2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

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).

978-0-7695-4395-6/11 $26.00 © 2011 IEEE
DOI 10.1109/CCGrid.2011.22

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 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 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.

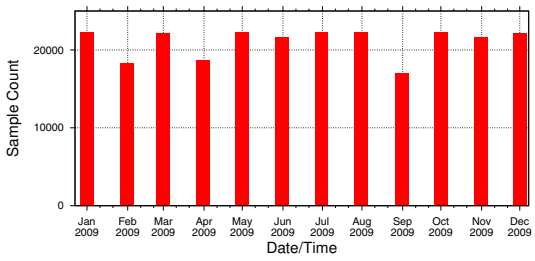

Fig. 1. Number of monthly data samples.

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). CloudStatus 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

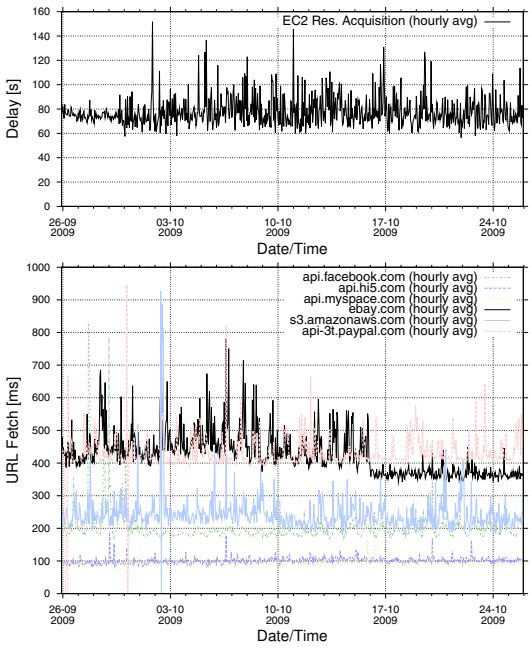

Fig. 2. Performance variability for two selected cloud services during the period Sep 26, 2009 to Oct 26, 2009: (top) for Amazon EC2, and (bottom) for Google URL Fetch.

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.

TABLE I
SUMMARY STATISTICS FOR AMAZON WEB SERVICES'S CLOUD SERVICES.

Service	Min	Q1	Median	Q3	Max	Mean	SD
EC2 [s]							
Deployment Latency	57.00	73.59	75.70	78.50	122.10	76.62	5.17
S3 [MBps]							
GET EU HIGH	0.45	0.65	0.68	0.70	0.78	0.68	0.30
GET US HIGH	8.60	15.50	17.10	18.50	25.90	16.93	2.39
PUT EU HIGH	1.00	1.30	1.40	1.40	1.50	1.38	0.10
PUT US HIGH	4.09	8.10	8.40	8.60	9.10	8.26	0.55
SDB [ms]							
Query Response Time	28.14	31.76	32.81	33.77	85.40	32.94	2.39
Update Latency	297.54	342.52	361.97	376.95	538.37	359.81	26.71
SQS [s]							
Lag Time	1.35	1.47	1.50	1.79	6.62	1.81	0.82
FPS [ms]							
Latency	0.00	48.97	53.88	76.06	386.43	63.04	23.22

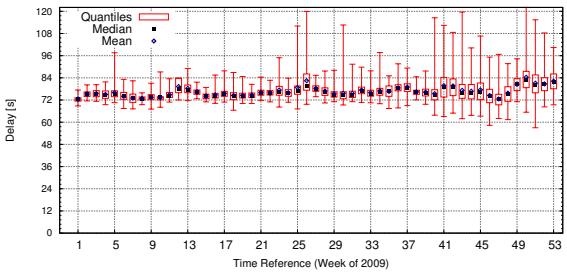

Fig. 3. Amazon EC2: The weekly statistical properties of the resource acquisition operation. The box and whiskers show min-Q_1-Q_3-max.

B. 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 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,

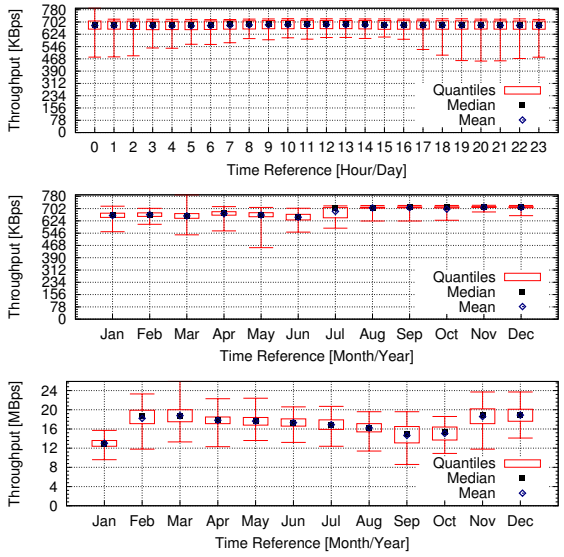

Fig. 4. Amazon S3: The hourly statistical properties of GET EU HI operations (top), and the monthly statistical properties of the GET EU HI operations (middle) and of GET US HI operations (bottom).

and the work day hours (from 8AM to 3PM) exhibiting much higher minimal transfer rates.

Figure 4 (middle) shows the monthly statistical properties of the 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 4 (bottom) shows the monthly statistical properties of the 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 4 (bottom) also shows that there exists a wide range of median monthly performance values, from 13 to 19 MBps over the year.

D. 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 5 shows the monthly statistical properties of the 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 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).

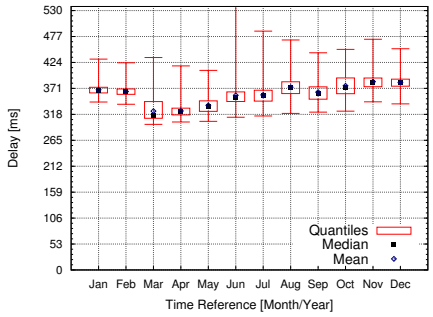

Fig. 5. Amazon SDB: The monthly statistical properties of the update operation.

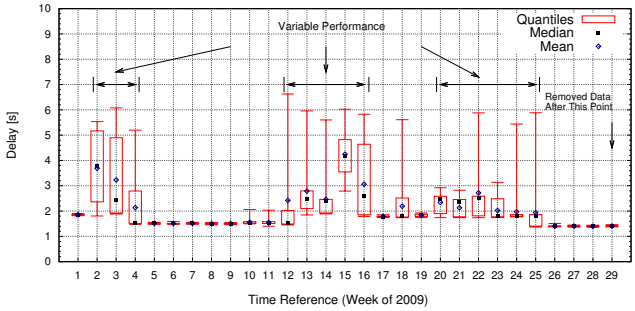

Fig. 6. Amazon SQS: The weekly statistical properties. The statistics for the weeks 30–53 (not shown) are very similar to those for weeks 26–29.

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.

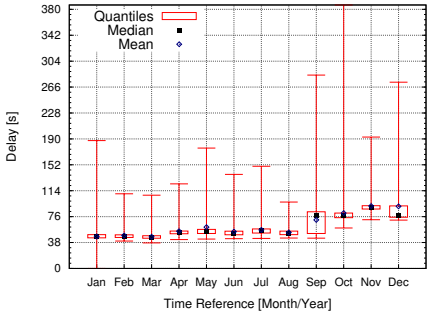

Fig. 7. Amazon FPS: The monthly statistical properties.

TABLE II
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
Amazon AWS					
EC2					Y
S3	Y		Y	Y	Y
SDB	Y			Y	
SQS					Y
FPS					Y

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.

Service	Min	Q1	Median	Q3	Max	Mean	SD
Python Runtime [ms]	1.00	284.14	302.31	340.37	999.65	314.95	76.39
Datastore [ms]							
Create	1040	1280	1420	1710	5590	1600	600
Delete	1.00	344.40	384.22	460.73	999.86	413.24	102.90
Read	1.00	248.55	305.68	383.76	999.27	336.82	118.20
Memcache [ms]							
Get	45.97	50.49	58.73	65.74	251.13	60.03	11.44
Put	33.21	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	1.01	198.60	226.13	245.83	983.31	214.21	64.10
ebay.com	1.00	388.00	426.74	460.03	999.83	412.57	108.31
api.facebook.com	1.00	172.95	189.39	208.23	998.22	195.76	44.40
api.hi5.com	71.31	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

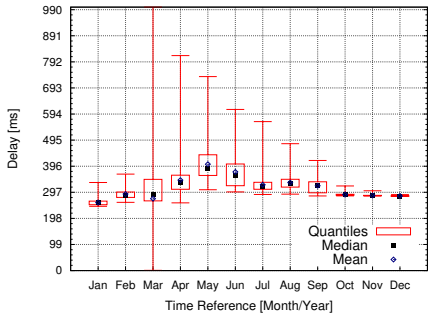

Fig. 8. Google Run: The monthly statistical properties of running an application in the Python Runtime Environment.

B. The Google Run Service

CloudStatus.com reports the following performance indicator 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 narrow range, and with steady month-to-month median. Similar to the Amazon SDB service (see Section IV-D), 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.

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.

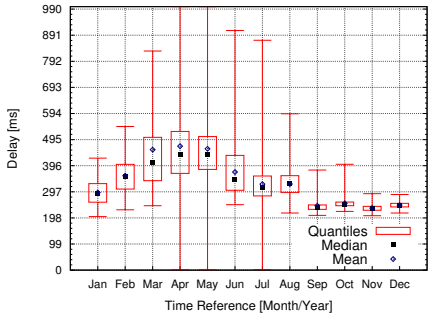

Fig. 9. Google Datastore: The monthly statistical properties of the read operation.

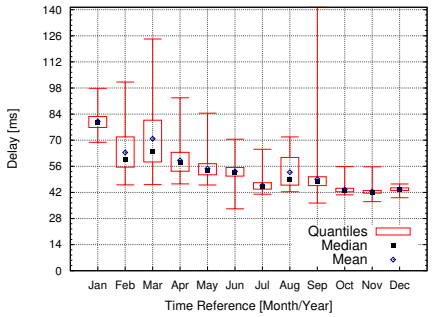

Fig. 10. Google Memcache: The monthly statistical properties of the PUT operation.

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

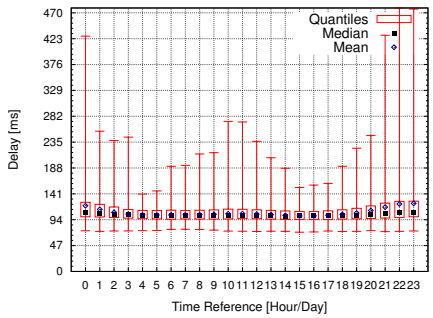

Fig. 11. Google URL Fetch: The hourly statistical properties; target web site is the Hi5 social network.

TABLE IV
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
Google App Engine					
Run				Y	Y
Datastore	Y				Y
Memcache					
URL Fetch			Y	Y	Y

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
LARGE-SCALE APPLICATIONS USED TO ANALYZE THE IMPACT OF VARIABILITY.

Section	Application	Used Service
Section VI-B	Job execution	GAE Run
Section VI-C	Selling virtual goods	AWS FPS
Section VI-B	Game status management	AWS SDB GAE Datastore

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* 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 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 $APR(t) =$

TABLE VI
JOB EXECUTION (GAE RUN SERVICE): THE CHARACTERISTICS OF THE INPUT WORKLOAD TRACES.

Trace ID, Source (Trace ID in Archive)	Trace Number of Mo.	Jobs	Users	System Size Sites	CPUs	Load [%]
Grid Workloads Archive [17], 3 traces						
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	-
Parallel Workloads Archive [18], 2 traces						
4. CTC SP2 (6)	11	0.1M	679	1	430	66
5. SDSC SP2 (9)	24	0.1M	437	1	128	83

$\frac{P(t)}{P_{ref}} \times \frac{U(t)}{U_{max}}$, where $P(t)$ is the performance at time t, P_{ref} is the reference performance, $U(t)$ is the number of users at time t, and U_{max} is the maximum number of users over the course of the trace; $P(t)$ 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.

B. 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 five long-term traces from real grids and PPEs as workloads; Table VI summarizes their characteristics, with the ID of each trace indicating the system from which the trace was taken; see [17], [18] 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 also assume that the GAE Run service can run the input workload, and exhibits the monthly variability evaluated in Section V-B. 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 VII 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 this scenario.

TABLE VII
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 ART [s]	ABSD (10s)	Cost	Variable Performance 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

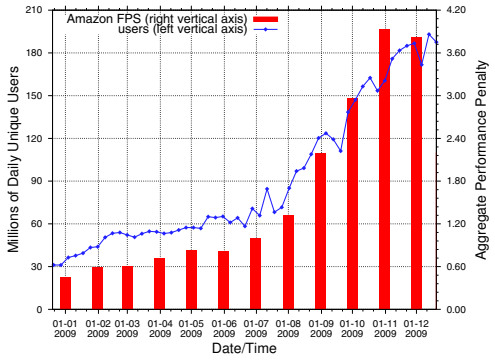

Fig. 12. Selling Virtual Goods in Social Networks (Amazon FPS): Aggregate Performance Penalty when using Amazon FPS as the micro-payment backend. (Data source for the number of FaceBook users: `http://www.developeranalytics.com/`)

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 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 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 clients respond to high payment latency similarly to other consumers of Internet newmedia [19], [20], that is, they become unsatisfied and quit, our APR metric is a clear indicator for the virtual goods vendor that the cloud provider

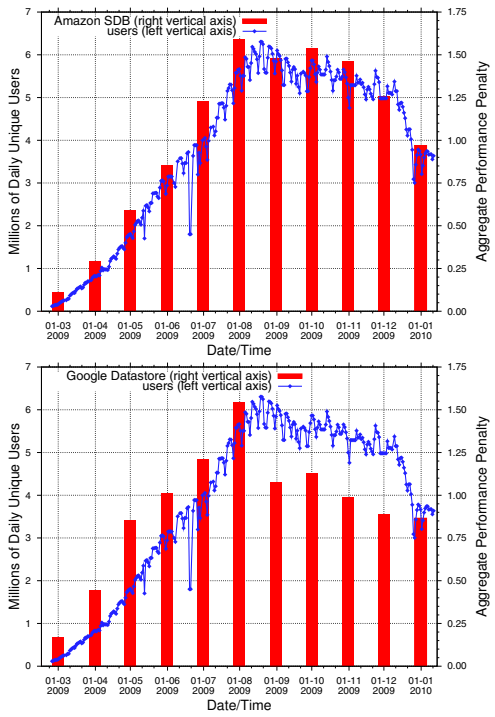

Fig. 13. Game Status Maintenance for Social Games (Amazon SDB and Google App Engine Datastore): Aggregate Performance Penalty (top) when using Amazon SDB as the database backend; (bottom) when using Google App Engine Datastore as the database backend. (Data source for the number of Farm Town users: `http://www.developeranalytics.com/`)

should be changed.

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 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 opeation (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.

VII. Related Work

Much effort has been put recently in assessing the performance of virtualized resources, in cloud computing environments [7]–[11], [22]–[24] and in general [25]–[28]. 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 performance variability of production cloud services.

Close to our work is the seminal study of Amazon S3 [8], which also includes a 40 days evaluation of the service availability. Our work complements this study by analyzing the performance of eight other AWS and GAE services over a 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 [9], the early comparative study of Eucalyptus and EC2 performance [10], the study of performance and cost of executing a scientific workflow in clouds [7], the study of file transfer performance between Amazon EC2 and S3, etc. Our results complement these studies and give more insight into the (variability of) performance of EC2 and other cloud services.

Recent studies using general purpose benchmarks have shown that virtualization overhead can be below 5% for computation [25] and below 15% for networking [25], [27]. Similarly, the performance loss due to virtualization for parallel I/O and web server I/O has been shown to be below 30% [29] and 10% [30], respectively. Our previous work [11], [15] has shown that virtualized resources in 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.

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 a similar period of better performance as the GAE running service, outperforms the AWS database service for a social gaming application.

References

[1] R. H. Arpaci-Dusseau, A. C. Arpaci-Dusseau, A. Vahdat, L. T. Liu, T. E. Anderson, and D. A. Patterson, "The interaction of parallel and sequential workloads on a network of workstations," in *SIGMETRICS*, 1995, pp. 267–278.

[2] A. Iosup, O. O. Sonmez, S. Anoep, and D. H. J. Epema, "The performance of bags-of-tasks in large-scale distributed systems," in *HPDC*. ACM, 2008, pp. 97–108.

[3] Y. Zhang, M. S. Squillante, A. Sivasubramaniam, and R. K. Sahoo, "Performance implications of failures in large-scale cluster scheduling," in *JSSPP*, 2004, pp. 233–252.

[4] D. Hilley, "Cloud computing: A taxonomy of platform and infrastructure-level offerings," Georgia Institute of Technology, Tech. Rep. GIT-CERCS-09-13, Dec 2008. [Online]. Available: www.cercs.gatech.edu/tech-reports/tr2009/git-cercs-09-13.pdf

[5] J. Maguire, J. Vance, and C. Harvey, "85 cloud computing vendors shaping the emerging cloud," Aug 2009, iTManagement Tech.Rep.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009. [Online]. Available: www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[7] E. Deelman, G. Singh, M. Livny, J. B. Berriman, and J. Good, "The cost of doing science on the cloud: the Montage example," in *SC*. IEEE/ACM, 2008, p. 50.

[8] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for science grids: a viable solution?" in *DADC '08: Proceedings of the 2008 international workshop on Data-aware distributed computing*. ACM, 2008, pp. 55–64.

[9] E. Walker, "Benchmarking Amazon EC2 for HP Scientific Computing," *Login*, vol. 33, no. 5, pp. 18–23, Nov 2008.

[10] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system," in *CCGRID*, 2009, pp. 124–131.

[11] 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*, ser. LNICST, vol. 34, 2009, pp. 115–31.

[12] RightScale, "Amazon usage estimates," Aug 2009, [Online] Available: blog.rightscale.com/2009/10/05/amazon-usage-estimates.

[13] G. Rosen, "Cloud usage analysis series," Aug 2009, [Online] Available: www.jackofallclouds.com/category/analysis.

[14] The Cloud Status Team, "Report on cloud performance and availability status," Jan. 2010, [Online] Available: www.cloudstatus.com.

[15] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. on Parallel and Distrib. Sys.*, 2010, (accepted Sep 2010, in print).

[16] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong, "Theory and practice in parallel job scheduling," in *JSSPP*, ser. LNCS, vol. 1291. Springer-Verlag, 1997, pp. 1–34.

[17] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. Epema, "The Grid Workloads Archive," *FGCS*, vol. 24, no. 7, pp. 672–686, 2008.

[18] The Parallel Workloads Archive Team, "The parallel workloads archive logs," Jan. 2010, [Online] Available: www.cs.huji.ac.il/labs/parallel/workload/logs.html.

[19] K. T. Chen, P. Huang, and C. L. Lei, "Effect of network quality on player departure behavior in online games," *IEEE TPDS*, vol. 20, no. 5, pp. 593–606, 2009.

[20] M. Claypool and K. T. Claypool, "Latency and player actions in online games," *CACM*, vol. 49, no. 11, pp. 40–45, 2006.

[21] 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*, 2008, p. 10.

[22] Y. El-Khamra, H. Kim, S. Jha, and M. Parashar, "Exploring the performance fluctuations of hpc workloads on clouds," in *CloudCom*. IEEE, 2010.

[23] N. Yigitbasi, A. Iosup, D. H. J. Epema, and S. Ostermann, "C-meter: A framework for performance analysis of computing clouds," in *CCGRID*, 2009, pp. 472–477.

[24] J. Dejun, G. Pierre, and C.-H. Chi, "EC2 performance analysis for resource provisioning of service-oriented applications," in *Proceedings of the 3rd Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing*, Nov. 2009.

[25] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *SOSP*. ACM, 2003, pp. 164–177.

[26] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J. N. Matthews, "Xen and the art of repeated research," in *USENIX ATC*, 2004, pp. 135–144.

[27] A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel, "Diagnosing performance overheads in the Xen virtual machine environment," in *VEE*. ACM, 2005, pp. 13–23.

[28] 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)*. ACM, 2009, pp. 25–30.

[29] W. Yu and J. S. Vetter, "Xen-based HPC: A parallel I/O perspective," in *CCGrid*. IEEE, 2008, pp. 154–161.

[30] L. Cherkasova and R. Gardner, "Measuring CPU overhead for I/O processing in the Xen virtual machine monitor," in *USENIX ATC*, 2005, pp. 387–390.