Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing

Alexandru Iosup, *Member, IEEE,* Simon Ostermann,Nezih Yigitbasi, *Member, IEEE,*
Radu Prodan, *Member, IEEE,* Thomas Fahringer, *Member, IEEE,* and Dick Epema, *Member, IEEE*

Abstract—Cloud computing is an emerging commercial infrastructure paradigm that promises to eliminate the need for maintaining expensive computing facilities by companies and institutes alike. Through the use of virtualization and resource time-sharing, clouds serve with a single set of physical resources a large user base with different needs. Thus, clouds have the potential to provide to their owners the benefits of an economy of scale and, at the same time, become an alternative for scientists to clusters, grids, and parallel production environments. However, the current commercial clouds have been built to support web and small database workloads, which are very different from typical scientific computing workloads. Moreover, the use of virtualization and resource time-sharing may introduce significant performance penalties for the demanding scientific computing workloads. In this work we analyze the performance of cloud computing services for scientific computing workloads. We quantify the presence in real scientific computing workloads of Many-Task Computing (MTC) users, that is, of users who employ loosely coupled applications comprising many tasks to achieve their scientific goals. Then, we perform an empirical evaluation of the performance of four commercial cloud computing services including Amazon EC2, which is currently the largest commercial cloud. Last, we compare through trace-based simulation the performance characteristics and cost models of clouds and other scientific computing platforms, for general and MTC-based scientific computing workloads. Our results indicate that the current clouds need an order of magnitude in performance improvement to be useful to the scientific community, and show which improvements should be considered first to address this discrepancy between offer and demand.

Index Terms—Distributed Systems, Distributed applications, Performance evaluation, Metrics/Measurement, Performance measures.

1 Introduction

Scientific computing requires an ever-increasing number of resources to deliver results for ever-growing problem sizes in a reasonable time frame. In the last decade, while the largest research projects were able to afford (access to) expensive supercomputers, many projects were forced to opt for cheaper resources such as commodity clusters and grids. Cloud computing proposes an alternative in which resources are no longer hosted by the researchers' computational facilities, but are leased from big data centers only when needed. Despite the existence of several cloud computing offerings by vendors such as Amazon [1] and GoGrid [2], the potential of clouds for scientific computing remains largely unexplored. To address this issue, in this paper we present a performance analysis of cloud computing services for many-task scientific computing.

The cloud computing paradigm holds great promise for the performance-hungry scientific computing community: Clouds can be a cheap alternative to supercomputers and specialized clusters, a much more reliable platform than grids, and a much more scalable platform than the largest of commodity clusters. Clouds also promise to "scale by credit card," that is, to scale up instantly and temporarily within the limitations imposed only by the available financial resources, as opposed to the physical limitations of adding nodes to clusters or even supercomputers and to the administrative burden of over-provisioning resources. Moreover, clouds promise good support for bags-of-tasks, which currently constitute the dominant grid application type [3]. However, clouds also raise important challenges in many aspects of scientific computing, including performance, which is the focus of this work.

There are three main differences between scientific computing workloads and the initial target workload of clouds: in required system size, in performance demand, and in the job execution model. Size-wise, top scientific computing facilities comprise very large systems, with the top ten entries in the Top500 Supercomputers List together totaling about one million cores, while cloud computing services were designed to replace the small-to-medium size enterprise data centers. Performance-wise, scientific workloads often require High Performance Computing (HPC) or High-Throughput Computing (HTC) capabilities. Recently, the scientific computing community has started to focus on Many-Task Computing (MTC) [4], that is, on high-performance execution of loosely coupled applications comprising many (possibly inter-related) tasks. With MTC, a paradigm at the intersection of HPC and HTC, it is possible to demand

- *A. Iosup, N. Yigitbasi, and D. Epema are with the Parallel and Distributed Systems, Delft University of Technology, Delft, the Netherlands.*
- *S. Ostermann, R. Prodan, and T. Fahringer are with the Institute for Computer Science, University of Innsbruck, Innsbruck, Austria.*
- *Corresponding author: Alexandru Iosup* A.Iosup@tudelft.nl.

systems to operate at high utilizations, similar to those of current production grids (over 80% [5]) and Parallel Production Infrastructures (PPIs) (over 60% [6]), and much higher than those of the systems that clouds originally intended to replace (servers with 10-20% utilization). The job execution model of scientific computing platforms is based on the exclusive, space-shared usage of resources. In contrast, most clouds time-share resources and use virtualization to abstract away from the actual hardware, thus increasing the concurrency of users but potentially lowering the attainable performance.

These three main differences between scientific computing workloads and the target workloads of clouds raise an important research question: *Is the performance of clouds sufficient for MTC-based scientific computing?*, or, in other words, *Can current clouds execute MTC-based scientific workloads with similar performance (that is, for traditional performance metrics [7]) and at lower cost?* Though early attempts to characterize clouds and other virtualized services exist [8], [9], [10], [11], [12], this question remains largely unexplored. Our main contribution toward answering it is threefold:

1) We investigate the presence of a (proto-)MTC component in scientific computing workloads and quantify the presence of these users in scientific computing environments.
2) We evaluate with well-known micro-benchmarks and application kernels the performance of four commercial cloud computing services that can be used for scientific computing, among which the Amazon Elastic Compute Cloud (EC2), the largest commercial computing cloud in production.
3) We compare the performance of clouds with that of scientific computing alternatives such as grids and parallel production infrastructures. Our comparison uses trace-based simulation and the empirical performance results of our cloud performance evaluation.

The remainder of the article is organized as follows. In Section 2 we give a general introduction to the use of cloud computing services for scientific computing, and select four exemplary clouds for use in our investigation. Then, in Section 3 we focus on finding the MTC component in existing scientific computing workloads, and in Section 4 we evaluate empirically the performance of four commercial clouds. In Section 5 we compare the performance of clouds and of other scientific computing environments. Last, we compare our investigation with related work in Section 6, and we present our conclusion and potential future research topics in Section 7.

2 Cloud Computing Services for Scientific Computing

In this section we provide a background to analyzing the performance of cloud computing services for scientific computing. We first describe the main characteristics of the common scientific computing workloads, based on previous work on analyzing and modeling of workload traces taken from PPIs [6] and grids [5], [13]. Then, we introduce the cloud computing services that can be used for scientific computing, and select four commercial clouds whose performance we will evaluate empirically.

2.1 Scientific Computing

Job structure and source PPI workloads are dominated by parallel jobs [6], while grid workloads are dominated by small bags-of-tasks (BoTs) [3] and sometimes by small workflows [14], [15] comprising mostly sequential tasks. Source-wise, it is common for PPI grid workloads to be dominated by a small number of users. We consider users that submit many tasks, often grouped into the same submission as BoTs, as proto-MTC users, in that they will be most likely to migrate to systems that provide good performance for MTC workload execution. We focus in Section 3 on a more rigorous definition of MTC workloads, and on demonstrating their presence in recent scientific workloads.

Bottleneck resources Overall, scientific computing workloads are highly heterogeneous, and can have either one of CPU, I/O, memory, and network as the bottleneck resource. Thus, in Section 4 we investigate the performance of these individual resources.

Job parallelism A large majority of the parallel jobs found in published PPI [16] and grid [13] traces have up to 128 processors [5], [6]. Moreover, the average scientific cluster size was found to be around 32 nodes [17] and to be stable over the past five years [18]. Thus, in Section 4 we look at the the performance of executing parallel applications of up to 128 processors.

2.2 Four Selected Clouds: Amazon EC2, GoGrid, ElasticHosts, and Mosso

We identify three categories of cloud computing services [19], [20]: Infrastructure-as-a-Service (IaaS), that is, raw infrastructure and associated middleware, Platform-as-a-Service (PaaS), that is, APIs for developing applications on an abstract platform, and Software-as-a-Service (SaaS), that is, support for running software services remotely. Many clouds already exist, but not all provide virtualization, or even computing services. The scientific community has not yet started to adopt PaaS or SaaS solutions, mainly to avoid porting legacy applications and for lack of the needed scientific computing services, respectively. Thus, in this study we are focusing only on IaaS providers. We also focus only on public clouds, that is, clouds that are not restricted within an enterprise; such clouds can be used by our target audience, scientists.

Based on our recent survey of the cloud computing providers [21], we have selected for this work four IaaS clouds. The reason for this selection is threefold. First, not all the clouds on the market are still accepting clients; FlexiScale puts new customers on a waiting list for over two weeks due to system overload. Second, not all the

TABLE 1
The resource characteristics for the instance types offered by the four selected clouds.

Name	Cores (ECUs)	RAM [GB]	Archi. [bit]	Disk [GB]	Cost [$/h]
Amazon EC2					
m1.small	1 (1)	1.7	32	160	0.1
m1.large	2 (4)	7.5	64	850	0.4
m1.xlarge	4 (8)	15.0	64	1,690	0.8
c1.medium	2 (5)	1.7	32	350	0.2
c1.xlarge	8 (20)	7.0	64	1,690	0.8
GoGrid (GG)					
GG.small	1	1.0	32	60	0.19
GG.large	1	1.0	64	60	0.19
GG.xlarge	3	4.0	64	240	0.76
Elastic Hosts (EH)					
EH.small	1	1.0	32	30	£0.042
EH.large	1	4.0	64	30	£0.09
Mosso					
Mosso.small	4	1.0	64	40	0.06
Mosso.large	4	4.0	64	160	0.24

clouds on the market are large enough to accommodate requests for even 16 or 32 co-allocated resources. Third, our selection already covers a wide range of quantitative and qualitative cloud characteristics, as summarized in Tables 1 and our cloud survey [21], respectively. We describe in the following Amazon EC2; the other three, GoGrid (GG), ElasticHosts (EH), and Mosso, are IaaS clouds with provisioning, billing, and availability and performance guarantees similar to Amazon EC2's.

The **Amazon Elastic Computing Cloud (EC2)** is an IaaS cloud computing service that opens Amazon's large computing infrastructure to its users. The service is elastic in the sense that it enables the user to extend or shrink its infrastructure by launching or terminating new virtual machines (*instances*). The user can use any of the *instance types* currently available on offer, the characteristics and cost of the five instance types available in June 2009 are summarized in Table 1. An ECU is the equivalent CPU power of a 1.0-1.2 GHz 2007 Opteron or Xeon processor. The theoretical peak performance can be computed for different instances from the ECU definition: a 1.1 GHz 2007 Opteron can perform 4 flops per cycle at full pipeline, which means at peak performance one ECU equals 4.4 gigaflops per second (GFLOPS).

To create an infrastructure from EC2 resources, the user specifies the instance type and the VM image; the user can specify any VM image previously registered with Amazon, including Amazon's or the user's own. Once the VM image has been transparently deployed on a physical machine (the resource status is *running*), the instance is booted; at the end of the boot process the resource status becomes *installed*. The installed resource can be used as a regular computing node immediately after the booting process has finished, via an `ssh` connection. A maximum of 20 instances can be used concurrently by regular users by default; an application can be made to increase this limit, but the process involves an Amazon representative. Amazon EC2 abides by a Service Level Agreement (SLA) in which the user is compensated if the resources are not available for acquisition at least 99.95% of the time. The security of the Amazon services has been investigated elsewhere [10].

3 MTC Presence in Scientific Computing Workloads

An important assumption of this work is that the existing scientific workloads already include Many Task Computing users, that is, of users that employ loosely coupled applications comprising many tasks to achieve their scientific goals. In this section we verify this assumption through a detailed investigation of workload traces taken from real scientific computing environments.

3.1 Method and Experimental Setup

MTC workloads may comprise tens of thousands to hundreds of thousands of tasks and BoTs [4], and a typical period may be one year or the whole trace. Our method for identifying proto-MTC users—users with a pronounced MTC-like workload, which are potential MTC users in the future—in existing system workloads is based on the identification of users with many submitted tasks and/or bags-of-tasks in the workload traces taken from real scientific computing infrastructures. We define an *MTC user* to be a user that has submitted at least J jobs and at least B bags-of-tasks. The user part of our definition serves as a coupling between jobs, under the assumption that a user submits jobs for execution towards an arbitrary but meaningful goal. The jobs part ensures that we focus on high-volume users; these users are likely to need new scheduling techniques for good system performance. The bag-of-tasks part ensures that task submission occurs within a short period of time; this submission pattern raises new challenges in the area of task scheduling and management [4]. Ideally, it should be possible to use a unique pair of values for J and B across different systems.

To investigate the presence of an MTC component in existing scientific computing infrastructures we analyze ten workload traces. Table 2 summarizes the characteristics of the ten traces; see [13], [16] for more details about each trace. The ID of the trace indicates the system from which it was taken. The traces have been collected from a wide variety of grids and parallel production environments. The traces precede the existence of MTC tools; thus, the presence of an MTC component in these traces indicates the existence of proto-MTC users, who will be likely to use today's MTC-friendly environments.

To identify MTC users, we first formulate the identification criterion by selecting values for J, B. If $B \geq 1$, we first identify the BoTs in the trace using the method that we introduced in our previous work [22], that is, we use the BoT identification information when it is

TABLE 2
The characteristics of the workload traces.

Trace ID, Source (Trace ID in Archive)	Trace Time [mo.]	Trace Number of Jobs	Trace Number of Users	System Size Sites	System Size CPUs	Load [%]
Grid Workloads Archive [13], 6 traces						
1. DAS-2 (1)	18	1.1M	333	5	0.4K	15+
2. RAL (6)	12	0.2M	208	1	0.8K	85+
3. GLOW (7)	3	0.2M	18	1	1.6K	60+
4. Grid3 (8)	18	1.3M	19	29	3.5K	-
5. SharcNet (10)	13	1.1M	412	10	6.8K	-
6. LCG (11)	1	0.2M	216	200+	24.4K	-
Parallel Workloads Archive [16], 4 traces						
7. CTC SP2 (6)	11	0.1M	679	1	0.4K	66
8. SDSC SP2 (9)	24	0.1M	437	1	0.1K	83
9. LANLO2K (10)	5	0.1M	337	1	2.0K	64
10. SDSC DS (19)	13	0.1M	460	1	1.7K	63

present in the trace, and identify BoTs as groups of tasks submitted by the same user at and during short time intervals, otherwise. (We have investigated the effect of the time frame in the identification of BoTs in our previous work [22].) Then, we eliminate the users that have not submitted at least B BoTs. Last, from the remaining users we select the users that have submitted at least J tasks.

3.2 Results

The number of MTC users decreases quickly with the increase of J and B. Figure 1 shows the results for our analysis where we use the number of submitted BoTs (left), and the number of submitted tasks (right) as criteria for identifying MTC users for the DAS-2 (top) and SDSC SP2 (bottom) traces. As expected, the number of MTC users identified in the workload traces decreases as the number of submitted BoTs/tasks increases. The number of MTC users identified in the trace decreases much faster in the SDSC trace than in the DAS-2 trace with the increase of the number of BoTs/tasks. In addition, since there are not many MTC users for large number of BoTs/tasks in PPI, we see evidence that there is more MTC activity in grids than in PPI.

Expectedly, **there is more MTC-like activity in grids than in PPIs.** To compare the MTC-like activity of grids and PPIs we analyze for each trace the percentage of MTC jobs from the total number of jobs, and the percentage of CPU time consumed by MTC jobs from the total CPU time consumption recorded in the trace. Table 3 presents the results for various simple and complex criteria for all traces. We use "number of BoTs submitted $\geq$ 100" and "number of jobs submitted $\geq$ 1,000" as the simple criteria, and "number of BoTs submitted $\geq$ 1,000 & number of tasks submitted $\geq$ 10,000" as the complex criterion. Even for the simple criteria, we observe that for PPIs, except for the LANL-O2K trace, there are no MTC jobs for large values of B (the number of BoTs). As the number of BoTs and tasks increases, the percentage of MTC jobs and their consumed CPU-time decrease for both PPI and grids, as expected. However, for the Grid3 and GLOW traces the MTC activity is highly present even for large values of J and B.It turns out that the complex criterion additionally selects mostly users who submit many single-node tasks (not shown). Since this type of proto-MTC workload has the potential to execute well in any environment, including clouds, we select and use this complex criterion for the remainder of this work.

4 Cloud Performance Evaluation

In this section we present an empirical performance evaluation of cloud computing services. Toward this end, we run micro-benchmarks and application kernels typical for scientific computing on cloud computing resources, and compare whenever possible the obtained results to the theoretical peak performance and/or the performance of other scientific computing systems.

4.1 Method

Our method stems from the traditional system benchmarking. Saavedra and Smith [23] have shown that benchmarking the performance of various system components with a wide variety of micro-benchmarks and application kernels can provide a first order estimate of that system's performance. Similarly, in this section we evaluate various components of the four clouds introduced in Section 2.2. However, our method is not a straightforward application of Saavedra and Smith's method. Instead, we add a cloud-specific component, select several benchmarks for a comprehensive platform-independent evaluation, and focus on metrics specific to large-scale systems (such as efficiency and variability).

Cloud-specific evaluation An attractive promise of clouds is that they can always provide resources on demand, without additional waiting time [20]. However, since the load of other large-scale systems varies over time due to submission patterns [5], [6] we want to investigate if large clouds can indeed bypass this

TABLE 3
The percentage of MTC jobs, and the CPU time consumed by these jobs from the total number of jobs and consumed CPU time for all traces, with various simple and complex criteria for identifying MTC users. CPUT stands for Total CPU Time.

Trace ID	Simple criteria: BoTs $\geq$ 100 Jobs [%]	Simple criteria: BoTs $\geq$ 100 CPUT [%]	Simple criteria: Tasks $\geq$ 1,000 Jobs [%]	Simple criteria: Tasks $\geq$ 1,000 CPUT [%]	Complex criterion: Jobs $\geq$ 10,000 & BoTs $\geq$ 1,000 Jobs [%]	Complex criterion: Jobs $\geq$ 10,000 & BoTs $\geq$ 1,000 CPUT [%]
DAS-2	90	65	95	73	57	26
RAL	81	78	98	94	33	28
GLOW	95	75	99	91	83	69
Grid3	100	97	100	97	97	95
SharcNet	40	35	95	85	15	9
LCG	61	39	87	61	0	0
CTC SP2	16	13	18	14	0	0
SDSC SP2	13	93	25	6	0	0
LANL O2	66	14	73	18	34	6
SDSC DS	29	7	44	18	0	0

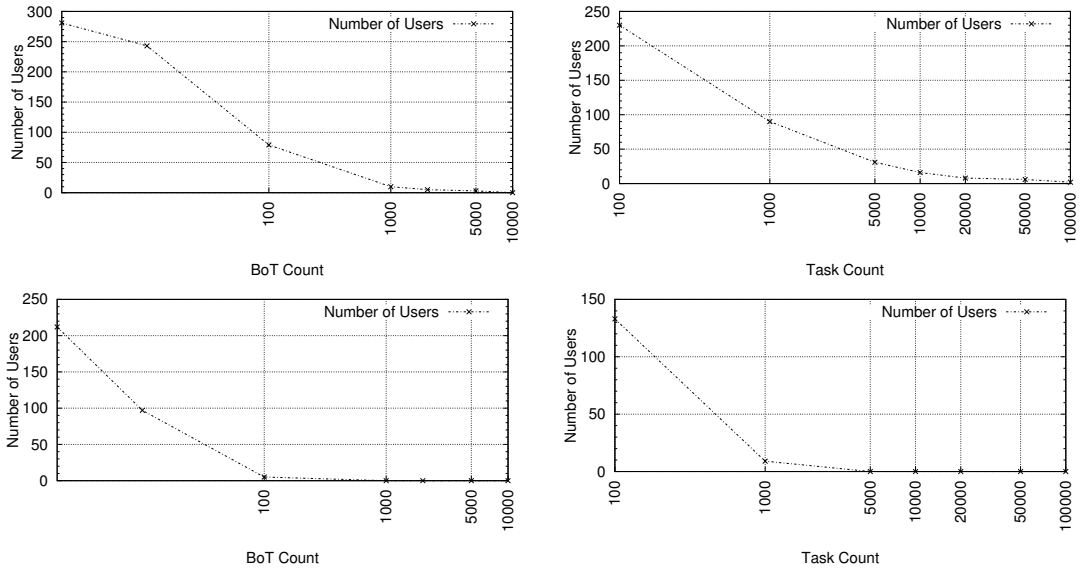

Fig. 1. Number of MTC users for the DAS-2 trace (*top*), and the San Diego Supercomputer Center (SDSC) SP2 trace (*bottom*) when considering only the submitted BoT count criterion (*left*), and only submitted task count criterion (*right*).

TABLE 4
The benchmarks used for cloud performance evaluation. B, FLOP, U, and PS stand for bytes, floating point operations, updates, and per second, respectively.

Type	Suite/Benchmark	Resource	Unit
SI	LMbench/all [24]	Many	Many
SI	Bonnie/all [25], [26]	Disk	MBps
SI	CacheBench/all [27]	Memory	MBps
MI	HPCC/HPL [28], [29]	CPU	GFLOPS
MI	HPCC/DGEMM [30]	CPU	GFLOPS
MI	HPCC/STREAM [30]	Memory	GBps
MI	HPCC/RandomAccess [31]	Network	MUPS
MI	HPCC/b_{eff}(lat.,bw.) [32]	Comm.	μs, GBps

TABLE 5
The VM images used in our experiments.

VM image	OS, MPI	Archi	Benchmarks
EC2/ami-2bb65342	FC6	32bit	SI
EC2/ami-36ff1a5f	FC6	64bit	SI
EC2/ami-3e836657	FC6, MPI	32bit	MI
EC2/ami-e813f681	FC6, MPI	64bit	MI
GG/server$_1$	RHEL 5.1, MPI	32&64bit	SI&MI
EH/server$_1$	Knoppix 5.3.1	32bit	SI
EH/server$_2$	Ubuntu 8.10	64bit	SI
Mosso/server$_1$	Ubuntu 8.10	32&64bit	SI

problem. To this end, one or more instances of the same instance type are repeatedly acquired and released during a few minutes; the resource acquisition requests follow a Poisson process with arrival rate $\lambda = 1s^{-1}$.

Infrastructure-agnostic evaluation There currently is no single accepted benchmark for scientific computing at large-scale. To address this issue, we use several traditional benchmark suites comprising micro-benchmarks and (scientific) application kernels. We further design two types of test workloads: SI–run one or more single-process jobs on a single instance (possibly with multiple cores), and MI–run a single multi-process job on multiple instances. The SI workloads execute in turn one of the *LMbench* [33], *Bonnie* [34], and *CacheBench* [35] benchmark suites. The MI workloads execute the *HPC Challenge Benchmark (HPCC)* [28] scientific computing benchmark suite. The characteristics of the used benchmarks and the mapping to the test workloads are summarized in Table 4; we refer to the benchmarks' references for more details.

Performance metrics We use the performance metrics defined by the benchmarks presented in Table 4. We also define and use the *HPL efficiency* of a virtual cluster based on the instance type T as the ratio between the HPL benchmark performance of the real cluster and the peak theoretical performance of a same-sized T-cluster, expressed as a percentage. Job execution at large-scale often leads to performance variability. To address this problem, in this work we report not only the average performance, but also the variability of the results.

4.2 Experimental Setup

We now describe the experimental setup in which we use the performance evaluation method presented earlier.

Performance Analysis Tool We have recently [36] extended the GrenchMark [37] large-scale distributed testing framework with new features which allow it to test cloud computing infrastructures. The framework was already able to generate and submit both real and synthetic workloads to grids, clusters, clouds, and other large-scale distributed environments. For this work, we have added to GrenchMark the ability to execute and analyze the benchmarks described in the previous section.

Environment We perform our measurements on homogeneous virtual environments built from virtual re-

sources belonging to one of the instance types described in Table 1; the used VM images are summarized in Table 5. The experimental environments comprise from 1 to 128 cores. Except for the use of internal IP addresses, described below, we have used in all our experiments the standard configurations provided by the cloud. Due to our choice of benchmarks, our Single-Job results can be readily compared with the benchmarking results made public for many other scientific computing systems, and in particular by the HPCC effort [38].

MPI library and network The VM images used for the HPCC benchmarks also have a working pre-configured MPI based on the `mpich2-1.0.5` [39] implementation. For the MI (parallel) experiments, the network selection can be critical for achieving good results. Amazon EC2 and GoGrid, the two clouds for which we have performed MI experiments, use internal IP addresses (IPs), that is, the IPs accessible only within the cloud, to optimize the data transfers between closely-located instances. (This also allows the clouds to better shape the traffic and to reduce the number of Internet-accessible IPs, and in turn to reduce the cloud's operational costs.) EC2 and GoGrid give strong incentives to their customers to use internal IP addresses, in that the network traffic between internal IPs is free, while the traffic to or from the Internet IPs is not. We have used only the internal IP addresses in our experiments with MI workloads.

Optimizations, tuning The benchmarks were compiled using GNU C/C++ 4.1 with the `-O3 -funroll-loops` command-line arguments. We did not use any additional architecture- or instance-dependent optimizations. For the HPL benchmark, the performance results depend on two main factors: the the Basic Linear Algebra Subprogram (BLAS) [40] library, and the problem size. We used in our experiments the GotoBLAS [41]library, which is one of the best portable solutions freely available to scientists. Searching for the problem size that can deliver peak performance is an extensive (and costly) process. Instead, we used a free analytical tool [42] to find for each system the problem sizes that can deliver results close to the peak performance; based on the tool advice we have used values from 13,000 to 110,000 for N, the size (order) of the coefficient matrix A [28], [43].

4.3 Results

The experimental results of the Amazon EC2 performance evaluation are presented in the following.

4.3.1 Resource Acquisition and Release

We study two resource acquisition and release scenarios: for single instances, and for multiple instances allocated at once.

Single instances We first repeat 20 times for each instance type a resource acquisition followed by a release as soon as the resource status becomes installed (see Section 2.2). Figure 2 shows the overheads associated with resource acquisition and release in EC2. The total resource acquisition time (*Total*) is the sum of the *Install* and *Boot* times. The *Release* time is the time taken to release the resource back to EC2; after it is released the resource stops being charged by Amazon. The `c1.*` instances are surprisingly easy to obtain; in contrast, the `m1.*` instances have for the resource acquisition time higher expectation (63-90s compared to around 63s) and variability (much larger boxes). With the exception of the occasional outlier, both the VM *Boot* and *Release* times are stable and represent about a quarter of *Total* each. Table 6 presents basic statistics for single resource allocation and release. Overall, **Amazon EC2 has one order of magnitude lower single resource allocation and release durations than GoGrid.** From the EC2 resources, the `m1.small` and `m1.large` instances have higher average allocation duration, and exhibit outliers comparable to those encountered for GoGrid. **The resource acquisition time of GoGrid resources is highly variable**; here, GoGrid behaves similarly to to grids [5] and unlike the promise of clouds.

TABLE 6
Statistics for single resource allocation/release.

Instance Type	Res. Allocation			Res. Release		
	Min	Avg	Max	Min	Avg	Max
m1.small	69	82	126	18	21	23
m1.large	50	90	883	17	20	686
m1.xlarge	57	64	91	17	18	25
c1.medium	60	65	72	17	20	22
c1.xlarge	49	65	90	17	18	20
GG.large	240	540	900	180	210	240
GG.xlarge	180	1,209	3,600	120	192	300

Multiple instances We investigate next the performance of requesting the acquisition of multiple resources (2,4,8,16, and 20) *at the same time*; a scenario common for creating homogeneous virtual clusters. When resources are requested in bulk, we record acquisition and release times for each resource in the request, separately. Figure 3 shows the basic statistical properties of the times recorded for `c1.xlarge` instances. The expectation and the variance are both higher for multiple instances than for a single instance.

4.3.2 Single-Machine Benchmarks

In this set of experiments we measure the raw performance of the CPU, I/O, and memory hierarchy using the Single-Instance benchmarks listed in Section 4.1. We run each benchmark 10 times and report the average results.

Compute performance We assess the computational performance of each instance type using the entire LMbench suite. The performance of int and int64 operations, and of the float and double-precision float operations is depicted in Figure 4 left and right, respectively. *The GOPS recorded for the floating point and double-precision float operations is six to eight times lower than the theoretical*

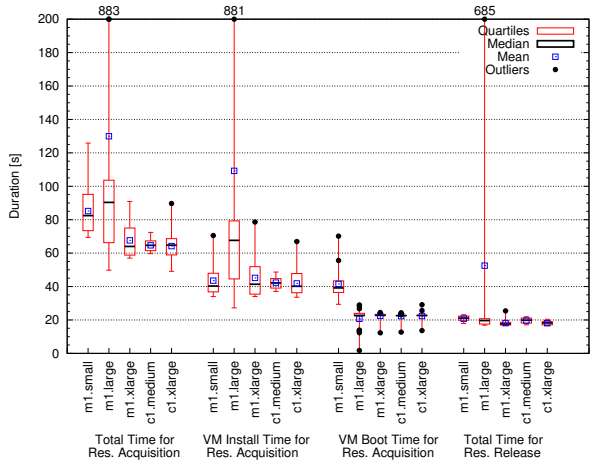

Fig. 2. Resource acquisition and release overheads for acquiring single EC2 instances.

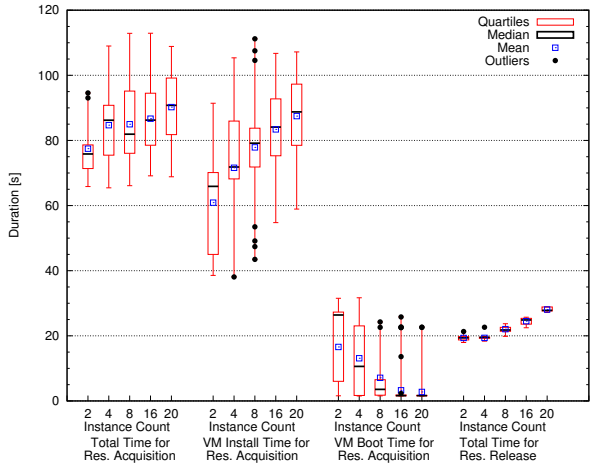

Fig. 3. Single-instance resource acquisition and release overheads when acquiring multiple `c1.xlarge` instances at the same time.

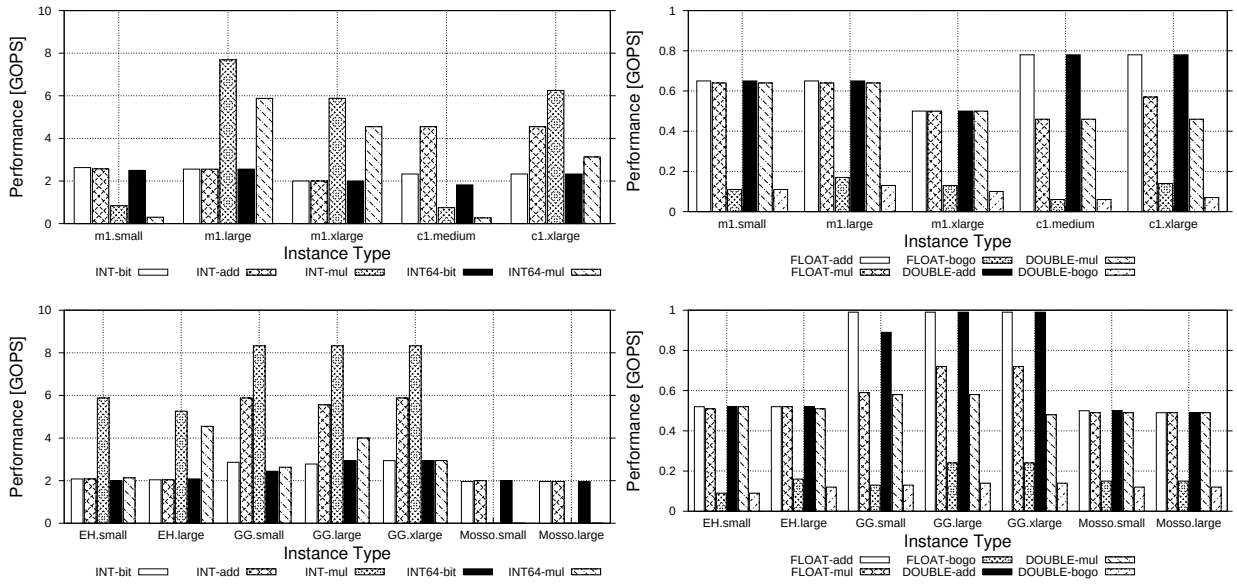

Fig. 4. LMbench results (*top*) for the EC2 instances, and (*bottom*) for the other instances. Each row depicts the performance of 32- and 64-bit integer operations in giga-operations per second (GOPS) (*left*), and of floating operations with single and double precision (*right*).

maximum of ECU (4.4 GOPS). A potential reason for this situation is the over-run or thrashing of the memory caches by the working sets of other applications sharing the same physical machines; a study independent from ours [44] identifies the working set size as a key parameter to consider when placing and migrating applications on virtualized servers. This situation occurs especially when the physical machines are shared among users that are unaware of each other; a previous study [45] has found that even instances of the same user may be located on the same physical machine. The performance evaluation results also indicate that the double-precision float performance of the `c1.*` instances, arguably the most important for scientific computing, is mixed: excellent addition but poor multiplication capabilities. Thus, as many scientific computing applications use heavily both of these operations, the user is faced with the difficult problem of selecting between two wrong choices. Finally, several double and float operations take longer on `c1.medium` than on `m1.small`. For the other instances, `EH.*` and `Mosso.*` instances have similar performance for both integer and floating point operations. `GG.*` instances have the best float and double-precision performance, and good performance for integer operations, which suggests the existence of better hardware support for these operations on these instances.

I/O performance We assess in two steps the I/O performance of each instance type with the Bonnie benchmarking suite. The first step is to determine the smallest file size that invalidates the memory-based I/O

TABLE 7
The I/O of clouds vs. 2002 [25] and 2007 [26] systems.

Instance Type	Seq. Output Char [MB/s]	Seq. Output Block [MB/s]	Seq. Output ReWr [MB/s]	Seq. Input Char [MB/s]	Seq. Input Block [MB/s]	Rand. Input [Seek/s]
m1.small	22.3	60.2	33.3	25.9	73.5	74.4
m1.large	50.9	64.3	24.4	35.9	63.2	124.3
m1.xlarge	57.0	87.8	33.3	41.2	74.5	387.9
c1.medium	49.1	58.7	32.8	47.4	74.9	72.4
c1.xlarge	64.8	87.8	30.0	45.0	74.5	373.9
GG.small	11.4	10.7	9.2	29.2	40.24	39.8
GG.large	17.0	17.5	16.0	34.1	97.5	29.0
GG.xlarge	80.7	136.9	92.6	79.26	369.15	157.5
EH.large	7.1	7.1	7.1	27.9	35.7	177.9
Mosso.sm	41.0	102.7	43.88	32.1	130.6	122.6
Mosso.lg	40.3	115.1	55.3	41.3	165.5	176.7
'02 Ext3	12.2	38.7	25.7	12.7	173.7	-
'02 RAID5	14.4	14.3	12.2	13.5	73.0	-
'07 RAID5	30.9	40.6	29.0	41.9	112.7	192.9

cache, by running the Bonnie suite for thirteen file sizes in the range 1024 Kilo-binary byte (KiB) to 40 GiB. The results of this preliminary step have been summarized in a technical report [46, pp.11-12]; we only summarize them here. For all instance types, a performance drop begins with the 100MiB test file and ends at 2GiB, indicating a capacity of the memory-based disk cache of 4-5GiB (twice 2GiB). Thus, the results obtained for the file sizes above 5GiB correspond to the real I/O performance of the system; lower file sizes would be served by the system with a combination of memory and disk operations. We analyze the I/O performance obtained for files sizes above 5GiB in the second step; Table 7 summarizes the results. We find that the I/O performance indicated by Amazon EC2 (see Table 1) corresponds to the achieved performance for random I/O operations (column 'Rand. Input' in Table 7). The `*.xlarge` instance types have the best I/O performance from all instance types. *For the sequential operations more typical to scientific computing all Amazon EC2 instance types have in general better performance when compared with similar modern commodity systems*, such as the systems described in the last three rows in Table 7; EC2 may be using better hardware, which is affordable due to economies of scale [20].

4.3.3 Multi-Machine Benchmarks

In this set of experiments we measure the performance delivered by homogeneous clusters formed with Amazon EC2 and GoGrid instances when running the Single-Job-Multi-Machine benchmarks. For these tests we execute 5 times the HPCC benchmark on homogeneous clusters of 1–16 (1–8) instances on EC2 (GoGrid), and present the average results.

HPL performance The performance achieved for the HPL benchmark on various virtual clusters based on the `m1.small` and `c1.xlarge` instance types is depicted in Figure 5. For the `m1.small` resources one node was able to achieve a performance of 1.96 GFLOPS, which is 44.54% from the peak performance advertised by Amazon. Then, the performance increased to up to 27.8 GFLOPS for 16 nodes, while the efficiency decreased slowly to 39.4%. The results for a single `c1.xlarge` instance are better: the achieved 49.97 GFLOPS represent 56.78% from the advertised peak performance. However, while the performance scales when running up to 16 instances to 425.82 GFLOPS, the efficiency decreases to only 30.24%. The HPL performance loss from one to 16 instances can therefore be expressed as 53.26% which results in rather bad qualification for HPC applications and their need for fast inter-node communication. We have obtained similar results the `GG.large` and `GG.xlarge` instances, as shown in Figure 5. For `GG.large` instances, the efficiency decreases quicker than for EC2 instances, down to 47.33% while achieving 45.44 GFLOPS on eight instances. The `GG.xlarge` performed even poorer in our tests. We further investigate the performance of the HPL benchmark for different instance types; Table 8 summarizes the results. *The efficiency results* presented in Figure 5 and Table 8 *place clouds below existing environments for scientific computing, for which the achieved performance is 60-70% of the theoretical peak even for demanding real applications [47], [48], [49].*

TABLE 8
HPL performance and cost comparison for various EC2 and GoGrid instance types.

Name	Peak Perf.	GFLOPS	GFLOPS /Unit	GFLOPS /$1
m1.small	4.4	2.0	2.0	19.6
m1.large	17.6	7.1	1.8	17.9
m1.xlarge	35.2	11.4	1.4	14.2
c1.medium	22.0	3.9	0.8	19.6
c1.xlarge	88.0	50.0	2.5	62.5
GG.large	12.0	8.8	8.8	46.4
GG.xlarge	36.0	28.1	7.0	37.0

HPCC performance To obtain the performance of virtual EC2 and GoGrid clusters we run the HPCC benchmarks on *unit clusters* comprising a single instance, and on *128-core clusters* comprising 16 `c1.xlarge` instances. Table 9 summarizes the obtained results and, for comparison, results published by HPCC for four modern and similarly-sized HPC clusters [38]. For HPL, only the performance of the `c1.xlarge` is comparable to that of an HPC system. However, for STREAM, and RandomAccess the performance of the EC2 clusters is similar or better than the performance of the HPC clusters. We attribute this mixed behavior mainly to the network characteristics: first, the EC2 platform has much higher latency, which has an important negative impact on the performance of the HPL benchmark; second, the network is shared among different users, a situation which often leads to severe performance degradation [50]. In particular, this relatively low network performance means that the ratio between the theoretical peak performance and achieved HPL performance increases with the number of instances, making the virtual EC2 clusters poorly scalable. Thus, for scientific computing

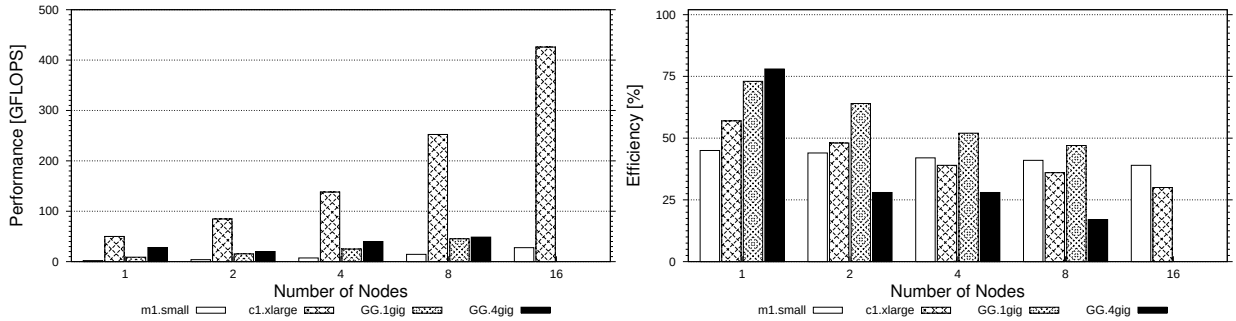

Fig. 5. The HPL (LINPACK) performance of virtual clusters formed with EC2 `m1.small`, EC2 `c1.xlarge`, GoGrid `large`, and GoGrid `xlarge` instances. (*left*) Throughput. (*right*) Efficiency.

TABLE 9
The HPCC performance for various platforms. HPCC-*x* is the system with the HPCC ID *x* [38]. The machines HPCC-224 and HPCC-227, and HPCC-286 and HPCC-289 are of brand TopSpin/Cisco and by Intel Endeavor, respectively. Smaller values are better for the Latency column and worse for the other columns.

Provider, System	Cores or Capacity	Peak Perf. [GFLOPS]	HPL [GFLOPS]	HPL N	DGEMM [GFLOPS]	STREAM [GBps]	RandomAccess [MUPs]	Latency [μs]	Bandwidth [GBps]
EC2, 1 x m1.small	1	4.40	1.96	13,312	2.62	3.49	11.60	-	-
EC2, 1 x m1.large	2	17.60	7.15	28,032	6.83	2.38	54.35	20.48	0.70
EC2, 1 x m1.xlarge	4	35.20	11.38	39,552	8.52	3.47	168.64	17.87	0.92
EC2, 1 x c1.medium	2	22.00	-	13,312	11.85	3.84	46.73	13.92	2.07
EC2, 1 x c1.xlarge	8	88.00	51.58	27,392	44.05	15.65	249.66	14.19	1.49
EC2, 2 x c1.xlarge	16	176.00	84.63	38,656	34.59	15.65	223.54	19.31	1.10
EC2, 4 x c1.xlarge	32	352.00	138.08	54,784	27.74	15.77	280.38	25.38	1.10
EC2, 8 x c1.xlarge	64	704.00	252.34	77,440	3.58	15.89	250.40	35.93	0.97
EC2, 16 x c1.xlarge	128	1,408.00	425.82	109,568	0.23	16.38	207.06	45.20	0.75
EC2, 16 x m1.small	16	70.40	27.80	53,376	4.36	11.95	77.83	68.24	0.10
GoGrid, 1 x GG.large	1	12.00	8.805	10,240	10.01	2.88	17.91	-	-
GoGrid, 4 x GG.large	4	48.00	24.97	20,608	10.34	20.17	278.80	110.11	0.06
GoGrid, 8 x GG.large	8	96.00	45.436	29,184	10.65	20.17	351.68	131.13	0.07
GoGrid, 1 x GG.xlarge	3	36.00	28.144	20,608	10.82	45.71	293.30	16.96	0.97
GoGrid, 4 x GG.xlarge	12	144.00	40.03	41,344	11.31	19.95	307.64	62.20	0.24
GoGrid, 8 x GG.xlarge	24	288.00	48.686	58,496	18.00	20.17	524.33	55.54	1.33
HPCC-227, TopSpin/Cisco	16	102.40	55.23	81,920	4.88	2.95	10.25	6.81	0.66
HPCC-224, TopSpin/Cisco	128	819.20	442.04	231,680	4.88	2.95	10.25	8.25	0.68
HPCC-286, Intel Endeavor	16	179.20	153.25	60,000	10.50	5.18	87.61	1.23	1.96
HPCC-289, Intel Endeavor	128	1,433.60	1,220.61	170,000	10.56	5.17	448.31	2.78	3.47

applications similar to HPL the virtual EC2 clusters can lead to an order of magnitude lower performance for large system sizes (1024 cores and higher). An alternative explanation may be the working set size of HPL, which would agree with the findings of another study on resource virtualization [44]. The performance of the GoGrid clusters with the single core instances is as expected, but we observe scalability problems with the 3 core `GG.xlarge` instances. In comparison with previously reported results, the DGEMM performance of `m1.large` (`c1.medium`) instances is similar to that of Altix4700 (ICE) [29], and the memory bandwidth of Cray X1 (2003) is several times faster than that of the fastest cloud resource currently available [30].

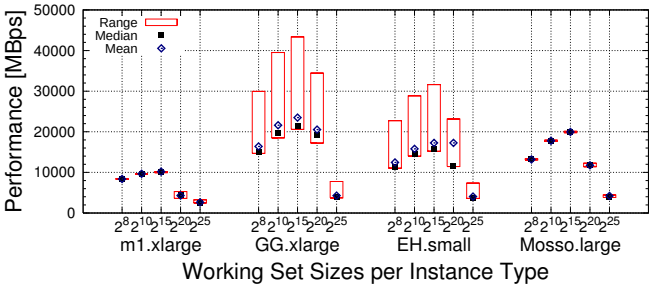

Fig. 6. Performance stability of cloud instance types with the CacheBench benchmark with Rd-Mod-Wr operations.

4.3.4 Performance Stability

An important question related to clouds is *Is the performance stable?* (*Are our results repeatable?*) Previous work on virtualization has shown that many virtualization packages deliver the same performance under identical tests for virtual machines running in an isolated environment [51]. However, it is unclear if this holds for virtual machines running in a large-scale cloud (shared) environment.

To get a first picture of the side-effects caused by the sharing with other users the same physical resource, we have assessed the stability of different clouds by running the LMBench (computation and OS) and CacheBench (I/O) benchmarks multiple times on the same type of virtual resources. For these experiments we have used `m1.xlarge`, `GG.xlarge`, `EH.small`, and

`Mosso.large` resources. Figure 6 summarizes the results for one example benchmark from the CacheBench suite, Rd-Mod-Wr. The `GG.large` and `EH.small` types have important differences between the min, mean, and max performance even for medium working set sizes, such as $10^{10}B$. The best-performer in terms of computation, `GG.xlarge`, is unstable; this makes cloud vendor selection an even more difficult problem. We have performed a longer-term investigation in other work [52].

5 Clouds vs. Other Scientific Computing Infrastructures

In this section we present a comparison between clouds and other scientific computing infrastructures using both complete workloads, and MTC workloads extracted from the complete workloads.

5.1 Method

We use trace-based simulation to compare clouds with scientific computing infrastructures. To this end, we first extract the performance characteristics from long-term workload traces of scientific computing infrastructures; we call these infrastructures *source environments*. Then, we compare these characteristics with those of a cloud execution.

System model We define two performance models of clouds, which differ by the factor that jobs are slowed down. The *cloud with source-like performance* is a theoretical cloud environment that comprises the same resources as the source environment. In this cloud model, the runtimes of jobs executed in the cloud are equal to those recorded in the source environment's workload traces (no slowdown). This model is akin to having a grid being converted into a cloud of identical performance and thus it is useful for assessing the theoretical performance of future and more mature clouds. However, as we have shown in Section 4, in real clouds performance is below the theoretical peak, and for parallel jobs the achieved efficiency is lower than that achieved in grids. Thus, we introduce the second model, the *clouds with real performance*, in which the runtimes of jobs executed in the cloud are extended by a factor, which we call the *slowdown factor*, derived from the empirical evaluation presented in Section 4. The system equivalence between clouds and source environments is assumed in this model, and ensured in practice by the complete system virtualization [53] employed by all the clouds investigated in this work.

Job execution model For job execution we assume exclusive resource use: for each job in the trace, the necessary resources are acquired from the cloud, then released after the job has been executed. We relax this assumption in Section 5.3.4.

System workloads To compare the performance of clouds with other infrastructures, we use both complete workloads, and MTC workloads extracted from the complete workloads using the method described in Section 3.1. Finally we evaluate the performance and the cost of executing MTC workloads in clouds with real performance for various slowdown factors.

Performance metrics We measure the performance of all environments using the three traditional metrics [7]: *wait time (WT)*, *response time (ReT)*, and *bounded slowdown (BSD)*)–the ratio between the job response time in the real vs. an exclusively-used environment, with a bound that eliminates the bias towards short jobs. The BSD is expressed as $BSD = \max(1, ReT/\max(10, ReT - WT))$, where 10 is the bound that eliminates the bias of jobs with runtime below 10 seconds. We compute for each job the three metrics and report for a complete workload the average values for these metrics, AWT, AReT, and ABSD, respectively.

Cost metrics We report for the two cloud models the total cost of workload execution, defined as the number of instance-hours used to complete all the jobs in the workload. This value can be converted directly into the cost for executing the whole workload for $/CPU-hour and similar pricing models, such as Amazon EC2's.

5.2 Experimental Setup

System setup We use the DGSim simulator [18] to analyze the performance of cloud environments. We have extended DGSim with the two cloud models, and used it to simulate the execution of real scientific computing workloads on cloud computing infrastructures. To model the slowdown of jobs when using clouds with real performance, we have used different slowdown factors. Specifically, single-processor jobs are slowed-down by a factor of 7, which is the average performance ratio between theoretical and achieved performance analyzed in Section 4.3.2, and parallel jobs are slowed-down by a factor up to 10 depending on the job size, due to the HPL performance degradation with job size described in Section 4.3.3. In Section 5.3.3, we also present the results of our performance evaluation by using various slowdown factors with the cloud real performance model.

Workload setup We use as input workload the ten workload traces described in Section 3. The traces Grid3 and LCG do not include the job waiting time information; only for these two traces we set the waiting time for all jobs to zero, which favors these two grids in comparison with clouds. The wait time of jobs executed in the cloud (also their AWT) is set to the resource acquisition and release time obtained from real measurements (see Section 4.3.1).

Performance analysis tools We use the Grid Workloads Archive tools [13] to extract the performance metrics from the workload traces of grids and PPIs. We extend these tools to also analyze cloud performance metrics such as cost.

5.3 Results

Our experiments follow four main aspects: performance for complete and MTC-only workloads, the effect of

cloud performance changes on performance and cost metrics, and the performance-cost-security trade-off. We present the experimental results for each main aspect, in turn.

5.3.1 Complete Workloads

We compare the execution in source environments (grids, PPIs, etc.) and in clouds of the ten workload traces described in Table 2. Table 10 summarizes the results of this comparison, on which we comment below.

An order of magnitude better performance is needed for clouds to be useful for daily scientific computing. The performance of the cloud with real performance model is insufficient to make a strong case for clouds replacing grids and PPIs as a scientific computing infrastructure. The response time of these clouds is higher than that of the source environment by a factor of 4-10. In contrast, the response time of the clouds with source-like performance is much better, leading in general to significant gains (up to 80% faster average job response time) and at worst to 1% higher AWT (for traces of Grid3 and LCG, which are assumed conservatively to always have zero waiting time[1]). We conclude that if clouds would offer an order of magnitude higher performance than the performance observed in this study, they would form an attractive alternative for scientific computing, not considering costs.

Price-wise, clouds are reasonably cheap for scientific computing, if the usage and funding scenarios allow it (but usually they do not). Looking at costs, and assuming the external operational costs in the cloud to be zero, one million EC2-hours equate to $100,000. Thus, to execute the total workload of RAL over one year (12 months) would cost $4,029,000 on Amazon EC2. Similarly, the total workload of DAS-2 over one year and a half (18 months) would cost $166,000 on Amazon EC2. Both these sums are much lower than the cost of these infrastructures, which includes resource acquisition, operation, and maintenance. To better understand the meaning of these sums, consider the scenario (disadvantageous for the clouds) in which the original systems would have been sized to accommodate strictly the average system load, and the operation and maintenance costs would have been zero. Even in this scenario using Amazon EC2 is cheaper. We attribute this difference to the economy of scale discussed in a recent study [20]: the price of the basic operations in a very large data center can be an order of magnitude lower than in a grid or data center of regular size. However, despite the apparent cost saving it is not clear that the transition to clouds would have been possible for either of these grids. Under the current performance exhibited by clouds, the use of EC2 would have resulted in response times three to four times higher than in the original system, which would have been in conflict with the mission of RAL as a production environment. A similar concern can be formulated for DAS-2. Moreover, DAS-2 is specifically targeting research in computer science, and its community would not have been satisfied to use commodity resources instead of a state-of-the-art environment comprising among others high-performance lambda networks; other new resource types, such as GPUs and Cell processors, are currently available in grids but not in clouds. Looking at the funding scenario, it is not clear if finance could have been secured for virtual resources; one of the main outcomes of the long-running EGEE project is the creation of a European Grid infrastructure. Related concerns have been formulated elsewhere [20].

Clouds are now a viable alternative for short deadlines. A low and steady job wait time leads to much lower (bounded) slow-down for any cloud model, when compared to the source environment. The average bounded slowdown (ABSD, see Section 5.1) observed in real grids and PPIs is for our traces between 11 and over 500!, but below 3.5 and even 1.5 for the cloud models with low and with high utilization. The meaning of the ABSD metric is application-specific, and the actual ABSD value may seem to over-emphasize the difference between grids and clouds. However, the presence of high and unpredictable wait times even for short jobs, captured here by the high ABSD values, is one of the major concerns in adopting shared infrastructures such as grids [5], [54]. We conclude that cloud is already a viable alternative for scientific computing projects with tight deadline and few short-running jobs remaining, if the project has the needed funds.

5.3.2 MTC Part of the Complete Workloads

We evaluate the performance of clouds using only the MTC workloads extracted from the complete workloads using the method described in Section 3.1. We assume that a user is an MTC user if $B \geq 1,000$ and $J \geq 10,000$; T is considered to be the duration of the workload trace. Table 11 summarizes the results of our evaluation. The results are similar to the results obtained for complete workloads, in the previous section. We observe that the response time of clouds with real performance is higher than that of grids/PPIs by a factor of 2-5. Hence, although the cost of using clouds seems reasonable, significant performance improvement is needed for clouds to be a viable alternative to grids/PPIs for MTC based scientific computing. In addition, similar to results for complete workloads, we observe low and steady wait times hence lower ABSD, and reduced time to solution which makes clouds attractive for MTC based scientific computing.

5.3.3 The Effect of the Cloud Slowdown Factor on Performance and Cost

The slowdown factor is the factor by which the job runtime changes between the source environment and

1. Although we realize the Grid3 and LCG grids do not have zero waiting time, we follow a conservative approach in which we favor grids against clouds, as the latter are the *new* technology.

TABLE 10
The results of the comparison between workload execution in source environments (grids, PPIs, etc.) and in clouds. The "-" sign denotes missing data in the original traces. For the two Cloud models AWT=80s (see text). The total cost for the two Cloud models is expressed in millions of CPU-hours.

	Source env. (Grid/PPI)			Cloud (real performance)			Cloud (source performance)		
Trace ID	AWT [s]	AReT [s]	ABSD (10s)	AReT [s]	ABSD (10s)	Total Cost [CPU-h,M]	AReT [s]	ABSD (10s)	Total Cost [CPU-h,M]
DAS-2	432	802	11	2,292	2.39	2	450	2	1.19
RAL	13,214	27,807	68	131,300	1	40	18,837	1	6.39
GLOW	9,162	17,643	55	59,448	1	3	8,561	1	0.60
Grid3	-	7,199	-	50,470	3	19	7,279	3	3.60
SharcNet	31,017	61,682	242	219,212	1	73	31,711	1	11.34
LCG	-	9,011	-	63,158	1	3	9,091	1	0.62
CTC SP2	25,748	37,019	78	75,706	1	2	11,351	1	0.30
SDSC SP2	26,705	33,388	389	46,818	2	1	6,763	2	0.16
LANL O2K	4,658	9,594	61	37,786	2	1	5,016	2	0.26
SDSC DS	32,271	33,807	516	57,065	2	2	6,790	2	0.25

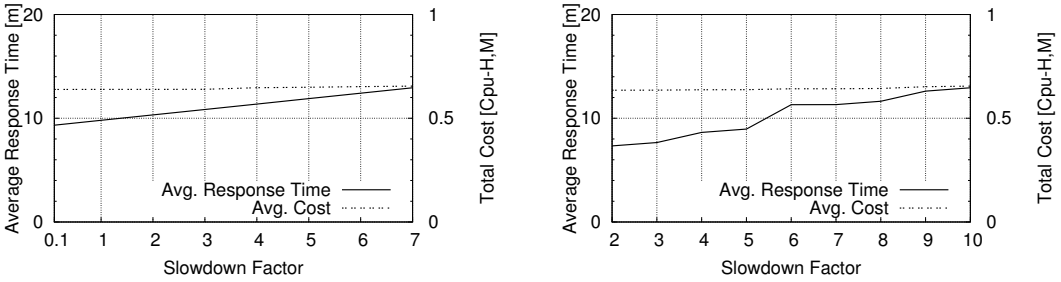

Fig. 7. Performance and cost of using cloud resources for MTC workloads with various slowdown factors for sequential jobs (*left*), and parallel jobs(*right*) using the DAS-2 trace.

the cloud (see Section 5.1). In previous sections, we have used a slowdown factor of 7 for sequential jobs, and 10 for parallel jobs for modeling clouds with real performance. We now evaluate the performance of clouds with real performance using only the MTC workloads with various slowdown factors for both sequential and parallel jobs. Similar to previous section, when extracting the MTC workload from complete workloads we assume that a user is an MTC user if $B \geq 1,000$ and $J \geq 10,000$.

Figure 7 shows the average response time and cost of clouds with real performance with various slowdown factors for sequential (left) and parallel (right) jobs using the DAS-2 trace. As the slowdown factor increases, we observe a steady but slow increase in cost and response time for both sequential and parallel jobs. This is expected: the higher the response time, the longer a cloud resource is used, increasing the total cost. The sequential jobs dominate the workload both in number of jobs and in consumed CPU time, and their average response time increases linearly with the performance slowdown; thus, improving the performance of clouds for sequential jobs should be the first priority of cloud providers.

TABLE 12
Relative strategy performance: resource bulk allocation (S2) vs. resource acquisition and release per job (S1). Only performance differences above 5% are shown.

Relative Cost	DAS-2	Grid3	LCG	LANL O2K
$\frac{\lvert S2-S1\rvert}{S1} \times 100$ [%]	30.2	11.5	9.3	9.1

5.3.4 Performance and Security vs. Cost

Currently, clouds lease resources but do not offer a resource management service that can use the leased resources. Thus, the cloud adopter may use any of the resource management middleware from grids and PPIs; for a review of grid middleware we refer to our recent work [55]. We have already introduced the basic concepts of cloud resource management in Section 4.2, and explored the potential of a cloud resource management strategy (*strategy S1*) for which resources are acquired

TABLE 11
The results of the comparison between workload execution in source environments (grids, PPIs, etc.) and in clouds with only the MTC part extracted from the complete workloads. The LCG, CTC SP2, SDSC SP2, and SDSC DS traces are not presented, as they do not have enough MTC users (the criterion is described in text).

	Source env. (Grid/PPI)			Cloud (real performance)			Cloud (source performance)		
Trace ID	AWT [s]	AReT [s]	ABSD (10s)	AReT [s]	ABSD (10s)	Total Cost [CPU-h,M]	AReT [s]	ABSD (10s)	Total Cost [CPU-h,M]
DAS-2	70	243	3	776	2	0.65	252	2	0.61
RAL	4,866	18,694	24	68,847	1	6.60	17,272	1	1.80
GLOW	6,062	13,396	41	29,416	1	1.40	7,413	1	0.42
Grid3	-	7,422	-	29,769	4	10.61	7,502	4	3.30
SharcNet	10,387	30,092	6	141,510	1	7.10	20,182	1	1.09
LANL O2K	635.78	1,715	4	94	2	0.10	1,171	1	<0.01

and released for each submitted job in Section 5. This strategy has good security and resource setup flexibility, but may incur high time and cost overheads, as resources that could otherwise have been reused are released as soon as the job completes. As an alternative, we investigate now the potential of a cloud resource management strategy in which resources are allocated in bulk for all users, and released only when there is no job left to be served (*strategy S2*). To compare these two cloud resource management strategies, we use the experimental setup described in Section 5.2; Table 12 shows the obtained results. The maximum relative cost difference between the strategies is for these traces around 30% (the DAS-2 trace); in three cases, around 10% of the total cost is to be gained. Given these cost differences, *we raise as a future research problem optimizing the application execution as a cost-performance-security trade-off*.

6 Related work

In this section we review related work from three areas: clouds, virtualization, and system performance evaluation. Our work also comprises the first characterization of the MTC component in existing scientific computing workloads.

Performance Evaluation of Clouds and Virtualized Environments There has been a recent spur of research activity in assessing the performance of virtualized resources, in cloud computing environments [9], [10], [11], [56], [57] and in general [8], [24], [51], [58], [59], [60], [61]. In contrast to this body of previous work, ours is different in scope: we perform extensive measurements using general purpose and high-performance computing benchmarks to compare several clouds, and we compare clouds with other environments based on real long-term scientific computing traces. Our study is also much broader in size: we perform in this work an evaluation using over 25 individual benchmarks on over 10 cloud instance types, which is an order of magnitude larger than previous work (though size does not simply add to quality).

Performance studies using general purpose benchmarks have shown that the overhead incurred by virtualization can be below 5% for computation [24], [51] and below 15% for networking [24], [58]. Similarly, the performance loss due to virtualization for parallel I/O and web server I/O has been shown to be below 30% [62] and 10% [63], [64], respectively. In contrast to these, our work shows that virtualized resources obtained from public clouds can have a much lower performance than the theoretical peak.

Recently, much interest for the use of virtualization has been shown by the HPC community, spurred by two seminal studies [8], [65] that find virtualization overhead to be negligible for compute-intensive HPC kernels and applications such as the NAS NPB benchmarks; other studies have investigated virtualization performance for specific HPC application domains [61], [66], or for mixtures of Web and HPC workloads running on virtualized (shared) resources [67]. Our work differs significantly from these previous approaches in target (clouds as black boxes vs. owned and controllable infrastructure) and in size. For clouds, the study of performance and cost of executing a scientific workflow, Montage, in clouds [9] investigates cost-performance trade-offs between clouds and grids, but uses a single application on a single cloud, and the application itself is remote from the mainstream HPC scientific community. Also close to our work is the seminal study of Amazon S3 [10], which also includes a performance evaluation of file transfers between Amazon EC2 and S3. Our work complements this study by analyzing the performance of Amazon EC2, the other major Amazon cloud service; we also test more clouds and use scientific workloads. Several small-scale performance studies of Amazon EC2 have been recently conducted: the study of Amazon EC2 performance using the NPB benchmark suite [11] or selected HPC benchmarks [68], the early comparative study of Eucalyptus and EC2 performance [56], the study of file transfer performance between Amazon EC2 and S3 [69], etc. An early comparative study of the DawningCloud and several operational models [12] extends the comparison method employed for Eucalyptus [56], but uses job emulation instead of job execution. Our performance evaluation results extend and complement these previous findings, and gives more insights into the performance of EC2 and other clouds.

Other (Early) Performance Evaluation Much work has been put into the evaluation of novel supercomputers [27], [29], [30], [31], [47], [48] and non-traditional systems [5], [32], [37], [49], [70] for scientific computing. We share much of the used methodology with previous work; we see this as an advantage in that our results are readily comparable with existing results. The two main differences between this body of previous work and ours are that we focus on a different platform (that is, clouds) and that we target a broader scientific computing community (e.g., also users of grids and small clusters).

Other Cloud Work Recent work [12], [71] has considered running mixtures of MTC with other workloads in cloud-like environments. For this direction of research, our findings can be seen as further motivation and source of realistic setup parameters.

7 Conclusion and Future Work

With the emergence of cloud computing as a paradigm in which scientific computing can done exclusively on resources leased only when needed from big data centers, e-scientists are faced with a new platform option. However, the initial target workloads of clouds does not match the characteristics of MTC-based scientific computing workloads. Thus, in this paper we seek to answer the research question *Is the performance of clouds sufficient for MTC-based scientific computing?* To this end, we first investigate the presence of an MTC component in existing scientific computing workloads, and find that

this presence is significant both in number of jobs and in resources consumed. Then, we perform an empirical performance evaluation of four public computing clouds, including Amazon EC2, one of the largest commercial clouds currently in production. Our main finding here is that the compute performance of the tested clouds is low. Last, we compare the performance and cost of clouds with those of scientific computing alternatives such as grids and parallel production infrastructures. We find that, while current cloud computing services are insufficient for scientific computing at large, they may still be a good solution for the scientists who need resources instantly and temporarily.

We will extend this work with additional analysis of the other services offered by clouds, and in particular storage and network; how do they respond to to the combined stress of workloads with different characteristics and requirements that the diverse population of cloud users are supposed to incur in the future? We will also extend the performance evaluation with other real and synthetic applications, toward creating a performance database for the scientific community.

Standing the test of time. The usefulness of our empirical evaluation part of this work (Section 4.3) may be reduced with the commercialization of new cloud services. For example, since mid-July 2010 a new commercial compute service by Amazon, the Cluster Compute instances, is targeted at HPC users. The increase in performance for this new service versus the Amazon instances tested in our work can be up to a factor of 8.5 [72], which is similar to the performance gap found by our performance evaluation. The difference in performance for the Cluster Compute instances cannot be explained only by the superior resource performance—the compute performance of the Cluster Compute instances is, for example, only a factor of 1.5 times better than that of the best-performing instance tested in our study. Another possible contributor may be that the new instance type offers dedicated infrastructure (that is, compute and network resources). Thus, these cloud instances are operated in a "shared-nothing" mode; historically, such clusters tend to have low utilization [73], which in turn threatens to cancel out the commercial benefits. Our performance evaluation results remain representative for clouds that multiplex their resources among their users, at least until an isolation technology is able to limit access to compute, memory, network, and I/O resources with low overhead; recent yet early attempts in this direction, such as the Linux containers [74], are promising. Our performance evaluation may also be indicative, as a cross-section analysis of the offerings available on the market, for the differences between the cloud operators present on the market at any given time.

Acknowledgements This work is partially funded by the European Union under grant agreement number 261585/SHIWA Project.

REFERENCES

[1] Amazon Inc., "Amazon Elastic Compute Cloud (Amazon EC2)," Dec 2008, [Online] Available: http://aws.amazon.com/ec2/.
[2] GoGrid, "GoGrid cloud-server hosting," Dec 2008, [Online] Available: http://www.gogrid.com.
[3] A. Iosup, O. O. Sonmez, S. Anoep, and D. H. J. Epema, "The performance of bags-of-tasks in large-scale distributed systems," in *HPDC*. ACM, 2008, pp. 97–108.
[4] I. Raicu, Z. Zhang, M. Wilde, I. T. Foster, P. H. Beckman, K. Iskra, and B. Clifford, "Toward loosely coupled programming on petascale systems," in *SC*. ACM, 2008, p. 22.
[5] 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, 2006, pp. 262–269.
[6] U. Lublin and D. G. Feitelson, "Workload on parallel supercomputers: modeling characteristics of rigid jobs," *J.Par.&Distr.Comp.*, vol. 63, no. 11, pp. 1105–1122, 2003.
[7] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong, "Theory and practice in parallel job scheduling," in *JSSPP*, ser. LNCS, vol. 1291. Springer-Verlag, 1997, pp. 1–34.
[8] L. Youseff, R. Wolski, B. C. Gorda, and C. Krintz, "Paravirtualization for HPC systems," in *ISPA Workshops*, ser. LNCS, vol. 4331. Springer-Verlag, 2006, pp. 474–486.
[9] E. Deelman, G. Singh, M. Livny, J. B. Berriman, and J. Good, "The cost of doing science on the cloud: the Montage example," in *SC*. IEEE/ACM, 2008, p. 50.
[10] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for science grids: a viable solution?" in *DADC '08: Proceedings of the 2008 international workshop on Data-aware distributed computing*. ACM, 2008, pp. 55–64.
[11] E. Walker, "Benchmarking Amazon EC2 for HP Scientific Computing," *Login*, vol. 33, no. 5, pp. 18–23, Nov 2008.
[12] L. Wang, J. Zhan, W. Shi, Y. Liang, and L. Yuan, "In cloud, do mtc or htc service providers benefit from the economies of scale?" in *SC-MTAGS*, 2009.
[13] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. Epema, "The Grid Workloads Archive," *Future Generation Comp. Syst.*, vol. 24, no. 7, pp. 672–686, 2008.
[14] D. Thain, J. Bent, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. Livny, "Pipeline and batch sharing in grid workloads," in *HPDC*. IEEE, 2003, pp. 152–161.
[15] S. Ostermann, A. Iosup, R. Prodan, T. Fahringer, and D. H. J. Epema, "On the characteristics of grid workflows," in *CGIW*, 2008, pp. 431–442.
[16] The Parallel Workloads Archive Team , "The parallel workloads archive logs," Jan. 2009, [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/logs.html.
[17] Y.-S. Kee, H. Casanova, and A. A. Chien, "Realistic modeling and svnthesis of resources for computational grids," in *SC*, 2004, p. 54.
[18] A. Iosup, O. O. Sonmez, and D. H. J. Epema, "DGSim: Comparing grid resource management architectures through trace-based simulation," in *Euro-Par*, ser. LNCS, vol. 5168. Springer-Verlag, 2008, pp. 13–25.
[19] L. Youseff, M. Butrico, and D. Da Silva, "Towards a unified ontology of cloud computing," in *Proc. of the Grid Computing Environments Workshop (GCE08)*, Nov 2008.
[20] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009.
[21] R. Prodan and S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers," in *GRID*, 2009, pp. 1–10.
[22] A. Iosup, M. Jan, O. O. Sonmez, and D. H. J. Epema, "The characteristics and performance of groups of jobs in grids," in *Euro-Par*, 2007, pp. 382–393.
[23] R. H. Saavedra and A. J. Smith, "Analysis of benchmark characteristics and benchmark performance prediction," *ACM Trans. Comput. Syst.*, vol. 14, no. 4, pp. 344–384, 1996.
[24] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *SOSP*. ACM, 2003, pp. 164–177.
[25] A. Kowalski, "Bonnie - file system benchmarks," Jefferson Lab," Tech.Rep., Oct 2002, [Online] Available: http://cc.jlab.org/docs/scicomp/benchmark/bonnie.html.

[26] M. Babcock, "XEN benchmarks, Tech.Rep.," Aug 2007, [Online] Available: mikebabcock.ca/linux/xen/.
[27] J. S. Vetter, S. R. Alam, T. H. D. Jr., M. R. Fahey, P. C. Roth, and P. H. Worley, "Early evaluation of the Cray XT3," in *IPDPS*, 2006.
[28] P. Luszczek, D. H. Bailey, J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi, "S12 - The HPC Challenge (HPCC) benchmark suite." in *SC*. ACM, 2006, p. 213.
[29] S. Saini, D. Talcott, D. C. Jespersen, M. J. Djomehri, H. Jin, and R. Biswas, "Scientific application-based performance comparison of SGI Altix 4700, IBM POWER5+, and SGI ICE 8200 supercomputers," in *SC*. IEEE/ACM, 2008, p. 7.
[30] T. H. Dunigan, M. R. Fahey, J. B. W. III, and P. H. Worley, "Early evaluation of the Cray X1," in *SC*. ACM, 2003, p. 18.
[31] S. R. Alam, R. F. Barrett, M. Bast, M. R. Fahey, J. A. Kuehn, C. McCurdy, J. Rogers, P. C. Roth, R. Sankaran, J. S. Vetter, P. H. Worley, and W. Yu, "Early evaluation of IBM BlueGene/P," in *SC*. ACM, 2008, p. 23.
[32] R. Biswas, M. J. Djomehri, R. Hood, H. Jin, C. C. Kiris, and S. Saini, "An application-based performance characterization of the Columbia Supercluster," in *SC*. IEEE, 2005, p. 26.
[33] L. McVoy and C. Staelin, "LMbench - tools for performance analysis," [Online] Available: http://www.bitmover.com/lmbench/, Dec 2008.
[34] T. Bray, "Bonnie," 1996, [Online] Available: http://www.textuality.com/bonnie/, Dec 2008.
[35] P. J. Mucci and K. S. London, "Low level architectural characterization benchmarks for parallel computers," U. Tennessee, Tech. Rep. UT-CS-98-394, 1998.
[36] N. Yigitbasi, A. Iosup, S. Ostermann, and D. Epema, "C-meter: A framework for performance analysis of computing clouds," in *Proceedings of CCGRID'09*, 2009, pp. 472–477.
[37] A. Iosup and D. H. J. Epema, "GrenchMark: A framework for analyzing, testing, and comparing grids," in *CCGrid*, 2006, pp. 313–320.
[38] The HPCC Team , "HPCChallenge results," Jan. 2009, [Online]. Available: http://icl.cs.utk.edu/hpcc/hpcc_results.cgi.
[39] J. Worringen and K. Scholtyssik, "MP-MPICH: User documentation & technical notes," Jun 2002.
[40] J. Dongarra et al., "Basic Linear Algebra Subprograms Technical Forum Standard," *Int'l. J. of High Perf. App. and Supercomputing*, vol. 16, no. 1, pp. 1–111, 2002.
[41] K. Goto and R. A. v. d. Geijn, "Anatomy of high-performance matrix multiplication," *ACM Trans. Math. Softw.*, vol. 34, no. 3, pp. 1–25, 2008.
[42] Advanced Clustering Tech., "Linpack problem size analyzer," Dec 2008, [Online] Available: http://www.advancedclustering.com/.
[43] J. Dongarra, P. Luszczek, and A. Petitet, "The linpack benchmark: past, present and future," *Concurrency and Computation: Practice and Experience*, vol. 15, no. 9, pp. 803–820, 2003.
[44] A. Verma, P. Ahuja, and A. Neogi, "Power-aware dynamic placement of hpc applications," in *ICS*. ACM, 2008, pp. 175–184.
[45] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments," in *OSDI*. USENIX, 2008, pp. 29–42.
[46] S. Ostermann, A. Iosup, N. M. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "An early performance analysis of cloud computing services for scientific computing," TU Delft, Tech. Rep., Dec 2008, [Online] Available: http://pds.twi.tudelft.nl/reports/2008/PDS-2008-006.pdf.
[47] F. Petrini, D. J. Kerbyson, and S. Pakin, "The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of ASCI Q," in *SC*. ACM, 2003, p. 55.
[48] D. J. Kerbyson, A. Hoisie, and H. J. Wasserman, "A performance comparison between the Earth Simulator and other terascale systems on a characteristic ASCI workload," *Concurrency - Practice and Experience*, vol. 17, no. 10, pp. 1219–1238, 2005.
[49] F. Petrini, G. Fossum, J. Fernández, A. L. Varbanescu, M. Kistler, and M. Perrone, "Multicore surprises: Lessons learned from optimizing Sweep3D on the Cell Broadband Engine," in *IPDPS*. IEEE, 2007, pp. 1–10.
[50] R. H. Arpaci-Dusseau, A. C. Arpaci-Dusseau, A. Vahdat, L. T. Liu, T. E. Anderson, and D. A. Patterson, "The interaction of parallel and sequential workloads on a network of workstations," in *SIGMETRICS*, 1995, pp. 267–278.
[51] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J. N. Matthews, "Xen and the art of repeated research," in *USENIX ATC*, 2004, pp. 135–144.
[52] A. Iosup, N. M. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," TU Delft, Tech. Rep. PDS-2010-002, Jan 2010, [Online] Available: http://pds.twi.tudelft.nl/reports/2010/PDS-2010-002.pdf.
[53] T. Killalea, "Meet the virts," *Queue*, vol. 6, no. 1, pp. 14–18, 2008.
[54] D. Nurmi, R. Wolski, and J. Brevik, "Varq: virtual advance reservations for queues," in *HPDC*. ACM, 2008, pp. 75–86.
[55] A. Iosup, D. H. J. Epema, T. Tannenbaum, M. Farrellee, and M. Livny, "Inter-operating grids through delegated matchmaking," in *SC*. ACM, 2007, p. 13.
[56] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system," 2008, uCSD Tech.Rep. 2008-10. [Online] Available: http://eucalyptus.cs.ucsb.edu/.
[57] B. Quétier, V. Néri, and F. Cappello, "Scalability comparison of four host virtualization tools," *J. Grid Comput.*, vol. 5, pp. 83–98, 2007.
[58] A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel, "Diagnosing performance overheads in the Xen virtual machine environment," in *VEE*. ACM, 2005, pp. 13–23.
[59] N. Sotomayor, K. Keahey, and I. Foster, "Overhead matters: A model for virtual resource management," in *VTDC*. IEEE, 2006, pp. 4–11.
[60] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott, "Proactive fault tolerance for HPC with Xen virtualization," in *ICS*. ACM, 2007, pp. 23–32.
[61] L. Youseff, K. Seymour, H. You, J. Dongarra, and R. Wolski, "The impact of paravirtualized memory hierarchy on linear algebra computational kernels and software," in *HPDC*. ACM, 2008, pp. 141–152.
[62] W. Yu and J. S. Vetter, "Xen-based HPC: A parallel I/O perspective," in *CCGrid*. IEEE, 2008, pp. 154–161.
[63] L. Cherkasova and R. Gardner, "Measuring CPU overhead for I/O processing in the Xen virtual machine monitor," in *USENIX ATC*, 2005, pp. 387–390.
[64] U. F. Minhas, J. Yadav, A. Aboulnaga, and K. Salem, "Database systems on virtual machines: How much do you lose?" in *ICDE Workshops*. IEEE, 2008, pp. 35–41.
[65] W. Huang, J. Liu, B. Abali, and D. K. Panda, "A case for high performance computing with virtual machines," in *ICS*. ACM, 2006, pp. 125–134.
[66] L. Gilbert, J. Tseng, R. Newman, S. Iqbal, R. Pepper, O. Celebioglu, J. Hsieh, and M. Cobban, "Performance implications of virtualization and hyper-threading on high energy physics applications in a grid environment," in *IPDPS*. IEEE, 2005.
[67] J. Zhan, L. Wang, B. Tu, Y. Li, P. Wang, W. Zhou, and D. Meng, "Phoenix cloud: Consolidating different computing loads on shared cluster system for large organization," in *CCA-08 Posters*, 2008, pp. 7–11.
[68] C. Evangelinos and C. N. Hill, "Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazons ec2," in *CCA-08*, 2008, pp. 1–6.
[69] M.-E. Bgin, B. Jones, J. Casey, E. Laure, F. Grey, C. Loomis, and R. Kubli, "Comparative study: Grids and clouds, evolution or revolution?" CERN" EGEE-II Report, June 2008, [Online] Available: https://edms.cern.ch/file/925013/3/EGEE-Grid-Cloud.pdf.
[70] S. Williams, J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. A. Yelick, "The potential of the Cell processor for scientific computing," in *Conf. Computing Frontiers*. ACM, 2006, pp. 9–20.
[71] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," UCBerkeley, Tech. Rep. UCB/EECS-2010-87, May 2010. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-87.html
[72] L. Vu, "Berkeley Lab contributes expertise to new Amazon Web Services offering," Jul 2010, [Online] Available: http://www.lbl.gov/cs/Archive/news071310.html.
[73] J. P. Jones and B. Nitzberg, "Scheduling for parallel supercomputing: A historical perspective of achievable utilization," in *JSSPP*, ser. LNCS, vol. 1659. Springer-Verlag, 1999, pp. 1–16.
[74] lxc Linux Containers Team, "Linux containers overview," Aug 2010, [Online] Available: http://lxc.sourceforge.net/.

Alexandru Iosup received his Ph.D. in Computer Science in 2009 from the Delft University of Technology (TU Delft), the Netherlands. He is currently an Assistant Professor with the Parallel and Distributed Systems Group at TU Delft. He was a visiting scholar at U.Wisconsin-Madison and U.California-Berkeley in the summers of 2006 and 2010, respectively. He is the co-founder of the Grid Workloads, the Peer-to-Peer Trace, and the Failure Trace Archives, which provide open access to workload and resource operation traces from large-scale distributed computing environments. He is the author of over 50 scientific publications and has received several awards and distinctions, including best paper awards at IEEE CCGrid 2010, Euro-Par 2009, and IEEE P2P 2006. His research interests are in the area of distributed computing (keywords: massively multiplayer online games, grid and cloud computing, peer-to-peer systems, scheduling, performance evaluation, workload characterization).

Simon Ostermann received Bakk.techn. and Dipl.-Ing. degrees from the University of Innsbruck, Austria, in 2006 and 2008, respectively. Since 2008 he is following a doctoral track in Computer Science with the Distributed and Parallel Systems Group at the Institute for Computer Science, University of Innsbruck. His research interests are in the areas of resource management and scheduling in the area of grid and cloud computing. He is the author of over 10 journal and conference publications.

M.Nezih Yigitbasi received BSc and MSc degrees from the Computer Engineering Department of the Istanbul Technical University, Turkey, in 2006 and 2008, respectively. Since September 2008, he is following a doctoral track in Computer Science within the Parallel and Distributed Systems Group, Delft University of Technology. His research interests are in the areas of resource management, scheduling, design and performance evaluation of large-scale distributed systems, in particular grids and clouds.

Radu Prodan received the Ph.D. degree from Vienna University of Technology, Vienna, Austria, in 2004. He is currently an Assistant Professor at the Institute of Computer Science, University of Innsbruck, Austria. His research in the area of parallel and distributed systems comprise programming methods, compiler technology, performance analysis, and scheduling. He participated in several national and European projects. He is currently coordinating three Austrian projects and was workpackage Leader in the IST-034601 (edutain@grid) and 26185 (SHIWA) projects. He is the author of over 70 journal and conference publications and one book. Radu Prodan was the recipient of an IEEE Best Paper Award.

Thomas Fahringer received the Ph.D. degree in 1993 from the Vienna University of Technology. Between 1990 and 1998, he worked as an assistant professor at the University of Vienna, where he was promoted as an associate professor in 1998. Since 2003, he has been a full professor of computer science in the Institute of Computer Science, University of Innsbruck, Austria. His main research interests include software architectures, programming paradigms, compiler technology, performance analysis, and prediction for parallel and distributed systems. He coordinated the IST-034601 edutain@grid project and was involved in numerous other Austrian and international European projects. He is the author of more than 100 papers, including three books. Prof. Fahringer was the recipient of two best paper awards from the ACM and the IEEE.

Dick H.J. Epema received the MSc and PhD degrees in mathematics from Leiden University, Leiden, the Netherlands, in 1979 and 1983, respectively. Since 1984, he has been with the Department of Computer Science of Delft University of Technology, where he is currently an associate professor in the Parallel and Distributed Systems Group. During 1987-1988, the fall of 1991, and the summer of 1998, he was a visiting scientist at the IBM T.J. Watson Research Center in New York. In the fall of 1992, he was a visiting professor at the Catholic University of Leuven, Belgium, and in the fall of 2009 he spent a sabbatical at UCSB. His research interests are in the areas of performance analysis, distributed systems, peer-to-peer systems, and grids. He has co-authored over 70 papers in peer-reviewed conferences and journals, and was a general co-chair of Euro-Par 2009 and IEEE P2P 2010.