1458 outperforms Beowulf up to a factor 10, when increasing the number of processes. This behaviour depends on the different requirements of the two applications. In the analyzed cases, IPB solely benefits from fast single CPU, while HPL tests the entire system and benefits from high number of processes linked with fast connections. Starting from these remarks, it is quite evident that the Beowulf Cluster is faster in the execution of IPB, while it poorly performs with respect to HPL. On the contrary, with respect to

HPL, SC1458 outdoes the Beowulf Cluster, but it does not achieve good results on the proposed image processing operations.

Following our methodology, it clearly emerges the differences in the performance of both resources in each level of benchmark. SC1458 definitely performs better than the Beowulf Cluster with respect to the micro-benchmark. However, considering the second level of benchmark, the Beowulf Cluster appears as the suitable choice for the execution of specific applications. This performance divergence also occurred in other similar comparisons we conducted for all the other benchmarks previously described, and thus testifies the appropriateness of our approach.

4 GREEN a Benchmark-Based Tool to Manage Grid Resources

To reduce the gap between users and resources, we designed GREEN, a Grid management tools mainly aimed to perform matchmaking based on a performance characterization of resources and jobs. GREEN bases on a distributed approach and leverages on a overlay network infrastructure to connect the various POs constituting a Grid [24]. GREEN introduces some features able to satisfactory fulfil the diverse needs of Grid stakeholder:

- Insertion of benchmark information by system administrators;
- Supporting users to the submission of Job to the Grid;
- Translation of job submission expressed into a JSDL document into the specific submission language accepted by the middleware;
- Execution of the (distributed) matchmaking process;

These functionalities rely on a proper description of resources required both on the job/user and on the owner side, necessary to accomplish the coupling task. In fact, according to our methodology, benchmarking outcomes are used to annotate (tagging) Grid resources. These tags are then compared with the benchmark-related requirements, contained in the job documents submitted by users. Analysing the main success proposals, carried out by different projects and research groups in the field of resources and job description, and aimed to deal with different middlewares transparently to Grid users, we defined two extensions capable of capturing the benchmark characterization of both resources and jobs.

4.1 Extending Languages for Job and Resource Characterization

As to resources characterization, we adopted the Grid resources vision offered by the Glue 2.0 specification language [25], which foresees that benchmark-value copies are represented as Glue entities according to the XML reference realizations of Glue 2.0 [26]. By employing the openness of `BenchmarkType_t`, the set of recognized benchmarks is extensible without any change to the document schema. This solution allows the seamless insertion of new benchmarks data as soon as they should appear relevant to the users of a VO. The specificity of our two-level methodology is modelled with the extension mechanism defined in Glue. We enriched the `Benchmark_t` type adding the `BenchLevel` element to specify the benchmark level (i.e. two string values `micro` and `application`). An excerpt from a document related to the execution of micro-benchmark Flops against the Beowulf Cluster, whose head node has IP 150.145.8.160, resulting in 480 MFlops is:

```
<Benchmark>
   <LocalID>150.145.8.160</LocalID>
   <Type>MFlops</Type>
   <Value>480</Value>
   <BenchLevel>micro</BenchLevel>
</Benchmark>
```

Listing 1 Example of the extension to the Benchmark element

The counterpart of benchmarking resources is the ability for users submitting a job to express their preferences about the performance of target machines. A job submission request, in addition to stating the application-related attributes (e.g. name and location of source code, input and output files), should express syntactic requirements (e.g. number of processors, main memory size) and ranking preferences (if any) to guide and constraint the matching process on resources. To this end, some mechanism is required to allow users to explicitly assess these requirements inside the job submission document.

The three main Job Submission Languages (JSL) currently used by Grid community are the Globus Job Description Document (JDD) [27], the EU-DataGrid Job Description Language (JDL) [28], and the Job Submission Description Language (JSDL) [29] proposed by one of the Working Group of Grid Forum. Evaluating their major properties and how they differentiate each others, e.g. in the support to express requirements on resource, we decided to extend JSDL, whose mission is to provide a standard language to be used on top of existing middlewares. Augmenting JSDL schema to take into account ranking specification, we introduced an element `Rank` (of complex type `Rank_Type`) devoted to this task. To maintain a desirable, although not mandatory, uniform lexicon between the JSDL constructs on job side and the Glue description on resource side, we borrowed from the Glue extension the definition of `BenchmarkType_t`, which is embedded as sub-element of `Rank`.

4.2 Components Description

GREEN is designed as a Grid service based on a distributed and cooperative approach for Grid resource discovery and ranking. For every PO in a Grid, a GREEN instance is responsible for the management of updated data about the state of its resources, and for its exchange with other GREEN instances to satisfy user's requests. Figure 4 depicts the main components of GREEN, along with some interactions with other middleware services, notably the Information Service (IS) and Execution Environment (EE), occurring after the submission of a job. In the following, we summarise the role and the behaviour of those components:

- The Job Submission (JS) component is the main gateway to GREEN functionalities; it receives requests of benchmark submission by PO administrator or jobs submission initiated by users. Depending on the activation mode (according to the different published signatures), it behaves just like a messages dispatcher or a translator of JSL documents carrying-out their subsequent submission to the EE, thus addressing interoperability issues.

- The main task of the Benchmark Evaluation (BE) component is to support administrator in the characterization of PO resources on the basis of benchmark-measured performance. Initially, for any relevant benchmark, the administrator submits a JSDL document to the JS component of the GREEN instance associated with his PO. After translating the JSDL document into the particular JSL document compliant with the middleware used by the PO (e.g. JDL for gLite, JDD for Globus), JS passes it to the Benchmark Evaluator port, which interacts with the EE to execute the benchmark against all resources/machines alive. When results are returned, an XML fragment, similar to the one reported in Listing 1, is created for each resource and inserted in a XML document (i.e. *Benchmark image*), which collects all benchmark evaluations for the PO.

- The Resource Discovery (RD) is in charge of feeding GREEN with the state of Grid resources. RD operates both locally and globally by carrying out two tasks: 1) to discover the state of the PO resources, 2) to dispatch requests to other GREEN instances. As to the first task, RD dialogues with the underlying IS (e.g. MDS, gLite IS) that periodically reports the state of the PO in the form of an XML file largely conformed to the Glue version adopted by the underlying middleware. This document (namely the *PO snapshot*) is stored, as it is, in memory and managed by GREEN to answer to external queries issued by other clients (e.g. other GREEN instances, meta-schedulers). To accomplish the dispatching task, RD handles the so called *neighbours view*, establishing network routes to other nodes. Depending of the number of POs, this *view* may be limited to a reduced set of network addresses to be contacted individually (as in the case of Figure 3), or deployed via complex data structures and algorithms like those used in Super-Peer networks such as DHT [30] or random walk [31].
- The Matchmaker performs the core feature of GREEN: the matching of resources in the Grid and their subsequent ranking, according to the benchmark preferences expressed by the users. Acting as a distributed matchmaker, GREEN manages and compares the benchmark-enriched view of resources with user-submitted jobs, and produces a list of feasible resources (see Figure 4). The task of selecting the “best” among this list, is left to a (meta)scheduler to which the resource set is passed, so allowing to apply the preferred scheduling policies to optimize Grid throughput or other target functions (e.g. response times, QoS,….). Once the “best” resource is chosen, GREEN will be re-invoked to carry-out the submission of the job on it, via the EE. To carry out the exchange of message with other GREEN instances, MM leverages on the services of RD.

Figure 4 exemplifies the submission of an extended JSDL document (i.e. including benchmark requirements) by a user via Grid portal (Step 1). The Resource Selector

(RS) forwards the document to the JS component of a randomly selected GREEN instance (2) (e.g. PO_1). JS activates the Matchmaker (MM) (3), which, through RD forwards the document to all the other known GREEN instances and contemporaneously checks its local memory (4). All the matchmakers filter their *PO snapshot* selecting the set of PO resources satisfying the query (including benchmark preferences). The resources identifiers and their corresponding benchmark values are included in a list, called *PO list*, which is returned back to MM, following the routes expressed by their neighbors' views (5). MM merges these lists with its own PO list and produce a Global List, ordered on the ranking values, that is passed to JS (7), which returns it back to RS (8). RS applies its scheduling policy to determine the resource to use, and calls the JS of the GREEN responsible of the PO owning the selected machine (GREEN PO_2's instance in our case), by sending it the extended JSDL document along with the data indentifying the selected resource (9). This JS translates the information regarding the job execution of the original JSDL document in the format proper of the specific PO middleware, stating the resource on which the computation takes place (producing a JDD document for GT4 resources or a JDL document for the gLite ones), and finally, activates the Execution Environment in charge of executing the job represented in the translated document (10).

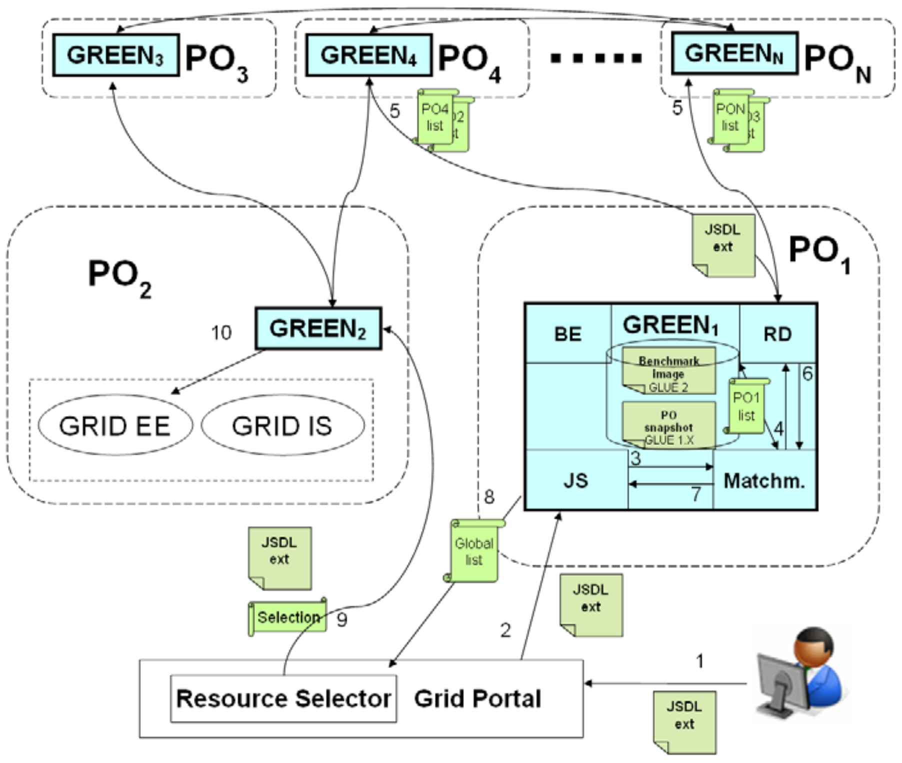

Figure 4 A user submitting an extended JSDL document via Grid portal

5 Concluding Remarks

To satisfactorily fulfil all the potentialities offered by Grids, users have to be supplied with practices and tools, able to overcome the difficulties and obstacles present in such rich but complex environments. In particular, distributed resources and users applications have to be orchestrated in such a way that user's objectives are addressed in the most seamless and effective way. We designed GREEN a management tool, primarily devoted to the matching of resources and jobs. It operates at intermediate level between users and Grid middleware, and in this way enables a simplified management of Grid resources. GREEN is based on a benchmarking methodology aimed to evaluate the performance of resources, and allowing users to express her performance preference, through an appropriate extension to Grid submission and description languages such as JSDL and Glue. The appropriateness of our methodological approach is documented by the presentation of some experimental results, which confirmed, in our opinion, the choice of adopting a double level of benchmark, as a means to reduce the gap between users' needs and resources offer.

References

1. I. Foster, C. Kesselman: The Grid 2: Blueprint for a New Computing, 2 edition, Morgan Kaufmann (2003)
2. The cancer Biomedical Informatics Grid Homepage, https://cabig.nci.nih.gov/
3. The Worldwide LHC Computing Grid Homepage, http://lcg.web.cern.ch/LCG/
4. The AstroGrid Homepage, http://www.astroGrid.org/
5. The Business Experiments in Grid, http://www.beinGrid.eu/
6. I. Foster, C. Kesselman, S. Tuecke: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications, V 15, 3, 200-222, (2001)
7. J. Yu, R. Buyya,.K. Ramamohanarao: Workflow Scheduling Algorithms for Grid Computing. Metaheuristics for Scheduling in Distributed Computing Environments, Springer, (2008)
8. X. Bai, H. Yu, Y. Ji, D.C. Marinescu: Resource matching and a matchmaking service for an intelligent Grid. Int. Journal of Computational Intelligence, v.1, n. 3, 163-171 (2004)
9. G. Tsouloupas, M. D. Dikaiakos, GridBench: A Tool for the Interactive Performance Exploration of Grid Infrastructures. Journal of Parallel and Distributed Computing, Elsevier, v. 67, pp. 1029-1045 (2007)
10. R.W. Hockney: The science of computer benchmarking. Software, environments, tools, SIAM, Philadelphia (1996)
11. F. Nadeem, R, Prodan, T. Fahringer and A. Iosup: Benchmarking Grid Applications for Performance and Scalability Predictions. In CoreGrid Workshop on Middleware, Springer Verlag, Dresden, Germany (2007)
12. M.D. Dikaiakos: Grid benchmarking: vision, challenges, and current status. Concurrency and Computation - Practice & Experience, v. 19, n. 1, pp. 89-105 (2007)
13. E. Huedo, R.S. Montero and I.M. Llorente, A Framework for Adaptive Scheduling and Execution on Grids, Software - Practice & Experience, v. 34, n. 7, pp. 631-651, John Wiley & Sons, (2004)
14. gLite 3.1 User Guide, Doc. CERN-LCG-GDEIS-722398, 28 April 2009, https://edms.cern.ch/file/722398/1.2/gLite-3-UserGuide.html

15. G. Chun, H. Dail, H. Casanova, and A. Snavely: Benchmark probes for Grid assessment. In: 18th International Parallel and Distributed Processing Symposium (IPDPS 2004), Santa Fe, New Mexico, USA. IEEE Computer Society (2004)
16. M. Frumking, R.F. Van der Wijngaart: NAS Grid Benchmarks: A tool for Grid space exploration. Cluster Computing, v. 5, n. 3, pp. 315–324, (2002)
17. E. Elmroth, J. Tordsson: Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions. Future Generation Computer Systems, v. 24 n. 6, pp. 585-593, (2008)
18. The Flop Benchmark: http://www.netlib.org/performance/papers/flops/flops_2/
19. The STREAM Benchmark: Computer Memory Bandwidth, www.streambench.org
20. Cachebench Home Page, http://icl.cs.utk.edu/projects/llcbench/cachebench.html
21. MPPTest - Measuring MPI Performance, http://www-unix.mcs.anl.gov/mpi/mpptest/).
22. Bonnie Home Page, http://www.textuality.com/bonnie/
23. The LINPACK Benchmark: http://www.netlib.org/benchmark/hpl/
24. A. Clematis, A. Corana, D. D'Agostino, V. Gianuzzi, A. Merlo, A. Quarati: A distributed approach for structured resource discovery on Grid, Proceeding of CISIS 2008, pp. 117-125, IEEE Computer Society (2008)
25. S. Andreozzi: GLUE Specification v. 2.0 (rev. 3) (2009), http://forge.Gridforum.org/sf/docman/do/downloadDocument/projects.glue-wg/docman.root.drafts/doc15023
26. GLUE v. 2.0 – Reference Realizations to Concrete Data Models, 2008, http://forge.Gridforum.org/sf/go/doc15221?nav=1
27. http://www.globus.org/toolkit/docs/4.2/4.2.0/user/gtuser-execution.html
28. Job Description Language Attributes Specification for the gLite Middleware, Doc. EGEE-JRA1-TEC-555796-JDL-Attributes-v0-8, 3/5/2006
29. A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, A. Savva: Job Submission Description Language (JSDL) Specification v1.0. Grid Forum Document GFD, 56 (2005)
30. M. Cai, M. Frank, J. Chen, and P. Szekely, Maan: A multiattribute addressable network for Grid Information Services, Proc. 4th Int. Workshop on Grid Computing, 2003.
31. C. Rabat, A. Bui, O. Flauzac, A Random Walk Topology Management Solution for Grid, Lecture Notes in Computer Science 3908, pp. 91-104, Springer, 2006.