Applying Design of Experiments (DOE) to Performance Evaluation of Commercial Cloud Services

Zheng Li, Australian National University and NICTA, Australia*
Liam O'Brien, Geoscience Australia, Australia
He Zhang, University of East London, UK
Rajiv Ranjan, CSIRO ICT Center, Australia

ABSTRACT

Appropriate performance evaluations of commercial Cloud services are crucial and beneficial for both customers and providers to understand the service runtime, while suitable experimental design and analysis would be vital for practical evaluation implementations. However, there seems to be a lack of effective methods for Cloud services performance evaluation. For example, in most of the existing evaluation studies, experimental factors (also called parameters or variables) were considered randomly and intuitively, experimental sample sizes were determined on the fly, and few experimental results were comprehensively analyzed. To address these issues, we suggest applying Design of Experiments (DOE) to Cloud services evaluation. To facilitate applying DOE techniques, this paper introduces an experimental factor framework and a set of DOE application scenarios. As such, new evaluators can explore and conveniently adapt our work to their own experiments for performance evaluation of commercial Cloud services.

Keywords: Cloud computing, commercial Cloud services, performance evaluation, experimental design, factor framework

INTRODUCTION

Along with the boom in Cloud computing, an increasing number of commercial providers have started to offer public Cloud services (Li et al., 2010; Prodan & Ostermann, 2009). Different commercial Cloud services have been supplied with different terminologies, qualities, and cost models (Prodan & Ostermann, 2009). Consequently, performance evaluation of those services would be crucial and beneficial for both service customers and providers (Li et al., 2010). For example, proper performance evaluation of candidate Cloud services can help customers perform cost-benefit analysis and decision making for service selection, while it can also help providers improve their service qualities against competitors. Given the diversity of Cloud services and the uncertainty of service runtime, however, implementing appropriate performance evaluation of Cloud services is not easy. In particular, since Cloud services evaluation belongs to the domain of experimental computer science, suitable experimental design and analysis would be vital for practically evaluating Cloud services (Stantchev, 2009).

Unfortunately, there seems to be a lack of effective methods for determining evaluation implementations in the Cloud computing domain. The current experimental design approaches vary significantly in the existing studies of Cloud services evaluation, and we have identified three main issues related to the current evaluation experiments. Firstly, the experimental sample sizes were determined arbitrarily, while inappropriate sample size could increase the probability of type II error in evaluation experiments (Montgomery, 2009). Secondly, most evaluators did not specify “experimental factors” when preparing evaluation experiments. In fact, identification of the relevant factors that may influence performance plays a prerequisite role in designing evaluation experiments (Jain, 1991). Thirdly, few Cloud services evaluation reports gave comprehensive analysis of experimental results. However, sound evaluation conclusions may require more objectivity by applying more statistical methods to experimental analysis (Montgomery, 2009).

To deal with these identified issues, we decided to apply Design of Experiments (DOE) strategies to performance evaluation of commercial Cloud services. DOE is traditionally applied to agriculture, chemical, and process industries (Antony, 2003; Montgomery, 2009). Considering the natural relationship between experiment and evaluation, we believe that the various DOE techniques of experimental design and statistical analysis can also benefit Cloud services evaluation. Therefore, we investigated two main activities of applying DOE: (1) selection of input factors (parameters of Cloud resources and workload) and response variables (indicators of service runtime qualities); (2) choice of experimental design and statistical analysis based on the selected factors/variables. To facilitate experimental factor selection, we established a factor framework after collecting, clarifying and rationalizing the key concepts and their relationships in the existing Cloud performance evaluation studies. To help identify suitable experimental design and analysis techniques, we performed a series of case studies to demonstrate a set of DOE application scenarios. As such, new evaluators can explore and refer to our work to design their own experiments for performance evaluation of commercial Cloud services.

Note that, as a continuation of our previous work (Li et al., 2012a, 2012b, 2012c, in press a, in press b), this study conventionally employed four constrains, as listed below.

- We focused on the evaluation of only commercial Cloud services, rather than that of private or academic Cloud services, to make our effort closer to industry's needs.
- We only investigated Performance evaluation of commercial Cloud services. The main reason is that not enough data about evaluating other service features could be found to support the generalization work. For example, there are little empirical studies in Security evaluation of commercial Cloud services due to the lack of quantitative metrics (Li et al., 2012b).
- We considered Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) without considering Software as a Service (SaaS). Since SaaS with special functionalities is not used to further build individual business applications (Binnig et al., 2009), the evaluation of various SaaS instances could comprise an infinite and exclusive set of factors that would be out of the scope of this investigation.
- We only explored empirical evaluation practices in academic publications. There is no doubt that informal descriptions of Cloud services evaluation in blogs and technical websites can also provide highly relevant information. However, on the one hand, it is impossible to explore and collect useful data from different study sources all at once. On the other hand, the published evaluation reports can be viewed as typical and peer-reviewed representatives of the existing ad hoc evaluation practices.

The remainder of this paper is organized as follows. Next section summarizes the related work following the sequence of the identified issues. Section 3 briefly introduces the procedure of establishing the proposed factor framework, and specifies the tree-structured framework branch by branch. Two real cases are employed in Section 4 to demonstrate three typical DOE application scenarios when evaluating performance of commercial Cloud services. Conclusions and some future work are discussed in the last section.

RELATED WORK

Evaluation is crucial and inevitably required for computing paradigms involving virtualization (Wang et al., 2010a, 2010b, 2010c, 2011), and Cloud computing is such a typical case. Cloud services evaluation has been recognized as being in the field of experimental computer science (Stantchev, 2009). Therefore, designing suitable experiments would be crucial before practically evaluating Cloud services. In fact, most evaluators have emphasized experimental design or setup in their evaluation reports. Some general activities are widely highlighted, such as identifying service feature to be evaluated, selecting benchmarks and metrics, configuring experimental environment, etc. (Hill et al., 2010; Juve et al., 2009; Li et al., 2010; Palankar et al., 2008; Stantchev, 2009). Unfortunately, the current experimental design approaches vary significantly, and three main issues can be found in the existing Cloud services evaluation studies.

First, the number of experimental replicates (sample sizes) was determined on the fly. It is clear that the repeat of experiments is vital particularly for observing variance of Cloud service features. Thus, Stantchev (2009) repeated an Amazon EC2 test six times on six consecutive days; Ostermann et al. (2009) chose 20 times when investigating the variability of Cloud resource acquisition and release; Palankar et al. (2008) downloaded data from Amazon S3 every 15 minutes; Hill et al. (2010) evaluated TCP communication between two Azure VMs by transferring data every half hour for several days; Hill & Humphrey (2010) passed messages between EC2 instances 1000 trials for the smaller data points up to 32K and 10 trials for 4MB; while some other works emphasized that their experiments were repeated multiple times without specifying exact numbers (Juve et al., 2009; Li et al., 2010). However, all these evaluators did not justify how and why they set those experimental sample sizes.

Second, there is a lack of systematic approaches to factor selection for experimental design. In most cases, evaluators identified factors either randomly or intuitively, and thus prepared evaluation experiments through an ad hoc way. For example, when it comes to the performance evaluation of Amazon EC2, different studies casually considered different EC2 instance factors in different experiments, such as VM type (Stantchev, 2009), number (Stantchev, 2009), geographical location (Iosup, Yigitbasi & Epema, 2010), operation system (OS) brand (Li et al., 2010), and even CPU architecture (Iosup, Yigitbasi & Epema, 2010) and brand (Napper & Bientinesi, 2009), etc. In fact, to the best of our knowledge, none of the current Cloud performance evaluation studies has used "experimental factors" deliberately to design evaluation experiments and analyze the experimental results.

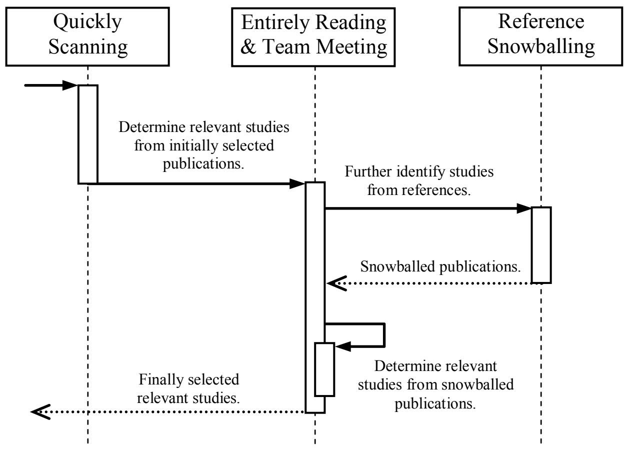
Third, few Cloud services evaluation studies analyzed experimental results comprehensively. In fact, statistical methods have been strongly suggested for experimental analysis (Montgomery, 2009). Although such methods do not directly prove any factor's effect, the statistical analysis adds objectivity to drawing evaluation conclusions and potential decision-making process. Nevertheless, it seems that most evaluators intend to mainly report their observations by visualizing or listing the experimental results (Juve et al., 2009; Stantchev, 2009). Only when evaluating the variability of a particular Cloud service feature, some studies employed simple graphical tools like Box Plot (Ostermann et al., 2009) and Cumulative Fraction (Hill et al., 2010; Li et al., 2010; Palankar et al., 2008), while some others limited themselves to giving the minimum, maximum and average values (Baun & Kunze, 2009). The lack of comprehensive data analysis could be a result of the aforementioned ad hoc design that did not consider "experimental factors".

THE TREE-STRUCTURED FACTOR FRAMEWORK

As mentioned previously, to facilitate experimental factor selection, we established an experimental factor framework based on our previous work. The whole work is mainly composed of four steps, as respectively specified below.

- **Conduct a systematic literature review (SLR).** The foundation for establishing this factor framework is a systematic literature review (SLR) on evaluating commercial Cloud services (Li et al., in press b). As the main methodology applied for Evidence-Based Software Engineering (EBSE) (Dybå, Kitchenham & Jørgensen, 2005), SLR has been widely accepted as a standard and systematic approach to investigation of specific research questions by identifying, assessing, and analyzing published primary studies. Following a rigorous selection process in this SLR, as illustrated in Figure 1, we have identified 82 Cloud services evaluation studies covering six commercial Cloud providers, such as Amazon, GoGrid, Google, IBM, Microsoft, and Rackspace, from a set of popular digital publication databases. The evaluation experiments in those identified 82 studies were thoroughly analyzed. In particular, the atomic experimental components, such as evaluation requirements, Cloud service features, metrics, benchmarks, experimental resources, and experimental operations, were respectively extracted and arranged.

  Figure 1. The study selection sequence in the SLR on evaluating commercial Cloud services.


Quickly Scanning
Entirely Reading & Team Meeting
Reference Snowballing
Determine relevant studies from initially selected publications.
Further identify studies from references.
Snowballed publications.
Determine relevant studies from snowballed publications.
Finally selected relevant studies.

- **Construct a taxonomy based on the SLR.** During the analysis of these identified evaluation studies, we found that there were frequent reporting issues ranging from non-standardized specifications to misleading explanations (Li et al., 2012a). Considering that those issues would inevitably obstruct comprehending and spoil drawing lessons from the existing evaluation work, we created a novel taxonomy to clarify and arrange the key concepts and terminology for Cloud services performance evaluation. The taxonomy is constructed along two dimensions: Performance Feature and Experiment. Moreover, the Performance Feature dimension is further split into *Physical Property* and *Capacity* parts, while the Experiment dimension is split into *Environmental Scene* and *Operational Scene* parts, as shown in Figure 2. The details of this taxonomy have been elaborated in (Li et al., 2012a).

- **Build a conceptual model based on the taxonomy.** Since a model is an abstract summary of some concrete object or activity in reality (Mellor, Clark & Futagami, 2003), the identification of real and concrete objects/activities plays a fundamental role in the corresponding modeling work. Given that the taxonomy has capsuled relevant key concepts and terminology, we further built a conceptual model of performance evaluation of commercial Cloud services to rationalize different abstract-level classifiers and their relationships (Li et al., in press a). In detail, we used a three-layer structure to host different abstract elements for the performance evaluation conceptual model. To save space, here we only portray the most generalized part hosted in the top classifier layer, as shown in Figure 3, which reflects the most generic reality of performance evaluation of a computing paradigm: essentially, performance evaluation can be considered as *exploring the capacity of particular computing resources with particular workloads driven by a set of operations*.

Figure 2. Two-dimensional taxonomy of Cloud services performance evaluation.

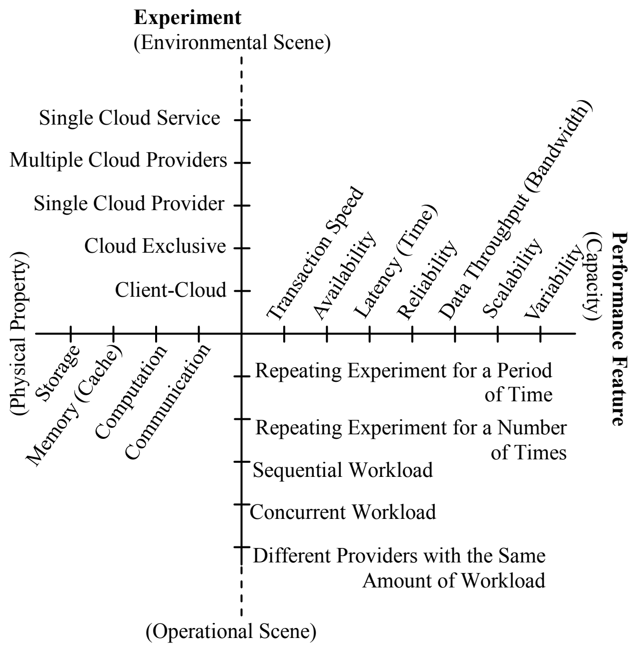
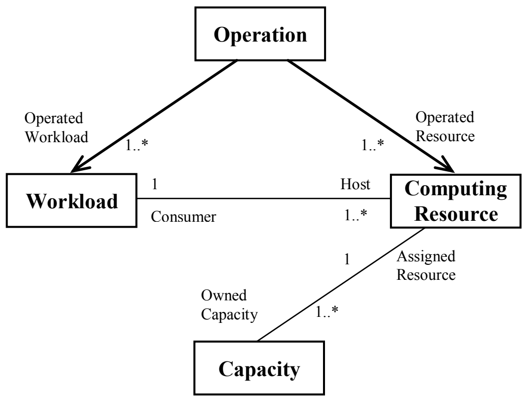


Figure 3. Conceptual model of Cloud services performance evaluation in the top classifier layer.

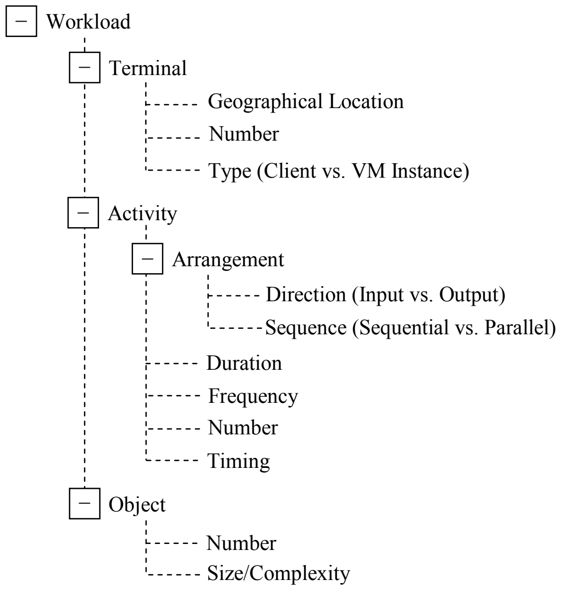


- **Establish an experimental factor framework.** In fact, the specific classifiers in the abovementioned conceptual model (Li et al., in press a) has implied the state-of-the-practice of performance evaluation factors that people currently took into account in the Cloud Computing domain. According to different positions in the process of an evaluation experiment (Antony, 2003), the specific classifiers of Workload and Computing Resource indicate input process factors; the specific classifiers of Capacity suggest output process factors; while the Operation classifiers are used to adjust values of input process factors. Consequently, the experimental factors for performance evaluation of commercial Cloud services can be categorized into two input process groups (Workload and Computing Resource) and one output process group (Capacity). Then, we naturally portrayed the factor framework as a tree with three branches (Li et al., 2012c). Each of the following subsections describes one branch of the factor tree.

Workload Factors

Based on our previous work (Li et al., 2012a; Li et al., in press a), we found that a piece of workload used in performance evaluation could be described through one of three different concerns or a combination of them, namely Terminal, Activity, and Object. As such, we can adjust the workload by varying any of the concerns through different experimental operations. The individual workload factors are listed in Figure 4.

Figure 4. The workload factors for experimental design.



Terminal

In contrast with services to be evaluated in the Cloud, clients and particular Cloud resource (usually VM instances) issuing workload activities can be viewed as terminals. Correspondingly, the *geographical location* or *number* of both clients (Garfinker, 2007a) and VM instances (Hill et al., 2010) have been used to depict the relevant workload. Meanwhile, the *terminal type* can also be used as a workload factor. For example, the authors evaluated Cloud network latency by using client and EC2 instance respectively to issue pings (Baun & Kunze, 2009). In this case, the *terminal type* has the equal essence to the factor *communication scope* (cf. Subsection *Communication*).

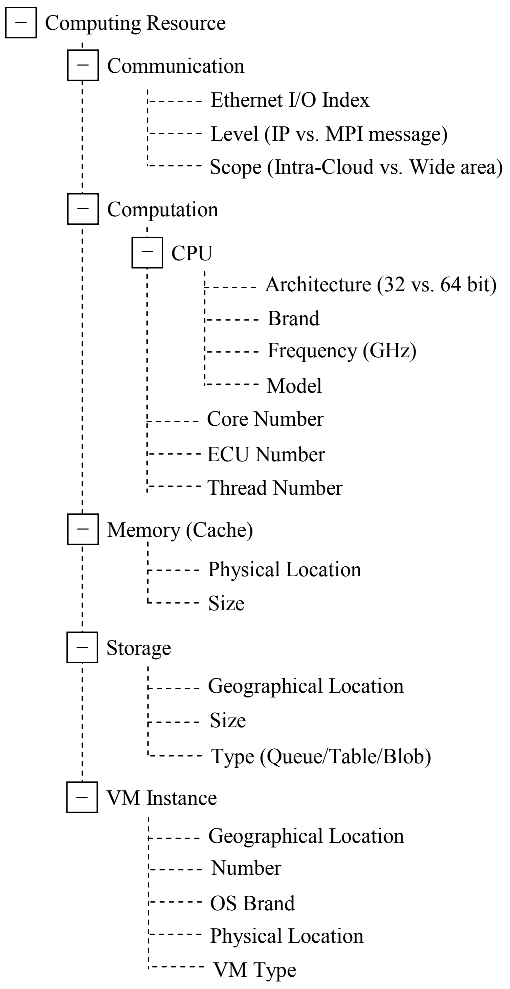
Activity

The concept "activity" here describes an inherent property of workload, which is different from, but adjustable by, experimental operations. For example, disk I/O request as a type of activity can be adjusted by operations like the number or time of the requests. In fact, the number- and time-related variables, such as *activity duration* (Garfinker, 2007a), *frequency* (Chiu & Agrawal, 2010), *number* (Chiu & Agrawal, 2010), and *timing* (Garfinker, 2007b), have been widely considered as workload factors in practice. Furthermore, by taking a particular Cloud resource being evaluated as a reference, the factor *activity direction* can be depicted as input or output (Baun & Kunze, 2009). As for the *activity sequence* in a workload, the arrangement generates either sequential (Baun & Kunze, 2009) or parallel (Hill et al., 2010) activity flows.

Object

In a workload for Cloud services performance evaluation, objects refer to the targets of the abovementioned activities. The concrete objects can be individual messages (Hill & Humphrey, 2009), data files (Hill et al., 2010), and transactional jobs/tasks (Deelman et al., 2008) in fine grain, while they can also be coarse-grained workflows or problems (Deelman et al., 2008). Therefore, the *object number* and *object size/complexity* are two typical workload factors in the existing evaluation studies. Note that we do not consider object location as a workload factor, because the locations of objects are usually hosted and determined by computing resources (cf. Subsection Computing Resource Factors). In particular, a workload may have multiple object size/complexity-related factors in one experiment. For example, a set of parameters of HPL benchmark, such as the block size and process grid size, should be tuned simultaneously when evaluating Amazon EC2 (Bientinesi, Iakymchuk & Napper, 2010).

Figure 5. The computing resource factors for experimental design.



Computing Resource Factors

According to the physical properties in the performance feature of commercial Cloud services (Li et al., 2012a), the Cloud Computing resource can be consumed by one or more of four basic styles: Communication, Computation, Memory (Cache), and Storage. In particular, the VM Instance resource is an integration of all the four basic types of computing resources. Overall, the computing resource factors can be organized as shown in Figure 5.

Communication

As explained in (Li et al., 2012a), Communication becomes a special Cloud Computing resource because commercial Cloud services are employed inevitably through Internet/Ethernet. As such, the *Ethernet I/O Index* is usually pre-supplied as a service-level agreement (SLA) by service providers. In practice, the scope and level of communication have been frequently emphasized in the performance evaluation studies. Therefore, we can summarize two practical factors: The factor *Communication Scope* considers intra-Cloud and wide-area data transferring respectively (Li et al., 2010), while the *Communication Level* distinguishes between IP-level and MPI-message-level networking (He et al., 2010).

Computation

When evaluating PaaS, the Computation resource is usually regarded as a black box (Iosup, Yigitbasi & Epema, 2010). Whereas, for IaaS, the practices of Computation evaluation of Cloud services have taken into account *Core Number* (Bientinesi, Iakymchuk & Napper, 2010), *Elastic Compute Unit (ECU) Number*, *Thread Number* (Baun & Kunze, 2009), and a set of CPU characteristics. Note that, compared to physical CPU core and thread, ECU is a logical concept introduced by Amazon, which is defined as the CPU power of a 1.0-1.2 GHz 2007 Opteron or Xeon processor (Ostermann et al., 2009). When it comes to CPU characteristics, the *Architecture* (e.g. 32 bit vs. 64 bit) (Iosup, Yigitbasi & Epema, 2010) and *Brand* (e.g. AMD Opteron vs. Intel Xeon) (Napper & Bientinesi, 2009) have been respectively considered in evaluation experiments. Processors with the same brand can be further distinguished between different *CPU Models* (e.g. Intel Xeon E5430 vs. Intel Xeon X5550) (Bientinesi, Iakymchuk & Napper, 2010). In particular, *CPU Frequency* appears also as an SLA of Cloud computation resources.

Memory (Cache)

Since Memory/Cache could closely work with the Computation and Storage resources in computing jobs, it is hard to exactly distinguish the affect to performance brought by Memory/Cache. Therefore, not many dedicated Cloud memory/cache evaluation studies can be found from the literature. In addition to the SLA *Memory Size*, interestingly, *Physical Location* and *Size* of cache (e.g. L1=64KB vs. L2=1MB in Amazon m1.* instances) (Ostermann et al., 2009) have attracted attentions when analyzing the memory hierarchy. However, in Ostermann et al. (2009), different values of these factors were actually revealed by performance evaluation rather than used for experimental design.

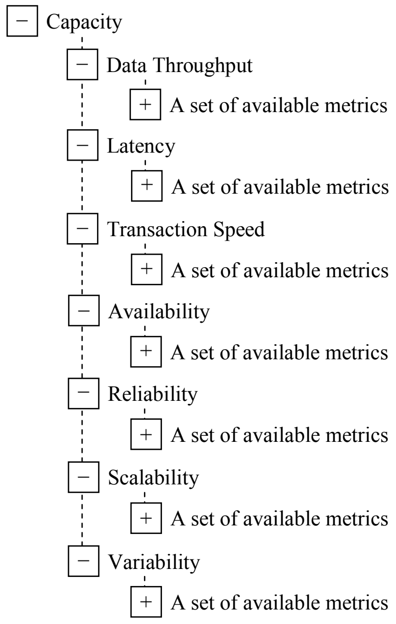
Storage

As mentioned in Li et al. (2012a), Storage can be either the only functionality or a component functionality of a Cloud service, for example Amazon S3 vs. EC2. Therefore, it can be often seen that disk-related storage evaluation also adopted experimental factors of evaluating other relevant resources like VM instances (cf. Subsection VM Instance). Similarly, the predefined *Storage Size* acts as an SLA, while a dedicated factor of evaluating Storage is the *Geographical Location*. Different geographical locations of Storage resources can result either from different service data centers (e.g. S3 vs. S3-Europe) (Palankar et al., 2008) or from different storing mechanisms (e.g. local disk vs. remote NFS drive) (Sobel et al., 2008). In addition, although not all of the public Cloud providers specified the definitions, the Storage resource has been distinguished among three types of offers: Blob, Table and Queue (Li et al., 2010). Note that different *Storage Types* correspond to different sets of data-access activities, as described in Li et al. (2012b).

VM Instance

VM Instance is one of the most popular computing resource styles in the commercial Cloud service market. The widely considered factors in current VM Instance evaluation experiments are *Geographical Location*, *Instance Number*, and *VM Type* (Bientinesi, Iakymchuk & Napper, 2010; Hill & Humphrey, 2009; Hill et al., 2010; Iosup, Yigitbasi & Epema, 2010; Li et al., 2010; Ostermann et al., 2009; Stantchev, 2009). The *VM Type* of a particular instance naturally reflects its corresponding provider, as demonstrated in (Li et al., 2010). Moreover, although not common, the *OS Brand* (e.g. Linux vs. Windows) (Li et al., 2010) and *Physical Location* (Dejun, Pierre & Chi, 2009) also emerged as experimental factors in some evaluation studies. Note that the physical location of a VM instance indicates the instance's un-virtualized environment, which is not controllable by evaluators in evaluation experiments (Dejun, Pierre & Chi, 2009). In particular, recall that a VM Instance integrates above four basic types of computing resources. We can therefore find that some factors of evaluating previous resources were also used in the evaluation of VM Instances, for example the *CPU Architecture* and *Core Number* (Bientinesi, Iakymchuk & Napper, 2010; Ostermann et al., 2009).

Figure 6. The capacity factors for experimental design.



Capacity
Data Throughput
A set of available metrics
Latency
A set of available metrics
Transaction Speed
A set of available metrics
Availability
A set of available metrics
Reliability
A set of available metrics
Scalability
A set of available metrics
Variability
A set of available metrics

Capacity Factors

As discussed about the generic reality of performance evaluation (cf. Figure 3), it is clear that the capacities of a Cloud computing resource are intangible until they are measured. Meanwhile, the measurement has to be realized by using measurable and quantitative metrics (Le Boudec, 2011). Therefore, we can treat the values of relevant metrics as tangible representations of the evaluated capacities. Moreover, a particular capacity of a commercial Cloud service may be reflected by a set of relevant metrics, and each metric provides a different lens into the capacity as a whole (Fortier & Michel, 2003). For example, Benchmark Transactional Job Delay (Luckow & Jha, 2010) and Benchmark Delay (Juve et al., 2009) are both Latency metrics: the former is from the individual perspective, while the latter from the global perspective. As such, we further regard relevant metrics as possible output process factors (Antony, 2003) when measuring a particular Cloud service capacity, and every single output process factor can be used as a candidate response (Antony, 2003) in the experimental design. Since we have clarified seven different Cloud service capacities (Li et al., 2012a), i.e. Data Throughput, Latency, Transaction Speed, Availability, Reliability, Scalability, and Variability, the possible capacity factors (metrics) can be correspondingly categorized as shown in Figure 6. Due to the limit of space, it is impossible and unnecessary to exhaustively list all the metrics in this paper. Some sample metrics for evaluating *Memory (Cache)* are shown in Table 1. In fact, the de facto metrics for performance evaluation of commercial Cloud services have been collected and summarized in our previous work (Li et al., 2012b).

Table 1. Sample metrics for evaluating Memory (Cache) (Li et al., 2012b).

Capacity	**Metrics**	**Benchmark**
Transaction Speed	Random Memory Update Rate (MUP/s, GUP/s)	HPCC: RandomAccess
Latency	Mean Hit Time (s)	Land Elevation Change App
	Memcache Get/Put/Response Time (ms)	Operate 1Byte/ 1MB data
Data Throughput	Memory bit/Byte Speed (MB/s, GM/s)	CacheBench
		HPCC: PTRANS
		HPCC: STREAM

THREE TYPICAL SCENARIOS OF APPLYING DOE TECHNIQUES

Recall that relevant factors play a prerequisite role in designing evaluation experiments. Benefitting from the pre-established factor framework, evaluators may employ suitable DOE techniques for experimental design and analysis when evaluating Cloud services. Here we use two cases to preliminarily explain how to apply DOE to Cloud services evaluation. First, we replicate a study of Google AppEngine evaluation (Iosup, Yigitbasi & Epema, 2010) to demonstrate the usage of two techniques for determining sample size and analyzing variance. Second, we adopt an existing study of Amazon EC2 disk I/O evaluation to illustrate the application of 2^3 Factorial Design.

Sample Size Determination

Determining sample size is critical in any experimental design problem (Montgomery, 2009). Unfortunately, most of the existing Cloud services evaluation studies did not justify how the number of experimental replicates was decided. In fact, the statistical approach in DOE can be used to facilitate sample size determination. We demonstrate this by replicating a straightforward study of Google AppEngine evaluation.

The overall objective of the original study (Iosup, Yigitbasi & Epema, 2010) is to evaluate the computation performance of the Google AppEngine Python runtime. In particular, the study investigated how variable the Google AppEngine's performance was during different time periods. Therefore, by exploring the experimental factor framework, we can identify that the only factor considered in the original evaluation work is *Timing* (cf. Figure 4). Although there are other potentially useful factors like *Workload Size*, we deliberately ignored them to make our study comparable with the original one. Similarly, following the original study, we directly selected the metric *Benchmark Runtime* to measure the computation performance of Google AppEngine. With regard to the benchmark, we coded a Python program to recursively calculate the 27th Fibonacci number, as implemented in (Iosup, Yigitbasi & Epema, 2010).

Furthermore, to make this demonstration simple and clear, we decided to choose seven consecutive days as the experimental period. In other words, we treated different dates as different levels of the factor *Timing*. As such, the evaluation requirement can be formally hypothesized as Equation (1) to test the equality of seven computation performance means, where μ_i refers to the Fibonacci calculation mean in the *i*th day.

$$\begin{cases} H_0 : \mu_1 = \mu_2 = ... = \mu_7 \\ H_1 : \mu_i \neq \mu_j \ \textit{for at least one pair(i, j)} \\ (i \neq j \ and \ i, j = 1,2,..7) \end{cases} \quad (1)$$

As suggested in (Montgomery, 2009), we performed a set of random and pilot Fibonacci calculations within Google AppEngine to estimate its performance standard deviation, and then used the Operating Characteristic (OC) Curves to find a suitable number of replicates for everyday. An OC curve is essentially "a plot of the type II error probability of a statistical test for a particular sample size versus a parameter that reflects the extent to which the null hypothesis is false" (Montgomery, 2009). The type II error is the failure to reject a false null hypothesis, and thus the type II error probability can be defined as Equation (2). The null hypothesis H_0 in this case is that the seven performance means are equal to each other.

$$\begin{aligned} \beta &= P(\textit{type II error}) \\ &= P(\textit{Fail to reject } H_0 \mid H_0 \textit{ is false}) \end{aligned} \quad (2)$$

Recall that the value of standard deviation must be specified before using OC curves (Montgomery, 2009). Given the estimated standard deviation is 34*ms* by pilot Fibonacci calculations, we finally decided to run 123 replicates per day (or replicate once per 720 seconds) to satisfy a target power ($1 - \beta$) of at least 0.9. To save space, here we replace the OC curve illustration with the Minitab output of finding sample sizes, as shown in Figure 7.

Figure 7. Sample size for performing Google AppEngine evaluation (by Minitab).

Power and Sample Size

```
One-way ANOVA

Alpha = 0.01  Assumed standard deviation = 34

Factors: 1  Number of levels: 7


   Maximum  Sample  Target
Difference    Size   Power  Actual Power
        21     123     0.9      0.900852

The sample size is for each level.
```

Analysis of Variance

We continue the previous case study to demonstrate the one-factor DOE technique for analyzing the variance of Google AppEngine performance. Given the determined sample size, the evaluation experiments were correspondingly deployed and implemented. Several typical indices of the experimental result are shown in Table 2, while the specific result can be visualized as shown in Figure 8.

Table 2. Experimental result of the 27th Fibonacci calculation with Google AppEngine Python runtime

Date	**Average**	**Minimum**	**Maximum**	**Standard Deviation**
Sept. 1	197.97ms	152.36ms	329.62ms	35.07ms
Sept. 2	194.65ms	151.65ms	311.38ms	30.43ms
Sept. 3	197.83ms	150.57ms	308.64ms	28.81ms
Sept. 4	199.95ms	151.13ms	329.29ms	34.82ms
Sept. 5	208.44ms	155.14ms	318.45ms	38.38ms
Sept. 6	226.39ms	153.91ms	313.66ms	45.48ms
Sept. 7	220.79ms	148.15ms	366.49ms	44.18ms
Total	206.58ms	148.15ms	366.49ms	38.84ms

Figure 8. Google AppEngine computation performance during seven days.

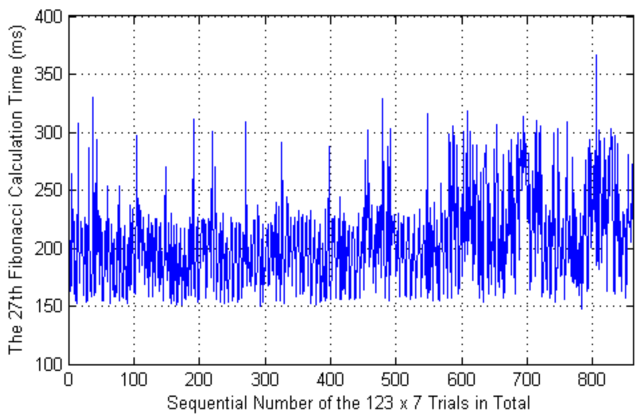


Table 2 and Figure 8 intuitively show that Google AppEngine takes 200 ± 50*ms* in general to calculate the 27th Fibonacci number. Moreover, the computation performance peak of Google AppEngine is relatively stable (around 150*ms* for the 27th Fibonacci calculation) everyday, while the worst-case calculation time varies largely. However, such observations do not advise whether or not we can expect a stable mean of the computation performance of Google AppEngine. Therefore, we employed Tukey's Test (Montgomery, 2009) to perform all pair-wise mean comparisons. Given the significance level α, the procedure of Tukey's Test constructs confidence intervals on the differences in all pairs of means, and the simultaneous confidence level is 100(1 − α) percent for those intervals. In this case, we directly show the output of Tukey's Test by using Minitab (cf. Figure 9).

Figure 9. Grouping information in Tukey's analysis result (by Minitab).

```
One-way ANOVA: Runtime versus Date

Grouping Information Using Tukey Method

Date        N    Mean  Grouping
Sept. 6   123  226.39  A
Sept. 7   123  220.79  A B
Sept. 5   123  208.44    B C
Sept. 4   123  199.95      C
Sept. 1   123  197.97      C
Sept. 3   123  197.83      C
Sept. 2   123  194.65      C

Means that do not share a letter are significantly different.
```

It can be seen that the seven days' Fibonacci calculation means are divided into three groups, which statistically confirms that it is impossible to achieve a stable performance when using Google AppEngine at different period of time. However, interestingly, Group B can be viewed as a linkage between Group A and C. We thus claim that, although not absolutely stable, the performance mean of Google AppEngine may fluctuate mildly.

2^3 Factorial Design

In general cases of Cloud services evaluation, one experiment could take into account more than one factor related to both the service to be evaluated and the workload. Suppose there is a requirement of evaluating Amazon EC2 with respect to its disk I/O. Given the factor framework proposed in this paper, we can quickly and conveniently lookup and choose experimental factors according to the evaluation requirement. To simplify the demonstration, here we constrain the terminal to be clients, while only consider the direction of disk I/O and data size to be read/write in workload factors, and only consider the EC2 VM type in computing resource factors. As for the capacity factors, we can employ multiple suitable metrics in this evaluation, for example disk I/O latency and data throughput. However, since only one metric should be determined as the response in an experimental design (Antony, 2003), we choose the disk data throughput in this case. Thus, we have identified *active direction*, *object size* and *VM type* as factors, while *data throughput* as response in the framework for designing experiments. In particular, we use two-level settings for the three factors: the value of *active direction* can be Write or Read; *object size* can be Char or Block; and *VM type* only covers M1.small and M1.large. In addition, we use "MB/s" as the unit of *data throughput*.

Since only three factors are considered, we can simply adopt the most straightforward design technique, namely Full-factorial Design (Antony, 2003), for this demonstration. This design technique adjusts one factor at a time, which results in an experimental matrix comprising eight trials, as shown in Matrix (1). For conciseness, we further assign aliases to those experimental factors, as listed below. Note that the sequence of the experimental trials has been randomized to reduce possible noises or biases (Antony, 2003) during the designing process.

- **A:** Activity Direction (Write vs. Read).
- **B:** Object Size (Char vs. Block).
- **C:** VM Type (M1.small vs. M1.large).
- **Response:** Data Throughput (MB/s).

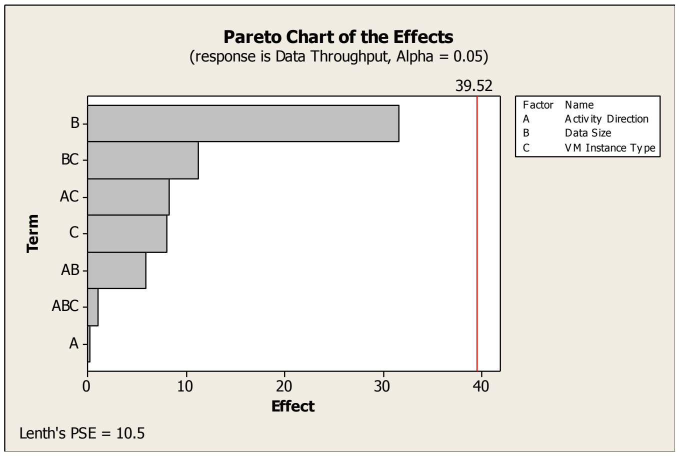
$$\begin{bmatrix} trial & A & B & C & Response \\ 1 & Write & Block & M1.small & ? \\ 2 & Read & Char & M1.large & ? \\ 3 & Write & Char & M1.small & ? \\ 4 & Read & Char & M1.small & ? \\ 5 & Read & Block & M1.large & ? \\ 6 & Read & Block & M1.small & ? \\ 7 & Write & Char & M1.large & ? \\ 8 & Write & Block & M1.large & ? \end{bmatrix} \tag{1}$$

Following the experimental matrix, we can implement evaluation experiments trial by trial, and fill the Response column with experimental results. For our convenience, here we directly employ the evaluation results reported in (Iosup et al., 2011), as listed in Matrix (2).

$$\begin{bmatrix} trial & A & B & C & Response \\ 1 & Write & Block & M1.small & 73.5MB/s \\ 2 & Read & Char & M1.large & 50.9MB/s \\ 3 & Write & Char & M1.small & 25.9MB/s \\ 4 & Read & Char & M1.small & 22.3MB/s \\ 5 & Read & Block & M1.large & 64.3MB/s \\ 6 & Read & Block & M1.small & 60.2MB/s \\ 7 & Write & Char & M1.large & 35.9MB/s \\ 8 & Write & Block & M1.large & 63.2MB/s \end{bmatrix} \tag{2}$$

Finally, different analytical techniques can be employed to reveal more comprehensive meanings of experimental results (Antony, 2003) for commercial Cloud services. For example, in this case, we can further investigate the significances of these factors to analyze their different influences on the disk I/O performance. In detail, by setting the significance level α as 0.05 (Jackson, 2011), we draw a Pareto plot to detect the factor and interaction effects that are important to the process of reading/writing data from/to EC2 disks, as shown in Figure 10.

Figure 10. The Pareto plot of factor effects.



Given a particular significance level, Pareto plot displays a red reference line besides the effect values. Any effect that extends past the reference line is potentially important (Antony, 2003). In Figure 10, none of the factor or interaction effects is beyond the reference line, which implies that none of the factors or interactions significantly influences the EC2 disk I/O performance. Therefore, we can claim that EC2 disk I/O is statistically stable with respect to those three factors. However, Factor B (Data Size to be read/written) has relatively significant influence on the performance of EC2 disk I/O. Since the throughput of small-size data (Char) is much lower than that of large-size data (Block), we can conclude that there is a bottleneck of transaction overhead when reading/writing small size of data. On the contrary, there is little I/O performance effect when switching activity directions, which means the disk I/O of EC2 is particularly stable no matter reading or writing the same size of data.

In particular, through the above demonstrations, we show that the proposed factor framework offers a concrete and rational foundation for implementing performance evaluation of commercial Cloud services. When evaluating Cloud services, there is no doubt that the techniques of experimental design and analysis can still be applied by using intuitively selected factors. Nevertheless, by referring to the existing evaluation experiences, evaluators can conveniently identify suitable experimental factors while excluding the others, which essentially suggest a systematic rather than ad hoc decision making process.

CONCLUSIONS AND FUTURE WORK

Cloud Computing has attracted a tremendous amount of attention from both customers and providers in the current computing industry, which leads to a competitive market of commercial Cloud services. As a result, different Cloud infrastructures and services may be offered with different terminology, definitions, and goals (Prodan & Ostermann, 2009). On the one hand, different Cloud providers have their own idiosyncratic characteristics when developing services (Li et al., 2010). On the other hand, even the same provider can supply different Cloud services with comparable functionalities for different purposes. For example, Amazon has provided several options of storage service, such as EC2, EBS, and S3 (Chiu & Agrawal, 2010). Consequently, performance evaluation of candidate services would be crucial and beneficial for many purposes ranging from cost-benefit analysis to service improvement (Li et al., 2010).

When it comes to performance evaluation of a computing system, proper experimental design and analysis should be performed with respect to a set of factors that may influence the system's performance (Jain, 1991; Montgomery, 2009). In the Cloud Computing domain, however, most of the evaluators intuitively employed experimental factors and implemented ad hoc experiments with few comprehensive analyses for evaluating performance of commercial Cloud services. Thus, we suggest applying DOE to systematically instruct Cloud services evaluation. In particular, considering factor identification plays a prerequisite role in experimental design, we collected experimental factors that people currently took into account in Cloud services performance evaluation, and arranged them into a tree-structured framework.

The main contribution of this work is twofold. On the one hand, the established factor framework supplies a dictionary-like approach to selecting experimental factors for Cloud services performance evaluation. Benefitting from the framework, evaluators can identify necessary factors in a concrete space instead of on the fly. Note that the experimental factor framework is supposed to supplement, but not replace, the expert judgment for experimental factor identification, which would be particularly helpful for Cloud services evaluation when there is a lack of a bunch of experts. On the other hand, based on the experimental factor framework, we initially suggested a series of DOE techniques for sample size determination, single-factor experimental analysis, and three-factor experimental design. Thus, new evaluators can conveniently refer to our work and adapt these DOE techniques to their own evaluation scenarios.

The future work of this research will be unfolded along two directions. First, we will gradually collect feedback from external experts to supplement this factor framework. As explained previously, Cloud Computing is still maturing and relatively chaotic (Stokes, 2011), it is therefore impossible to exhaustively identify the relevant experimental factors all at once. Through smooth expansion, we can make this factor framework increasingly suit the more general area of evaluation of Cloud Computing. Second, given the currently available experimental factors, we plan to further introduce and adapt suitable DOE techniques to evaluating commercial Cloud services. As demonstrated in Section 4, the adapted DOE techniques together with the experimental factor framework would effectively support systematic implementations of Cloud services evaluation.

ACKNOWLEDGEMENT

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

REFERENCE

Antony, J. (2003). *Design of Experiments for Engineers and Scientists*. Burlington, MA: Butterworth-Heinemann.

Baun, C., & Kunze, M. (2009). *Performance Measurement of a Private Cloud in the OpenCirrus^{TM} Testbed*. Paper presented at the Proceedings of the 4th Workshop on Virtualization in High-Performance Cloud Computing (VHPC 2009) in conjunction with the 15th International European Conference on Parallel and Distributed Computing (Euro-Par 2009), Delft, The Netherlands.

Bientinesi, P., Iakymchuk, R., & Napper, J. (2010) HPC on competitive Cloud resources. In B. Furht, & A. Escalante (Eds.), *Handbook of Cloud Computing* (pp. 493-516). New York: Springer-Verlag.

Binnig, C., Kossmann, D., Kraska, T., & Loesing, S. (2009). *How is the Weather Tomorrow? Towards a Benchmark for the Cloud*. Paper presented at the Proceedings of the 2nd International Workshop on Testing Database Systems (DBTest 2009) in conjunction with ACM SIGMOD/PODPS International Conference on Management of Data (SIGMOD/PODS 2009), Providence, USA.

Chiu, D., & Agrawal, G. (2010). *Evaluating Caching and Storage Options on the Amazon Web Services Cloud*. Paper presented at the Proceedings of the 11th ACM/IEEE International Conference on Grid Computing (Grid 2010), Brussels, Belgium.

Deelman, E., Singh, G., Livny, M., Berriman, B., & Good, J. (2008). *The Cost of Doing Science on the Cloud: The Montage Example*. Paper presented at the Proceedings of the 2008

International Conference on High Performance Computing, Networking, Storage and Analysis (SC 2008), Austin, Texas, USA.

Dejun, J., Pierre, G., & Chi, C.-H. (2009). *EC2 Performance Analysis for Resource Provisioning of Service-Oriented Applications*. Paper presented at the Proceedings of the 2009 International Conference on Service-Oriented Computing (ICSOC/ServiceWave 2009), Stockholm, Sweden.

Dybå, T., Kitchenham, B. A., & Jørgensen, M. (2005). Evidence-based software engineering for practitioners. *IEEE Software, 22*(1), 58-65.

Fortier, P. J., & Michel, H. E. (2003). *Computer Systems Performance Evaluation and Prediction*. Burlington, MA: Digital Press.

Garfinkel, S. (2007a). *An evaluation of Amazon's grid computing services: EC2, S3 and SQS* (Tech. Rep. TR-08-07). Cambridge, MA: Harvard University, School of Engineering and Applied Sciences.

Garfinkel, S. (2007b). Commodity grid computing with Amazon's S3 and EC2. *;Login, 32*(1), 7-13.

He, Q., Zhou, S., Kobler, B., Duffy, D., & McGlynn, T. (2010). *Case Study for Running HPC Applications in Public Clouds*. Paper presented at the Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC 2010), Chicago, Illinois, USA.

Hill, Z., & Humphrey, M. (2009). *A Quantitative Analysis of High Performance Computing with Amazon's EC2 Infrastructure: The Death of the Local Cluster?* Paper presented at the Proceedings of the 10th IEEE/ACM International Conference on Grid Computing (Grid 2009), Banff, Alberta, Canada.

Hill, Z., Li, J., Mao, M., Ruiz-Alvarez, A., & Humphrey, M. (2010). *Early Observations on the Performance of Windows Azure*. Paper presented at the Proceedings of the 19th ACM International Symposium High Performance Distributed Computing (HPDC 2010), Chicago, Illinois, USA.

Iosup, A., Ostermann, S., Yigitbasi, N., Prodan, R., Fahringer, T., & Epema, D. H. J. (2011). Performance analysis of Cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems, 22*(6), 931-945.

Iosup, A., Yigitbasi, N., & Epema, D. (2010). *On the performance variability of production Cloud services* (Tech. Rep. PDS-2010-002). Netherlands: Delft University of Technology.

Jackson, S. L. (2011). *Research Methods and Statistics: A Critical Thinking Approach*, 4th ed. Belmont, CA: Wadsworth Publishing.

Jain, R. K. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. New York, NY: Wiley Computer Publishing, John Wiley & Sons, Inc.

Juve, G., Deelman, E., Vahi, K., Mehta, G., Berriman, B., Berman, B. P., & Maechling, P. (2009). *Scientific Workflow Applications on Amazon EC2*. Paper presented at the Proceedings of the Workshop on Cloud-based Services and Applications in conjunction with the 5th IEEE International Conference on e-Science (e-Science 2009), Oxford, UK.

Le Boudec, J.-Y. (2011). *Performance Evaluation of Computer and Communication Systems*. Lausanne, Switzerland: EFPL Press.

Li, Z., O'Brien, L., Cai, R., & Zhang, H. (2012a). *Towards a Taxonomy of Performance Evaluation of Commercial Cloud Services*. Paper presented at the Proceedings of the 5th International Conference on Cloud Computing (IEEE CLOUD 2012), Honolulu, Hawaii, USA.

Li, Z., O'Brien, L., Zhang, H., & Cai, R. (2012b). *On a Catalogue of Metrics for Evaluating Commercial Cloud Services*. Paper presented at the Proceedings of the 13th ACM/IEEE International Conference on Grid Computing (Grid 2012), Beijing, China.

Li, Z., O'Brien, L., Zhang, H., & Cai, R. (2012c). *A Factor Framework for Experimental Design for Performance Evaluation of Commercial Cloud Services*. Paper presented at the Proceedings of the 4th International Conference on Cloud Computing Technology and Science (CloudCom 2012), Taipei, Taiwan.

Li, Z., O'Brien, L., Zhang, H., & Cai, R. (in press a). On a taxonomy-based conceptual model of performance evaluation of Infrastructure as a Service. *IEEE Transactions on Service Computing*.

Li, Z., Zhang, H., O'Brien, L., Cai, R., & Flint, S. (in press b) On evaluating commercial Cloud services: A systematic review. *Journal of Systems and Software*.

Li, A., Yang, X., Kandula, S., & Zhang, M. (2010). *CloudCmp: Comparing Public Cloud Providers*. Paper presented at the Proceedings of the 10th Annual Conference on Internet Measurement (IMC 2010), Melbourne, Australia.

Luckow, A., & Jha, S. (2010). *Abstractions for Loosely-coupled and Ensemble-based Simulations on Azure*. Paper presented at the Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2010), Indianapolis, USA.

Mellor, S. J., Clark, A. N., & Futagami, T. (2003). Model-driven development - guest editor's introduction. *IEEE Software, 20*(5), 14-18.

Montgomery, D. C. (2009). *Design and Analysis of Experiments*, 7th ed. Hoboken, NJ: John Wiley & Sons, Inc.

Napper, J., & Bientinesi, P. (2009). *Can Cloud Computing Reach the Top500?* Paper presented at the Proceedings of the Combined Workshops on UnConventional High Performance Computing Workshop plus Memory Access Workshop (UCHPC-MAW 2009), Ischia, Italy.

Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., & Epema, D. H. J. (2009). *A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing*. Paper presented at the Proceedings of the 1st International Conference on Cloud Computing (CloudComp 2009), Munich, Germany.

Palankar, M. R., Iamnitchi, A., Ripeanu, M., & Garfinkel, S. (2008). *Amazon S3 for Science Grids: A Viable Solution?* Paper presented at the Proceedings of the 2008 International Workshop on Data-aware Distributed Computing (DADC 2008), Boston, MA, USA.

Prodan, R., & Ostermann, S. (2009). *A Survey and Taxonomy of Infrastructure as a Service and Web Hosting Cloud Providers*. Paper presented at the Proceedings of the 10th IEEE/ACM International Conference on Grid Computing (Grid 2009), Banff, Alberta, Canada.

Sobel, W., Subramanyam, S., Sucharitakul, A., Nguyen, J., Wong, H., Klepchukov, A., Patil, S., Fox, A., & Patterson, D. (2008). *Cloudstone: Multi-platform, Multi-language Benchmark and Measurement Tools for Web 2.0*. Paper presented at the Proceedings of the 1st Workshop on Cloud Computing and Its Applications (CCA 2008), Chicago, IL.

Stantchev, V. (2009). *Performance Evaluation of Cloud Computing Offerings*. Paper presented at the Proceedings of the 3rd International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2009), Sliema, Malta.

Stokes, J. (2011). The PC is order, the Cloud is chaos. Retrieved March 23, 2013, from http://www.wired.com/cloudline/2011/12/the-pc-is-order/.

Wang, L., Laszewski, G., Chen, D., Tao, J., & Kunze, M. (2010a). Provide virtual machine information for Grid computing. *IEEE Transactions on Systems, Man, and Cybernetics, Part A 40*(6), 1362-1374.

Wang, L., Laszewski, G., Kunze, M., Tao, J., & Dayal, J. (2010b). Provide virtual distributed environments for Grid computing on demand. *Advances in Engineering Software 41*(2), 213-219.

Wang, L., Laszewski, G., Tao, J., & Kunze, M. (2010c). Virtual Data System on distributed virtual machines in computational grids. *International Journal of Ad Hoc and Ubiquitous Computing 6*(4), 194-204.

Wang, L., Chen, D., & Huang, F. (2011). Virtual workflow system for distributed collaborative scientific applications on Grids. *Computers & Electrical Engineering 37*(3), 300-310.