Performance analysis of cloud computing services for many-tasks scientific computing

Citation for published version (APA):

DOI:
10.1109/TPDS.2011.66

Document status and date:
Published: 01/01/2011

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 07. Jun. 2019
Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing

Alexandru Iosup, Member, IEEE, Simon Ostermann, M. Nezih Yigitbasi, Member, IEEE, Radu Prodan, Member, IEEE, Thomas Fahringer, Member, IEEE, and Dick H. J. 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 (BoTs), 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 interrelated) tasks. With MTC, a
paradigm at the intersection of HPC and HTC, it is possible
to demand systems to operate at high utilizations, similar to
those of current production grids (over 80 percent [5]) and
Parallel Production Infrastructures (PPIs) (over 60 percent
[6]), and much higher than those of the systems that clouds
originally intended to replace (servers with 10-20 percent
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 hard-
ware, thus increasing the concurrency of users but poten-
tially lowering the attainable performance.

These three main differences between scientific comput-
ing 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 work-
loads with similar performance (that is, for traditional perfor-
ance 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 microbenchmarks
   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 per-
   formance 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 commer-
cial 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 pre-
vious 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 [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 work-
load 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 applica-
tions 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 clouds on
the market are large enough to accommodate requests for
even 16 or 32 coallocated resources. Third, our selection
already covers a wide range of quantitative and qualitative
cloud characteristics, as summarized in Table 1 and our
cloud survey [21], respectively. We describe in the follow-
ing Amazon EC2; the other three, GoGrid (GG), Elasti-
chHosts (EH), and Mosso, are IaaS clouds with provisioning,
billing, and availability and performance guarantees similar
to Amazon EC2’s.
The Amazon Elastic Computing Cloud 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 percent of the time. The security of the Amazon services has been investigated elsewhere [10].

### Table 1: The Resource Characteristics for the Instance Types Offered by the Four Selected Clouds

<table>
<thead>
<tr>
<th>Name</th>
<th>Cores (ECUs)</th>
<th>RAM [GB]</th>
<th>Archi. [bit]</th>
<th>Disk [GB]</th>
<th>Cost [$/h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EC2</td>
<td>1 (1)</td>
<td>1.7</td>
<td>32</td>
<td>160</td>
<td>0.1</td>
</tr>
<tr>
<td>m1.small</td>
<td>2 (4)</td>
<td>7.5</td>
<td>64</td>
<td>850</td>
<td>0.4</td>
</tr>
<tr>
<td>m1.large</td>
<td>4 (8)</td>
<td>15.0</td>
<td>64</td>
<td>1,690</td>
<td>0.8</td>
</tr>
<tr>
<td>m1.xlarge</td>
<td>2 (5)</td>
<td>1.7</td>
<td>32</td>
<td>350</td>
<td>0.2</td>
</tr>
<tr>
<td>c1.xlarge</td>
<td>8 (20)</td>
<td>7.0</td>
<td>64</td>
<td>1,690</td>
<td>0.8</td>
</tr>
</tbody>
</table>

### Table 2: The Characteristics of the Workload Traces

<table>
<thead>
<tr>
<th>Trace ID, Source (Trace ID in Archive)</th>
<th>Time [hrs]</th>
<th>Trace ID in Archive</th>
<th>Number of Jobs</th>
<th>Number of Users</th>
<th>System Size</th>
<th>System Load [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid Workloads Archive [13], 6 traces</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. DAS-2 (1)</td>
<td>18</td>
<td>1.1M</td>
<td>333</td>
<td>5</td>
<td>0.4K</td>
<td>15</td>
</tr>
<tr>
<td>2. RAL (6)</td>
<td>12</td>
<td>0.2M</td>
<td>208</td>
<td>1</td>
<td>0.8K</td>
<td>85</td>
</tr>
<tr>
<td>3. GLOW (7)</td>
<td>3</td>
<td>0.2M</td>
<td>16</td>
<td>1</td>
<td>1.6K</td>
<td>60</td>
</tr>
<tr>
<td>4. Grid (8)</td>
<td>18</td>
<td>1.3M</td>
<td>19</td>
<td>29</td>
<td>3.5K</td>
<td></td>
</tr>
<tr>
<td>5. ShareNet (10)</td>
<td>13</td>
<td>1.1M</td>
<td>412</td>
<td>10</td>
<td>6.8K</td>
<td>-</td>
</tr>
<tr>
<td>6. LCG (11)</td>
<td>1</td>
<td>0.2M</td>
<td>216</td>
<td>200+</td>
<td>24.4K</td>
<td>-</td>
</tr>
</tbody>
</table>

Parallel Workloads Archive [16], 4 traces

<table>
<thead>
<tr>
<th>Trace ID, Source (Trace ID in Archive)</th>
<th>Time [hrs]</th>
<th>Trace ID in Archive</th>
<th>Number of Jobs</th>
<th>Number of Users</th>
<th>System Size</th>
<th>System Load [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>7. CTC SPZ (6)</td>
<td>11</td>
<td>0.1M</td>
<td>679</td>
<td>1</td>
<td>0.4K</td>
<td>86</td>
</tr>
<tr>
<td>8. SDSC SP2 (9)</td>
<td>24</td>
<td>0.1M</td>
<td>437</td>
<td>1</td>
<td>0.1K</td>
<td>83</td>
</tr>
<tr>
<td>9. LANLNO3 (10)</td>
<td>5</td>
<td>0.1M</td>
<td>337</td>
<td>1</td>
<td>2.0K</td>
<td>64</td>
</tr>
<tr>
<td>10. SDSC OS (19)</td>
<td>13</td>
<td>0.1M</td>
<td>460</td>
<td>1</td>
<td>1.7K</td>
<td>63</td>
</tr>
</tbody>
</table>

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 toward 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 ≥ 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 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 \( B \). Fig. 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 tasks 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 microbenchmarks 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 microbenchmarks 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.

<table>
<thead>
<tr>
<th>Trace ID</th>
<th>Simple criterion</th>
<th>Complex criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BoTs ( \geq 100 ) Jobs [%]</td>
<td>Tasks ( \geq 1,000 ) Jobs [%]</td>
</tr>
<tr>
<td>DAS-2</td>
<td>90</td>
<td>95</td>
</tr>
<tr>
<td>RAL</td>
<td>81</td>
<td>78</td>
</tr>
<tr>
<td>GLOW</td>
<td>95</td>
<td>75</td>
</tr>
<tr>
<td>Grid3</td>
<td>100</td>
<td>97</td>
</tr>
<tr>
<td>ShareNet</td>
<td>40</td>
<td>95</td>
</tr>
<tr>
<td>LCG</td>
<td>61</td>
<td>39</td>
</tr>
<tr>
<td>CTC SP2</td>
<td>16</td>
<td>13</td>
</tr>
<tr>
<td>SDSC SP2</td>
<td>13</td>
<td>93</td>
</tr>
<tr>
<td>LANL O2</td>
<td>66</td>
<td>14</td>
</tr>
<tr>
<td>SDSC D5</td>
<td>29</td>
<td>7</td>
</tr>
</tbody>
</table>

CPUT stands for Total CPU Time.
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 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 $\lambda = 1 \text{s}^{-1}$.

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 microbenchmarks 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 multiprocess 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 resources 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 preconfigured 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 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].

### TABLE 4

<table>
<thead>
<tr>
<th>Type</th>
<th>Suite/Benchmark</th>
<th>Resource</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI</td>
<td>LMBench/All [24]</td>
<td>Many</td>
<td>Many</td>
</tr>
<tr>
<td>SI</td>
<td>Bonnie/All [25], [26]</td>
<td>Disk</td>
<td>MBps</td>
</tr>
<tr>
<td>SI</td>
<td>CacheBench/All [27]</td>
<td>Memory</td>
<td>MBps</td>
</tr>
<tr>
<td>MI</td>
<td>HPC/HP/L [28], [29]</td>
<td>CPU</td>
<td>GFLOPS</td>
</tr>
<tr>
<td>MI</td>
<td>HPC/DGEMM [30]</td>
<td>CPU</td>
<td>GFLOPS</td>
</tr>
<tr>
<td>MI</td>
<td>HPC/STREAM [30]</td>
<td>Memory</td>
<td>GBps</td>
</tr>
<tr>
<td>MI</td>
<td>HPC/RandomAccess [31]</td>
<td>Network</td>
<td>MUPS</td>
</tr>
<tr>
<td>MI</td>
<td>HPC/bachif(lat, bw) [32]</td>
<td>Comm.</td>
<td>$\mu$s, GBps</td>
</tr>
</tbody>
</table>

B, FLOP, U, and PS stand for bytes, floating point operations, updates, and per second, respectively.

### TABLE 5

<table>
<thead>
<tr>
<th>VM image</th>
<th>OS, MPI</th>
<th>Archi</th>
<th>Benchmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC2/ami-2b65342</td>
<td>FC6</td>
<td>32bit</td>
<td>SI</td>
</tr>
<tr>
<td>EC2/ami-36f11a5f</td>
<td>FC6</td>
<td>64bit</td>
<td>SI</td>
</tr>
<tr>
<td>EC2/ami-3e56657</td>
<td>FC6, MPI</td>
<td>32bit</td>
<td>MI</td>
</tr>
<tr>
<td>EC2/ami-e13f681</td>
<td>FC6, MPI</td>
<td>64bit</td>
<td>MI</td>
</tr>
<tr>
<td>GG/server1</td>
<td>RHEL 5.1, MPI</td>
<td>32k48bit</td>
<td>S4&amp;M</td>
</tr>
<tr>
<td>EI/server1</td>
<td>Knoppix 5.3.1</td>
<td>32bit</td>
<td>SI</td>
</tr>
<tr>
<td>EH/server2</td>
<td>Ubuntu 8.10</td>
<td>64bit</td>
<td>SI</td>
</tr>
<tr>
<td>Mossa/server2</td>
<td>Ubuntu 8.10</td>
<td>32k48bit</td>
<td>SI</td>
</tr>
</tbody>
</table>

The VM images used for the HPCC benchmarks also have a working preconfigured 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.
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). Fig. 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: /C3 instances are surprisingly easy to obtain; in contrast, the m1: /C3 instances have for the resource acquisition time higher expectation (63-90 s compared to around 63 s) 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 grids [5] and unlike the promise of clouds.

**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. Fig. 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.

![Fig. 2. Resource acquisition and release overheads for acquiring single EC2 instances.](image)

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

<table>
<thead>
<tr>
<th>Instance Type</th>
<th>Res. Allocation</th>
<th>Res. Release</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Avg</td>
</tr>
<tr>
<td>m1.small</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m1.large</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c1.medium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c1.xlarge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m1.x-large</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m1.large</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c1.medium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c1.xlarge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GoGrid.large</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GoGrid.xlarge</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 Figs. 4a and 4b, respectively. The GOPS recorded for the floating point and double-precision float operations is six to eight times lower than the theoretical maximum of ECU (4.4 GOPS). A potential reason for this situation is the overrun 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 cache, by running the Bonnie suite for thirteen file sizes in the range 1,024 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 100 MiB test file and ends at 2 GiB, indicating a capacity of the memory-based disk cache of 4-5 GiB (twice 2 GiB). Thus, the results obtained for the file sizes above 5 GiB 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 5 GiB 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 Multimachine 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-Multimachine 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 Fig. 5. For the m1.small resources one node was able to achieve a performance of 1.96 GFLOPS, which is 44.54 percent 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 percent. The results for a single c1.xlarge instance are better: the achieved 49.97 GFLOPS represent 56.78 percent 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 percent. The HPL performance loss from one to 16 instances can, therefore, be expressed as 53.26 percent which results in rather bad qualification for HPC applications and their need for fast

---

**TABLE 7**

<table>
<thead>
<tr>
<th>Instance Type</th>
<th>Seq. Output</th>
<th>Seq. Input</th>
<th>Rand. Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.small</td>
<td>22.3</td>
<td>60.2</td>
<td>33.3</td>
</tr>
<tr>
<td>m1.large</td>
<td>50.9</td>
<td>64.3</td>
<td>24.4</td>
</tr>
<tr>
<td>m1.xlarge</td>
<td>57.0</td>
<td>87.8</td>
<td>33.3</td>
</tr>
<tr>
<td>c1.medium</td>
<td>49.1</td>
<td>58.7</td>
<td>32.8</td>
</tr>
<tr>
<td>c1.xlarge</td>
<td>64.8</td>
<td>87.8</td>
<td>30.0</td>
</tr>
<tr>
<td>Gg.small</td>
<td>11.4</td>
<td>10.7</td>
<td>9.2</td>
</tr>
<tr>
<td>Gg.large</td>
<td>17.0</td>
<td>17.5</td>
<td>16.0</td>
</tr>
<tr>
<td>Gg.xlarge</td>
<td>80.7</td>
<td>136.9</td>
<td>92.6</td>
</tr>
<tr>
<td>Eh.large</td>
<td>7.1</td>
<td>7.7</td>
<td>7.1</td>
</tr>
<tr>
<td>Mosso.large</td>
<td>41.0</td>
<td>102.7</td>
<td>43.8</td>
</tr>
<tr>
<td>Mosso.large</td>
<td>103.0</td>
<td>151.7</td>
<td>35.3</td>
</tr>
<tr>
<td>'02 Ext5</td>
<td>12.2</td>
<td>35.7</td>
<td>25.7</td>
</tr>
<tr>
<td>'02 RAID5</td>
<td>14.4</td>
<td>14.3</td>
<td>12.2</td>
</tr>
<tr>
<td>'07 RAID5</td>
<td>30.9</td>
<td>40.6</td>
<td>29.0</td>
</tr>
</tbody>
</table>
internode communication. We have obtained similar results with the GG.large and GG.xlarge instances, as shown in Fig. 5. For GG.large instances, the efficiency decreases quicker than for EC2 instances, down to 47.33 percent 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 Fig. 5 and Table 8 place clouds below existing environments for scientific computing, for which the achieved performance is 60-70 percent of the theoretical peak even for demanding real applications [47], [48], [49].

**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 applications similar to HPL the virtual EC2 clusters can lead to an order of magnitude lower performance.

**TABLE 8**

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

**TABLE 9**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EC2, x m1.small</td>
<td>1</td>
<td>4.40</td>
<td>1.96</td>
<td>13,312</td>
<td>2.62</td>
<td>11,60</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EC2, x m1.large</td>
<td>2</td>
<td>17.60</td>
<td>7.15</td>
<td>28,032</td>
<td>6.83</td>
<td>54.35</td>
<td>20.48</td>
<td>0.70</td>
</tr>
<tr>
<td>EC2, x m1.xlarge</td>
<td>4</td>
<td>35.20</td>
<td>11.38</td>
<td>39,552</td>
<td>8.52</td>
<td>168.64</td>
<td>17.87</td>
<td>0.92</td>
</tr>
<tr>
<td>EC2, x c1.medium</td>
<td>2</td>
<td>22.00</td>
<td>-</td>
<td>13,312</td>
<td>11.85</td>
<td>46.73</td>
<td>13.92</td>
<td>2.07</td>
</tr>
<tr>
<td>EC2, x c1.xlarge</td>
<td>8</td>
<td>88.00</td>
<td>51.58</td>
<td>27,392</td>
<td>44.05</td>
<td>249.66</td>
<td>14.19</td>
<td>1.49</td>
</tr>
<tr>
<td>EC2, x m1.large</td>
<td>16</td>
<td>176.00</td>
<td>84.53</td>
<td>38,656</td>
<td>34.99</td>
<td>223.34</td>
<td>19.31</td>
<td>1.10</td>
</tr>
<tr>
<td>EC2, x c1.xlarge</td>
<td>32</td>
<td>352.00</td>
<td>138.08</td>
<td>54,784</td>
<td>27.74</td>
<td>280.38</td>
<td>25.38</td>
<td>1.10</td>
</tr>
<tr>
<td>EC2, x c1.xlarge</td>
<td>64</td>
<td>704.00</td>
<td>252.34</td>
<td>77,440</td>
<td>3.58</td>
<td>250.40</td>
<td>35.93</td>
<td>0.97</td>
</tr>
<tr>
<td>EC2, x m1.large</td>
<td>128</td>
<td>1,408.00</td>
<td>425.82</td>
<td>109,568</td>
<td>0.23</td>
<td>207.06</td>
<td>45.20</td>
<td>0.75</td>
</tr>
<tr>
<td>EC2, x c1.xlarge</td>
<td>16</td>
<td>70.40</td>
<td>27.80</td>
<td>53,376</td>
<td>4.36</td>
<td>77.83</td>
<td>68.24</td>
<td>0.10</td>
</tr>
<tr>
<td>GoGrid, x GG.large</td>
<td>1</td>
<td>12.00</td>
<td>8.05</td>
<td>10,260</td>
<td>10.01</td>
<td>2.88</td>
<td>17.91</td>
<td>-</td>
</tr>
<tr>
<td>GoGrid, x GG.large</td>
<td>4</td>
<td>48.00</td>
<td>24.97</td>
<td>20,608</td>
<td>10.34</td>
<td>278.80</td>
<td>110.11</td>
<td>-</td>
</tr>
<tr>
<td>GoGrid, x GG.large</td>
<td>8</td>
<td>96.00</td>
<td>45.436</td>
<td>29,184</td>
<td>10.65</td>
<td>351.68</td>
<td>131.13</td>
<td>0.07</td>
</tr>
<tr>
<td>GoGrid, x GG.large</td>
<td>3</td>
<td>36.00</td>
<td>28.144</td>
<td>20,608</td>
<td>10.62</td>
<td>293.30</td>
<td>16.96</td>
<td>0.97</td>
</tr>
<tr>
<td>GoGrid, x GG.large</td>
<td>12</td>
<td>144.00</td>
<td>40.03</td>
<td>41,344</td>
<td>11.31</td>
<td>307.64</td>
<td>62.20</td>
<td>0.24</td>
</tr>
<tr>
<td>GoGrid, x GG.large</td>
<td>24</td>
<td>288.00</td>
<td>48.686</td>
<td>58,496</td>
<td>18.00</td>
<td>524.33</td>
<td>55.54</td>
<td>1.33</td>
</tr>
<tr>
<td>HPCCC-227, TopSpin/Cisco</td>
<td>16</td>
<td>102.40</td>
<td>55.23</td>
<td>81,920</td>
<td>4.88</td>
<td>2.95</td>
<td>10.25</td>
<td>0.66</td>
</tr>
<tr>
<td>HPCCC-224, TopSpin/Cisco</td>
<td>128</td>
<td>819.20</td>
<td>442.04</td>
<td>231,680</td>
<td>4.88</td>
<td>2.95</td>
<td>10.25</td>
<td>8.25</td>
</tr>
<tr>
<td>HPCCC-286, Intel Endevor</td>
<td>16</td>
<td>179.20</td>
<td>153.25</td>
<td>60,000</td>
<td>10.50</td>
<td>5.18</td>
<td>87.61</td>
<td>1.23</td>
</tr>
<tr>
<td>HPCCC-289, Intel Endevor</td>
<td>128</td>
<td>1,435.60</td>
<td>1,220.61</td>
<td>170,000</td>
<td>10.56</td>
<td>448.31</td>
<td>2.78</td>
<td>3.47</td>
</tr>
</tbody>
</table>

HPCCC-<x> is the system with the HPCC ID <x> [38]. The machines HPCCC-224 and HPCCC-227, and HPCCC-286 and HPCCC-289 are of brand TopSpin/ Cisco and by Intel Endevor, respectively. Smaller values are better for the Latency column and worse for the other columns.
for large system sizes (1,024 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].

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. Fig. 6 summarizes the results for one example benchmark from the CacheBench suite, Rd-Mod-Wr. The GG.xlarge 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 versus 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 versus an exclusively—used environment, with a bound that eliminates the bias toward 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 s. 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

![Performance stability of cloud instance types with the CacheBench benchmark with Rd-Mod-Wr operations.](image-url)
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 clouds 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. 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 percent faster average job response time) and at worst to 1 percent 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

\(^1\) 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.
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) slowdown 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 overemphasize 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 $B/C21 > 1,000$ and $J/C21 > 10,000$; $T$ 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 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 $B/C21 > 1,000$ and $J/C21 > 10,000$.

Fig. 7 shows the average response time and cost of clouds with real performance with various slowdown factors for sequential (Fig. 7a) and parallel (Fig. 7b) 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
TABLE 12
Relative Strategy Performance: Resource Bulk Allocation (S2) versus Resource Acquisition and Release per Job (S1)

<table>
<thead>
<tr>
<th>Relative Cost</th>
<th>DAS-2</th>
<th>Grid3</th>
<th>LCG</th>
<th>LANL O2K</th>
</tr>
</thead>
<tbody>
<tr>
<td>L32 - 311 x 100 [%]</td>
<td>30.2</td>
<td>11.5</td>
<td>9.3</td>
<td>9.1</td>
</tr>
</tbody>
</table>

Only performance differences above 5 percent are shown.

performance of clouds for sequential jobs should be the first priority of cloud providers.

5.3.4 Performance and Security versus 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 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 percent (the DAS-2 trace); in three cases, around 10 percent 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 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 percent for computation [24], [51] and below 15 percent 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 percent [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 versus 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 nontraditional 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 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 service 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.

**Acknowledgments**

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

**References**

Alexandru Iosup received the PhD degree 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 cofounder 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 more than 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, workflow characterization). He is a member of the IEEE and the IEEE Computer Society.

Simon Ostermann received the 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. He is the author of more than 10 journal and conference publications. His research interests are in the areas of resource management and scheduling in the area of grid and cloud computing.

M. Nezih Yigitbasi received the 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. He is a member of the IEEE.

Radu Prodan received the PhD degree from the Vienna University of Technology, Vienna, Austria, in 2004. He was involved in 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 26185 (SHIWA) projects. He is the author of more than 70 journal and conference publications and one book. He was the recipient of an IEEE Best Paper Award. He is a member of the IEEE.

Thomas Fahringer received the PhD degree in 1993 from the Vienna University of Technology. Between 1990 and 1996, 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. He coordinated the IST-034601 education project and was workpackage Leader in the IST-034601 (edutain@grid) and other Austrian and international European projects. He is the author of more than 100 papers, including three books. He was the recipient of two best paper awards from the ACM and the IEEE. His main research interests include software architectures, programming paradigms, compiler technology, performance analysis, and prediction for parallel and distributed systems. He is a member of 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. He has coauthored more than 70 papers in peer-reviewed conferences and journals, and was a general cochair of Euro-Par 2009 and IEEE P2P 2010. His research interests are in the areas of performance analysis, distributed systems, peer-to-peer systems, and grids. He is a member of the IEEE.