CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2007; **19**:89–105
Published online 13 June 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1086

Grid benchmarking: vision, challenges, and current status

Marios D. Dikaiakos*,†

Department of Computer Science, University of Cyprus, 1678 Nicosia, Cyprus

SUMMARY

Grid benchmarking is an important and challenging topic of Grid computing research. In this paper, we present an overview of the key challenges that need to be addressed for the integration of benchmarking practices, techniques, and tools in emerging Grid computing infrastructures. We discuss the problems of performance representation, measurement, and interpretation in the context of Grid benchmarking, and propose the use of ontologies for organizing and describing benchmarking metrics. Finally, we present a survey of ongoing research efforts that develop benchmarks and benchmarking tools for the Grid. Copyright © 2006 John Wiley & Sons, Ltd.

Received 27 March 2006; Accepted 7 April 2006

KEY WORDS: Grid benchmarking; ontologies; performance evaluation

1. INTRODUCTION

Benchmarks are standardized programs or detailed specifications of programs designed to investigate well-defined performance properties of computer systems according to a widely accepted set of methods and procedures [1]. For many years, benchmarks have been used to characterize a large variety of systems ranging from CPU architectures and caches to file-systems, databases, parallel systems, Internet infrastructures, and middleware [2–10]. Computer benchmarking provides a commonly accepted basis for comparing the performance of different computer systems in a fair manner [11].

Benchmarking can be just as beneficial in Grid computing. Benchmarking metrics published on the Grid can provide a basis for users to assess the 'quality of service' expected of a Grid resource or a Virtual Organization (VO), by providing computational services at a given cost or to rank

*Correspondence to: Marios D. Dikaiakos, Department of Computer Science, University of Cyprus, 1678 Nicosia, Cyprus.
†E-mail: mdd@cs.ucy.ac.cy

Contract/grant sponsor: European Union, Working Group APART

resource centers in terms of performance and reliability. Grid benchmarks can be used by middleware developers to compare different middleware solutions such as job submission services, resource allocation policies, and scheduling algorithms. Grid benchmarks can serve as an evaluation of the fitness of a collection of Grid resources for running a specific application or class of applications. Benchmarks can also help study the effect of the dynamic nature of Grids to application performance and gain insights into the properties of Grid architectures.

However, the complexity, the heterogeneity, and the dynamic nature of Grids raise serious questions about the overall applicability and cost-effectiveness of Grid benchmarking. Performance measurements are affected by a variety of factors, including the characteristics of resources allocated for a particular run, the time-dependent latency and bandwidth of shared Internet links, and the performance capacity of middleware libraries used at the application level. Existing platforms are largely under continuous re-design and development, with very limited cross-platform interoperability, making the specification, submission, and management of jobs a tedious process.

In this paper, we examine the challenges that arise in the context of Grid benchmarking and provide a survey of ongoing Grid benchmarking research. The paper is organized as follows: Section 2 provides a short overview of benchmarking principles; Section 3 presents the key characteristics of Grids and explains why Grid benchmarking is important for Grid infrastructures; Section 4 discusses the key challenges that need to be addressed in the Grid context in order to enhance the integration of Grid benchmarking in Grid operations; Section 5 surveys ongoing research efforts in the area of Grid benchmarking; we conclude in Section 6.

2. BENCHMARKS AND BENCHMARKING

2.1. Performance benchmarking

Benchmarks are used to investigate the performance capacity and behavior of computer systems under carefully tuned workloads that stress particular aspects of system performance. Moreover, they are used extensively to guide the optimization and assessment of system design and implementation, and to help researchers establish quantitative arguments in systems research. A benchmark program can be: (i) a small probe (micro-benchmark) designed to measure an isolated aspect of system performance; (ii) an application kernel, which corresponds to the computationally demanding part of a real application; (iii) a micro-kernel, that is a synthetic program mimicking some real workload; or (iv) a full application, representative of an important class of applications.

Benchmarking is the process of running some benchmark(s) on a particular system in order to measure the resulting performance of the system under conditions representative of a real-world workload (see Figure 1). Benchmarking experiments should be carried out according to well-defined rules for execution, measurement, and reporting. Furthermore, benchmarks have to meet certain requirements [1]: For a benchmark to be easily portable to different platforms and accepted by wider audiences, it should be *easy to obtain and use*. The specification of the benchmark and its execution context should be *openly available*, *self-contained*, and *concise*. Moreover, its compilation, execution, performance measurement, metrics' storage, and management should be *affordable* in terms of costs such as the management of the benchmarking process, hardware or software involved, execution time, and storage. Performance metrics derived from benchmarking experiments should be easily associated with the structure of the corresponding benchmarks and the characteristics of the system under scrutiny,

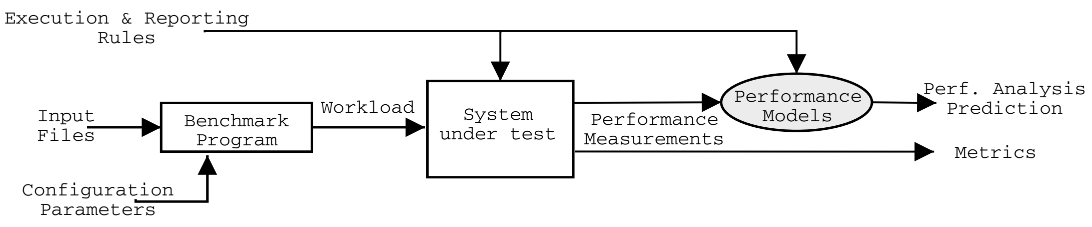

Figure 1. The process of performance benchmarking.

in order to *facilitate the derivation of useful conclusions*. Benchmarks are expected to be *fair*, in the sense that they should not favor particular system designs or implementations. Last, but not least, a benchmark program and its associated input parameters and files should result to workloads that are *representative of realistic conditions* of use for the systems under study.

The use of benchmarking for computer system performance analysis, however, comes with various shortcomings [1,12]. These shortcomings are usually related to the fairness and relevance of benchmarks, the usefulness and the cost of benchmark measurements: designers often optimize their systems according to well-known benchmarks in order to achieve improved benchmark results, even if this does not translate to improved performance over real-life applications. Several studies have shown that the configuration of a benchmarking experiment (choice of compilation flags, input parameters, input files, etc.) may have a significant effect on how representative or relevant its measurements are. Furthermore, typical benchmarks produce a few performance metrics, which do not constitute a thorough characterization of the systems under study. The design of benchmarks producing realistic workloads for network-centric systems, such as those that comprise the Internet infrastructure, involves complex, multi-component setups, which render benchmarking overly complicated and expensive.

2.2. Dependability benchmarking

In the case of distributed, service-oriented systems, such as those that support Internet services and Grid computing, *dependability* becomes equally important to performance, because of the very high cost that service outage and degraded operation have to service-level objectives and the user-perceived quality of service [13,14]. The dependability of a system is defined as its ability to deliver specified services, and can be described in terms of metrics such as: (i) the *robustness* that the system displays under stress conditions; (ii) system *availability*, i.e. the probability that a system works properly at some point in time; (iii) system *reliability*, which is a measure of the system's ability to provide proper service over a period of time; (iv) system *maintainability*, which estimates the time it takes to restore proper system function following a failure; and (v) system *performability*, which quantifies the system's performance in the presence of perturbations that affect its operation, including hardware, software, and external network failures, operator errors, and unusual workload conditions, such as those triggered by flash crowds, denial of service attacks, and targeted attacks that exploit system vulnerabilities.

A few studies have proposed benchmarking as a mechanism for assessing system robustness by creating systematically workloads that stress system capabilities and unveil their vulnerabilities [15,16].

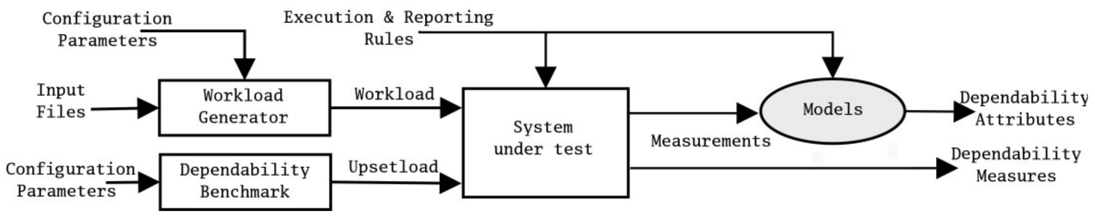

Figure 2. The process of dependability benchmarking.

Going beyond robustness benchmarking, a number of studies are investigating the development of benchmarking frameworks for evaluating system dependability and performability [17–19]. The goal of performability/dependability benchmarking is to 'provide cost-effective ways to evaluate the behavior of components of computer systems in the presence of faults, allowing the quantification of dependability attributes or the characterization of the systems in well defined dependability classes' [18]. To this end, performability benchmarking frameworks try to define the experimental setting, the workload, the perturbation load, and the metrics required to capture and represent dependability in a fair and systematic way (see Figure 2).

3. THE GRID SETTING

The Grid is emerging as a very large, distributed computing infrastructure that seeks to support resource sharing and coordinated problem solving in dynamic, multi-institutional VOs [20]. Typical Grid infrastructures, such as EGEE [21], comprise large numbers of heterogeneous resources (hardware and software), distributed across multiple administrative domains and interconnected through an open network. Access to those resources is provided to VO members through the Grid middleware, which exposes high-level programming and communication functionalities to application programmers and end-users, enforcing some level of *resource virtualization* [22]. VO membership and service brokerage is regulated by policies of access and usage agreed among the infrastructure operators, the VOs, the resource providers, and the recourse consumers.

The Grid infrastructure operator manages a number of central services that handle the authentication and authorization of resource consumers, monitor resource usage, and collect accounting information (Figure 3). Within a Grid infrastructure, the matchmaking between resource requests and available resources is done by the *Resource Broker*, a middleware component belonging to the VO's *Workload Management Services* (WMS) [23,24]. The Resource Broker interprets the resource requirements of a job submitted for execution to the VO. Then, it chooses a set of appropriate resources, using resource usage and availability information retrieved from the Grid information service. Subsequently, the WMS reserves the required resources, which may span across multiple administrative domains, and schedules the execution of the submitted job. During job execution, and with the help of the VO's monitoring services, the WMS monitors resource consumption and compliance to VO policies, providing feedback to the end-user; if necessary, it stops the job or manages its migration to a different set of resources.

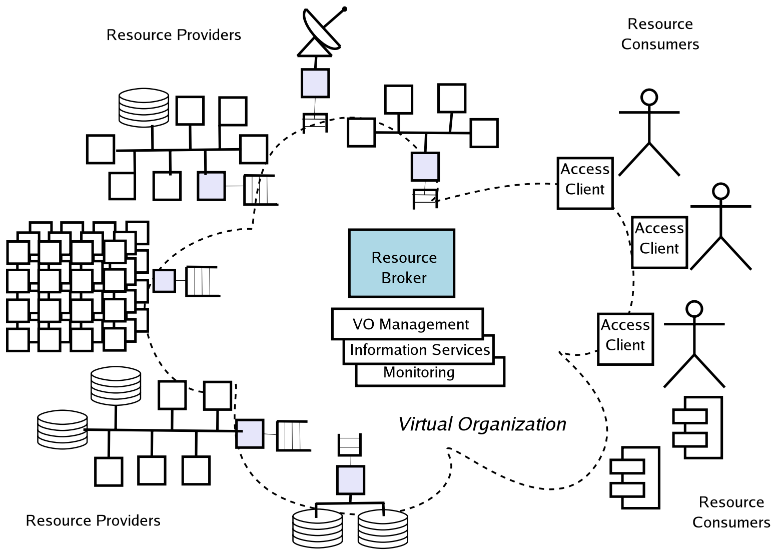

Figure 3. Grid architecture.

It is expected that the Grid will operate as an open marketplace, where different VOs offer access to a variety of computational resources at a certain price. In this context, a resource provider that plans to offer computational services through some Grid infrastructure will first have to negotiate the terms of its participation to the infrastructure with infrastructure operator (see Figure 4). At the other end of the marketplace, a resource consumer that wishes to join a VO will have to first evaluate and compare the resource offerings of different infrastructures and VOs. These negotiations will take into account the 'quality' of the resources contributed by a resource provider to the VO, and the resources made available by the VO to the resource consumer. Quality should incorporate aspects such as the functionality, the performance capacity, the availability, and the performability of those resources. The outcome of these negotiations will be encoded in contracts (Service Level Agreements (SLAs)) between the resource provider and the VO Operator, and between the VO Operator and the resource consumer.

The negotiation process outlined above is essentially a dynamic procurement procedure, the results of which have to be established upon some commonly accepted set of quantitative metrics, and on commonly accepted ground rules on how to produce those metrics. The traditional practice for deriving such metrics, as discussed in Section 2, is through benchmarking. Nevertheless, the key defining principles of Grids raise several challenges and issues vis-à-vis the requirements that need to be satisfied so that benchmarks and benchmarking results become acceptable and useful in the context of the Grid.

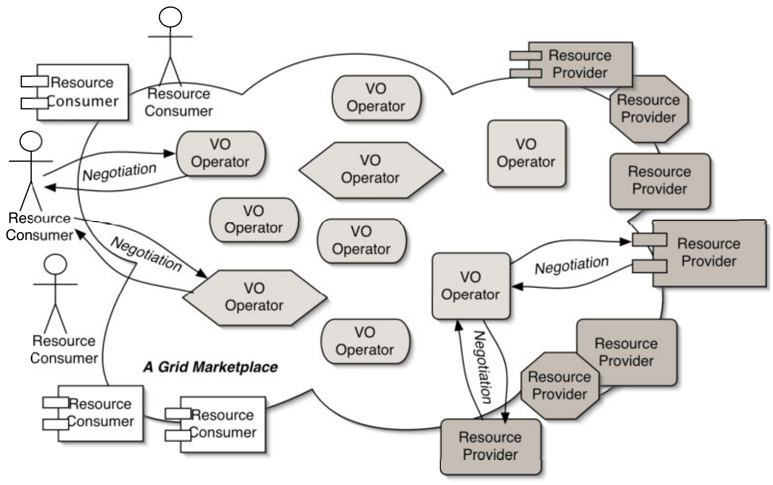

Figure 4. A Grid marketplace.

4. GRID BENCHMARKING CHALLENGES

We define as *Grid benchmarking* the use of benchmark programs for the fair, concise, and affordable performance characterization of different aspects of a Grid infrastructure. The applicability of Grid benchmarking in the context of the Grid, and issues such as the choice, the proper interpretation, and the reliability of benchmarking metrics and the cost of administering Grid benchmarks, have to be addressed while taking into account issues raised by the openness and scale of Grid infrastructures and by the virtualization imposed from the Grid middleware. Our conjecture is that traditional benchmarks and their context of use cannot be ported directly to a Grid setting, for the reasons described below.

4.1. Performance representation

Grids are inherently complex structures, consisting of hierarchical collections of heterogeneous resources and being managed by layers of interacting software components [25,26]. A Grid infrastructure comprises geographically distributed *Grid sites*, which belong to different administrative domains, communicate through Internet, and make their resources available to one or more VOs (see Figure 5) [20,21]. Typically, a *Grid site* has a cluster of computing and storage nodes interconnected through a high-speed local-area network. Each Grid site also hosts *local services*, which manage the site's membership to a Grid infrastructure, providing remote access to local resources and information. The operation of Grid infrastructures is supported by *central services*, which provide information about the configuration and state of infrastructure resources, facilitate job submission, job control, etc.

This multi-layered structure of the Grid suggests that the observed performance of Grid services and applications is affected by a variety of factors, such as [27]: (i) the performance capacity of

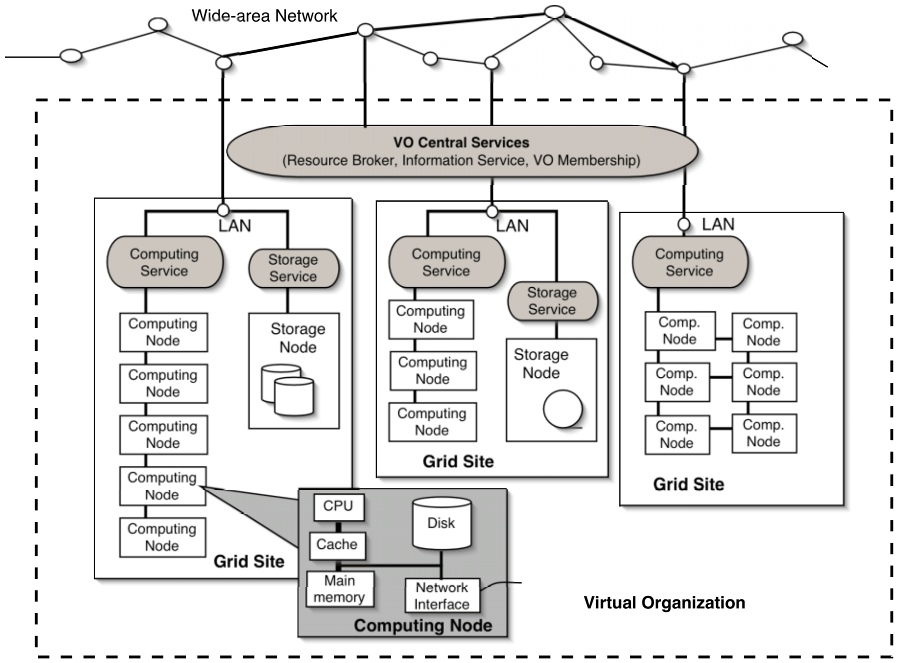

Figure 5. A typical Grid architecture.

local Grid sites (this depends on the performance of hardware entities, such as CPUs, memory hierarchies, computing nodes, storage devices, local networking, and clusters); (ii) the performance of the wide-area networks connecting Grid sites into a Grid infrastructure; (iii) the performance and overhead of libraries and services providing Grid applications with support for communication, synchronization, bulk data transfer, database querying, and other higher-level Grid programming abstractions; (iv) the performance and overhead of Grid services supporting job submission and management, such as workload management systems, resource brokers, and Grid information services; and (v) the reliability and the robustness of Grid middleware and services.

One of the principal goals of benchmarking is to enable the characterization and comparison of competing systems under a variety of realistic conditions. Owing to the complexity of the Grid, however, meaningful comparisons of Grids cannot be established upon a small set of performance metrics. Instead, we need an *ontology of performance metrics* describing different aspects of Grid performance at different levels of abstraction. An *ontology* is a 'formal explicit description of concepts (also called *classes*) in a domain of discourse, properties (also called *slots* or roles) of each concept describing various features and attributes of the concept, and restrictions on slots (also called *facets*)' [28]. An ontology together with a set of individual instances of classes constitutes a *knowledge base* [28]. We provide a simple example of such an ontology in Figure 6. According to this example, the representation of Grid performance is organized in classes of metrics, which correspond to the

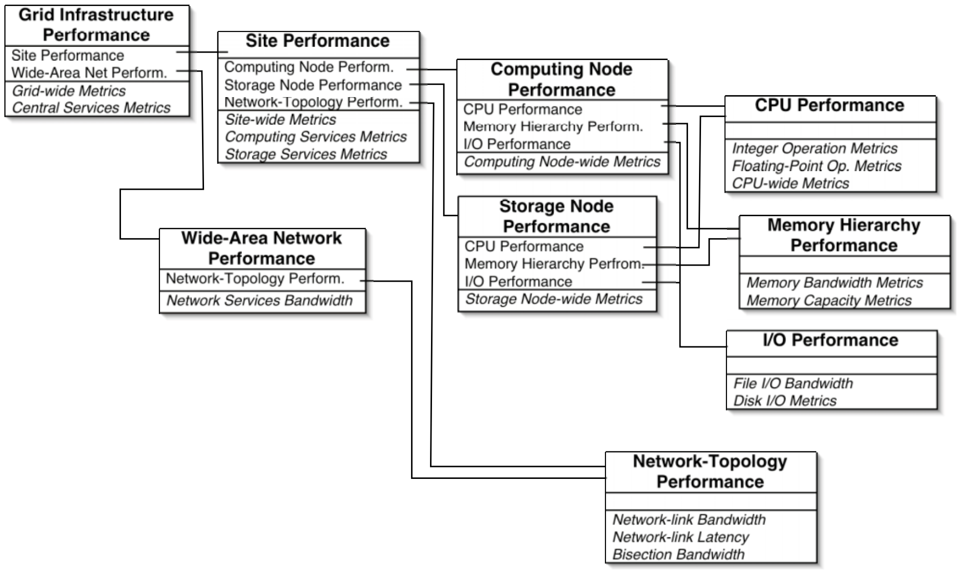

Figure 6. A simple Grid performance ontology: each class contains as attributes aggregated and direct metrics; direct metrics are presented with the oblique font.

basic abstractions of the Grid architecture of Figure 5. The performance of each class in our ontology is specified through the slots (properties) of the class. Our ontology supports two types of slots.

1. *Extrinsic* slots correspond to metrics that can be derived through direct performance measurements of the respective Grid entities. For example, the measured completion time of the High-Performance Linpack benchmark [5] and a measured performance indicator of some Grid service can be adopted as extrinsic performance metrics for Grid sites. The former is a metric representing the performance capacity of the site running parallel numerical calculations (a 'site-wide' metric), whereas the latter is a metric representing the performance of a service associated with that particular site.
2. *Instance* slots of an ontology class refer to the performance of its constituent Grid-entity abstractions. For example, the instance slots of a Grid site include the performance of that site's computing nodes, storage nodes, and local network (see Figure 6).

We refer to extrinsic slots as *direct performance metrics* and instance slots as *aggregated performance metrics*.

When measuring direct performance metrics, measurements may have to be repeated in order to derive values of sufficient statistical significance that can be distilled into cumulative distribution functions. Hence, we will be able to provide more understandable and concise metrics losing, however, potentially useful information. Finally, when striving to characterize higher-level abstractions of the Grid architecture, we need to go beyond direct and aggregated metrics and describe qualitative factors that affect observable Grid performance, such as the heterogeneity, the robustness, the reliability, and the level and quality of maintenance of Grid infrastructures. Such descriptions could be derived by models that combine direct and aggregated metrics, providing *derived metrics*.

The definition, organization, and storage of Grid-performance metrics represents an important challenge. To address this challenge, we need to come up with expressive data models, amenable to statistical, data-mining or artificial intelligence post-processing. These models must be described in open and extensible formats and must be widely adopted by different tools and middleware systems, in order to encourage the wide acceptance and use of metrics by other Grid subsystems.

4.2. Measuring performance

4.2.1. Choosing the benchmarks

A Grid performance ontology should be instantiated into a *Grid performance knowledge-base* through a series of benchmarking experiments that produce instances for the classes of the ontology. To this end, we need to design benchmarks that investigate the performance behavior of Grid entities belonging to different layers of the Grid architecture, from CPUs and memory hierarchies to whole sites and central Grid services (Figure 5) [27,29]. In particular, we can use the following.

- *Micro-benchmarks* for 'stress-testing' and measuring the performance of some Grid entity in isolation. For example, using a micro-benchmark we can estimate the performance of a network link in terms of the file-transfer time between two different Grid sites, or the performance of a CPU in terms of the floating point operations per second achieved on some chosen program.
- *Micro-kernels*, for measuring the performance of some composite Grid entity under some synthetic workload designed to stress-test simultaneously several aspects of the entity's performance. For example, we can estimate the performance of a Grid site's cluster in terms of the running time of the parallel High-Performance Linpack benchmark [30] or the performance of a single-processor computing node using a sequential version of Linpack.
- *Application kernels* and *Grid applications* used to investigate the performance of some Grid entity (node, site, infrastructure) under realistic workload conditions. It should be noted, however, that on the Grid there is still a lack of widely accepted applications and prevalent programming paradigms. Consequently, it may be too early to identify a set of 'representative' Grid applications after which application benchmarks can be modeled, and to choose an appropriate programming paradigm for developing such benchmarks.

The success of benchmarking as a performance characterization approach depends not only on the choice of benchmarks but also on benchmark tuning, that is, on the selection of input values, files, and configuration parameters. The right tuning is important for the benchmarking experiments to be realistic and relevant with respect to the measured resources (main memory size, nominal network bandwidth, etc.). Owing to the Grid's heterogeneity and scale, however, the choice of parameter values and input files can vary between different parts of an infrastructure. In the context of large Grids, it may be very expensive to tune benchmarks manually; therefore, we need to have systems and algorithms that tune the benchmarks automatically, taking into account the configuration of the underlying Grid infrastructure.

We also need common and open standards for representing benchmark specification and benchmark execution conditions. The existence and wider adoption of such standards will enhance the trust that Grid operators and end-users have on published benchmarking results, and will allow the interoperability of benchmarking tools with other Grid middleware components. The definitions and properties of benchmarks can be a part of an extended performance ontology, going beyond that presented in Figure 6.

4.2.2. Measuring a moving target

The underlying assumption when benchmarking a large-scale computing system (such as a massively parallel machine) is that the system is readily available, properly functioning, and that its configuration is well known and not subject to change; furthermore, that the system is homogeneous in terms of hardware and software setup. These assumptions do not hold for Grids. Owing to their scale and lack of central administrative control, the likelihood that some resources are *not* available for benchmarking and/or functioning properly at any given time is increased. Furthermore, Grid jobs are susceptible to total or partial failure and degraded performance, due to problems that may occur across the Grid infrastructure; usually, such problems are prompted by operator error, maintenance activities, and various faults (network, hardware, or software), which lie outside end-user control. Finally, the heterogeneity of Grids may raise the need for a different customization of benchmarking parameters for the differing parts of a Grid infrastructure. Consequently, it is expected that the collection of representative and acceptable performance metrics will require repeated and carefully tuned benchmarking sessions. Therefore, we need to do the following.

- Validate the advertised configuration, state, and functionality of Grid resources, possibly running quick end-to-end tests designed to discover problems that affect the functionality, robustness, and overall performance of a Grid.
- Capture the actual subset of Grid resources assigned to a running benchmark, along with their status; store this representation together with the measured metrics in order to facilitate further analyses.
- Conduct successive benchmarking experiments until we get a complete and up-to-date performance snapshot of a whole infrastructure. Merge individual performance measurements into a coherent set of metrics.
- Repeat benchmarking experiments in order to derive measurements of adequate statistical significance and to keep stored metrics up to date, when the infrastructure changes.

4.2.3. Trusting the measurements

Programs submitted to the Grid are executed on 'virtual' machines, dynamically assembled immediately after job submission; the precise composition and architecture of those machines might not be known beforehand and may change between different runs or even within a single run. Also, Grid users do not have full control over the resources assigned to their jobs and have very limited influence on scheduling or re-scheduling decisions, on available bandwidth, and on network latency. Therefore, benchmarks submitted for execution to a Grid infrastructure may end-up sharing the same resources with other tasks, which have been co-allocated by a Resource Broker, or run without authorization from the controlling VO ('free-riders'), or are runaway processes. These tasks can affect the benchmark execution time and 'pollute' the derived metrics. In such a non-dedicated and possibly transient infrastructure, it is not easy to obtain sets of reproducible measurements that can be easily trusted and widely accepted.

The problem of measurement pollution can be addressed through *policy mechanisms* applied at the Grid infrastructure and VO level. Such mechanisms could handle benchmarking as a maintenance activity, during which the middleware ensures the exclusive use of resources by suspending non-benchmarking jobs or withholding the submission thereof, and by killing runaway or free-rider processes. In cases where such policy mechanisms cannot be implemented or enforced, we need to come up with approaches for assessing the trustworthiness of measured metrics and filtering out polluted measurements.

4.3. Metrics interpretation

Benchmarking metrics, as such, do not provide direct insights about the causes of a particular performance behavior. In fact, the more complex a system is, the more difficult it becomes to combine individual measurements into a meaningful system analysis and to predict the performance of real applications. Consequently, the proper interpretation of Grid benchmarking metrics is a challenging task, which requires the combination of benchmarking metrics with appropriate models for the Grid and its applications. To the best of our knowledge, however, the field of Grid modeling is still at its infancy, although some promising results have already appeared in the literature [31].

Therefore, we need new and mature theoretical and practical tools that will support the interpretation of information stored in Grid performance knowledge-bases, guiding the mapping of applications to Grid resources, the prediction of Grid application performance, the performance debugging of applications, the re-configuration of Grid infrastructures, etc.

5. A BRIEF SURVEY OF CURRENT APPROACHES

Grid benchmarking has been an active field of research in recent years. A Research Group is dedicated specifically to Grid benchmarking [32] within Global Grid Forum, the organization that coordinates standardization of Grid architectures and protocols [33]. A number of research groups and projects have focused on different aspects of Grid benchmarking, proposing benchmark specifications, benchmarking suites, and benchmarking tools for Grids. In this section, we provide a brief overview of these efforts.

5.1. Benchmarking Grid infrastructures

As mentioned earlier, the multi-layered structure of the Grid architecture calls for benchmarks that investigate performance aspects of the different Grid layers. A number of research efforts focus on the highest layer of the Grid architecture, considering Grid infrastructure as a single, unified resource that should be characterized through carefully selected and tuned benchmarks. This approach follows naturally from one of the key goals of Grid computing, which is the complete virtualization of distributed resources behind simple APIs and communication protocols. Research efforts adopting this approach seek to develop and validate benchmarks representative of demanding Grid workloads. Such benchmarks can be used to derive metrics of Grid-infrastructure performance or to investigate the behavior of Grid subsystems.

5.1.1. NAS Grid Benchmarks

One of the first proposed Grid benchmarks is the NAS Grid Benchmark (NGB) suite, also known as the suite of ALU Intensive Grid Benchmarks (AIGB) [34,35]. The NGB suite comprises paper-and-pencil specifications of different computations, along with recommendations for data-structure sizes. NGB authors also provide a reference implementation of their benchmarks [36]. The NGB specifications define a set of computationally intensive, synthetic Grid benchmarks, which are representative of scientific, post-processing, and visualization workloads. The benchmarks are structured as dataflow graphs of communicating components. The components (nodes) of the NGB graphs are programs selected from the NAS Parallel Benchmark (NPB) suite [2]. The communication between these components carries initialization and control information rather than actual data results. NGB comprises four different classes of data-flow graphs, each corresponding to a different communication pattern: Embarrassingly Distributed, Helical Chain, Visualization Pipeline, and Mixed Bag. The selection of the computation and communication patterns implemented by the NGB is supposed to represent characteristic Grid workloads arising from loosely coupled and pipelined Grid applications [35]. *Job turnaround time* is proposed as the performance metric to be reported with NGB experiments [35], although the NGB authors recommend the reporting of performance measurements for individual NGB tasks and communications.

NGB was designed to serve as a ‘uniform tool for testing functionality and efficiency of Grid environments’ [35]. The mapping and scheduling of NGB jobs to some Grid infrastructure, however, as well as the conditions of their execution, are left to the Grid environment and are not a part of the configuration or the output of the benchmarks.

A few recent studies have published experiments with implementations of the NGB [37,38]. In [37], the NGB authors report experiments conducted with two implementations of NGB, deployed on the NASA Ames nodes of NASA’s Information Power Grid (IPG) [39]. The discussion presented in [37] shows that the configuration, administration, and analysis of NGB experiments requires an extensive manual effort. Furthermore, the results reported show that the job turnaround time metric provides little if any insights about a Grid, even if it is derived through restricted benchmarking experiments that involve machines under the same administrative domain and job submissions overwriting Grid scheduler decisions. The reason is that a large number of factors, which are outside the NGB specification, determine the success or failure of a submitted NGB job and its performance measurements. For example, the orchestration of the data-flow computation, the use (or not) of local queuing systems, network traffic, file-system configuration, the IPG service setup, and the effect of other jobs running on the same resources. All of these factors, however, need to be taken into account and reported when trying to analyze job turnaround time. Consequently, it is not clear whether there is anything to be learned from such a simple performance metric for the efficiency and the functionality of a Grid infrastructure.

In [38], the authors use NGB to estimate the overhead of job submission Grid services. To this end, they submit NGB jobs to a local cluster that comprises two multiprocessor nodes and runs the Sun Grid Engine and Globus. Measurements show that job submission through the Grid middleware incurs extra delays, but that these delays become minimal for longer computations. Furthermore, experiments show that submission through Globus results in a higher overhead than submission through the Sun Grid Engine. The paper, however, provides little information about the NGB implementation used and the handling of NGB data-flow dependencies. Furthermore, it is not clear how these results would scale in larger and open Grids.

5.1.2. Grid Assessment Probes

An approach complementary to NGB proposes the use of micro-benchmarks instead of synthetic applications for measuring Grid infrastructures. This approach is developed in the context of the *Grid Assessment Probes (GRASP)* project of the San Diego Supercomputing Center [40]. GRASP times the lifecycle of a Grid micro-benchmark running on Globus, including authentication, resource discovery, and validation. GRASP codes implement three alternative patterns of data exchange (3-node, circle, gather) between different Grid nodes and capture the performance of basic Grid operations such as file transfers and remote execution. A series of experiments using the benchmark probes have been executed on three nodes of the GrADS testbed [41], providing insights into the effects that network variability has on measured end-to-end performance and uncovering testbed configuration problems.

5.2. Benchmarking Grid services

The performance of Grid middleware services affects the overall quality of a Grid environment by adding extra overhead to job turnaround times and delaying crucial operations, such as workload management, job monitoring, and control. Furthermore, badly designed or mis-configured Grid services may not be able to sustain a high-rate of job submissions, the timely update of published Grid state information, and the effective allocation of Grid resources.

In an effort to isolate and evaluate quantitatively the performance of Grid services, researchers have recently proposed synthetic micro-kernels that target Grid Information and Monitoring Services (GIS) [42,43]. The GIS micro-kernels generate sequences of queries, submitted to different GISs, and report measurements of the *query response time* and *throughput* (average number of requests processed by a service per second). Additional metrics supported are the *ease of use*, which quantifies the work that a client must undertake to obtain the desired information from the Grid service [42], and the average load of the machine running the service [43]. Implementations of the GIS micro-kernels were used to study the performance of alternative database systems for storing Grid information [42], and to compare the performance and scalability of three different GISs [43].

Grid-service benchmarks are important tools for assessing the performance of Grid services and for comparing alternative services or different service configurations. Therefore, it is necessary to have more benchmarks targeting critical Grid services beyond information and monitoring, such as queuing systems, security services, and resource brokers. We anticipate that the need for such benchmarks will be increased as Grids move towards a fully service-oriented architecture. The design of adequate benchmarks, however, will require a better understanding of Grid service workloads, which can be made possible through Grid characterization studies. Furthermore, it will be necessary to address the high cost of Grid-service benchmarking administration and results analysis. Finally, new benchmarks are required to measure Grid-service dependability, which is a primary concern in current large Grid infrastructures.

5.3. Benchmarking Grid resources

An important factor that needs to be taken into account when mapping applications to the Grid is the performance of hardware resources made available through its VOs (sites, computing nodes, file servers, network links, etc). Benchmarking represents the obvious choice for characterizing the performance of Grid resources quantitatively. The performance characterization of Grid resources at different layers of the Grid architecture has been examined in the context of the CrossGrid project [44] and led to the development of the GridBench tool and suite of benchmarks [45].

GridBench follows a layered approach to characterize Grid resources at different layers of abstraction, in accordance to the Grid architecture of Figure 3 [27,46]. Most of the codes included in the GridBench suite are not new; they are open-source, sequential, and parallel benchmarks, which have been tested and used extensively in the past and have been ported to the Grid for the needs of GridBench (e.g. STREAM, Linpack, Whetstone, MPPTest). In addition to pre-existing benchmarks, the GridBench suite incorporates kernels extracted from the computationally intensive applications developed in the context of CrossGrid [44].

GridBench benchmarks were chosen in order to provide measurements for an ontology of metrics similar to that presented in Figure 6. The GridBench benchmarks have been used to characterize the performance capacity of resources belonging to the CrossGrid testbed [47], a large Grid infrastructure consisting of 18 sites distributed throughout Europe [49]. Also, GridBench was recently ported and demonstrated on the pan-European Grid infrastructure of EGEE [21].

The usefulness of GridBench-derived metrics was demonstrated in a scenario involving the deployment of a computational hemodynamics Grid application developed by the University of Amsterdam [48] on the CrossGrid testbed. In particular, we used GridBench micro-benchmarks to conduct a detailed characterization of the CrossGrid infrastructure, and used the metrics gathered to guide the selection of resources on which the hemodynamics application could run and achieve satisfactory performance [50].

Early demonstrations have shown that Grid-resource performance metrics can be used successfully to guide resource allocation [29,23] and application scheduling [51]. However, more work is required in the context of very large heterogeneous infrastructures where the collection, administration, and update of performance metrics is a difficult task.

5.4. Grid benchmarking tools

As explained earlier, benchmarking Grids can be hard and very expensive. Therefore, we need powerful, Grid-enabled tools to facilitate Grid benchmarking experimentation. Issues related to the development of a tool for Grid benchmarking have been addressed in the context of the GridBench work [45]. GridBench is an integrated tool designed to facilitate the configuration and administration of Grid benchmarks, the easy integration of new Grid benchmarks, the validation of collected metrics, the integration of metrics with benchmark specifications, and the analysis of benchmarking results.

GridBench is built around the GridBench Definition Language (GBDL) [46], which is an XML language that integrates benchmark specifications (code, parameters, metrics), the configuration of benchmarking experiments, and infrastructure monitoring data associated with particular experiments. GBDL records are created through the GridBench graphical-user interface, and populated with data extracted from GISs during benchmark submission and execution.

GridBench translates GBDL benchmark specifications to Grid job definitions encoded in a job submission language of choice (for instance, RSL for Globus or JDL for Condor/EDG) and manages the submission and monitoring of benchmarking jobs, the collection of metrics, and the storage of results. An analysis module allows for the graphical visualization of metrics in various forms and supports the performance comparison between different Grid sites.

6. CONCLUSIONS

The transition of the Grid into an open marketplace of computational resources requires, among other things, the operation of performance-aware Grid services and tools. These services should rely on up-to-date, reliable, widely acceptable, and accessible performance metadata, characterizing the performance capacity of Grid services and hardware resources. The existence of Grid benchmarking services and tools is necessary for consolidating such metadata. However, the design and development of benchmarks for Grids faces several challenging obstacles. Furthermore, Grid benchmarking is a difficult and costly endeavor. Therefore, several research advances are required in order to integrate benchmarking practices and techniques in the Grid middleware. In particular they include the following.

- The development of a wide range of Grid benchmarks that are representative of the various classes of emerging Grid applications and can be used to measure the performance at the different layers of the Grid architecture. The introduction of benchmarks measuring the dependability of Grids and Grid subsystems.
- The definition of ontologies of metrics that are widely acceptable and can be easily combined with specifications of benchmarks and benchmarking conditions, and with ontologies for Grid resources, services, and applications. The development of open and extensible data models for encoding these ontologies, allowing the easy integration of performance metrics with Grid configuration and monitoring information.
- The development of user-friendly tools that facilitate the management of Grid benchmarking experiments by Grid administrators, application developers, and end-users. Also, the development of techniques for automated and adaptable administration and management of benchmarking experiments.
- New models and modeling tools to enable the interpretation of benchmarking results and the mining of interesting properties out of knowledge bases that comprise information about benchmarking experiments, metrics, and infrastructure architecture, configuration and state.

ACKNOWLEDGEMENTS

Work supported in part by the European Union under the Working Group on Automatic Performance Analysis: Real Tools (APART).

REFERENCES

1. Weicker R, Benchmarking. *Proceedings of Performance 2002* (*Lecture Notes in Computer Science*, vol. 2459), Calzarossa MC, Tucci S (eds.). Springer: Berlin, 2002; 179–207.

2. Bailey DH *et al*. The NAS parallel benchmarks. *The International Journal of Supercomputer Applications* 1991; **5**(3):63–73.
3. Dikaiakos MD, Kyriakou M, Samaras G. Performance evaluation of mobile-agent middleware: A hierarchical approach. *Proceedings of the 5th International Conference on Mobile Agents (MA 2001)* (*Lecture Notes in Computer Science*, vol. 2240), Picco GP (ed.). Springer: Berlin, 2002; 244–259.
4. Dongarra JJ, Hey T, Strohmaier E. PARKBENCH: Methodology, relations, and results. *High-Performance Computing and Networking (HPCN'96 Europe)* (*Lecture Notes in Computer Science*, vol. 1067), Liddell H, Colbrook A, Hertzberge B, Sloot P (eds.). Springer: New York, 1996; 770–777.
5. Dongarra JJ. Performance of various computers using standard linear equations software (Linpack Benchmark Report). *Technical Report CS-89-85* (Updated version), Department of Computer Science, University of Tennessee, 2004.
6. Gray J. *The Benchmark Handbook for Database and Transaction Processing Systems* (2nd edn). Morgan Kaufmann: San Mateo, CA, 1993.
7. Manley S, Seltzer M, Courage M. A self-scaling and self-configuring benchmark for Web servers. *Proceedings of the 1998 Sigmetrics Conference on Measurement and Modeling of Computer Systems*. ACM Press: New York, 1998; 270–272.
8. Samaras G, Dikaiakos MD, Spyrou C, Liverdos A. Mobile agent platforms for Web-databases: A qualitative and quantitative assessment. *Proceedings of the Joint Symposium ASA/MA'99 1st International Symposium on Agent Systems and Applications (ASA '99) and 3rd International Symposium on Mobile Agents (MA '99)*. IEEE Computer Society Press: Los Alamitos, CA, 1999; 50–64.
9. Transaction Processing Performance Council (TPC). *TPC Benchmark W (Web Commerce)—Draft Specification*, December 1999.
10. Woo SC, Ohara M, Torrie E, Singh JP, Gupta A. The SPLASH-2 programs: Characterization and methodological considerations. *Proceedings of the 22nd Annual International Symposium on Computer Architecture*. ACM Press: New York, 1995; 24–37.
11. Dongarra J, Gentzsch W (eds.). *Computer Benchmarks*. North-Holland: Amsterdam, 1993.
12. Gustafson J. Purpose-based benchmarks. *International Journal of High Performance Computing Applications* 2004; **18**(4):475–487.
13. Patterson DA. A simple way to estimate the cost of downtime. *Proceedings of LISA '02: 16th Systems Administration Conference*. USENIX Association: Berkeley, CA, 2002; 185–188.
14. Patterson DA. Availability and Maintainability $\gg$ Performance: New focus for a new century. *Keynote Address, USENIX Conference on File and Storage Technologies (FAST '02)*, January 2002. USENIX Association: Berkeley, CA, 2002.
15. Koopman P, Sung J, Dingman C, Siewiorek D, Marz T. Comparing operating systems using robustness benchmarks. *Proceedings of the 16th IEEE Symposium on Reliable Distributed Systems*. IEEE Press: Piscataway, NJ, 1997; 72–79.
16. Miller B, Fredriksen L, So B. An empirical study of the reliability of UNIX utilities. *Communications of the ACM* 1990; **33**(12):32–44.
17. Brown A, Patterson DA. Towards availability benchmarks: A case study of software RAID systems. *Proceedings of the 2000 USENIX Annual Technical Conference*. USENIX Association: Berkeley, CA, 2000; 263–276.
18. Madeira H, Koopman P. Dependability benchmarking: Making choices in an n-dimensional problem space. *Proceedings of the Workshop on Evaluating and Architecting System dependabilitY (EASY), International Conference on Dependable Systems and Networks (DSN) 2001*, July 2001. Available at: http://www.crhc.uiuc.edu/EASY/easy01-program.html [May 2006].
19. Oppenheimer D, Brown AB, Traupman J, Broadwell P, Patterson DA. Practical issues in dependability benchmarking. *Proceedings of the 2nd Workshop on Evaluating and Architecting System Dependability (EASY)*, 2002. Available at: http://www.crhc.uiuc.edu/EASY/easy02-final-program.html [May 2006]
20. Foster I, Kesselman C, Tuecke S. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications* 2001; **15**(3):200–222.
21. EGEE: Enabling Grids for eScience in Europe. http://www.eu-egee.org [April 2005].
22. Xu M, Hu Z, Long W, Liu W. Service virtualization: Infrastructure and applications. *The Grid: Blueprint for a New Computing Infrastructure*. Elsevier: Amsterdam, 2004; 179–190.
23. Heymann E, Fernandez A, Senar MA, Saltv J. The EU-CrossGrid approach for Grid application scheduling. *Proceedings of Grid Computing: 1st European AcrossGrids Conference*, Santiago de Compostella, Spain, February 2003 (*Lecture Notes in Computer Science*, vol. 2970), Rivera F, Bubak M, Gomez-Tato A, Doallo R (eds.). Springer: Berlin, 2004; 17–24.
24. Thain D, Livny M. Building reliable clients and services. *The Grid: Blueprint for a New Computing Infrastructure*, Foster I, Kesselman C (eds.). Elsevier: Amsterdam, 2004; 285–318.
25. Bubak M, Malawski M, Zajac K. The CrossGrid Architecture: Applications, Tools, and Grid Services. *Proceedings of Grid Computing: 1st European AcrossGrids Conference*, Santiago de Compostella, Spain, February 2003 (*Lecture Notes in Computer Science*, vol. 2970), Rivera F, Bubak M, Gomez-Tato A, Doallo R (eds.). Springer: Berlin, 2004; 309–316.
26. Foster I, Kesselman C. Concepts and architecture. *The Grid: Blueprint for a New Computing Infrastructure*, Foster I, Kesselman C (eds.). Elsevier: Amsterdam, 2004; 37–64.

27. Tsouloupas G, Dikaiakos MD. GridBench: A tool for benchmarking Grids. *Proceedings of the 4th International Workshop on Grid Computing (Grid2003)*. IEEE Computer Society Press: Los Alamitos, CA, 2003; 60–67.
28. Fridman Noy N, McGuinness D. Ontology development 101: A guide to creating your first ontology. *Technical Report KSL-01-05*, Stanford Knowledge Systems Laboratory, October 2001.
29. Nudd GR, Jarvis SA. Performance-based middleware for Grid computing. *Concurrency and Computation: Practice and Experience* 2004; **17**:215–234.
30. Petitet A, Whaley RC, Dongarra J, Cleary A. HPL—a portable implementation of the High-Performance Linpack Benchmark for distributed-memory computers. http://www.netlib.org/benchmark/hpl/ [March 2005].
31. Hey AJG, Papay J, Surridge M. The role of performance engineering techniques in the context of the Grid. *Concurrency and Computation: Practice and Experience* 2005; **17**:297–316.
32. Grid Benchmarking Research Group. Global Grid Forum. https://forge.gridforum.org/projects/gb-rg [March 2005].
33. Global Grid Forum. http://www.ggf.org [April 2005].
34. Van der Wijngaart RF. *ALU Intensive Grid Benchmarks*. Global Grid Forum, Research Group in Grid Benchmarking, GWD-I, February 2004.
35. Frumking M, Van der Wijngaart RF. NAS Grid Benchmarks: A tool for Grid space exploration. *Cluster Computing* 2002; **5**(3):315–324.
36. NAS Grid Benchmarks. http://www.nas.nasa.gov/Software/NPB [May 2006].
37. Van der Wijngaart RF, Frumkin MA. Evaluating the Information Power Grid using the NAS Grid Benchmarks. *Proceedings of the High-Performance Grid Computing Workshop, 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Santa Fe, NM, April 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004.
38. Peng L, See S, Song J, Stoelwinder A, Neo HK. Benchmarks performance on Cluster Grid with NGB. *Proceedings of the High-Performance Grid Computing Workshop, 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Santa Fe, NM, April 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004.
39. NASA's Information Power Grid. http://www.ipg.nasa.gov [March 2005].
40. Chun G, Dail H, Casanova H, Snavely A. Benchmark probes for Grid assessment. *Proceedings of the High-Performance Grid Computing Workshop, 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Santa Fe, NM, April 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004.
41. Berman F *et al.* The GrADS Project: Software support for high-level Grid application development. *International Journal of Supercomputing Applications* 2001; **15**(4):327–344.
42. Plale B, Jacobs C, Liu Y, Moad C, Parab R, Vaidya P. Understanding Grid resource information management through a synthetic database benchmark/workload. *Proceedings of the 4th IEEE/ACM International Symposium on Cluster Computing and the Grid*. ACM Press: New York, 2004.
43. Zhang X, Freschl JL, Schopf JM. A performance study of monitoring and information services for distributed systems. *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*. IEEE Computer Society Press: Los Alamitos, CA, 2003.
44. European CrossGrid Project. http://www.crossgrid.org [April 2005].
45. GridBench: A tool for benchmarking Grids. http://grid.ucy.ac.cy/GridBench [April 2005].
46. Tsouloupas G, Dikaiakos MD. GridBench: A workbench for Grid benchmarking. *Proceedings of the European Grid Conference on Advances in Grid Computing (EGC 2005)*, Amsterdam, The Netherlands, 14–16 February 2005 (*Lecture Notes in Computer Science*, vol. 3470). Springer: Berlin, 2005; 211–225.
47. Marco J *et al.* First prototype of the CrossGrid testbed. *Proceedings of the Grid Computing 1st European AcrossGrids Conference*, Santiago de Compostella, Spain, February 2003 (*Lecture Notes in Computer Science*, vol. 2970), Rivera F, Bubak M, Gomez-Tato A, Doallo R (eds.). Springer: Berlin, 2004; 67–77.
48. Sloot PMA. Tirado-Ramos A, Hoekstra AG, Bubak M. An interactive Grid environment for non-invasive vascular reconstruction. *Proceedings of the 2nd International Workshop on Biomedical Computations on the Grid (BioGrid'04), in Conjunction with the 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2004)*, Chicago, IL, April 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004.
49. Tsouloupas G, Dikaiakos MD. Characterization of computational Grid resources using low-level benchmarks. *Technical Report TR-2004-5*, Department of Computer Science, University of Cyprus, December 2004. Available at: http://grid.ucy.ac.cy/reports/TR-04-5.pdf.
50. Tiramo-Ramos A, Tsouloupas G, Dikaiakos MD, Sloot P. Grid resource selection by application benchmarking: A computational haemodynamics case study. *Proceedings of the 5th International Conference on Computational Science (ICCS 2005)*, Atlanta, GA, May 2005 (*Lecture Notes in Computer Science*, vol. 3514). Springer: Berlin, 2005; 22–25.
51. Wolski R, Spring N, Hayes J. The Network Weather Service: A distributed resource performance forecasting service in metacomputing. *Journal of Future Generation Computer Systems* 1999; **15**(5–6):757–768.