

# GrenchMark: A Framework for Testing Large-Scale Distributed Computing Systems

Alexandru Iosup

Faculty of EEMCS, Delft University of Technology

email: A.Iosup@tudelft.nl

Supervisors: Prof. Dr. Henk Sips and Assoc.Prof. Dr. Dick Epema

## I. PROBLEM AND MOTIVATION

Today's large-scale distributed computing systems (LSDCSs) integrate (tens of) thousands of resources. Infrastructures such as the WLCG, the NorduGrid, the TeraGrid, the Open Science Grid etc., offer similar or better throughput when compared with large-scale parallel production environments [1], [2]. However, two problems threaten to cancel out the benefits of this integration. First, the integration brings about performance, reliability, and functionality issues. Second, there is a lack of published test data that the community can use to compare and improve existing solutions. To address these two problems, we present in this work GRENCHMARK, a framework for testing LSDCSs.

The complexity, the heterogeneity, or simply the sheer scale of the production LSDCSs expose their current problems. Early testing in real grids reveals much lower performance than expected from simulations [3], failure rates from 10% and up to 45% [4], [5], [6], [1]<sup>1</sup>, and functionality problems of around one in every three tests for widely-installed grid services [8]. Repeated and realistic testing is a proven industrial way to alleviate these problems, but such testing is difficult in LSDCSs: the requirements are poorly understood, there are few real workload traces to be used as input, and there is no tool that can generate realistic workloads for testing purposes.

The lack of commonly-used testing practices is wide-spread in LSDCSs. As a consequence, there are no comprehensive comparative studies, and there is no effort to publish anonymized test results in a standard format. This reduces the ability of LSDCSs to gain industrial acceptance, makes their tuning difficult and case-by-case only, and prevents scientists from identifying and focusing on meaningful research problems. For example, we show using GRENCHMARK that tuning the system to reduce trivial overheads can significantly improve the system's performance, and that for some classes of workloads relatively little performance remains to be gained by improving the scheduling algorithms alone [9].

The GRENCHMARK framework completely automates the testing process from workload specification to obtaining results. Compared with related work [10], [11], [12], [13], [14], it covers the complete LSDCS testing process, and has unique workload analysis and modeling, and results analysis and

storage features. Our reference implementation addresses the practical problems of testing, and, in particular, it can create appropriate and repeatable experimental conditions for a large variety of environments.

## II. BACKGROUND AND RELATED WORK

### A. Design Goals for LSDCS Testing Frameworks

In this section we synthesize the design goals for testing large-scale distributed computing systems. Our motivation is twofold. First, large-scale distributed computing systems (LSDCSs) have specific testing requirements, e.g., unique workload characteristics, the problem of scale. Second, in spite of last decade's evolution of tools for various testing purposes (see Section II-B), there is still place for improvement, especially in managing tests for the testing community, e.g., sharing best-practices, and test and provenance data storage. Below we present seven design goals (in four categories) specific to frameworks for testing LSDCSs. In Section II-B we evaluate the related work according to these design goals.

- 1) **Realism** To perform realistic tests, there is a need for realistic LSDCS workload generation. However, the realistic workloads are not a goal *per-se*; a workload may be realistic for one scenario, and unrealistic for another.
- 2) **Reproducibility** Compared with existing computing systems, LSDCSs have dynamic availability and varying load. However, unlike the Internet, in many cases the full system state can be recorded. Thus:
  - (a) **Same Environment** There is a need to ensure that tests are (re-)performed in experimental conditions that match the test design (e.g., test workload, background load).
  - (b) **Data Provenance** There must exist support for provenance – where a piece of data comes from, and through which process [15] – and result annotations. In some cases the experimental conditions must be recorded as a complement to the test results.
- 3) **High-Performance** The testing framework must cope with the scale of the tested system. In particular:
  - (a) **Workload Generation and Submission** The workload generation and submission are the only part of a testing framework whose performance is related to that of the tested system's. For example,

<sup>1</sup>The failure rate in today's grids is much higher than that of contemporary large-scale parallel production installations [7].

TABLE I

A SUMMARY OF TESTING PROJECTS IN DISTRIBUTED COMPUTING.

Project	Area	Sub-Area	Design Goals (see Section II-A)							
			1 realistic	2 reproducible		3 high-perf.		4 extensible		
			a	b	a	b	a	b		
GRECHMARK	G,P2P	all	all	+	+	+	+	+	+	
<i>Grid/Cluster benchmarking</i>										
HPCC [10]	CI	all	$\mu$ K,AB	+	-	-	+	-	-	
NAS NGB [16]	G	C,D	$\mu$ K	+	-	-	-	-	-	
GridBench [11]	G	C,D	$\mu$ K,AB	+	-	-	-	+	+	
<i>Grid performance evaluation</i>										
DtPerF [17]	G	C	SWL	+	-	+	-	+	+	
Quake [12]	G	C	SWL	+	-	-	-	-	+	
<i>Grid functionality testing</i>										
INCA [18]	G	C	P	+	~	-	+	-	-	
GRADS [13]	G	C,D	AB	+	-	-	-	-	-	
NMI Test [8]	G	C	AB	+	-	-	-	+	+	
<i>Internet testing</i>										
NIMI [14]	I	D	CWL	~	~	+	-	+	-	
WAWM [19]	I	D	CWL	~	~	+	+	+	+	

The +, ~, and - signs denote a feature that is present, for which an insufficient approach has been taken, or which lacks, respectively. Area acronyms: CI - cluster, G - grid, I - Internet, P2P - peer-to-peer.  
Sub-Area acronyms: C - computing, D - data transfer, S - data storage. "Realism" acronyms: P - probe,  $\mu$ K - micro-kernel, AB - application benchmark, SWL - simple workload, CWL - complex workload.

in load-testing a system, the workload submitter must ensure a submission rate higher than that of any existing user. Rates of  $2 \times 10^4$  submissions/hour and  $10^6$  submissions/day must be supported for testing grids and other LSDCSs [3]. The workload generation process must support similar rates if the testing framework generates the workload on-line.

- (b) **Test Results Management** Large amounts of data are produced during testing in an LSDCS. As a first step towards Online Analytical Processing (OLAP) capabilities, there must exist a hierarchy for results storage and aggregation, i.e., by project, by scenario, by test run. There may also be a need for coordination of test tasks and subtasks, where the results of one test (or their aggregates) are used as input to subsequent runs.
- 4) **Extensibility** There are many types of LSDCSs, e.g., grid computing, volunteer computing, peer-to-peer computing. As a result:
  - (a) **New Environments and Metrics** The design of a testing tool should be extensible, so that new environments and metrics can be supported, i.e., through plug-ins or configurable policies.
  - (b) **Test Templates** With so many alternatives, there must exist a set of templates (best practices) for commonly used scenarios. The scenarios are configured by the user at test time.

### B. Related Work

We survey here studies related to testing in real LSD(C)Ss (i.e., from grids to the Internet). Table I presents a summary of these studies; more details about four directions in LSD(C)S testing are given below.

*Grid/Cluster benchmarking* Following results from the parallel systems community, e.g., the NASA Parallel Benchmarks, several grid performance evaluation and benchmarking approaches focus on tests using micro-kernels, and application benchmarks [10], [16], [11]. These approaches have a limited

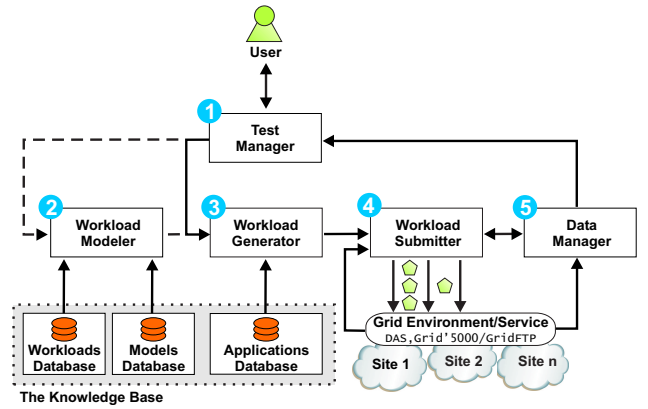


Fig. 1. An overview of the GRECHMARK framework.

applicability in LSDCSs, due to the heterogeneity and to the dynamic state of these systems.

*Grid performance evaluation* Few performance evaluation tools have been proposed in the context of grids [17], [12]. Even for these few, the research effort has been directed to providing solutions for distributed deployment and testing. Thus, relatively little focus has been given to the other goals presented at the beginning of this section.

*Grid functionality testing* Several testing suites are available for functionality testing, based on the submission of unrealistically simple workloads to the tested grid [18], [13].

*Internet testing* The research road for testing the Internet and the World Wide Web started with simple observations (and characterization) from a single point in the network, and is still evolving towards completely decentralized testing using realistic workloads [20]. The Internet testing tools do not address the following four design goals: ability to generate realistic workloads for LSDCSs, ability to collect and store annotation data, high rates for the submission of real applications, and support for LSDCS environments and metrics.

## III. THE GRECHMARK FRAMEWORK: APPROACH AND UNIQUENESS

In this section we present an overview of the GrenchMark framework, and describe two of its unique features: workload analysis and modeling, and results analysis and storage.

### A. Overview

Figure 1 shows an overview of the GrenchMark framework design. The five main components are numbered; early versions of the components 3, 4, and 5 have been presented in our previous work [5]. The framework description presented below follows the design goals introduced in Section II-A. To support the "realism" goal, the Workload Modeler (component number 2) makes use of databases of workloads and of workload models, and additionally provides mechanisms for truncation, scaling, and filtering of representative data. The database of real applications provides the Workload Generator (component number 3) with application that can be executed in real systems. To support the "reproducibility" goal, the Test Manager (component number 1) and the Workload Submitter (component number 4) can replay tests and sub-tests, and

test provenance and annotation data are stored with the test results by the Data Manager (component number 5). The Workload Submitter receives feedback from the tested environment, allowing tests that are LSDCS-specific, i.e., in which the submission depends on dynamic or delayed information (e.g., testing with grid workflows, or testing with service-level agreements). As annotation data, the Data Manager typically stores resource availability and consumption information, and the logs of different grid middleware.

This design specifies only the minimal set of capabilities present in a GRECHMARK implementation. These can be extended to accommodate more testing environments and metrics. Besides plug-ins, which require the installation of new sub-components, extensions can be made through *test templates*, that is, non-binary configuration files interpreted by the Test Manager to define how a test is conducted.

Our approach distinguishes itself from other approaches (see the previous section) in several key aspects. Compared to previous grid testing tools, GRECHMARK supports the complete process of testing LSDCSs, with richer workload modeling and generation, more detailed results analysis and storage, and support for test templates. Compared to Internet testing tools, GRECHMARK takes specific steps to ensure repeatable testing (rare in the Internet), and adapts to the testing specifics of LSDCSs.

### B. Unique Workload Analysis and Modeling Features

From the (real) traces stored in the Workloads Database, the workload analysis tools that are part of the Workload Modeler can extract automatically the parameters of a workload model, and also assess if the model selected by the user fits well the existing traces. This unique approach reduces one of the biggest risks in testing: that of using an unrealistic workload model.

The contents of LSDCS workloads have been characterized only recently. Unlike the workloads of parallel production environments (e.g., supercomputers, homogeneous clusters) [21], the workloads of LSDCSs are dominated by single-processor jobs, many of which are logically inter-dependent, that is, either part of bags-of-tasks or of workflows [3], [22]. The GRECHMARK framework is the first to support these two classes of workloads for LSDCSs.

### C. Unique Results Analysis and Storage Features

Most previous work has been dedicated to assessing the raw performance of a system. However, for LSDCSs the high rate of failures and other abnormal behavior means that performance should not be the only or even the primary concern of testing. The GRECHMARK framework looks beyond raw performance, and is able to assess the stability, the scalability, and the reliability of LSDCSs.

Similar to the Internet, LSDCSs have complex, multi-layer middleware stacks. Unlike the Internet, due to the timescales involved in executing jobs, LSDCSs can afford to log most of the state transitions occurring in their middleware stacks. As a result, it is possible to extract metrics at four levels: the job-

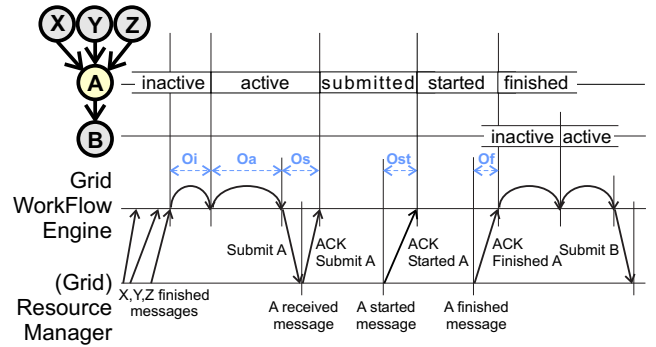


Fig. 2. The execution of job A. Boxes represent states. Continuous arrows represent information flows. Five overhead components, i.e., the  $O_*$ , are also depicted.

the operational-, the application-, and the service-level metrics. The GRECHMARK is the first framework that can assess for LSDCSs metrics at each of these levels. To this end, besides the analytical part, the Data Manager stores provenance and annotation data with the test results.

When the performance of the system is under investigation, few testing tools are able to find the removable bottlenecks by explaining their overhead. Without this ability, system tuning remains accessible to few, and is bound to a case-by-case basis. The GRECHMARK supports assessing the overheads that occur in the generic LSDCS job lifetime depicted in Figure 2. A workflow comprising five jobs (i.e., X, Y, Z, A, and B) is executed by the top layer in a middleware stack, the grid workflow engine (GWFE). Initially, job A is "inactive": it cannot be submitted for execution until its predecessor jobs (i.e., X, Y, and Z) finish executing. After they do, and after some time needed for post-processing and internal state updates, the GWFE changes A's state to "active", and starts A's submission process. The job remains "active" until the lower layer in the middleware stack, the grid resource manager, acknowledges the submission; afterwards, A becomes "submitted". The job waits in the grid resource manager's queue, then starts executing, and after some delay due to messaging A's state becomes "started". Finally, the job finishes executing and after some more messaging delay A's state becomes "finished". From that point, the GWFE can begin executing job B. The five independent overhead components (i.e., the  $O_*$ ) are supported by GRECHMARK for the whole middleware stack and for its individual layers.

## IV. RESULTS AND CONTRIBUTIONS

In this section we present a selection of important challenges related to and results obtained with the GRECHMARK framework.

### A. Challenges in Building the GrenchMark

For the past few years, grids were the most used type of LSDCSs. At the beginning of our work on the GRECHMARK framework, while large grids were already supporting the work of thousands of scientists, very little was known about their actual use. Because of strict organizational permissions, there were few or no traces of grid workloads available to

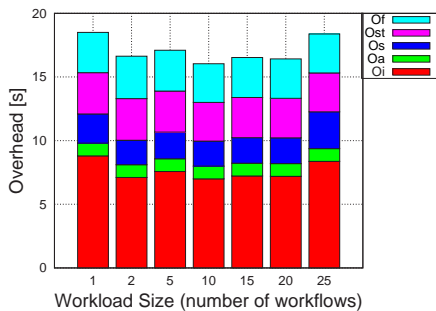


Fig. 3. The overhead breakdown for seven workload sizes.

the grid researcher and practitioner. To address this problem, we have created the Grid Workloads Archive (GWA) [23], an instance of the Workloads Database subcomponent of the GRENCHEM, and currently the largest source of grid traces (more than 7 million jobs over a period of over 13 operational years). For the GWA we have developed a common data format for LSDCS traces.

By analyzing the GWA data, we were able to establish that the workloads of LSDCSs are dominated by single-processor jobs often logically grouped in bags-of-tasks or workflows. We have built and validated a model for bags-of-tasks in LSDCS [24], which is the first to model bags-of-tasks explicitly rather than modeling only independent jobs [25]. As a result, our model is able to fit the existing traces with high accuracy while using only simple probabilistic distributions; thus, it is also more intuitive and tractable than previous alternatives. We have also investigated several workflow-based workloads that await validation against more real traces [22], [9].

### B. Beyond Raw Performance with the GrenchMark

Over the past 18 months, we have used the reference implementation in over 25 testing scenarios in grids (e.g., Globus-based), in peer-to-peer systems (e.g., BitTorrent-based), and in heterogeneous computing environments (e.g., Condor-based). Globus is arguably the most used grid middleware: it counts over 5,000 file transfer servers and about 100,000 service head-nodes (a service head-node can manage anything from one cluster to one processor) [26]. BitTorrent is currently the most used peer-to-peer file-sharing network [27]. Condor is one of the most used resource management systems: in 2006, Condor was used to manage at least 60,000 resources grouped in over 1750 pools [28]. We have also used the GRENCHEM capabilities for testing and two real LSD(C)Ss before their deployment: the Koala grid scheduler [29], and the Tribler peer-to-peer file-sharing network [27]. Below we present three results that are important for both researchers and system developers.

By generating a wide variety of workloads, we were able to show evidence that the real performance obtained by grid middleware stacks was lower than the performance predicted by simulation approaches [5]. In a recent study focused on GWFEs, we were also able to attribute the performance loss to overhead, and to show which overhead components are the most important [9]. Figure 3 shows the total overhead

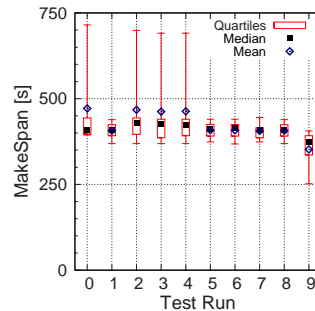


Fig. 4. The makespan distribution for 10 independent runs of the same workload. Each distribution is depicted as a *box-and-whiskers* plot with an additional point for the mean. Different runs exhibit similar mean and median, but very different value ranges.

and the overhead breakdown for one middleware stack. The overhead is significant: over 15 seconds per job. From the overhead components, the most important is  $O_i$ , followed by  $O_f$  and  $O_{st}$ ; these overhead components correspond to various state updates for the workflow jobs. The relative influence of each overhead component does not vary greatly with the workload size. Thus, better performance can be expected through improvements in the speed of updating the state of workflow jobs.

One of the often ignored characteristics of a tested system is its stability, here, its ability to behave identically when faced with the same workload. Using the ability of GRENCHEM to replay workloads under identical conditions, we assess the stability of a middleware stack by running independently ten identical test runs. Each run comprises the execution of the same workload with a size of 10 workflows/workload. Figure 4 depicts the workflow makespan distribution for each of the ten runs: five out of the ten runs (runs 1 and 5–8) hint towards the overall middleware stack’s stability. However, the other five runs exhibit wide ranges of makespans, which makes the system overall unstable. We have obtained similar stability results for various middleware stacks and for various levels of metrics, i.e., for the operational-level workflow makespan, for the job-level wait time. This result gives further evidence why the simulation of LSDCSs is difficult: the system performance cannot be realistically considered as fixed.

Another aspect often ignored in simulations is the scalability of the system. We have shown in our previous work that the scalability of a grid is often limited by the scalability of the grid middleware stack operating on the head-node [5]; the simulation assumption is that a head-node would be able to manage the concurrent execution of thousands of jobs. We have assessed the head-node scalability for four grid middleware stacks that include GWFEs [9]. Figure 5 shows the resource consumption for the four middleware stacks under a moderate workload (300 jobs grouped in 10 workflows). Stack 2 leads to the full utilization of the processor, to a number of 300-500 concurrently opened sockets (a common limit in today’s operating systems is 1024), and to 50-60% memory usage. This indicates that Stack 2 would scale with difficulty to larger workload sizes. In contrast to Stack 2,

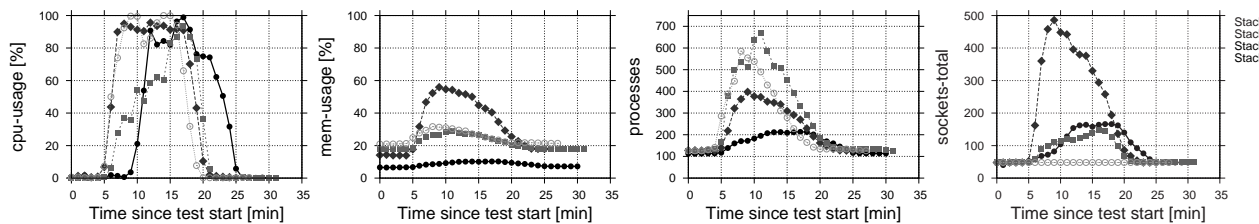


Fig. 5. The CPU consumption on the head-node increases with the workload size.

Stack 1 would scale much easier, in particular with regard to the memory usage (the consumption of which it heavily optimizes). The Stacks 3 and 4 offer trade-offs between the resource consumption of the Stacks 1 and 2.

## V. ONGOING AND FUTURE WORK

Today's LSDCSs, e.g., grids, are promising to provide a computational infrastructure for thousands of scientists. Until now, researchers have focused on creating interfaces to make this infrastructure easy-to-use, or on developing algorithms for more efficient use of the grid infrastructures. However, the deployed LSDCSs are exhibiting low performance, high failure rate, and complex functionality problems. Furthermore, there is a lack of in-depth and comparative studies of the existing solutions. To address these two problems, in this work we have presented the GRENCHMARK framework for testing LSDCSs that focuses on realistic and repeatable testing. We have shown how our framework fulfills the goals of testing LSDCSs, and presented GRENCHMARK's workload analysis and modeling, and results analysis and storage features. Using the GRENCHMARK reference implementation, we were able to assess the characteristics of real LSDCSs beyond the raw performance, and for various metric levels.

We believe the GRENCHMARK can become the basis for common performance evaluation framework for LSDCSs. We are currently following two research directions: enabling testing capabilities on large-scale experimental platforms, and building an LSDCS performance database.

## REFERENCES

- [1] C. Dumitrescu, I. Raicu, and I. T. Foster, "Experiences in running workloads over grid3." in *GCC*. IEEE CS, 2005, pp. 274–286.
- [2] A. Iosup, D. H. J. Epema, C. Franke, A. Papaspyrou, L. Schley, B. Song, and R. Yahyapour, "On grid performance evaluation using synthetic workloads," in *JSSPP*, ser. LNCS, E. Frachtenberg and U. Schwiegelshohn, Eds., vol. 4376. Springer, 2006, pp. 232–255.
- [3] A. Iosup, C. Dumitrescu, D. H. J. Epema, H. Li, and L. Wolters, "How are real grids used? the analysis of four grid traces and its implications," in *GRID*. IEEE CS, 2006, pp. 262–269.
- [4] H. Li, D. Groep, L. Wolters, and J. Templon, "Job failure analysis and its implications in a large-scale production grid." in *e-Science*. IEEE CS, Dec. 2006, pp. 84–84.
- [5] A. Iosup and D. H. J. Epema, "GrenchMark: A framework for analyzing, testing, and comparing grids," in *CCGRID*. IEEE CS, 2006, pp. 313–320.
- [6] O. Khalili, J. He, C. Olschanowsky, A. Snavely, and H. Casanova, "Measuring the performance and reliability of production computational grids," in *GRID*. IEEE CS, 2006, pp. 293–300.
- [7] B. Schroeder and G. A. Gibson, "A large-scale study of failures in high-performance computing systems," in *DSN*. IEEE CS, jun 2006, pp. 249–258.
- [8] A. Iosup, D. H. J. Epema, P. Couvares, A. Karp, and M. Livny, "Build-and-test workloads for grid middleware: Problem, analysis, and applications," in *CCGRID*. IEEE CS, 2007, pp. 205–213.
- [9] C. Stratan, A. Iosup, and D. H. J. Epema, "A performance study of grid workflow engines," TU Delft, Tech.Rep. PDS-2008-004, Apr 2008.
- [10] P. Luszczyk, D. H. Bailey, J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi, "The HPC Challenge (HPCC) benchmark suite." in *SC*. ACM Press, 2006, p. 213.
- [11] G. Tsouloupas and M. D. Dikaikakos, "GridBench: A workbench for grid benchmarking." in *EGC*, ser. LNCS, vol. 3470. Springer, 2005, pp. 211–225.
- [12] W. Hoarau, S. Tixeuil, and L. Silva, "Fault-injection and dependability benchmarking for grid computing middleware," in *CGIW*. Springer, 2007, pp. 119–134.
- [13] G. Chun, H. Dail, H. Casanova, and A. Snavely, "Benchmark probes for grid assessment." in *IPDPS*. IEEE CS, 2004.
- [14] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, "An architecture for large-scale internet measurement," *IEEE Communications*, vol. 36, no. 8, pp. 48–54, August 1998.
- [15] P. Buneman, S. Khanna, and W. C. Tan, "Why and where: A characterization of data provenance." in *ICDT*, ser. LNCS, J. V. den Bussche and V. Vianu, Eds., vol. 1973. Springer, 2001, pp. 316–330.
- [16] R. F. Van Der Wijngaart and M. Frumkin, "NAS Grid Benchmarks version 1.0," NASA, Tech.Rep. NAS-002-005, 2002.
- [17] I. Raicu, C. Dumitrescu, M. Ripeanu, and I. T. Foster, "The design, performance, and use of DiPerF: An automated distributed performance evaluation framework." *J. Grid Comput.*, vol. 4, no. 3, pp. 287–309, 2006.
- [18] S. Smullen, C. Olschanowsky, K. Ericson, P. Beckman, and J. M. Schopf, "The inca test harness and reporting framework." in *SC*. IEEE CS, 2004, p. 55.
- [19] P. Barford and M. Crovella, "Measuring web performance in the wide area." *Performance Evaluation Review*, vol. 27, no. 2, pp. 37–48, 1999.
- [20] M. Andreolini, V. Cardellini, and M. Colajanni, "Benchmarking models and tools for distributed web-server systems," in *Performance*, ser. LNCS, M. Calzarossa and S. Tucci, Eds., vol. 2459. Springer, 2002, pp. 208–235.
- [21] U. Lublin and D. G. Feitelson, "The workload on parallel supercomputers: modeling the characteristics of rigid jobs." *JPDC*, vol. 63, no. 11, pp. 1105–1122, 2003.
- [22] S. Ostermann, A. Iosup, R. Prodan, T. Fahringer, and D. Epema, "On the characteristics of grid workflows," in *CGIW*, Apr 2008, pp. 431–442, crete, GR. ISBN: 978-960-524-260-2.
- [23] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. Epema, "The grid workloads archive," Feb. 2008, (in print).
- [24] A. Iosup, O. Sonmez, S. Anoep, and D. H. Epema, "The performance of bags-of-tasks in large-scale distributed systems," in *HPDC*, 2008, (accepted).
- [25] H. Li, M. Muskulus, and L. Wolters, "Modeling job arrivals in a data-intensive grid," in *JSSPP*, ser. LNCS, E. Frachtenberg and U. Schwiegelshohn, Eds., vol. 4376. Springer, 2006, pp. 210–231.
- [26] The Metrics Project, "Globus metrics," Globus, Tech.Rep. v1.4, May 2007. [Online]. Available: [incubator.globus.org/metrics/reports/2007-02](http://incubator.globus.org/metrics/reports/2007-02)
- [27] J. Pouwelse et al., "Tribler: a social-based peer-to-peer system," *CC:PE*, vol. 20, no. 2, pp. 127–138, 2008.
- [28] D. Thain, T. Tannenbaum, and M. Livny, "How to measure a large open-source distributed system," *CC:PE*, vol. 18, no. 15, pp. 1989–2019, 2006.
- [29] H. H. Mohamed and D. H. J. Epema, "Experiences with the Koala co-allocating scheduler in multiclustes." in *CCGrid*. IEEE CS, 2005, pp. 784–791.