



Delft University of Technology
Parallel and Distributed Systems Report Series

**On the Performance Variability of
Production Cloud Services**

Alexandru Iosup, Nezhir Yigitbasi, and Dick Epema
A.Iosup@tudelft.nl, M.N.Yigitbasi@tudelft.nl, D.H.J.Epema@tudelft.nl

Completed January 2010.
To be submitted after revision

report number PDS-2010-002



ISSN 1387-2109

Published and produced by:
Parallel and Distributed Systems Section
Faculty of Information Technology and Systems Department of Technical Mathematics and Informatics
Delft University of Technology
Zuidplantsoen 4
2628 BZ Delft
The Netherlands

Information about Parallel and Distributed Systems Report Series:
reports@pds.twi.tudelft.nl

Information about Parallel and Distributed Systems Section:
<http://pds.twi.tudelft.nl/>

© 2010 Parallel and Distributed Systems Section, Faculty of Information Technology and Systems, Department of Technical Mathematics and Informatics, Delft University of Technology. All rights reserved. No part of this series may be reproduced in any form or by any means without prior written permission of the publisher.





Abstract

Cloud computing is an emerging commercial infrastructure paradigm that promises to eliminate the need for companies to maintain expensive computing hardware. Through the use of virtualization and resource time-sharing, clouds address 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 both the industry and the scientific community to self-owned clusters, grids, and parallel production environments. For this potential to become reality, the first generation of commercial computing and data storage clouds need to prove to be dependable. In this work we analyze the dependability of cloud computing services. Towards this end, we analyze long-term performance traces from two Amazon Web Services and Google App Engine, currently two of the largest commercial clouds in production. We find that the performance of about half of the cloud services investigated in this work exhibits yearly and daily patterns, but also that most services have periods of especially stable performance. Last, through trace-based simulation we assess the impact of the variability observed for the studied cloud services on three large-scale applications, job execution in scientific computing, virtual goods trading in social networks, and state management in social gaming. We show that the impact of performance variability depends on the application, and give evidence that performance variability can be an important factor in cloud provider selection.



Contents

1	Introduction	4
2	Production Cloud Services	4
2.1	Amazon Web Services	5
2.2	Google App Engine	5
3	The Analysis of the AWS Dataset	5
3.1	Summary Statistics	5
3.2	Amazon Elastic Compute Cloud (EC2)	6
3.3	Amazon Simple Storage Service (S3)	6
3.4	Amazon Simple DB (SDB)	7
3.5	Amazon Simple Queue Service (SQS)	8
3.6	Amazon Flexible Payment Service (FPS)	9
3.7	Summary of the AWS Dataset	9
4	The Analysis of the Google App Engine Dataset	10
4.1	Summary Statistics	10
4.2	The Google Run Service	10
4.3	The Google Datastore Service	11
4.4	The Google Memcache Service	12
4.5	The Google URL Fetch Service	12
4.6	Summary of the Google App Engine Dataset	12
5	The Impact of Variability on Large-Scale Applications	13
5.1	Experimental Setup	14
5.2	Grid and PPE Job Execution	15
5.3	Selling Virtual Goods in Social Networks	15
5.4	Game Status Maintenance for Social Games	16
6	Related work	17
7	Conclusion and Future Work	18



List of Figures

1	Amazon EC2: The weekly statistical properties of the resource acquisition operation.	6
2	Amazon S3: The hourly statistical properties of GET EU HI operations (top), and the monthly statistical properties of the GET EU HI operations (middle) and of GET US HI operations (bottom).	7
3	Amazon SDB: The monthly statistical properties of the update operation.	8
4	Amazon SQS: The weekly statistical properties. The statistics for the weeks 30–53 (not shown) are very similar to those for weeks 26–29.	8
5	Amazon FPS: The monthly statistical properties.	9
6	Google Run: The monthly statistical properties of running an application in the Python Runtime Environment.	11
7	Google Datastore: The monthly statistical properties of the read operation.	11
8	Google Memcache: The monthly statistical properties of the PUT operation.	12
9	Google URL Fetch: The hourly statistical properties; target web site is the Hi5 social network.	13
10	Impact of performance variability for the Selling Virtual Goods scenario	16
11	Impact of performance variability for the Social Game Status Maintenance scenario	17

List of Tables

1	Summary statistics for Amazon Web Services’s cloud services.	6
2	Presence of time patterns or special periods for the AWS services. A cell value of Y indicates the presence of a pattern or a special period.	9
3	Summary statistics for Google App Engine’s cloud services.	10
4	Presence of time patterns or special periods for the GAE services. A cell value of Y indicates the presence of a pattern or a special period.	13
5	Large-scale applications used to analyze the impact of variability.	13
6	Job Execution (GAE Run Service): The characteristics of the input workload traces.	14
7	Job Execution (GAE Run Service): Head-to-head performance of workload execution in clouds delivering steady and variable performance. The ”Cost” column presents the total cost of the workload execution, expressed in millions of CPU-hours.	15

1 Introduction

Cloud computing is emerging as a commercial alternative to traditional computing and software services such as grid computing and online payment. In the cloud computing approach resources and software are no longer hosted and operated by the user, but instead leased from large-scale data centers and service specialists strictly when needed. An important hurdle to cloud adoption is trusting that the cloud services are dependable, for example that their performance is stable over long periods of time. However, cloud providers do not disclose their infrastructure characteristics or how they change, and operate their physical resources in time-sharing; this situation may cause significant performance variability. To find out if the variability in the performance of production cloud services is significant and important, in this work we present the first long-term study on the variability of performance as exhibited by ten production cloud services of two popular cloud service providers, Amazon and Google.

Ideally, clouds should provide services such as running a user-given computation with performance equivalent to that of dedicated environments with similar characteristics. However, the performance characteristics of a cloud may vary over time as a result of structural changes that are not discussed with the users. Moreover, unlike current data centers and grids, clouds time-share their resources, and time-shared platforms have been shown [19, 3, 2] since the 1990s to cause complex performance variability and even performance degradation.

Although it would be beneficial to both researchers and system designers, there currently exists no investigation of the variability of performance for cloud services. Understanding performance variability guides in many ways research and system design. For example, it can help in selecting the service provider, designing and tuning schedulers [14], and detecting and predicting failures [31, 27]. Tens of clouds [12, 15] have started to offer services in the past few years; of these, Amazon Web Services (AWS) and Google App Engine (GAE) are two popular providers of cloud services [1, 24]. A number of studies [10, 23, 29, 5, 21, 23, 22, 1], including previous work from the authors of this work [22], investigate the performance of AWS, but none investigate the performance variability or even system availability for a period of over two months.

Our goal is to perform a comprehensive investigation of the long-term variability of performance for production cloud services. Towards this end, our main contribution is threefold:

1. We collect performance traces corresponding to ten production cloud services provided by Amazon Web Services and Google App Engine, currently two of the largest commercial clouds (Sections ??);
2. We analyze the collected traces, revealing for each service both summary statistics and the presence or absence of performance time patterns (Section 3 and 4);
3. We evaluate through trace-based simulation the impact of the variability observed in the studied traces on three large-scale applications that are executed today or may be executed in the cloud in the (near) future: executing scientific computing workloads on cloud resources, selling virtual goods through cloud-based payment services, and updating the virtual world status of social games through cloud-based database services.

2 Production Cloud Services

Cloud computing comprises both the offering of infrastructure and software services [1, 12, 24]. A cloud offering infrastructure services such as computing cycles, storage space, and resource management middleware (e.g., load balancers, queueing services) acts as Infrastructure as a Service (IaaS). A cloud offering platform services such as a runtime environment for compiled or interpreted application code (Java, Python, etc.) operating on top of virtualized resources acts as Platform as a Service (PaaS). A third category of clouds, Software as a Service (SaaS), incorporate the old idea of providing entire applications to users, over the Internet.

To accommodate this broad definition of clouds, in our model each cloud provides a set of *services*, and each service a set of *operations*. In our terminology, a *production cloud* is a cloud that operates on the market, that

is, it has real customers that use its services. Tens of cloud providers have entered the market in the last five last years, including Amazon Web Services (since 2002, effectively in production since 2006), ENKI (2003), Joyent (2004), Mosso (2006), RightScale (2008), GoGrid (2008), and Google App Engine (2008); more are scheduled to enter the market from 2010 onwards, including Microsoft Azure. From the clouds already in production, Amazon Web Services and Google App Engine are reported to have the largest number of clients [15]. We describe in turn the Amazon Web Services and Google App Engine clouds in the remainder of this section.

2.1 Amazon Web Services

Amazon Web Services (AWS) is an IaaS cloud comprising services such as the Elastic Compute Cloud (EC2, performing computing resource provisioning or web hosting operations), Elastic Block Storage and its frontend Simple Storage Service (S3, storage), Simple Queue Service (SQS, message queuing and synchronization), Simple DB (SDB, database), and the Flexible Payments Service (FPS, micro-payments). As examples of operation, the Elastic Compute Cloud (EC2) provides three main operations, for resource acquisition, resource release, and resource status query.

Through its services EC2 and S3, AWS can rent infrastructure resources located in 6 locations; the EC2 offering comprises 7 types of virtual resources (*instance types*) and the S3 offering comprises 2 types of resources. Estimates based on the numerical properties of identifiers given to provided services indicate that Amazon EC2 rents over 40,000 virtual computing boxes per day [25, 26], which is two orders of magnitude more than its competitors GoGrid and RightScale [26], and around the size of the largest scientific grid in production.

2.2 Google App Engine

The Google App Engine (GAE) is an PaaS cloud comprising services such as Java and Python Runtime Environments (Run, providing application execution operations), the Datastore (database), Memcache (caching), and URL Fetch (web crawling). Although through its Java and Python Runtime Environments users also consume computing and storage resources from the infrastructure underlying GAE, GAE does not provide root access to these resources, like the AWS.

Some of the services provided by GAE overlap with the services provided by AWS. For example, Amazon's Simple DB (SDB) and Google's Datastore both provide non-relational data stores (tuplespaces) under an Entity-Attribute-Value model.

3 The Analysis of the AWS Dataset

In this section, we present the analysis of the AWS Dataset which includes performance data for Amazon's widely used infrastructure and payment services. Each service comprises several operations. For each operation, we investigate the performance indicators in detail to understand the performance delivered by these operations.

3.1 Summary Statistics

In this section we follow the second step of our analysis method and analyze the summary statistics for Amazon Web Services; Table 1 summarizes the results. Although the Amazon EC2 deployment latency has low IQR, it has a high range. We observe higher range and IQR for the performance of S3 measured from small EC2 instances (see Section 3.3) compared to performance measured from large and extra large EC2 instances. Similar to EC2, Amazon SDB also has low IQR but a high range especially for the update operations. Finally, Amazon FPS latency is highly variable which has implications for the applications using this service for payment operations as we present in Section 5.3.

Table 1: Summary statistics for Amazon Web Services’s cloud services.

Service	Min	Q1	Median	Q3	Max	IQR	Mean	Std.Dev.
EC2								
Deployment Latency [s]	57.00	73.59	75.70	78.50	122.10	4.90	76.62	5.17
S3								
GET EU HIGH [KBps]	453.80	656.70	684.70	709.30	789.20	52.59	680.94	30.90
GET US HIGH [MBps]	8.60	15.50	17.10	18.50	25.90	3.00	16.93	2.39
PUT EU HIGH [MBps]	1.00	1.30	1.40	1.40	1.50	0.09	1.38	0.10
PUT US HIGH [MBps]	4.09	8.10	8.40	8.60	9.10	0.50	8.26	0.55
SDB								
Query Response Time [ms]	28.14	31.76	32.81	33.77	85.40	2.00	32.94	2.39
Update Latency [ms]	297.54	342.52	361.97	376.95	538.37	34.42	359.81	26.71
SQS								
Lag Time [s]	1.35	1.47	1.50	1.79	6.62	0.32	1.81	0.82
FPS								
Latency [ms]	0.00	48.97	53.88	76.06	386.43	27.09	63.04	23.22

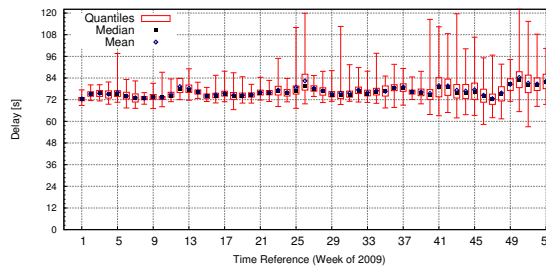


Figure 1: Amazon EC2: The weekly statistical properties of the resource acquisition operation.

3.2 Amazon Elastic Compute Cloud (EC2)

CloudStatus.com reports the following performance indicator for the EC2 service:

1. **Deployment Latency** - The time it takes to start an m1.small instance, from the time startup is initiated to the time that the instance is available.

Figure 1 shows weekly statistical properties of the Amazon EC2 service Resource Acquisition operation. We observe higher IQR and range for deployment latency from week 41 till the end of the year compared to the remainder of the year probably due to increased user base of the EC2 service. Steady performance for EC2 deployment latency is especially important for applications which uses the auto-scaling functionality of EC2.

3.3 Amazon Simple Storage Service (S3)

CloudStatus.com reports the throughput of Amazon S3 where the throughput is measured by issuing S3 requests from US-based EC2 instances to S3 buckets in the US and Europe. "High I/O" metrics reflect throughput for operations on Large and Extra Large EC2 instances.

The following performance indicators are reported:

1. **Get Throughput (bytes/second)** - Estimated rate at which an object in a bucket is read (GET).
2. **Put Throughput Per Second (bytes/second)** - Estimated rate at which an object in a bucket is written (PUT).

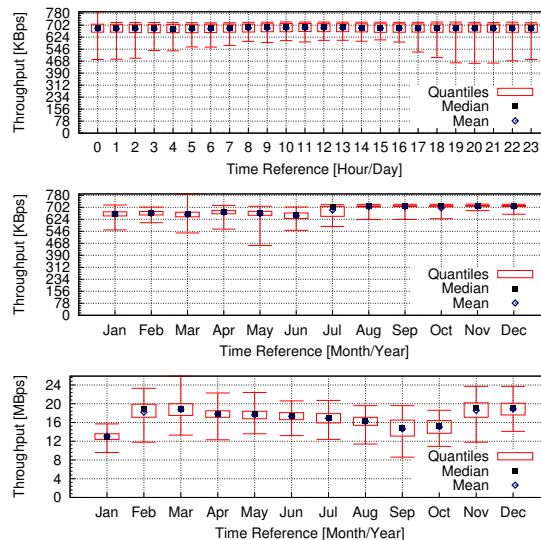


Figure 2: Amazon S3: The hourly statistical properties of GET EU HI operations (top), and the monthly statistical properties of the GET EU HI operations (middle) and of GET US HI operations (bottom).

Figure 2 (top) depicts the hourly statistical properties of the Amazon S3 service GET EU HI operation. The range has a pronounced daily pattern, with evening and night hours (from 7PM to 2AM the next day) exhibiting much lower minimal transfer rates, and the work day hours (from 8AM to 3PM) exhibiting much higher minimal transfer rates.

Figure 2 (middle) shows the monthly statistical properties of the Amazon S3 service GET EU HI operation. The operation’s performance changes its pattern in August 2009: the last five months of the year exhibit much lower IQR and range, and have significantly better performance – the median throughput increases from 660 KBps to 710 KBps.

Figure 2 (bottom) shows the monthly statistical properties of the Amazon S3 service GET US HI operation. The operation exhibits pronounced yearly patterns, with the months January, September, and October 2009 having the lowest mean (and median) performance. Figure 2 (bottom) also shows that there exists a wide range of median monthly performance values, from 13 to 19 MBps over the year.

The results suggest that when designing an application that uses Amazon S3 as a key-based object store, application architects should consider several design decisions including which data center to store the buckets (US/EU), and whether the performance of S3 will be satisfactory for read/write intensive workloads.

3.4 Amazon Simple DB (SDB)

CloudStatus.com reports the following performance indicators for the SDB service:

1. **Query Response Time (ms)** - The time it takes to execute a GetAttributes operation that returns 100 attributes.
2. **Update Latency (ms)** - The time it takes for the updates resulting from a PutAttributes operation to be available to a subsequent GetAttributes operation.

Figure 3 shows the monthly statistical properties of the Amazon SDB Update operation. The monthly median performance has a wide range, from 315 to 383 ms. There is a sudden jump in range in June 2009; the

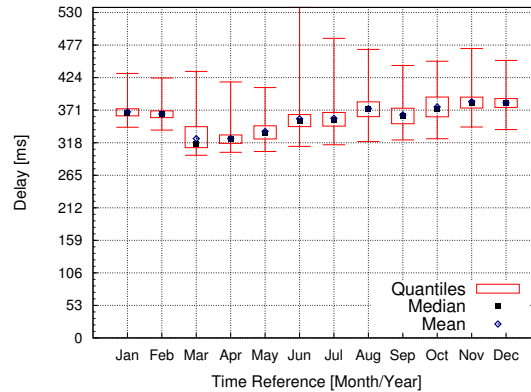


Figure 3: Amazon SDB: The monthly statistical properties of the update operation.

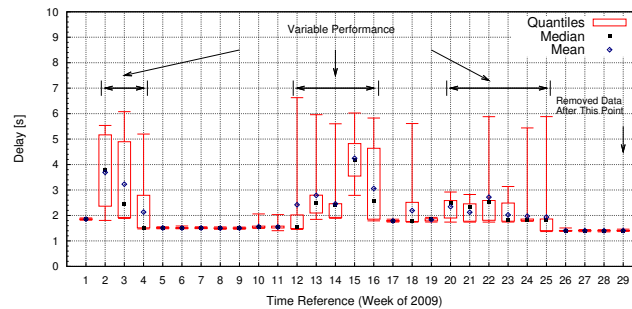


Figure 4: Amazon SQS: The weekly statistical properties. The statistics for the weeks 30–53 (not shown) are very similar to those for weeks 26–29.

range decreases steadily from June to December to the nominal values observed in the first part of the year. This is significant for applications such as online gaming, in which values above the 99% performance percentile are important, as unhappy users may trigger massive customer departure through their social links (friends and friends-of-friends).

3.5 Amazon Simple Queue Service (SQS)

CloudStatus.com reports the following performance indicators for the Amazon SQS service:

1. **Average Lag Time (s)** - The time it takes for a posted message to become available to be read. Lag time is monitored for multiple queues that serve requests from inside the cloud. The average is taken over the lag times measured for each monitored queue.

Figure 4 depicts the weekly statistical properties of Amazon SQS service. The service exhibits long periods of stability (low IQR and range, similar median performance week after week), for example weeks 5–9 and 26–53, but also periods of high performance variability, especially in weeks 2–4, 13–16, and 20–23. The periods with high performance variability are not always preceded by weeks of moderate variability. The duration of a period with high performance variability can be as short as a single week, for example during week 18. Application operators and architects should take this performance variability into account when designing and deploying

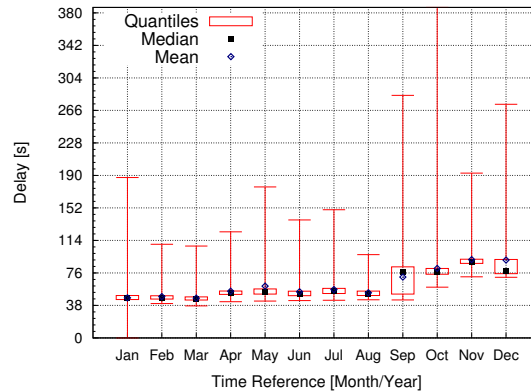


Figure 5: Amazon FPS: The monthly statistical properties.

Table 2: Presence of time patterns or special periods for the AWS services. A cell value of Y indicates the presence of a pattern or a special period.

Perf. Indicator	Yearly (Month)	Monthly (Day)	Weekly (Day)	Daily (Hour)	Special Period
<i>Amazon AWS</i>					
EC2					Y
S3	Y		Y	Y	Y
SDB	Y			Y	
SQS					Y
FPS					Y

applications that use the asynchronous messaging capabilities of the SQS service especially if the application should deliver steady performance.

3.6 Amazon Flexible Payment Service (FPS)

CloudStatus.com reports the following performance indicators for the Amazon FPS service:

1. **Response Time (s)** - The time it takes to execute a payment transaction. The response time does not include the round trip time to the FPS service nor the time taken to setup Pay tokens. Since Amazon reports the response time to the nearest second, payments that complete in less than a second will be recorded as zero.

Figure 5 depicts the monthly statistical properties of the Amazon FPS service. There is a sudden jump in the monthly median performance in September 2009, from about 50 to about 80 ms; whereas the median is relatively constant before and after the jump. We also observe high variability in the maximum performance values of the FPS service across months.

3.7 Summary of the AWS Dataset

The performance results indicate that all Amazon services that we have analyzed in this section exhibit one or more time patterns and/or periods of time where the service provides special behavior, as summarized in Table 2. Amazon EC2 exhibits periods of special behavior for the resource acquisition operation (Section 3.2). Both storage services of Amazon, namely the SDB and S3 services present daily, yearly, and monthly patterns

Table 3: Summary statistics for Google App Engine’s cloud services.

Service	Min	Q1	Median	Q3	Max	IQR	Mean	Std.Dev.
App Engine Python Runtime [ms]	1.00	284.14	302.31	340.37	999.65	56.22	314.95	76.39
Datastore								
Create [s]	1.04	1.28	1.42	1.71	5.59	0.42	1.60	0.60
Delete [ms]	1.00	344.40	384.22	460.73	999.86	116.32	413.24	102.90
Read [ms]	1.00	248.55	305.68	383.76	999.27	135.20	336.82	118.20
Memcache								
Get [ms]	45.97	50.49	58.73	65.74	251.13	15.24	60.03	11.44
Put [ms]	33.21	44.21	50.86	60.44	141.25	16.23	54.84	13.54
Response [ms]	3.04	4.69	5.46	7.04	38.71	2.35	6.64	3.39
URL Fetch								
s3.amazonaws.com [ms]	1.01	198.60	226.13	245.83	983.31	47.22	214.21	64.10
ebay.com [ms]	1.00	388.00	426.74	460.03	999.83	72.03	412.57	108.31
api.facebook.com [ms]	1.00	172.95	189.39	208.23	998.22	35.28	195.76	44.40
api.hi5.com [ms]	71.31	95.81	102.58	113.40	478.75	17.59	107.03	25.12
api.myspace.com [ms]	67.33	90.85	93.36	103.85	515.88	13.00	97.90	14.19
paypal.com [ms]	1.00	406.57	415.97	431.69	998.39	25.11	421.76	35.00

for different set of operations (Section 3.4 and Section 3.3). Finally, Amazon SQS and FPS services show special behavior for specific time periods (Section 3.5 and Section 3.6).

4 The Analysis of the Google App Engine Dataset

In this section, we present the analysis of the Google App Engine Dataset which includes performance data for Google App Engine services. Each service comprises several operations. For each operation, we investigate the performance indicators in detail to understand the performance delivered by these operations.

4.1 Summary Statistics

In this section we follow the second step of our analysis method and analyze the summary statistics for Google App Engine; Table 3 summarizes the results. The Google App Engine Python runtime and datastore have high range and IQRs leading to highly variable performance. However, we observe relatively stable performance of the memcache caching service.

4.2 The Google Run Service

CloudStatus.com reports the following performance indicator for the Run service:

1. **Fibonacci (ms)** - The time it takes to calculate the 27th Fibonacci number in the Python Runtime Environment.

Figure 6 depicts the monthly statistical properties of the Google App Engine Python Runtime. The last three months of the year exhibit stable performance, with very low IQR and narrow range, and with steady month-to-month median. Similar to the Amazon SDB service (see Section 3.4), the monthly median performance has a wide range, from 257 to 388 ms. Independently of the evolution of the median, there is a sudden jump in range in March 2009; the maximum response time (lowest performance) decreases steadily up to October, from which point the performance becomes steady.

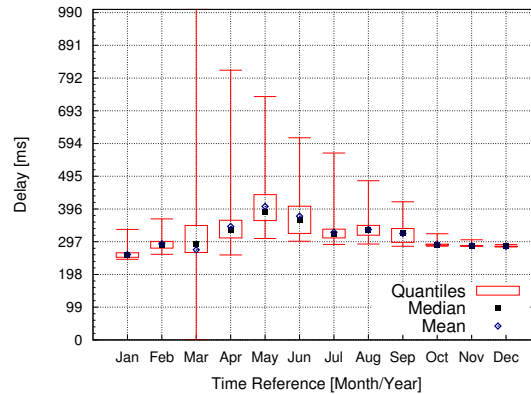


Figure 6: Google Run: The monthly statistical properties of running an application in the Python Runtime Environment.

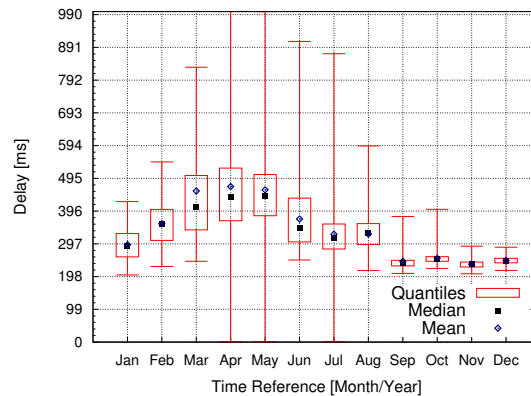


Figure 7: Google Datastore: The monthly statistical properties of the read operation.

4.3 The Google Datastore Service

To measure create/delete/read times CloudStatus uses a simple set of data which we refer to the combination of all these entities as a 'User Group'. CloudStatus.com reports the following performance indicators for the Datastore service:

1. **Create Time (s)** - The time it takes for a transaction that creates a User Group.
2. **Read Time (ms)** - The time it takes to find and read a User Group. Users are randomly selected, and the user key is used to look up the user and profile picture records. Posts are found via a GQL (Google Query Language) ancestor query.
3. **Delete Time (ms)** - The time it takes for a transaction that deletes a User Group.

Figure 7 depicts the monthly statistical properties of the Google App Engine Datastore service read performance. The last four months of the year exhibit stable performance, with very low IQR and relatively narrow range, and with steady month-to-month median. In addition we observe yearly patterns for the months January through August. Similar to Amazon S3 GET operations, the Datastore service exhibits a high IQR with yearly

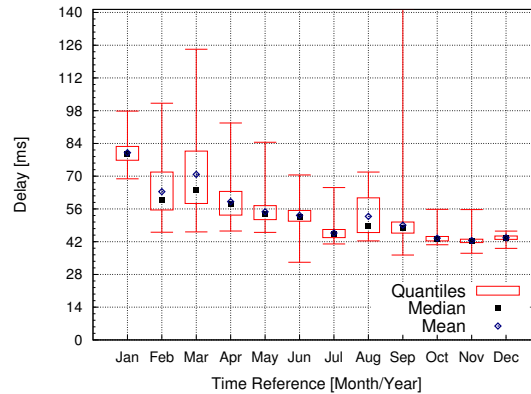


Figure 8: Google Memcache: The monthly statistical properties of the PUT operation.

patterns (Section 3.3), and in contrast to S3, the Datastore service read operations exhibit a higher range. Overall, the Update operation exhibits a wide yearly range of monthly median values, from 315 to 383 ms.

4.4 The Google Memcache Service

CloudStatus.com reports the following performance indicators for the Memcache service:

1. **Get Time (ms)** - The time it takes to get 1 MB of data from memcache.
2. **Put Time (ms)** - The time it takes to put 1 MB of data in memcache.
3. **Response Time (ms)** - The round-trip time to request and receive 1 byte of data from cache. This is analogous to Get Time, but for a smaller chunk of data.

Figure 8 depicts the monthly statistical properties of the Memcache service PUT operation performance. The last three months of the year exhibit stable performance, with very low IQR and relatively narrow range, and with steady month-to-month median. The same trend can be observed for the Memcache GET operation. Uniquely for the Memcache PUT operation, the median performance per month has an increasing trend over the first ten months of the year, with the response time decreasing from 79 to 43 ms.

4.5 The Google URL Fetch Service

CloudStatus.com reports the response time (ms) which is obtained by issuing web service requests to several web sites: api.facebook.com, api.hi5.com, api.myspace.com, ebay.com, s3.amazonaws.com, and paypal.com.

Figure 9 depicts the hourly statistical properties of the URL Fetch service when the target web site is the Hi5 social network. The ranges of values for the service response times vary greatly over the day, with several peaks. We have observed a similar pattern for every other target web site for which a URL Fetch web service request is issued.

4.6 Summary of the Google App Engine Dataset

The performance results indicate that all Google App Engine services that we have analyzed in this section exhibit one or more time patterns and/or periods of time where the service provides special behavior, as summarized in Table 4. The Python Runtime exhibits periods of special behavior and daily patterns (Section 4.2).

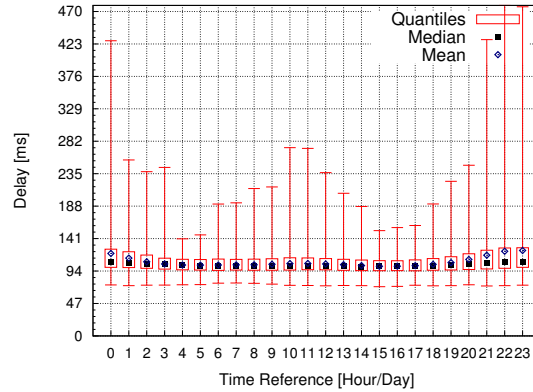


Figure 9: Google URL Fetch: The hourly statistical properties; target web site is the Hi5 social network.

Table 4: Presence of time patterns or special periods for the GAE services. A cell value of Y indicates the presence of a pattern or a special period.

Perf. Indicator	Yearly (Month)	Monthly (Day)	Weekly (Day)	Daily (Hour)	Special Period
<i>Google App Engine</i>					
Run				Y	Y
Datastore	Y				Y
Memcache					
URL Fetch			Y	Y	Y

The Datastore service presents yearly patterns and periods of time with special behavior (Section 4.3). The Memcache service performance has also monthly patterns and time patterns of special behavior for various operations (Section 4.4). Finally, the URL Fetch service presents weekly and daily patterns, and also shows special behavior for specific time periods for different target websites (Section 4.5).

5 The Impact of Variability on Large-Scale Applications

In this section we assess the impact of the variability of cloud service performance on the performance of large-scale applications. Our method for this task is to evaluate this impact using trace-based simulation. Since there currently exists no accepted traces or models of cloud workloads, we propose scenarios in which three realistic large-scale applications would leverage specific cloud services. Table 5 summarizes the three applications and the main cloud service that they leverage.

Table 5: Large-scale applications used to analyze the impact of variability.

Section	Application	Used Service
Section 5.2	Job execution	GAE Run
Section 5.3	Selling virtual goods	AWS FPS
Section 5.2	Game status management	AWS SDB GAE Datastore

Table 6: Job Execution (GAE Run Service): The characteristics of the input workload traces.

Trace ID, Source (Trace ID in Archive)	Trace Number of			System Size		Load [%]
	Mo.	Jobs	Users	Sites	CPUs	
<i>Grid Workloads Archive [13], 3 traces</i>						
1. RAL (6)	12	0.2M	208	1	0.8K	85+
2. Grid3 (8)	18	1.3M	19	29	3.5K	-
3. SharcNet (10)	13	1.1M	412	10	6.8K	-
<i>Parallel Workloads Archive [28], 2 traces</i>						
4. CTC SP2 (6)	11	0.1M	679	1	430	66
5. SDSC SP2 (9)	24	0.1M	437	1	128	83

5.1 Experimental Setup

Input Data For each application scenario, we use the real system traces described in the section corresponding to the scenario (column "Section" in Table 5), and the monthly performance variability of the main service leveraged by the "cloudified" application (column "Used Service" in Table 5).

Simulator We design for each application a simple simulator that considers from the real system trace each unit of information, that is, a job record for the Job Execution scenario and the number of daily unique users for the other two scenarios, and assesses the execution performance for a cloud with stable performance vs a cloud with variable performance. For each of the three considered applications we select one performance indicator, corresponding to the main cloud service that the "cloudified" application would use. In our simulations, the variability of this performance indicator, which, given as input data to the simulator, is the monthly performance variability analyzed earlier in this work. We define the *reference performance* P_{ref} as the average of the twelve monthly medians, and attribute this performance to the cloud with stable performance. To ensure that results are representative, we run each simulation 100 times and report the average results.

Metrics We report the following metrics:

- For the Job Execution scenario, which simulates the execution of compute-intensive jobs coming from grid and parallel production environments (PPEs), we first report two traditional job execution metrics for the grid and PPE communities: the average response time (**ART**), the average bounded slowdown (**ABSD**) with a threshold of 10 seconds [11]; the ABSD threshold of 10 eliminates the bias of the average toward jobs with runtime below 10 seconds. We also report one cloud-specific metric, **Cost**, which is the total cost for running the complete workload, expressed in millions of consumed CPU-hours.
- For the other two scenarios, which do not have traditional metrics, we devise a performance metric that aggregates two components, the relative performance and the relative number of users. We design our metric so that the lower values for the relative performance are better. We define the **Aggregate Performance Penalty** as $APR(t) = \frac{P(t)}{P_{ref}} \times \frac{U(t)}{\max U(t)}$, where $P(t)$ is the performance at time t , P_{ref} is the reference performance, and $U(t)$ is the number of users at time t ; $P(t)$ is a random value sampled from the distribution corresponding to the current month at time t . The relative number of users component is introduced because application providers are interested in bad performance only to the extent it affects their users; when there are few users of the application, this component ensures that the $APR(t)$ metric remains low for small performance degradation. Thus, the APR metric does not represent well applications for which good and stable performance is important at all times. However, for such applications the impact of variability can be computed straightforwardly from the monthly statistics of the cloud service; this is akin to excluding the user component from the APR metric.

Table 7: Job Execution (GAE Run Service): Head-to-head performance of workload execution in clouds delivering steady and variable performance. The "Cost" column presents the total cost of the workload execution, expressed in millions of CPU-hours.

Trace ID	Cloud with					
	Stable Performance			Variable Performance		
	ART [s]	ABSD (10s)	Cost	ART [s]	ABSD (10s)	Cost
RAL	18,837	1.89	6.39	18,877	1.90	6.40
Grid3	7,279	4.02	3.60	7,408	4.02	3.64
SharcNet	31,572	2.04	11.29	32,029	2.06	11.42
CTC SP2	11,355	1.45	0.29	11,390	1.47	0.30
SDSC SP2	7,473	1.75	0.15	7,537	1.75	0.15

5.2 Grid and PPE Job Execution

Scenario In this scenario we analyze the execution of compute-intensive jobs typical for grids and PPEs on cloud resources.

Input Traces We use as input workloads five long-term traces taken from real grids and PPEs; Table 6 summarizes their characteristics, with the ID of each trace indicating the system from which the trace was taken; see [13, 28] for more details about each trace.

Variability We assume that the execution performance for the cloud with steady performance is equivalent to the performance of the grid from which the trace was obtained. We assume that the GAE Run service can run the input workload, and exhibits the monthly variability evaluated in Section 4.2. Thus, we assume that the cloud with variable performance introduces for each job a random slowdown factor derived from the real performance distribution of the service for the month in which the job was submitted.

Results Table 7 summarizes the results for the job execution scenario. The performance metrics ART, ABSD, and Cost differ by less than 2% between the cloud with stable performance and the cloud with variable performance. Thus, the main finding is that the impact of service variability is low for the job execution scenario.

5.3 Selling Virtual Goods in Social Networks

Scenario In this scenario we look at selling virtual goods by a company operating a social network such as FaceBook, or by a third party associated with such a company. For example, FaceBook facilitates selling virtual goods through its own API, which in turn could make use of Amazon's FPS service for micro-payments.

Input Traces We assume that the number of payment operations depends linearly with the number of daily unique users, and use as input traces the number of daily unique users present on FaceBook (Figure 10).

Variability We assume that the cloud with variable performance exhibits the monthly variability of Amazon FPS, as evaluated in Section 3.6.

Results The main result is that our Aggregate Performance Penalty metric can be used to trigger and motivate the decision of switching cloud providers. Figure 10 shows the APR when using Amazon's FPS as the micro-payment backend of the virtual goods vendor. The significant performance decrease of the FPS service during the last four months of the year, combined with the significant increase in the number of daily FaceBook users, is well captured by the APR metric—it leads to APR values well above 1.0, to a maximum of 3.9 in November 2009. If the virtual goods clients respond to high payment latency similarly to other consumers of Internet newmedia [6, 9], that is, they become unsatisfied and quit, our APR metric is a clear indicator for the virtual goods vendor that the cloud service provider should be changed.

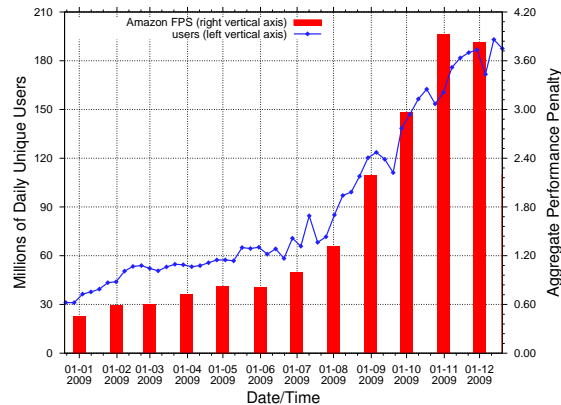


Figure 10: Selling Virtual Goods in Social Networks (Amazon FPS): Aggregate Performance Penalty when using Amazon FPS as the micro-payment backend. (Data source for the number of FaceBook users: <http://www.developeranalytics.com/>)

5.4 Game Status Maintenance for Social Games

Scenario In this scenario we investigate the maintenance of game status for a large-scale social game. Social games such as Farm Town, Mafia Wars, and Bejeweled Blitz currently have millions of unique users daily. In comparison with traditional massively multiplayer online games such as World of Warcraft and Runescape, which also gather millions of unique players daily, social games have very little player-to-player interaction (except for messaging, performed externally to the game, for example through FaceBook channels). Hence, maintaining the game status for social gaming is based on simpler database operations, without the burden of cross-updating information for concurrent players, as we have observed for Runescape in our previous work [20]. Thus, this scenario allows us to compare a pair of cloud database services, Amazon’s SDB and Google’s Datastore.

Input Traces Similarly to the previous scenario, we assume that the number of operations, database accesses in this scenario, depends linearly on the number of daily unique users. We use as input trace the number of daily unique users for the Farm Town social game (Figure 11).

Variability We assume, in turn, that the cloud with variable performance exhibits the monthly variability of Amazon SDB (Section 3.4) and of Google Datastore (Section 4.3). The input traces span the period March 2009 to January 2010; thus, we do not have a direct match between the variability data, which corresponds to only to months in 2009, and the month January 2010 in the input traces. Since the Datastore operations exhibit yearly patterns (Section 4.6), we use in simulation the variability data of January 2009 as the variability data for January 2010.

Results The main finding is that there is a big discrepancy between the two cloud services, which would allow the application operator to select the most suitable cloud provider. Figures 11 depicts the Aggregate Performance Penalty for the application using the Amazon SDB Update operation (top) and for the application using the Google Datastore Read operation (bottom). During September 2009–January 2010, the bars depicting the APR of Datastore are well below the curve representing the number of users. This corresponds to the performance improvements (lower median) of the Datastore Read performance indicator in the last part of 2009 (see also Figure 7). In contrast, the APR values for SDB Update go above the users curve. These visual clues indicate that, for this application, Datastore is superior to SDB over a long period of time. An inspection of the APR values confirms the visual clues: the APR for the last five depicted months is around 1.00 (no performance penalty) for Datastore and around 1.4 (40% more) for SDB. The application operator has solid grounds for using the Datastore services for the application studied in this scenario.

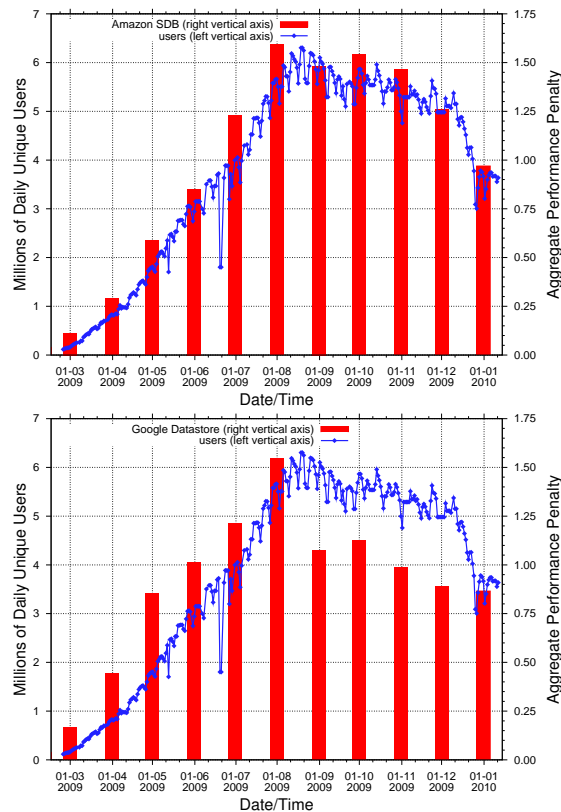


Figure 11: Game Status Maintenance for Social Games (Amazon SDB and Google App Engine Datastore): Aggregate Performance Penalty (top) when using Amazon SDB as the database backend; (bottom) when using Google App Engine Datastore as the database backend. (Data source for the number of Farm Town users: <http://www.developeranalytics.com/>)

6 Related work

In this section we review related work from the areas of clouds and virtualization. Much effort has been put recently in assessing the performance of virtualized resources, in cloud computing environments [10, 23, 29, 21, 22] and in general [4, 8, 17, 16]. In contrast to this body of previous work, ours is different in scope: we do not focus on the (average) performance values, but on their variability and evolution over time. In particular, our work is the first to characterize the long-term variability of the performance of production cloud services.

Close to our work is the seminal study of Amazon S3 [23], which also includes a 40 days evaluation of the availability of service. Our work complements this study by analyzing the performance of eight other AWS and GAE cloud services over a complete operational year; we also focus on different applications. Several small-scale performance studies of Amazon EC2 have been recently conducted: the study of Amazon EC2 performance using the NPB benchmark suite [29], the early comparative study of Eucalyptus and EC2 performance [21], the study of performance and cost of executing a scientific workflow, Montage, in clouds [10], the study of file transfer performance between Amazon EC2 and S3 [5], etc. Our results complement these previous findings, and give more insight into the (variability of) performance of EC2 and other cloud services. Finally, at the infrastructure level, automated performance management of virtual machines has been investigated by recent

research [16].

Recent performance studies using general purpose benchmarks have shown that the overhead incurred by virtualization can be below 5% for computation [4, 8] and below 15% for networking [4, 17]. Similarly, the performance loss due to virtualization for parallel I/O and web server I/O has been shown to be below 30% [30] and 10% [7, 18], respectively. Our own previous work [22] has shown that virtualized resources obtained from public clouds can have a much lower performance than the theoretical peak, especially for computation and network-intensive applications. In contrast to these studies, we investigate in this work the performance variability, and find several examples of performance indicators whose monthly median's variation is above 50% over the course of the studied year. Thus, our current study complements well the findings of our previous work, that is, the performance results obtained for small virtualized platforms are optimistic estimations of the performance observed in clouds.

7 Conclusion and Future Work

Production cloud services may incur high performance variability, due to the combined and non-trivial effects of system size, variability of workloads, virtualization overheads, and resource time-sharing. In this work we have set to identify the presence and extent of this variability, and to understand its impact on large-scale cloud applications. Toward this end, we have presented the first long-term study on the variability of performance exhibited by the services offered by two popular cloud service providers, Amazon and Google. Our study is based on the year-long traces that we have collected from CloudStatus and which comprise performance data for Amazon Web Services and Google App Engine cloud services. The two main achievements of our study are described in the following.

First, we have analyzed the overall and time-dependent characteristics exhibited by the traces, and found that the performance of the investigated cloud services exhibits on the one hand yearly and daily patterns, and on the other hand periods of particularly stable performance. We have also found that several cloud services exhibit high variation in the monthly median values, which indicates large performance changes over the course of the year. We hope that the details presented for these variations during the course of the analysis, and in particular the tables comprising summary statistics, will be put to good use in the future by the performance-aware community.

Second, we have found that the impact of the observed variability of cloud service performance varies greatly from one type of large-scale application to another. For example, we found that the service of running applications of GAE, which exhibits high performance variability and a three-months period of low variability and improved performance, has a negligible impact if the service is used to run grid and parallel production workloads. In contrast, we found that and explained the reasons for which the database service of GAE, having exhibited a similar period of better performance as the GAE running service, outperforms the AWS database service for a large social gaming application.

For the future, we will extend this work with additional analysis of other production cloud services, and in particular the services of Microsoft Azure and other clouds mentioned in the introductory part of this work.

Last but not least, we intend to make the performance data used as input for this work publicly accessible (pending acceptance from CloudStatus). We hope that this will stimulate more cloud providers to publish historical performance information, and spark interesting future work.

Acknowledgments

The authors would like to thank CloudStatus.com team for taking the financial burden and the time to make the status of the Amazon Web Services and Google App Engine available to the general public.

References

- [1] 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. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009. [4](#)
- [2] 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*, pages 267–278, 1995. [4](#)
- [3] I. Ashok and J. Zahorjan. Scheduling a mixed interactive and batch workload on a parallel, shared memory supercomputer. In *SuperComputing*, pages 616–625, 1992. [4](#)
- [4] 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*, pages 164–177. ACM, 2003. [17](#), [18](#)
- [5] M.-E. Bgin, B. Jones, J. Casey, E. Laure, F. Grey, C. Loomis, and R. Kubli. Comparative study: Grids and clouds, evolution or revolution? EGEE-II Report, CERN, June 2008. [Online] Available: <https://edms.cern.ch/file/925013/3/EGEE-Grid-Cloud.pdf>. [4](#), [17](#)
- [6] K. T. Chen, P. Huang, and C. L. Lei. Effect of network quality on player departure behavior in online games. *IEEE TPDS*, 20(5):593–606, 2009. [15](#)
- [7] L. Cherkasova and R. Gardner. Measuring CPU overhead for I/O processing in the Xen virtual machine monitor. In *USENIX ATC*, pages 387–390, 2005. [18](#)
- [8] 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*, pages 135–144, 2004. [17](#), [18](#)
- [9] M. Claypool and K. T. Claypool. Latency and player actions in online games. *CACM*, 49(11):40–45, 2006. [15](#)
- [10] 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*, page 50. IEEE/ACM, 2008. [4](#), [17](#)
- [11] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong. Theory and practice in parallel job scheduling. In *JSSPP*, volume 1291 of *LNCSS*, pages 1–34. Springer-Verlag, 1997. [14](#)
- [12] D. Hilley. Cloud computing: A taxonomy of platform and infrastructure-level offerings. Technical Report GIT-CERCSS-09-13, Georgia Institute of Technology, Dec 2008. [4](#)
- [13] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. Epema. The Grid Workloads Archive. *FGCS*, 24(7):672–686, 2008. [14](#), [15](#)
- [14] 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*, pages 97–108. ACM, 2008. [4](#)
- [15] J. Maguire, J. Vance, and C. Harvey. 85 cloud computing vendors shaping the emerging cloud, Aug 2009. ITManagement Tech.Rep. [4](#), [5](#)
- [16] J. Matthews, T. Garfinkel, C. Hoff, and J. Wheeler. Virtual machine contracts for datacenter and cloud computing environments. In *Workshop on Automated control for datacenters and clouds (ACDC)*, pages 25–30. ACM, 2009. [17](#), [18](#)
- [17] A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel. Diagnosing performance overheads in the Xen virtual machine environment. In *VEE*, pages 13–23. ACM, 2005. [17](#), [18](#)
- [18] U. F. Minhas, J. Yadav, A. Aboulnaga, and K. Salem. Database systems on virtual machines: How much do you lose? In *ICDE Workshops*, pages 35–41. IEEE, 2008. [18](#)
- [19] M. W. Mutka and M. Livny. The available capacity of a privately owned workstation environment. *Perform. Eval.*, 12(4):269–284, 1991. [4](#)
- [20] V. Nae, A. Iosup, S. Podlipnig, R. Prodan, D. H. J. Epema, and T. Fahringer. Efficient management of data center resources for massively multiplayer online games. In *SC*, page 10, 2008. [16](#)
- [21] D. Nurmi, R. Wolski, C. Grzegorzczak, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus open-source cloud-computing system. In *IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 124–131, 2009. [4](#), [17](#)
- [22] S. Ostermann, A. Iosup, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. An early performance analysis of cloud computing services for scientific computing. In *CloudComp 2009*, pages 1–10. [4](#), [17](#), [18](#)
- [23] 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*, pages 55–64. ACM, 2008. [4](#), [17](#)
- [24] R. Prodan and S. Ostermann. A survey and taxonomy of infrastructure as a service and web hosting cloud providers. In *GRID*, pages 17–25, 2009. [4](#)



- [25] RightScale. Amazon usage estimates, Aug 2009. [Online] Available: blog.rightscale.com/2009/10/05/amazon-usage-estimates. 5
- [26] G. Rosen. Cloud usage analysis series, Aug 2009. [Online] Available: www.jackofallclouds.com/category/analysis. 5
- [27] O. O. Sonmez, N. Yigitbasi, A. Iosup, and D. H. J. Epema. Trace-based evaluation of job runtime and queue wait time predictions in grids. In *HPDC*. ACM, 2009. 4
- [28] The Parallel Workloads Archive Team. The parallel workloads archive logs, Jan. 2010. [Online] Available: www.cs.huji.ac.il/labs/parallel/workload/logs.html. 14, 15
- [29] E. Walker. Benchmarking Amazon EC2 for HP Scientific Computing. *Login*, 33(5):18–23, Nov 2008. 4, 17
- [30] W. Yu and J. S. Vetter. Xen-based HPC: A parallel I/O perspective. In *CCGrid*, pages 154–161. IEEE, 2008. 18
- [31] Y. Zhang, M. S. Squillante, A. Sivasubramaniam, and R. K. Sahoo. Performance implications of failures in large-scale cluster scheduling. In *JSSPP*, pages 233–252, 2004. 4