On the performance variability of production cloud services

Citation for published version (APA):

DOI:
10.1109/CCGrid.2011.22

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.
On the Performance Variability of Production Cloud Services

Alexandru Iosup  
Parallel and Distributed Systems Group  
Delft University of Technology, Mekelweg 4, 2628 CD Delft  
A.Iosup@tudelft.nl

Nezih Yigitbasi  
Parallel and Distributed Systems Group  
Delft University of Technology, Mekelweg 4, 2628 CD Delft  
M.N.Yigitbasi@tudelft.nl

Dick Epema  
Parallel and Distributed Systems Group  
Delft University of Technology, Mekelweg 4, 2628 CD Delft  
D.H.J.Epema@tudelft.nl

Abstract—Cloud computing is an emerging 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 diverse needs. Thus, clouds have the potential to provide 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 clouds need to be proven to be dependable. In this work we analyze the dependability of cloud services. Towards this end, we analyze long-term performance traces from 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 we investigate 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.

I. INTRODUCTION

Cloud computing is emerging as an alternative to traditional computing and software services such as grid computing and online payment. With cloud computing 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 time periods. However, 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 performance variability of cloud services is significant, 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 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 [1] 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 performance variability for cloud services. Understanding this variability guides in many ways research and system design. For example, it can help in selecting the service provider, designing and tuning schedulers [2], and detecting and predicting failures [3]. Tens of clouds [4], [5] started to offer services in the past few years; of these, Amazon Web Services (AWS) and Google App Engine (GAE) are two popular production clouds [6]. A number of studies [6]–[8], [8]–[11], including our previous work [11], 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 III);
2) We analyze the collected traces, revealing for each service both summary statistics and the presence or absence of performance time patterns (Section IV and V);
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.

II. PRODUCTION CLOUD SERVICES

Cloud computing comprises both the offering of infrastructure and software services [4], [6]. A cloud offering infrastructure services such as computing cycles, storage space or queueing services acts as Infrastructure as a Service (IaaS).
A cloud offering platform services such as a runtime environment for compiled/interpreted application code operating on 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 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 (2006), ENKI (2003), Joyent (2004), Mosso (2006), RightScale (2008), GoGrid (2008), Google App Engine (2008) and recently Microsoft Azure(2010). From the clouds already in production, Amazon Web Services and Google App Engine are reported to have the largest number of clients [5] which we describe in turn.

A. 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 operation examples, the 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; the EC2 offering comprises more than 10 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 resources per day [12], [13], which is two orders of magnitude more than its competitors GoGrid and RightScale [13], and around the size of the largest scientific grid in production.

B. Google App Engine

The Google App Engine (GAE) is a 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 the Run service users consume computing and storage resources from the underlying GAE infrastructure, GAE does not provide root access to these resources, like the AWS.

III. METHOD

To characterize the long-term performance variability of cloud services we first build meaningful datasets from performance traces taken from production clouds, and then we analyze these datasets and characterize the performance variability.

Our method is built around the notion of performance indicators. We call a performance indicator the stochastic variable that describes the performance delivered by one operation or by a typical sequence of operations over time. For example, the performance indicators for Amazon include the response time of the resource acquisition operation of the EC2 service.

A. Performance Traces of Cloud Services

Data Source To characterize AWS and GAE we first acquire data from the performance database created by Hyperic’s CloudStatus team [14]. CloudStatus provides real-time values and weekly averages of about thirty performance indicators for AWS and GAE. In particular, it provides performance indicators for five main services provided by AWS (EC2, S3, SDB, SQS, and FPS) and for four main services provided by GAE (Run, Datastore, Memcache, and URL Fetch). Cloud-Status obtains values for the various performance indicators by running performance probes periodically, with a sampling rate of under 2 minutes. The CloudStatus probes can be reimplemented easily; we have repeated some of the CloudStatus experiments in our previous work [11], [15], with similar results. We conclude that using CloudStatus data reduces the cost of our study, but does not reduce the applicability of the results.

Data Sanitation We have acquired data from CloudStatus through a sequence of web crawls (samples). The availability and robustness of our crawling setup resulted in 253,174 useful samples, or 96.3% of the maximum number of samples possible for the year. Figure 1 shows the number of samples taken every month; during February, April, and September 2009 our crawling infrastructure did not manage to obtain useful samples repeatedly (indicated by the reduced height of the "Sample Count" bars). Mostly during these month we have lost 9,626 samples due to missing or invalid JSON data; however, we have obtained 76–96% of the maximum number of samples during these three months.

B. Method of Analysis

For each of the traces we extract the performance indicators, to which we apply independently an analysis method with three steps: find out if variability is present at all, find out the main characteristics of the variability, and analyze in detail the variability time patterns. We explain each step in the following, in turn.

To find out if variability is present at all we select one month of data from our traces and plot the values of the performance
indicator where a wide range of values may indicate variability. The month selection should ensure that the selected month does not correspond to a single calendar month (to catch some human-scheduled system transitions), is placed towards the end of the year 2009 (to be more relevant) but does not overlap with December 2009 (to avoid catching Christmas effects).

To find out the characteristics of the variability we compute six basic statistics, the five quartiles ($Q_0$–$Q_4$) including the median ($Q_2$), the mean, and the standard deviation. We also compute one derivative statistic, the Inter-Quartile Range (IQR, defined as $Q_3 - Q_1$). We thus characterize for each studied parameter its location (mean and median), and its variability or scale (the standard deviation, the IQR, and the range). Either a relative difference between the mean and the median of over 10 percent, or a coefficient of variation above 1.10 indicate high variability and possibly a non-normal distribution of values which impacts negatively the ability to enforce soft performance guarantees. Similarly, a ratio between the IQR and the median above 0.5 indicates that the bulk of the performance observations have high variability, and a ratio between range and the IQR above 4 indicates that the performance outliers are severe.

Finally, to analyze the variability over time we investigate for each performance indicator the presence of yearly (month-of-year and week-of-year), monthly (day-of-month), weekly (day-of-week and workday/weekend), and daily patterns (hour-of-day). To this end, we first split for each time pattern investigated the complete dataset into subsets, one for each category corresponding to the time pattern. For example, to investigate the monthly time pattern we split the complete dataset into twelve subsets comprising the performance value samples observed during a specific month. Then, we compute for each subset the basic and derivative statistics performed over the complete dataset in the second step, and plot them for visual inspection. Last, we analyze the results and the plots, record the absence/presence of each investigated time pattern, and attempt to detect new time patterns.

C. Is Variability Present?

An important assumption of this work is that the performance variability of production cloud services indeed exists. We follow in this section the first step of our analysis method and verify this assumption.

Towards this end, we present the results for the selection of data from Sep 26 to Oct 26, 2009. For this month, we present here only the results corresponding to one sample service from each of the Amazon and Google clouds. Figure 2 shows the performance variability exhibited by the Amazon EC2 service (top of the figure, one performance indicator) and by the Google URL Fetch service (bottom of the figure, six performance indicators) during the selected month. For EC2, the range of values indicates moderate-to-high performance variability. For URL Fetch, the wide ranges of the six indicators indicate high variability for all URL Fetch operations, regardless of the target URL. In addition, the URL Fetch service targeting eBay web pages suffers from a visible decrease of performance around Oct 17, 2009. We have also analyzed the results for the selected month for all the other cloud services we investigate in this work, and have experimented with multiple one-month selections that follow the rules stated by our analysis method; in all cases we have obtained similar results (for brevity reasons not shown).

To conclude, the effects observed in this section give strong evidence of the presence of performance variability in cloud services, and motivate an in-depth analysis of the performance variability of both Amazon and Google cloud services.

IV. THE ANALYSIS OF THE AWS DATASET

In this section, we present the analysis of the AWS dataset. Each service comprises several operations, and for each operation, we investigate the performance indicators to understand the performance variability delivered by these operations.

A. Summary Statistics

In this section we follow the second step of our analysis method and analyze the summary statistics for AWS; Table I summarizes the results. Although the 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 IV-C) compared to performance measured from large and extra large EC2 instances. Similar to EC2, SDB also has low IQR but a high range especially for the update operations. Finally, FPS latency is highly variable which has implications for the applications using this service for payment operations as we present in Section VI-C.
A. Deployment Latency - The time it takes to start an
EC2 instance from the time startup is initiated to the time that
the instance is available.

Figure 3 shows weekly statistical properties of the EC2
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 increasing user base of EC2. Steady performance for
the deployment latency is especially important for applications
which uses the EC2 for auto-scaling.

C. Amazon Simple Storage Service (S3)

CloudStatus.com reports the throughput of 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 4 (top) depicts the hourly statistical properties of the
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,
Fig. 5. Amazon SDB: The monthly statistical properties of the update operation.

E. Amazon Simple Queue Service (SQS)

CloudStatus.com reports the following performance indicators for the 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 6 depicts the weekly statistical properties of the 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.

F. Amazon Flexible Payment Service (FPS)

CloudStatus.com reports the following performance indicators for the 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 7 depicts the monthly statistical properties of the 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.

G. Summary of the AWS Dataset

The performance results indicate that all Amazon services we analyzed in this section exhibit one or more time patterns and/or periods of time where the service shows special behavior, as summarized in Table II. EC2 exhibits periods of special behavior for the resource acquisition operation (Section IV-B). Both storage services of Amazon, SDB and S3, present daily, yearly, and monthly patterns for different operations (Section IV-D and Section IV-C). Finally, SQS and FPS show special behavior for specific time periods (Section IV-E and Section IV-F).

V. THE ANALYSIS OF THE GOOGLE APP ENGINE DATASET

In this section, we present the analysis of the Google App Engine dataset. Each service comprises several operations, and for each operation, we investigate the performance indicators in detail to understand the performance variability delivered by these operations.

A. Summary Statistics

In this section we follow the second step of our analysis method and analyze the summary statistics for GAE; Table III summarizes the results. The GAE Python runtime and Datastore have high range and IQRs leading to highly variable performance. However, we observe relatively stable performance for the Memcache service.
TABLE III
SUMMARY STATISTICS FOR GOOGLE APP ENGINE’S CLOUD SERVICES.

<table>
<thead>
<tr>
<th>Service</th>
<th>Min</th>
<th>Q1</th>
<th>Median</th>
<th>Q3</th>
<th>Max</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python Runtime (ms)</td>
<td>1.00</td>
<td>159.49</td>
<td>329.83</td>
<td>450.31</td>
<td>1.00</td>
<td>124.90</td>
<td>70.39</td>
</tr>
<tr>
<td>Datastore (ms)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>1.00</td>
<td>344.40</td>
<td>460.73</td>
<td>999.86</td>
<td>413.24</td>
<td>102.90</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>1.00</td>
<td>248.55</td>
<td>383.76</td>
<td>999.27</td>
<td>336.82</td>
<td>118.20</td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td>1.00</td>
<td>344.40</td>
<td>460.73</td>
<td>999.86</td>
<td>413.24</td>
<td>102.90</td>
<td></td>
</tr>
</tbody>
</table>

| Memcache (ms)    |      |      |        |      |      |      |     |
| Get              | 1.00 | 59.49 | 58.73 | 65.74 | 251.13 | 60.03 | 11.44 |
| Put              | 1.00 | 44.21 | 50.86 | 60.44 | 141.25 | 54.84 | 13.54 |
| Response         | 3.04 | 4.69  | 5.46  | 7.04  | 38.71  | 6.64  | 3.39  |

| URL Fetch [ms]   |      |      |        |      |      |      |     |
| s3.amazonaws.com | 1.00 | 198.66 | 226.13 | 245.85 | 396.35 | 431.24 | 64.10 |
| ebay.com         | 1.00 | 388.00 | 426.74 | 440.03 | 399.93 | 412.57 | 108.71 |
| api.facebook.com | 1.00 | 172.95 | 189.39 | 208.23 | 998.22 | 195.76 | 44.40 |
| api.hi5.com      | 73.13 | 95.81 | 102.58 | 113.40 | 478.75 | 107.03 | 25.12 |
| api.myspace.com  | 67.33 | 90.85 | 93.36 | 103.85 | 515.88 | 97.90  | 14.19 |
| paypal.com       | 1.00 | 406.57 | 415.97 | 431.69 | 998.39 | 421.76 | 35.00 |

Fig. 8. Google Run: The monthly statistical properties of running an application in the Python Runtime Environment.

B. The Google Run Service
CloudStatus.com reports the following performance indicators for the Run service:

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

Figure 8 depicts the monthly statistical properties of the GAE Python Runtime. 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. Similar to Amazon S3, the Datastore service exhibits a high IQR with yearly patterns (Section IV-C), 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.

C. 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 9 depicts the monthly statistical properties of the GAE 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 patterns (Section IV-C), 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.

D. 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 10 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
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.

Table IV

<table>
<thead>
<tr>
<th>Perf. Indicator</th>
<th>Yearly (Month)</th>
<th>Monthly (Day)</th>
<th>Weekly (Day)</th>
<th>Daily (Hour)</th>
<th>Special Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google App Engine</td>
<td>Run</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Datastore</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Memcache</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>URL Fetch</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

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.

E. 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 11 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 other target web sites for which a URL Fetch request is issued.

F. Summary of the Google App Engine Dataset

The performance results indicate that all GAE services we 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 IV. The Python Runtime exhibits periods of special behavior and daily patterns (Section V-B). The Datastore service presents yearly patterns and periods of time with special behavior (Section V-C). The Memcache service performance has also monthly patterns and time patterns of special behavior for various operations (Section V-D). Finally, the URL Fetch service presents weekly and daily patterns, and also shows special behavior for specific time periods for different target websites (Section V-E).

TABLE V

<table>
<thead>
<tr>
<th>Application</th>
<th>Used Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job execution</td>
<td>GAE Run</td>
</tr>
<tr>
<td>Game status management</td>
<td>AWS SDB</td>
</tr>
</tbody>
</table>

VI. THE IMPACT OF VARIABILITY ON LARGE-SCALE APPLICATIONS

In this section we assess the impact of the variability of cloud service performance on large-scale applications using trace-based simulations. Since there currently exists no accepted traces or models of cloud workloads, we propose scenarios in which three realistic applications would use specific cloud services. Table V summarizes these applications and the main cloud service that they use.

A. Experimental Setup

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

Simulator We design for each application a simulator that considers from the 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 performance for a cloud with stable performance vs variable performance. For each application 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 to the simulator, is the monthly performance variability analyzed earlier in this work. We define the reference performance indicator, which is 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 from grid and parallel production environments (PPEs), we first report two traditional metrics for the grid and PPE communities: the average response time (ART), the average bounded slowdown (ABSD) with a threshold of 10 seconds [16]; 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 $APR(t) =$
In this scenario we analyze the execution of CPUs. 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 this scenario.

### Table VI

<table>
<thead>
<tr>
<th>Source (Trace ID)</th>
<th>Grid Workloads Archive [17], 3 traces</th>
<th>CPU-Hours</th>
<th>Number of Users</th>
<th>System CPU</th>
<th>Load [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAL (6)</td>
<td>1.1M</td>
<td>412</td>
<td>10</td>
<td>6.8K</td>
<td>85.7</td>
</tr>
<tr>
<td>Grid (8)</td>
<td>1.3M</td>
<td>412</td>
<td>10</td>
<td>6.8K</td>
<td>85.7</td>
</tr>
<tr>
<td>RAL (10)</td>
<td>1.1M</td>
<td>412</td>
<td>10</td>
<td>6.8K</td>
<td>85.7</td>
</tr>
<tr>
<td>CTC SP2 (6)</td>
<td>1.1M</td>
<td>412</td>
<td>10</td>
<td>6.8K</td>
<td>85.7</td>
</tr>
<tr>
<td>SDSC SP2 (9)</td>
<td>1.1M</td>
<td>412</td>
<td>10</td>
<td>6.8K</td>
<td>85.7</td>
</tr>
</tbody>
</table>

\[ P(t) \leq P(t) \leq U(t) \text{max}, \] where \( P(t) \) is the performance at time \( t \), \( P(t) \text{ref} \) is the reference performance, \( U(t) \text{max} \) is the maximum number of users over the course of the trace, and \( P(t) \) 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 VII

<table>
<thead>
<tr>
<th>Source (Trace ID)</th>
<th>Cloud with Variable Performance</th>
<th>Cloud with Stable Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAL (6)</td>
<td>18,837</td>
<td>18,937</td>
</tr>
<tr>
<td>Grid (8)</td>
<td>18,837</td>
<td>18,937</td>
</tr>
<tr>
<td>RAL (10)</td>
<td>18,837</td>
<td>18,937</td>
</tr>
<tr>
<td>CTC SP2 (6)</td>
<td>18,837</td>
<td>18,937</td>
</tr>
<tr>
<td>SDSC SP2 (9)</td>
<td>18,837</td>
<td>18,937</td>
</tr>
</tbody>
</table>

#### C. 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 12).

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

**Results** The main result is that our APR metric can be used to trigger and motivate the decision of switching cloud providers. Figure 12 shows the APR when using Amazon’s FPS as the micro-payment backend. (Data source for the number of FaceBook users: http://www.developeranalytics.com/).
The main finding is that there is a big discrepancy between the variability of Amazon SDB and Google Datastore (Section IV-D) and of Google Datastore (Section V-C). The performance exhibits the monthly variability of Amazon SDB for the Farm Town social game (Figure 13).

We use as input trace the number of daily unique users for each month January 2010 in the input traces. Since the Datastore operations exhibit yearly patterns (Section V-F), 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 provider. Figures 13 depicts the APR 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 9). 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.

**D. Game Status Maintenance for Social Games**

**Scenario** In this scenario we investigate the maintenance of game status for a large-scale social game such as Farm Town or Mafia Wars which 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 [21]. 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 13).

**Variability** We assume, in turn, that the cloud with variable performance exhibits the monthly variability of Amazon SDB (Section IV-D) and of Google Datastore (Section V-C). 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

![Figure 13: 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/)](image)
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.

VIII. CONCLUSION

Production cloud services may incur high performance variability, due to the combined and non-trivial effects of system size, workload variability, 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. 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 services. The two main achievements of our study are described in the following.

First, we have analyzed the time-dependent characteristics exhibited by the traces, and found that the performance of the investigated services exhibits on the one hand yearly and daily patterns, and on the other hand periods of stable performance. We have also found that many services exhibit high variation in the monthly median values, which indicates large performance changes over time.

Second, we have found that the impact of the performance variability varies greatly across application types. For example, we found that the service of running applications on GAE, which exhibits high performance variability and a three-months period of low variability and improved performance, has a negligible impact for running grid and parallel production workloads. In contrast, we found that and explained the reasons for which the GAE database service, having exhibited the monthly median values, which indicates large performance fluctuations of hpc workloads on clouds, in CloudCom.

REFERENCES


