# Jawari - A Grid Benchmarking Service

Ely de Oliveira[1]

Fraunhofer Institut für Techno-und Wirtschaftsmathematik, Fraunhofer-Platz 1,
67663, Kaiserslautern, Germany
*email:* ely.oliveira@itwm.fhg.de
*phone:* (+49 0631) 31600 4472, *fax:* (+49 0631) 31600 1099

**Abstract**

Over the past decade, computational grids have become a popular computing platform for large-scale, resource-intensive applications. Several technologies have been developed and many infrastructures built across the globe to support the increasing number of such applications. However, their strengths and weaknesses as well as their influence on applications performance are not yet well known. There is not enough support for users to choose grids that match their needs nor for system architects and administrators to improve them so that user's requirements can be met. In this paper, we present Jawari, an open source, free of charge service which aims at providing means of benchmarking grids around the world in order to enable further analysis and comparisons.

## 1 Introduction

Over the past decade, computational grids have become a popular computing platform for large-scale, resource-intensive applications. Several technologies have been developed and many infrastructures [5, 4, 7, 2] built across the globe to support the increasing number of such applications. However, the result of such a variety of solutions combined with the relative novelty of the subject is that grids are today complex environments, difficult to be maintained, whose characteristics are not completely known by their administrators or their users.

In order to continue to deliver the quality of service expected by their users and become even more ubiquitous, grids need to be evaluated and have their strengths and weaknesses identified, so that they can evolve.

Benchmarks allow us to quantify and understand capabilities of computing platforms. Hence, they would be an instrumental tool for grid evolution, helping us to assess and improve grid platforms. However, like any other grid issue, grid benchmarking has the challenge to address the complexity, large scale, and the dynamics of the environment. Moreover, it has to avoid to address previously covered areas, and focus on grid new features. But just like any other grid issue, the benefits of having such a task done are huge and motivate further efforts toward this goal.

In this paper we report the current status of jawari [1] [2] , an extensible, open source, platform independent, free of charge grid benchmarking service. Jawari aims to provide the grid community with support to assess and improve the quality of service delivered by grid infrastructures, making grid benchmarking accessible and easy to be done by users around the world. Doing so, it will help grids to evolve into more reliable and even more powerful platforms.

The reminder of this article is organized as follows: in Section 2 we present related works. In Section 3 a list of major requirements for grid benchmarking is presented. Section 4 and 5 presents the project design and implementation. Finally, future work and concluding remarks are presented in Section 6.

## 2 Related Works

In High Performance Computing realm, the most public visible benchmarking project is the TOP500 [6] list. It is based on the Linpack benchmark suite, known by its scalability over a large range of computer platforms. TOP500 allows the HPC community to publish benchmark results obtained by computer systems around the world. Twice a year, a list of the 500 most powerful systems is released, containing a variety of information such as system specifications. The TOP500 web site makes it possible for manufacturers to show off their products for competitive comparisons. It also permits decision makers to compare platforms and administrators to tune their systems.

Although grid benchmarking has not yet produced any tool which can compete in popularity with TOP500, it has been an active field of reseach in the recent years. A Research Group is dedicated specially to this subject within Global Grid Forum [3] and a number of projects have produced specifications, suites and even tools.

One of the most known works in this area is the NAS Grid Benchmark (NGB) suite [11], that comprises specifications of computationally intensive benchmarks, which represent classes of workflow grid applications, each corresponding to a different communication pattern. *Job turnaround time* is addopted as the performance metric. Some recent studies have criticized the extensive manual effort for the execution and analysis of NGB experiments as well as the large number of factors that can influence the job turnaround time, such as: local queuing systems, network traffic, file-system configuration and other concurrent job executions. They say that such a metric provides little if any insights about grid performance.

Another well known work is the GRASP [9], a set of three simple benchmark specifications that represent compositions of both computing and file-transfer tasks commonly found in grid applications. The so called probes focus on fundamental operations, and their results are affected mostly by network bandwidth.

[1]This project is sponsored by the German Research Ministry (BMBF).

[2]Jawari is a Brazilian indigenous ritual where representatives from different tribes confront their abilities in a series of disputes in order to identify the strongest group.

GridBench [13, 14] is an example of a functional grid benchmarking tool. Its current benchmark suite focus on low-level resource performance measurements, such as CPU speed, memory capacity and cache performance. It includes adaptations of well known HPC benchmarks such as *Wetstone CPU Benchmark* and addopts low-level metrics such as *Available Memory* and *FLOP/s.* Such metrics are assumed to offer better information about grid performance than high-level ones such as job turnaround time, since they are result of much less factors [10]. In addition, GridBench is based on an extensible framework which allows the incorporation of other types of benchmarks. It also provides the user with desktop tools for benchmarks management and results analysis.

## 3 Grid Benchmarking Requirements

Basically all benchmarking projects share reproducibility as a fundamental requirement. Two executions of the same benchmark are expected to expose a system to identical well-defined workloads, and produce the same results. In order to archive this goal, nondeterminism should be tightly controlled, so that the system behavior can reflect the performed tests. However, in parallel systems, this requirement is difficult to be met in practice. Even ordinary personal computers can have some sporadic intrusions (like interruptions), difficult to be controlled that influence benchmark results. This situation is even worse for grids. Considering their dynamism, decentralized administration, large-scale distribution, high heterogeneity, they are awash in nondeterminism. Moreover, traditional benchmarks assume that their target systems can be used exclusively for benchmarking. Such an assumption is not feasible for grids considering their distributed nature and quality of service requirements that must be met.

Besides, benchmark execution and management should be affordable in terms of costs such as the management of the benchmarking process, execution time, hardware and software involved. The complexity of the environment can boost this costs. It can also lead the benchmark results to be voluminous, expressed in different measurement units, presented in different formats, difficult to be understood. All these issues pose challenging primary questions: Is grid benchmarking really possible? How can it be done?

Grid benchmarking can be compared with photography. Single benchmark executions are like pictures - static images of a dynamic object. They may not offer any insights about object's behavior and can be blurred by several factors. A movie would offer a much more complete vision about the object. However, the process of making it could be too costly and the time to watch it could be too long. The solution could be to take several pictures in different moments. Some objects would need more pictures to be taken than others, depending on their level of dynamism. Ultimately, they would reveal the average behavior and tendencies of the observed object.

This can be translated to the use of statistical techniques to collect and process benchmark results, such as *random sampling.* A series of benchmark executions would produce snapshots that would reveal the average behavior and tendencies of the target system. Analisis of *histograms* could indicate the proper frequency of executions, *medians* could aggregate massive results in sumaries that better represent tendencies. By adopting such techniques, benchmarks can finally reproduce equivalent results which are meaningful and scientifically valid.

Furthermore, benchmark specifications should be platform independent, openly available, concise, self-contained and represent typical tasks executed on the target system [11]. By analyzing existing grid applications, it is possible to identify patterns in the way how tasks are organized or how frequently grid services are invoked. These patterns can be used to create benchmarks that expose the grid to usual situations. The benchmarks should also be composed by well defined quantities of work so that two different executions can produce comparable results.

Another important requirement is the right focus. Grid Computing, frequently described as an extension of the Internet, embraces many other subfields. Nevertheless, they deliver new services and qualities of services and therefore, grid benchmarks should explore these new features, avoiding to address previously covered areas [11]. Fast computing resources, broad network bandwidth usually are fundamental for a good performance in a computing environment. But they can contribute little in a very dynamic grid where resource brokering does not scale well, for example. Good results from traditional benchmarks can blind benchmarkers about poor quality delivered by grid services. In a grid, definitely the whole is more than the sum of its parts.

## 4 Design

Jawari assumes an underlying infrastructure consisting of a set of geographically distributed, heterogeneous grid services, organized in sites, connected over a shared network (e.g. the Internet) and participating of one or more virtual organizations. The Jawari architecture is shown in Figure 1, and its key components are described in the following.

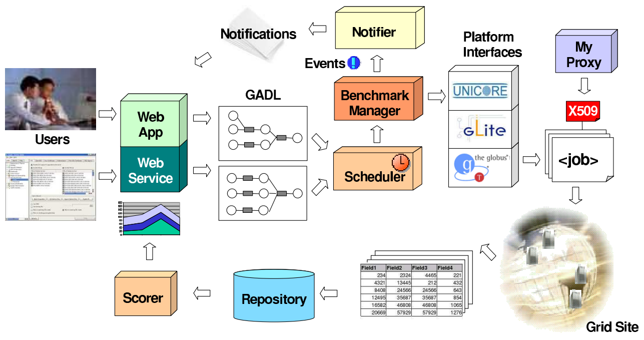

Figure 1: Jawari Architecture

**Web Site:** The Web Site is the interface meant for human users that allows them to manager benchmarks and analyze their results.

**Web Service:** The Web service, on the other hand, is meant for client software. Any other service or application that needs to benchmark grids or just query information about their resources can invoke it.

**Platform Interfaces:** Each grid platform is represented by a platform Interface object, that encapsulates all the communication with the platform underneath. The service can be extended to support new platforms, simply with the incorporation of new Platform Interface objects.

**GADL:** The Grid Application Description Language, so called GADL, is a platform independent XML-based language, proposed by Hoheisel *et al.* [12] that permits the definition of grid application workflows. It is used in Jawari to specify the sequence of tasks that composes a benchmark. Both GADL and Platform Interface objects permit the other Jawari components to work in a platform independent way.

**Scheduler:** The Scheduler is responsible for managing the execution of benchmarks according to the time constraints specified by users. Similarly to an operating system scheduler, it periodically executes 2 tasks: (i) suspends benchmarks no longer allowed to execute on the current time; (ii) starts (or resumes) the execution of benchmarks scheduled for the current time. A parameter defines a limit of benchmarks that can be running in parallel. Only one benchmark is allowed to execute per group of resources at the same time.

**Benchmark Manager:** This object coordinates a set of benchmarks that have to be executed in a grid. It receives from the Scheduler the user definitions: (i) a set of benchmarks to be executed; (ii) information about the target grid site such as grid services, adopted middleware and credentials location. Then, it discovers resources needed by each benchmark, defined in its GADL specification. After that, a number of benchmark instances are composed, while it is possible to use different resources and the limit $\tau$, defined as the benchmark sample size, is not reached. The sample sizes must be calculated using statistics techniques so that a representative amount of data can be collected.

Once started, it contacts the remote site, through the suitable Platform Interface and submits jobs that represent the benchmark instances. In fact, each benchmark instance is executed a number $\tau'$ of times. These executions happen serially, in a random order, separated by a minimum time interval $\Delta_t$, in order to reduce possible interactions between them.

The executions return several measurements and one of them is marked as the final result. Then, the collection of measurements is analyzed by a validation function. For instance, it only checks if they are complete and have values in their respective domains. Once validated, the benchmark instance final result $r$ is calculated as the median of the executions final results. Finally, the measurements and final results are sent to the repository.

**MyProxy:** By default, Jawari uses its own X509 certificate to access grid services. Nevertheless, other certificate can be defined for this purpose. The user only needs to inform its location in a MyProxy server from where it can be retrieved and used to contact the sites. Depending on the middleware, this certificate can be converted into a proper format before usage.

**Repository:** The Repository is where all the information used and produced by the service is stored.

**Scorer:** Whenever the user queries the benchmark results, the Scorer object is invoked to calculate a score for the grid performance. The formula used for this calculation is explained in the Section 4.2.

**Notifier:** The user can also define events he wants to be notified of, such as failures, benchmark completion and low scores. The Notifier receives notifications from other objects and sends emails to the interested users. Eventually, it will also permit the invocation of web services operations, automatizing maintenance procedures that can support quality of service assurance.

## 4.1 Benchmark Suite

The current suite includes twelve benchmarks. Differently from some other suites, they keep focus on grid new features rather than previously covered areas. Its approach is similar to the one addopted by other well known works such as NAS Grid Benchmarking. Roughly speaking, they are synthetic grid applications - workflows of possibly interdependent tasks, that represent classes of real grid applications, or usual fragments of those applications.

Firstly, there are three low-level benchmarks representing data exchange patterns. The 3-node Probe represents the common situation when a large data file (100 MB) is transfered from a data source host to a compute host, where it is processed, producing a result file which is transfered to a third location. The Gather Probe works in very much the same way as the 3-node Probe, except that multiple files are transfered parallelly from multiple data sources to the center compute host. The Circle Probe transfers and processes the file through a set of hosts, following a circular order. All these benchmarks addopt job turnaround time as metric, and focus the way how the middleware manages parallel large files transfers and network bandwidth. In our experiments, the robustness of services such as GridFTP is clearly noticed when compared with other solutions such as the Java stream based file transfers mechanism available in Unicore 5.

Another group of benchmarks focus on grid services throughput. The Discovery Overload, Job Manager Overload and File Transfer Overload execute a voluminous series of simple requests targeting single services from 4 different client machines. The metric used is the number of requests attended per second. By using these benchmarks, some users have been able to identify problems of robustness and improper configurations.

The other group of benchmarks represents grid applications workflow patterns. Job turnaround time is the addopted metric. Bag of Tasks represents the class of application whose tasks are independent of each other. Long Pipe represents a chain of tasks, such as a set of flow computation, that are run one after another. Compound Pipe is a collection of interconnected task flows that produce a single final result. Mixed Bag is a collection of highly interdependent tasks, emphasizing asymmetry, what makes it hard for a grid scheduler to map them to grid resources efficiently.

Two complementary benchmarks are part of the suite: the Single File Transfer, which transfers a large file (100 MB) between two hosts, and Single task submission, which represents the simplest possible grid application. The addopted metrics are, respectively, Job turnaround time and MB transfered per second.

While some benchmarks are new specifications, some others are variations of previous works [9, 11], slightly modified to focus certain grid features. Originally, they have real computation tasks as part of their workflows. In Jawari suite however, no computation is performed at all. Only very simple command line binaries such as *echo* are executed on the target environments. Doing so, resources computational capabilities influence on the benchmark execution is significantly reduced, so that the benchmarks can better reflect other environment aspects, such as the middleware overhead. Nevertheless, the Jawari design permits the suite to be extended and incorporate other benchmarks that might be identified as relevant for grid assessment.

## 4.2 Scoring Algorithm

The Jawari score calculation adopts an approach based on variables normalization, also found in other composite indexes such as the Human Development Index [8]. It is described in the following.

The grid overall performance is the result of the performance of several features of the environment. Each benchmark focuses one of these features and represents a variable of grid performance. The first step is to calculate the value of each one of these variables.

$$B = (r_i \in \mathbb{R} \mid i \in \mathbb{N}_q) \tag{1}$$

The successful benchmark results are defined by the Expression 1, where $r$ represents a measurement and $q$ is the number of successful results. The median $\mu$ of the results $B$ is calculated for each benchmark and then, normalized through a process of scaling, using the Expression 2. The denominator contains the difference between the maximum $\mu_{max}$ and the minimum value $\mu_{min}$ of the benchmark, and represents its scale. The numerator contains the difference between the values $\mu$ and $\mu_{min}$, and represents the performance achieved by the grid on a particular benchmark. The resulting percentage $\nu$ represents the path covered by the grid in the variable in question.

$$\nu = \frac{\mu - \mu_{min}}{\mu_{max} - \mu_{min}} \tag{2}$$

$$\nu = \frac{\mu_{max} - \mu}{\mu_{max} - \mu_{min}} \tag{3}$$

For some benchmarks (that usually measure execution time) the semantics of $\mu_{min}$ and $\mu_{max}$ are inverted: $\mu_{min}$ represents the best performance, and the $\mu_{max}$ the worst one. In this case, the Expression 3 is used instead, which results the proper percentage.

The scales work as baselines - references for all further benchmark results. Naturally, the calculation of $\nu$ can result percentages higher than 100% or even negative, in case the benchmarks reveal performances better or worse than their baselines, respectively.

The next step is to take into account the faulty benchmarks. The Expression 4 is used to calculate the percentage of success of benchmark executions, where $q$ is the number of successes, and $\tau$ is the total of executions.

$$\rho = \frac{q}{\tau} \tag{4}$$

Finally, the score is calculated using the Expression 5, where $n$ is the number of benchmarks and $\omega$ is the weight, that reflects the level of importance of the benchmark for grid assessment. The standard algorithm assumes equal weights ($\omega = 1$) for all benchmarks. However, the same expression can be used in situations where different weights are specified. The role of $\rho$ is to represent faulty benchmark executions. The higher the number of failures, the lower the value $\rho$. The sum of the products results the score $\lambda$.

$$\lambda = \sum_{i=1}^{n} \nu_i \cdot \rho_i \cdot \omega_i \tag{5}$$

Besides performance and availability, two other aspects can influence the score. In case the user specifies only a sub-set of benchmarks (that are known to have a good performance, for example), the final score will not include results from the omitted ones, tending to be lower than if all of them had been chosen. The score also reflects the diversity of grid resources. In a grid with a high number of different services, more benchmarks (that rely on different types of services) can be executed which increases the score.

It is also true that no single number will ever be self-sufficient to express a complex concept such as quality of service. Yet, it can be seen as a start point for a comprehensive assessment. Other traditional metrics will certainly continue to be useful in this process.

## 5 Implementation

At the time of writing, basically all the components of the architecture are implemented, and 2 middleware systems are being supported: Unicore 5 and Globus 4. The following technologies have been used: JBoss as J2EE container, Globus as grid service container, Struts as web framework, Hibernate as persistence framework, Postgres as database, JUnit and JMeter as test tools. A number of design patterns have been addopted including MVC2 and DAO/Model.

Jawari offers two possibilities of usage. The user can either access the public web application or set up a local installation. It depends mainly on whether the grid services are reachable from the central web site or not.

The user can get an account filling a form accessible from the web site front page. After the first login, he must register the sites and their resources that will be benchmarked. Optionally, he can register only main services such as Globus MDS or Unicore gateways, and let other resources to be automatically discovered. After that, the user can schedule and monitor benchmarks, define events to be notified of, browse historical data and compare sites performance over time. The results are presented in multiple levels of abstraction so that the user can go from high level charts down to detailed tables with raw data such as low-level measurements, and involved resources. The web site provides the user with a comprehensive help that describes all its functionalities.

The initial values of the design parameters were defined based on experiments. Each benchmark was executed a number of times in the D-Grid [1] environment composed by hundreds of Globus and Unicore resources spread over several cities in Germany over the course of several weeks. Then, the results were analyzed and values were identified for each one.

**Scale:** Each benchmark had their minimum and maximum final result values identified, and multiplied by 1.5 and 0.5 respectively. The resulting products were taken as the scale limits $\mu_{min}$ and $\mu_{max}$.

**Sample Size:** The result values of each benchmark were ordered by execution time, and divided into groups of 1000 elements. Then, it was possible to notice that their medians were less than 2% different from the median of the whole set of results. Then, smaller groups were formed and their medians calculated, repeatedly, while that difference was less than 2%. Finally, the size of the smallest formed group was taken as the benchmark sample size $\tau$.

During the development process, a big obstacle that we faced was the lack of comprehensive documentation for developers on both Globus and Unicore platforms. Although there has been improvement over the past few years, some essential support for developers is still missing. The Java API documentation sometimes simply does not exist, which forces the developer to read source code in order to understand their behavior.

## 6 Conclusion

Jawari is an ongoing project. Therefore, new features are constantly being added to the service and others perfected, in order to meet the project objectives. We intend to enrich the project with more benchmarks, a larger number of supported platforms, and a more sophisticated user interface. It is also planned the integration with monitoring services and other benchmark suites.

We also intend to calculate extra scores for different grid application categories (biomedical, physics, etc). The importance of each benchmark depends on each application category and, therefore, Jawari must allow the specification of different levels of weights for each benchmark result in the final score calculation. This could be done either by choosing standard application categories, or by informing a customized specification for the weights. This would allow us to rank grids per category, enabling users to search for grids that have the best performance for the kind of application they intend to run.

Although the current scoring algorithm has been able to reflect important aspects of quality of service, just like any composite index, it can have some changes in the future with the addoption of new variables, definition of new weights and new variable scales. These changes are inevitable and also desirable so that it can better reflect the grid overall performance. Nevertheless, as this algorithm is encapsuled by a single object, the Scorer, further changes will be harmless for the service and the user. All existing benchmark executions will have their scores automatically recalculated and new results will be available for the users. Naturally any adjustment in this formula will be announced and justified in advance so that the users can be aware of the changes.

As an open source project, any user can define new values for these parameters and even different scoring algorithms by simply changing its source code. However, we encourage users to contribute with the project joining the emails list and becoming an independent developer. They have been fundamental for the project making comments and suggestions of improvement. More information can be obtained at the web site: *jawari.itwm.fraunhofer.de*.

## References

1. D-Grid Initiative. http://www.d-grid.de.
2. gLite Middleware. http://glite.web.cern.ch/glite.
3. Global Grid Forum. http://www.ogf.org.
4. Globus Toolkit. http://www.globus.org.
5. OurGrid Project. http://www.ourgrid.org.
6. Top 500 Supercomputers. http://www.top500.org.
7. Unicore - Uniform Interface to Computing Resources. http://www.unicore.eu.
8. S. Anand and A. K. Sen. Human Development Index: Methodology and Measurement. Technical report, United Nations, New York, 1994.
9. G. Chun, H. Dail, H. Casanova, and A. Snavely. Benchmark Probes for Grid Assessment. Technical report, University of California, 2003.
10. Marios D. Dikaiakos. Grid benchmarking: vision, challenges, and current status: Research articles. *Concurr. Comput. : Pract. Exper.*, 19(1):89–105, 2007.
11. M. Frumkin and R. F. V. der Wijngaart. NAS Grid Benchmarks: A tool for Grid Space Exploration. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, 2001.
12. A. Hoheisel and U. Der. An XML-based Framework for Loosely Coupled Applications on Grid Environments. Technical report, Fhg FIRST, Berlin, 2003.
13. G. Tsouloupas and M. Dikaiakos. Design and Implementation of GridBench. In *European GRID 2005 Conference*, volume 3470, pages 211–225, Amsterdam, The Netherlands, June 2005.
14. George Tsouloupas and Marios D. Dikaiakos. Characterization of computational grid resources using low-level benchmarks. In *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, page 70, Washington, DC, USA, 2006. IEEE Computer Society.