**Boosting Metrics: Measuring Cloud Services from the Holistic Perspective**

Zheng Li¹, Liam O’Brien², and He Zhang³

¹School of Computer Science, Australian National University and NICTA, Australia
²ICT Innovation and Services, Geoscience Australia, Canberra, Australia
³Software Institute, Nanjing University, Jiangsu, China
zheng.li@anu.edu.au
liamob99@hotmail.com
dr.hezhang@gmail.com

**Abstract**

Studies have shown that Cloud services evaluation would be crucial and beneficial for both service customers and providers, and metrics would play a vital role in any evaluation implementation. Considering the numerous and various aspects of Cloud services, a frequent suggestion is to perform evaluation from a holistic view. The currently normal strategy of holistic evaluation is to use a set of metrics along with a suite of benchmarks to conduct separated experiments. Given the separated, diverse, and even possibly conflicting measurement criteria, it could be still hard for customers with such evaluation reports to understand an evaluated Cloud service from a global perspective. Inspired by the boosting approaches to machine learning, we proposed the concept *Boosting Metrics* to represent all the potential approaches that are able to deliver summary measurement of Cloud services. Essentially, the idea of boosting metrics is to holistically measure Cloud services with concern of service properties, which supplements the strategy of employing benchmark suites that is to holistically evaluate Cloud services with concern of different workloads. This paper introduces two types of preliminary approaches, and unifies a set of sophisticated measurements into the notion of boosting metrics. In particular, we show that boosting metrics can be used as a summary *Response* for applying experimental design to Cloud services evaluation. Although the concept *Boosting Metrics* was refined based on our work in the Cloud Computing domain, we believe it can be easily adapted to the evaluation work of other computing paradigms.

**Keywords:** Cloud Computing; Cloud Services Evaluation; Measurement Criteria; Boosting Metric; Experimental Design

---

1. **INTRODUCTION**

As Cloud Computing becomes one of the most promising computing paradigms in industry (Buyya et al., 2009), numerous vendors have started to supply public Cloud infrastructures and services with different terminologies, qualities, and cost models (Prodan and Ostermann, 2009). Since most providers do not reveal details about their infrastructures (Brooks, 2010), customers have little knowledge and control over the precise nature of public Cloud services even in the “locked down” environment (Sobel et al., 2008). As such, Cloud services evaluation 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.

When it comes to evaluating a Cloud service, suitable measurement criteria or metrics must be chosen. In fact, according to the rich research in the evaluation of traditional computer systems, the selection of metrics plays an essential role in evaluation implementations (Li et al., 2012b). Particularly, it is often useful and significant to evaluate Cloud services from a holistic view (Iosup et al., 2011; Rabl et al., 2010), and using single measurement index would be helpful and convenient for comparing alternatives and drawing conclusions (Islam et al., 2012). More importantly, a single index of an overall measurement can play a summary *Response* role in experimental design and analysis (Montgomery, 2009) for evaluating Cloud services.

Unfortunately, given the numerous and various aspects of Cloud services, the existing evaluation studies usually adopted multiple metrics to measure Cloud services feature by feature. To perform a holistic evaluation, the normal strategy is to employ a benchmark suite to cover and test various aspects of Cloud services from the perspective of workload. Through reviewing the relevant studies (Li et al., 2013c), we found that most evaluators intended to report individual benchmarking results with a lack of visibly integrated measurements. As a result, customers with such evaluation reports could have to further summarize various evaluation results by themselves. Although it is possible and sometimes flexible for customers to balance tradeoffs in employing a Cloud service, a single and overall index can significantly facilitate customers’ decision making by supplementing separated benchmarking results.
Therefore, it is valuable and necessary to investigate approaches to summary measurement of Cloud services. Moreover, considering different evaluation circumstances, we may expect different measurement approaches ranging from integrating homogeneous benchmarking results (Evangelinos and Hill, 2008) to catering for conflicting criteria (Zhang et al., 2012), e.g., the combination of performance and cost. Inspired by the boosting approaches to machine learning that combine weak rules into a single more accurate one (Schapire, 2002), we proposed the concept Boosting Metric to represent all the potential approaches that are able to integrate a set of separated measures, as defined below.

A boosting metric is a secondary measurement criterion by manipulating the primary metrics that directly measure individual Cloud service aspects.

This paper has been significantly extended from our previous work (Li et al., 2013a) that only discussed preliminary approaches integrating a suite of benchmarking results. Firstly, we have essentially expanded the concept of Boosting Metric to cover the integration of not only homogeneous benchmarking results but also diverse and even conflicting measurement criteria. Secondly, in addition to suggesting preliminary approaches, we also identified the same backend essence of a set of sophisticated Cloud service measurements, and unified them into the notion of Boosting Metric. Thirdly, to perform convincing demonstration, we followed a formal Cloud evaluation methodology and updated the case study about how the boosting metrics can help facilitate experimental design and analysis. In summary, the contributions of this paper are threefold, as listed below.

- The proposed concept Boosting Metric essentially suggests holistic measurement of Cloud services from the output (service property) perspective, which supplements the input (workload) perspective of employing benchmark suites. Note that, although this idea was extracted from our work in the Cloud computing domain, we believe it can be easily adapted to the evaluation work of other computing paradigms.
- The summarized preliminary and sophisticated boosting metrics can be directly used in the future Cloud services evaluation work. Evaluators can also use them to inspire and develop new measurement approaches.
- The case study can be viewed as a template for the usage of boosting metrics in a rigorous procedure of Cloud services evaluation. Meanwhile, this case study also acts as a concrete application for validating the employed Cloud evaluation methodology.

The remainder of this paper is organized as follows. Section 2 briefly summarizes the related work about holistic measurement of Cloud services. By employing a simple application scenario of using benchmark suites to evaluate Cloud services, Section 3 introduces two types of preliminary approaches to boosting metrics. Section 4 unifies four different and sophisticated Cloud service measurements into the notion of Boosting Metric. Section 5 employs a case study of evaluating Amazon EC2 to demonstrate how a boosting metric can be used in evaluation and how it can help analyze experimental results. Conclusions and some future work are discussed in Section 6.

2. RELATED WORK

Since the selection of measurement criteria or metrics plays an essential role in evaluation implementations (Obaidat and Boudriga, 2009), a set of suitable metrics must be chosen when evaluating Cloud services. Li et al. (2012b) have investigated the existing relevant studies and established a metric catalogue to facilitate metrics selection for Cloud services evaluation. In essence, the metric catalogue accommodates the de facto evaluation metrics in the Cloud computing domain, and the metrics have been categorized according to different Cloud service features. As such, evaluators can use a particular feature as the retrieval key to quickly locate candidate evaluation metrics in this catalogue. However, such a catalogue also shows that Cloud services are usually measured feature by feature. There seems a lack of approaches to summary measurement of multiple and even overall Cloud service features. For example, Cloud elasticity is related not only to the resource scaling time but also to the resource charging basis (Islam et al., 2012); consequently, it has become a challenge to explicitly quantify the amount of elasticity of a Cloud service (Li et al., 2012b, 2013c).

Therefore, to cover and test various service aspects from a holistic view, the current practitioners normally suggest employing benchmark suites for Cloud services evaluation (Iosup et al., 2011; Rabl et al., 2010). For example, the kernel benchmarks in NPB have been used to reveal different micro features of Amazon EC2 like computation, communication and storage respectively (Akioka and Muraoka, 2010); while six scale-out workloads are collected to simulate different macro application scenarios in today’s Cloud infrastructure (Ferdman et al., 2012). In particular, for verifying scientific computing in the Cloud, HPCC seems a popular benchmark suite to show high performance computing capabilities of Cloud services (Jackson et al., 2010; Ostermann et al., 2006). In addition to those predefined benchmark suites, the evaluator-selected application sets were also commonly adopted to evaluate Cloud services (Dejun et al., 2009; Jackson et al., 2010). Essentially, each application set here can be viewed as an individual benchmark suite.
However, even if benchmark suites were adopted to evaluate Cloud services, most evaluators intended to report individual benchmarking results without visibly integrated measurements (Li et al., 2013c). In fact, although it is often useful and convenient to compare alternatives by using a single index (Islam et al., 2012), the overall measurement of Cloud services would be significantly challenging because of the diverse and even conflicting criteria (Zhang et al., 2012).

3. PRELIMINARY BOOSTING METRICS FOR CLOUD SERVICES EVALUATION

As mentioned previously, we borrowed the “boosting” idea from the machine learning field to our Cloud services evaluation work. In machine learning, boosting refers to the method of producing a more accurate prediction rule by combining a set of rough and less accurate rules of thumb (Schapire, 2002). By analogy, in Cloud services evaluation, we treat “boosting” as integrating a set of local measures into a single global one, namely a boosting metric, to reflect the overall measurement of a Cloud service. A simple and typical application scenario is the employment of benchmark suites. When measuring different Cloud service aspects, a benchmark suite may adopt homogeneous primary metrics (e.g. NPB (NASA, 2012)) or inhomogeneous primary metrics (e.g. HPCC (Lusczek et al., 2006)). Correspondingly, here we show two types of preliminary boosting metrics, as described in the following two subsections respectively.

3.1 MEAN AS A BOOSTING METRIC FROM A SPATIAL PERSPECTIVE

In a benchmark suite, a set of different benchmarks are generally expected to be able to reflect different aspects of a Cloud service in an evaluation. If we imagine every single service aspect as an individual dimension, a Cloud service with n aspects can be represented as a Euclidean n-space. As such, when evaluating a Cloud service, the benchmarking results together would identify a particular point in the Euclidean n-space, which essentially uses a tuple to reflect the overall feature of the Cloud service with respect to the corresponding benchmark suite, as illustrated in Figure 1.

As previously mentioned, a boosting metric is supposed to represent overall service feature by using a single number instead of using a tuple. Thus, seeking boosting metrics here is to find single-number representations of the benchmarking point in the service-aspect space. Since the benchmarking point and the origin can determine a rectangular parallelepiped in the Euclidean n-space (cf. Figure 1), we can switch our focus from the coordinates of the point to the attributes of the corresponding rectangular parallelepiped, such as the perimeter, surface area, volume, etc. Furthermore, given the related work, we tried to rationalize several “classic” means (Cantrell, 2012) to suit the rectangular parallelepiped’s attributes, rather than reinventing “new” measures. The equations of the selected means are listed below, where Benchmarking_i denotes the benchmarking result by using the ith benchmark in a suite.

![Figure 1. The benchmarking results point in a Cloud service aspect space](image)

**Arithmetic Mean.** Corresponding to that the perimeter of a rectangular parallelepiped is the sum of its side lengths, we may use the arithmetic mean as a potential boosting metric, as shown in Equation (1).

\[
\text{Arithmetic Mean} = \frac{\sum_{i=1}^{n} \text{Benchmarking}_i}{n}
\]

**Geometric Mean.** Corresponding to that the volume of a rectangular parallelepiped is the product of its side lengths, we may use the geometric mean as a candidate boosting metric, as shown in Equation (2).

\[
\text{Geometric Mean} = \sqrt[n]{\prod_{i=1}^{n} \text{Benchmarking}_i}
\]

**Harmonic Mean.** In particular, corresponding to the rate between the volume and surface area of a rectangular parallelepiped, we may use the harmonic mean as a candidate boosting metric, as shown in Equation (3).

\[
\text{Harmonic Mean} = \frac{n \times \prod_{i=1}^{n} \text{Benchmarking}_i}{\sum_{j=1}^{n} \frac{\text{Benchmarking}_j}{\text{Benchmarking}_i}}
\]

\[
= \frac{n}{\sum_{i=1}^{n} \frac{1}{\text{Benchmarking}_i}}
\]
**Quadratic Mean.** Corresponding to the distance between the benchmarking point and the origin, we may use the quadratic mean as a candidate boosting metric, as shown in Equation (4).

\[
\text{Quadratic Mean} = \sqrt{\frac{\sum_{i=1}^{n} \text{Benchmark}_{i}^2}{n}}
\]  

(4)

As can be seen, it is convenient to calculate these means of a set of benchmarking results to reflect the summary feature of a Cloud service. Interestingly, the Geometric Mean seems the most popular one in practice (Evangelinos and Hill, 2008; Jackson et al., 2010). Nevertheless, there is a default constraint when employing means as boosting metrics: different Cloud service aspects should be homogeneously measured by using different benchmarks in a suite. In other words, to calculate means (secondary metrics), different benchmarking results for different Cloud service aspects must adopt the same primary metric. If the constraint cannot be satisfied, we may employ a more generic solution – Radar Plot, as specified in the following subsection.

### 3.2 Radar Plot as a Boosting Metric

Radar plot is a simple but intuitive graphical tool that can simultaneously depict a group of different types of values relative to a central point. When a benchmark suite uses different primary metrics to measure different Cloud service aspects, we can use radar plot to represent the benchmarking results over a predefined baseline. In particular, we can also portray several groups of standardized benchmarking results in one radar plot without predefining any baseline (cf. Figure 2). Given the analysis of the existing metrics for Cloud services evaluation (Li et al., 2012b), here we elaborate two standardization methods only for Higher Better (HB) metrics and Lower Better (LB) metrics (Obaidat and Boudriga, 2009) respectively.

\[
\text{HB}_{\text{Standardized}}_i = \frac{\text{Benchmark}_{i}}{\text{MAX}(\text{Benchmark}_{1...n})}
\]

(5)

\[
\text{LB}_{\text{Standardized}}_i = \frac{1}{\text{MAX}(\frac{1}{\text{Benchmark}_{i}})}
\]

(6)

Equation (5) is for the standardization of HB metrics, while Equation (6) for LB metrics. Here Standardized$_{i}$ refers to the standardized ith benchmarking result Benchmark$_{i}$. In fact, Equation (6) offers LB metrics a higher better representation through reciprocal standardization, so that all the standardized benchmarking results can be settled homogeneously higher better in a radar plot, and meanwhile constructs a bigger-area better polygon. As such, we can intuitively contrast the areas of different polygons to compare different groups of benchmarking results. Moreover, the area of a polygon can be regarded as a single numerical Response to facilitate experimental design and analysis. Suppose there are $n$ benchmarking results standardized and marked in a radar plot, we can calculate the area of the corresponding polygon by summing up areas of the $n$ adjacent triangles, as shown in Equation 7.

\[
\text{Area} = \sum_{i=1}^{n} \frac{2\sin\left(\frac{\pi}{n}\right) \times \text{Standardized}_{i} \times \text{Standardized}_{mod(i,n)}}{2}
\]

(7)

Here we employ a real case to demonstrate the radar plot as a boosting metric. For our convenience, the evaluation data reported in (Ostermann et al., 2006) are directly reused, as shown in Table 1. Given the various types of benchmarking results, such as HPL, STREAM, RandomAccess, Latency, and Bandwidth, within the HPCC benchmark suite (Luszczek et al., 2006), it is hard to compare the summary performance as a whole when evaluating different types of EC2 instances.

### Table 1. Original HPCC Benchmarking Results for Various EC2 Instance Types

| Name               | m1.large | m1.xlarge | c1.medium | c1.xlarge |
|--------------------|----------|-----------|-----------|-----------|
| HPL (GFLOPS)       | 7.15     | 11.38     | 3.91      | 51.58     |
| STREAM (GBps)      | 2.38     | 3.47      | 3.84      | 15.65     |
| RandomAccess       | 54.35    | 168.64    | 46.73     | 249.66    |
| (MUPS)             |          |           |           |           |
| Latency (μs)       | 20.48    | 17.87     | 13.92     | 14.19     |
| Bandwidth (GBps)   | 0.7      | 0.92      | 2.07      | 1.49      |

### Table 2. Standardized HPCC Benchmarking Results for Various EC2 Instance Types

| Name               | m1.large | m1.xlarge | c1.medium | c1.xlarge |
|--------------------|----------|-----------|-----------|-----------|
| HPL                | 0.1386   | 0.2206    | 0.0758    | 1         |
| STREAM             | 0.1521   | 0.2217    | 0.2454    | 1         |
| RandomAccess       | 0.2177   | 0.6755    | 0.1872    | 1         |
| Latency            | 0.6797   | 0.779     | 1         | 0.981     |
| Bandwidth          | 0.3382   | 0.4444    | 1         | 0.7198    |

Thus, we first standardize those benchmarking results respectively, as listed in Table 2. Note that the generated numbers in Table 2 do not come with any benchmarking unit. Then, the standardized benchmarking results are represented in a radar plot, as illustrated in Figure 2. Through this radar plot, we can intuitively and conveniently identify that: c1.xlarge has absolutely better overall performance than m1.large and m1.xlarge; c1.xlarge is also better than c1.medium in general, while slightly poorer in
terms of Bandwidth and Latency. In particular, the areas of
different benchmarking polygons in the radar plot are
further calculated to quantitatively reflect the summary
performance of the four types of EC2 instances, as
bracketed beside the legend entries. Essentially, the
numerical areas may play a Response role in the design of
experiments for Cloud services evaluation. A complete
experimental design and analysis sample by using boosting
metrics is elaborated in the next section.

4. Sophisticated Boosting Metrics
   for Cloud Services Evaluation

Although not common, the idea of boosting metrics has
been intuitively employed in some Cloud services
evaluation work, together with a little preliminary
discussion about the merits of employing boosting metrics.
For example, the geometric mean of eight NAS Parallel
Benchmarks (NPB) results (BT, CG, FT, IS, LU, MG, SP,
UA) was used to measure the computational performance of
Amazon EC2 on a wide set of model applications and
kernels (Evangelinos and Hill, 2008). Interestingly, some
sophisticated measurements of Cloud services can also be
unified into the notion of Boosting Metric. In other words,
we may identify the same essence behind some completely
different approaches to Cloud service measurement. Four
samples are demonstrated in the following subsections.

4.1 Sustained System Performance (SSP)

Considering the diversity of users’ requirements in a
supercomputer center, Jackson et al. (2010) employed seven
typical scientific applications as a benchmark suite to
evaluate the Amazon Cloud services for high performance
computing. These applications span a range of science
domain, parallelization schemes, concurrences, and
machine-based characteristics (e.g., communication,
computation, memory, and storage). To better represent the
effectiveness of Cloud services for delivered performance
on applications rather than peak FLOP rates, the authors
proposed an aggregate measure of computing system
performance, namely Sustained System Performance (SSP)
metric, as shown in Equation 8.

\[
SSP = N \left( \prod_{i=1}^{M} P_i \right)^{1/M}
\]

In detail, evaluators may select \(M\) applications as a set
of benchmarks; \(P_i\) indicates the application \(i\)’s performance
expressed in units of GFlops per second per core; the size of
the evaluated computing system is considered as the number
\(N\) of its computational cores. As such, the calculation of
SSP is to multiply the geometric mean of individual
applications’ performance per CPU core by the number of
computational cores, which can be viewed as an extended-
Geometric Mean-based boosting metric.

4.2 Penalty Model

As mentioned previously, it has been identified that
evaluating elasticity of a Cloud service is not trivial
(Kossmann and Kraska, 2010), because Cloud elasticity is
related to both the resource scaling time and the resource
cost (Islam et al., 2012). By integrating relevant basic
Quality of Service (QoS) metrics to monitor the requested
Cloud resources, Islam et al. (2012) suggested a Penalty
Model to measure the imperfections in elasticity of Cloud
services for a given workload, as shown in Equation 9.

\[
P = \frac{P_o(t_e, t_s) + P_u(t_s, t_e)}{t_e - t_s}
\]

Where \(P_o(t_p, t_e)\) refers to the penalty for over-
provisioning, which captures the cost of provisioned but
unutilized resources for a period starting at \(t_s\) and ending at
\(t_e\); while \(P_u(t_e, t_s)\) measures opportunity cost from the
performance degradation that arises with under-provisioning
during the same period. Note that, the authors assume that
each Cloud resource type can be allocated in units, and users
can learn the resource allocation level with relevant QoS
measurement results. Eventually, the Penalty Model acts as
a boosting metric, and delivers a single penalty score \(P\) in
monetary units over the time interval \([t_s, t_e]\).

4.3 Customer Satisfaction

Given the uncertainty in the runtime of Cloud services,
the existing Service Level Agreements (SLAs) often lack
providing comprehensive information about the overall
performance of a service regarding specific tasks (Lenk et
al., 2011). Similar to the aforementioned Penalty Model, a
utility theory-inspired model was developed for measuring
Customer Satisfaction in the Cloud, while Customer
Satisfaction was treated as an explicit metric to support utility-based SLAs in order to balance the performance of applications and the cost of running them (Chen et al., 2011). In this case, the authors considered different customer satisfactions as different combinations of service price and request processing time, as shown in Equation 10.

\[ U(p,t) = U_0 - ap - \beta t \]  

(10)

in which, the satisfaction or utility \( U \) of using a service is defined as a function of the service price \( p \) and the response time \( t \), \( U_0 \) is the maximum utility that the service delivers to the customer, and it is proportional to the size of the service request; while \( a \) and \( \beta \) are coefficients, and \( a/\beta \) is known as marginal rate of substitution in economics, denoting the rate at which the customer is willing to give up response time in exchange for service price without any satisfaction change.

### 4.4 Analytic Hierarchy Process (AHP) Based Decision

The overall measurement of Cloud services could have to cater for a number of conflicting criteria, e.g., performance and cost, and the problem would be further aggravated by the fact that different applications have heterogeneous QoS requirements. By treating the Cloud service measurement as a decision-making problem, Zhang et al. (2012) suggested the multi-criteria decision-making technique Analytic Hierarchy Process (AHP) to handle various and mixed qualitative and quantitative criteria. AHP is based on pair-wise comparisons of the criteria. For each pair of criteria, the administrator is required to provide a subjective opinion of their relative importance. By converting the subjective opinions to numerical values, AHP finally generates numerical priorities for all the decision alternatives. In other words, AHP can supply straightforward indexes as measures of candidate Cloud services to customers for their service selection. Due to the limit of space, we do not elaborate the detailed mechanism of AHP in this paper.

### 5. A Case Study of Using Boosting Metrics in Experimental Design and Analysis

1. **Requirement Recognition**: Recognize the problem, and state the purpose of a proposed evaluation.
2. **Service Feature Identification**: Identify Cloud services and their features to be evaluated.
3. **Metrics and Benchmarks Listing**: List all the metrics and benchmarks that may be used for the proposed evaluation.
4. **Metrics and Benchmarks Selection**: Select suitable metrics and benchmarks for the proposed evaluation.
5. **Experimental Factors Listing**: List all the factors that may be involved in the evaluation experiments.
6. **Experimental Factors Selection**: Select limited factors to study.

\[ \text{and also choose levels/ranges of these factors.} \]

7. **Experimental Design**: Design experiments based on the above work. Pilot experiments may also be done in advance to facilitate the experimental design.
8. **Experimental Implementation**: Prepare experimental environment and perform the designed experiments.
9. **Experimental Analysis**: Statistically analyze and interpret the experimental results.
10. **Conclusion and Reporting**: Draw conclusions and report the overall evaluation procedure and results.

**Figure 3. CEEM for Cloud services evaluation (Li et al., 2013b)**

Here we use a case study of evaluating Amazon EC2 to demonstrate how a boosting metric can be used in evaluation and how it can help analyze experimental results. To achieve convincing demonstration, we followed the ten-step Cloud Evaluation Experiment Methodology (CEEM, cf. Figure 3) (Li et al., 2013b) to perform the evaluation.

### 5.1 Requirement Recognition and Service Feature Identification

**Problem and Motivation.** We proposed to use a set of Amazon EC2 instances to perform a small-scale parallel computing project. According to the estimation of our project and the predefined EC2 instance types (Amazon, 2013), we initially selected m1.xlarge and m2.xlarge as two candidate types of parallel computing nodes. The specifications and prices of these two EC2 instance types are listed in Table 3. When making decision to choose the most suitable alternative from these two options, we found that it was hard to directly distinguish the better one based on their specifications. Unlike the clear differences between the other types of EC2 instances, each of these two options has its own distinctions. For example, m1.xlarge seems overall better than m2.xlarge, while m2.xlarge has faster single CPU core, larger memory, and lower price. As such, we decided to evaluate these two types of EC2 instances to roughly compare their potential performance in our project. It is then suitable to consider boosting metrics for summary measurement in this case.

**Table 3. Specifications and Prices of Two EC2 Instance Types**

| Specification        | m1.xlarge | M2.xlarge |
|----------------------|-----------|-----------|
| Core Amount          | 4         | 2         |
| ECU Amount           | 8         | 6.5       |
| Network I/O          | High      | Moderate  |
| Performance          |           |           |
| Memory Size          | 15 GB     | 17.1 GB   |
| Platform             | 64 bit    | 64 bit    |
| Storage Size         | 1690 GB   | 420 GB    |
| Windows Usage Cost   | $0.92 per Hour | $0.57 per Hour |

**Cloud Services to be Evaluated.** As mentioned previously, the evaluation requirement in this case can be viewed as merely a rough understanding of the parallel...
computing capability of those two instance types. To save
time, we decided to perform evaluation for each option only
on a single EC2 instance rather than on a real parallel cluster
environment. In detail, we applied one m1.xlarge instance
and one m2.xlarge instance respectively from Amazon’s
US-EAST-1 region, and both instances came with the same
quick launch Amazon Machine Image (AMI) – 64 bit
Windows Server 2008 Base.

5.2 BENCHMARK & METRICS LISTING AND SELECTION

As a well-known and well-accepted parallel computing
benchmark suite, NPB has been widely used for Scientific
Computing evaluation in the public Cloud (Akioka and
Muraoka, 2010; Carlyle et al., 2010; Evangelinos and Hill,
2008; He et al., 2010; Walker, 2008). Therefore, we also
employed NPB as the benchmark suite to evaluate the
summary performance of EC2 instances for our project. In
particular, since the software system in our project was
implemented using JAVA, we selected the latest JAVA
version of NPB, namely NPB3.0-JAV (NASA, 2012).
Although different benchmarks in NPB are used to reflect
different features of a computing system like computation,
communication and storage, all the NPB benchmarking
results adopt the same format with homogeneous metrics,
such as benchmark runtime (time in seconds) and
benchmark FLOP rate (floating point Mops total).
Following the popular choice (cf. Section 4), we also chose
Geometric Mean as the boosting metric over the primary
metrics benchmark runtime and benchmark FLOP rate in
this case study.

5.3 EXPERIMENTAL FACTORS LISTING AND SELECTION

Before evaluating a system, experimental factors
identification is a tedious but necessary task (Le Boudec,
2010). Factors here refer to the elements in the system or the
workload that may influence the evaluation result. In fact,
our previous work has established a factor framework for
Cloud services evaluation, and the latest framework version
capsules the state-of-the-practice of performance evaluation
factors that people currently take into account in the Cloud
Computing domain (Li et al., 2012a). Since this evaluation
work would also measure performance of EC2 instances, we
conventionally identified experimental factors within the
proposed framework. In detail, we explored experimental
factors related to Cloud resource and benchmark’s workload
respectively: Instance Type (m1.xlarge vs. m2.xlarge),
Thread Number (2 vs. 4), and Workload Size (Class W vs.
Class A).

5.4 EXPERIMENTAL DESIGN

When it comes to experimental design, there are three
basic principles: Randomization, Replication, and Blocking
(Montgomery, 2009). In this case, we only focus on the
Randomization and Replication. Although an entire NPB
suite run can be treated as a block, here we try to simplify
the demonstration without elaborating sophisticated design
approaches. The detailed designing process is then
composed of three steps, as specified below.

Determining Individual Experimental Trials. In this
evaluation work, an experimental trial indicates a specific
benchmark run on an EC2 instance. In practice, we used one
batch command to drive a single NPB benchmark run
during the experiments. Thus, a series of batch commands
were listed to represent different individual experimental
trials.

Determining Amount of Experimental Trials. As
mentioned previously, we decided to investigate two levels
of Workload Size (Class A and W) and two levels of Thread
Number (2 and 4) for two Instance Types (m1.xlarge and
m2.xlarge). To facilitate the investigation, we also planned
benchmarking with single thread as a reference baseline.
On the other hand, the JAVA-version NPB suite comprises
seven benchmarks. According to our pilot test of running
those seven benchmarks on a local machine, we decided to
replicate all the different trials five times. Therefore, there
are $2 \times 3 \times 7 \times 5 = 210$ experimental trials in total on each
instance.

Determining Sequence of Experimental Trials. To
achieve a randomized trial sequence, we assigned two
random numbers to each trial-associated batch command in
EXCEL. The 210 batch commands can be ordered by one
random number and another in turn, to run experiments on
the m1.xlarge and m2.xlarge instances respectively.
Through such a randomization, we made individual trials as
independent as possible between each other to reduce the
experimental sequence-related bias.

5.5 EXPERIMENTAL IMPLEMENTATION

Due to the limit of space, the specific experimental
results from individual NPB benchmarks are not reported in
this paper. In summary, by averaging results of identical
experimental trials, and dividing the trials into different
workload-size, thread-number, and instance-type groups, we
obtained a set of Runtime and FLOP Rate geometric means
with respect to the NPB suite under different conditions, as
listed in Table 4.

Table 4. Geometric Means of NPB3.0-JAV Benchmarking
Results with Different Circumstances

| Cloud Resource | Workload | Boosting Metric (Geometric Mean) |
|----------------|---------|---------------------------------|
|                |         | NPB Runtime (second) | NPB FLOP Rate (Mops) |
| 1 Thread       | 1 Thread| 6.215                 | 179.706             |
|                | Class W | 60.889                | 153.017             |
|                | Class A | 31.176                | 299.813             |
| 2 Thread       | 2 Thread| 3.727                 | 298.949             |
|                | Class W | 31.176                | 298.949             |
|                | Class A | 2.73                  | 412.717             |
| 4 Thread       | 4 Thread| 18.138                | 513.873             |
To intuitively show the instances’ performance changing when varying conditions, we also used four line charts to represent the boosting metric’s measurements, as illustrated in Figure 4. It is not surprising that, benefiting from the faster single core, the m2.xlarge instance defeats the m1.xlarge instance for running NPB suite before over-saturating its CPU cores, while the m1.xlarge instance performs better with four-thread trials. Nevertheless, it is still hard to tell whether Instance Type is a significant factor or not in general. Therefore, we employed formal experimental-analysis techniques to unfold further investigation, as explained in the following subsection.

(a) Geometric means of NPB runtime with workload Class W
(b) Geometric means of NPB runtime with workload Class A
(c) Geometric means of NPB FLOP rate with workload Class W
(d) Geometric means of NPB FLOP rate with workload Class A

Figure 4. Illustration of geometric means of NPB3.0-JAV benchmarking results with different circumstances.

5.6 EXPERIMENTAL ANALYSIS

Since only two levels of an experimental factor were particularly concerned in this case (cf. Subsection 5.3), we naturally adopted the optimal design and analysis technique, namely Full-factorial $2^k$ Design (Montgomery, 2009), to analyze the experimental results. Given the three factors considered, a pseudo $2^3$ design matrix was generated as shown in Table 5. The response columns in the matrix were filled with pseudo-trial results that correspond to eight geometric means in Table 4. For conciseness, we further assigned aliases to those experimental factors and responses, as listed below.

- Factor A: Instance Type (m1.xlarge vs. m2.xlarge).
- Factor B: Thread Number (2 vs. 4).
- Factor C: Workload Size (Class W vs. Class A).
- Response R1: NPB Runtime (seconds).
- Response R2: NPB FLOP Rate (Mops).

Recall that the analysis is to investigate if Instance Type (A) (or other factors) significantly influences the benchmarking results. By setting the significance level $\alpha$ as 0.05 (Jackson, 2011), we can draw Pareto plots (Antony, 2003) to detect the factor and interaction effects that are important to the parallel computing (NPB suite in this case), as shown in Figure 5. To save space, we do not elaborate the backend statistics here. In brief, 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).
Since the effect of factor Workload Size (C) is beyond the reference line in Figure 5a, it is apparent that Workload Size (C) dominates the runtime of NPB suite. On the contrary, the factor Instance Type (A) has little influence on the benchmark runtime in this case. As for the FLOP Rate analysis in Figure 5b, we show that none of the factor or interaction effects significantly influences the transaction speed. However, relatively speaking, Thread Number (B) is the most important to the NPB FLOP Rate, while Instance Type (A) is still the least important factor. Therefore, for our proposed parallel computing project, we are now suggested to pay more attention to the workload size to distinguish between those two EC2 instance types.

From Figure 4, it is clear that increasing thread numbers will not increase computing performance if an instance’s CPU cores are already saturated, especially with larger workload. However, increasing workload size seems to be able to continually increase the performance difference between two instances at 4 or more threads, which was further confirmed by running a supplementary experiment with NPB’s 4X larger workload Class B. To facilitate analysis, we calculated different performance improvements of switching from m2.xlarge to m1.xlarge at 4 threads by using Equation (11), as listed in Table 6. Note that we use the minimum performance value between the two instances as the denominator to avoid the Ratio Game bias (Jain, 1991).

\[
I = \frac{|\text{Performance}_{m2} - \text{Performance}_{m1}|}{\text{MIN(Performance}_{m1}, \text{Performance}_{m2})} \times 100\% \tag{11}
\]

Given the price increase of switching from m2.xlarge to m1.xlarge ((0.95–0.57)/0.57x100%=61.4%), the instance-hour for running m2.xlarge is 1.614 times higher than running m1.xlarge within the same budget. In other words, m2.xlarge always has a cost advantage over m1.xlarge until the performance improvement reaches 61.4%, although the total runtime may be longer. According to the previous analysis, we finally decided to choose m2.xlarge as the cost-wise option for our small-scale parallel computing project.

| Workload | Performance Improvement | NPB FLOP Rate |
|----------|-------------------------|---------------|
| Class W  | 9.4%                    | 10.4%         |
| Class A  | 39.6%                   | 39.5%         |
| Class B  | 48.4%                   | 48.4%         |

6. CONCLUSIONS AND FUTURE WORK

Metrics play a vital role in any evaluation implementation. To evaluate Cloud services from a holistic view, a widely adopted strategy is to use a set of metrics along with a suite of benchmarks to perform separated experiments. With those separated measurements, customers would have to further summarize various evaluation results by themselves. In particular cases, an evaluation requirement may comprise diverse and even conflicting measurement criteria. As a result, it could become hard for customers to trade off between performance and cost. Considering that delivering a single score is also a usual benchmarking strategy to facilitate drawing simple conclusions from evaluation results (Islam et al., 2012), we suggest using boosting metrics to depict summary measurements of Cloud services. In other words, boosting metrics are supposed to measure one complex Cloud service feature involving multiple service properties, or even

---

**Table 5. A Full-factorial (2^3) Design Matrix for This Case Study**

| Trial | A | B | C | R1 (second) | R2 (Mops) |
|-------|---|---|---|-------------|-----------|
| 1     | m1| 2 | W | 3.727       | 299.813   |
| 2     | m1| 4 | A | 18.138      | 513.873   |
| 3     | m2| 2 | W | 3.401       | 351.003   |
| 4     | m1| 2 | A | 31.176      | 298.949   |
| 5     | m2| 2 | A | 24.537      | 379.765   |
| 6     | m2| 4 | A | 25.32       | 368.289   |
| 7     | m1| 4 | W | 2.73        | 412.717   |
| 8     | m2| 4 | W | 2.987       | 373.948   |

---

Figure 5. The Pareto plots of experimental factor effects (generated by Minitab)
measure the overall service features, so as to help customers realize the overall quality of a Cloud service. For example, we may calculate different means or draw radar plot to integrate a suite of benchmarking results into a summary measurement index. Moreover, we can use the idea of boosting metrics to unify and organize a set of different and sophisticated approaches to Cloud service measurement, such as SSP, penalty model, customer satisfaction, and AHP-based decision. Benefiting from the unification, practitioners could conveniently understand, locate, or develop proper boosting metrics for Cloud services evaluation. More importantly, the usage of boosting metrics can further facilitate applying experimental design and analysis to the evaluation work, as demonstrated in Section 5.

Our future work will be unfolded along two directions. Firstly, we plan to gradually collect, propose, and report new boosting metrics to supplement primary measures of individual Cloud service features. Secondly, we will concentrate on the Elasticity of Cloud services, and help improve the current approach (Islam et al., 2012) to Elasticity evaluation by employing more suitable boosting metrics.

7. ACKNOWLEDGMENT

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. NICTA is also funded and supported by the Australian Capital Territory, the New South Wales, Queensland and Victorian Governments, the Australian National University, the University of New South Wales, the University of Melbourne, the University of Queensland, the University of Sydney, Griffith University, Queensland University of Technology, Monash University and other university partners.

8. REFERENCES

Akioka, S. and Muraoka, Y. (2010). HPC Benchmarks on Amazon EC2, Proceedings of the IEEE 24th International Conferences on Advanced Information Networking and Application Workshops (WAINA 2010), IEEE Computer Society, Perth, Australia, pp. 1029–1034.

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

Amazon EC2 Instance Types. (2013). Retrieved June 4, 2013, from http://aws.amazon.com/ec2/instance-types/.

Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J. and Brandic, I. (2009). Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computing Systems, 25(6), pp. 599–616.

Brooks, C. (2010). Cloud Computing Benchmarks on the Rise. Retrieved May 26, 2013, from http://searchcloudcomputing.techtarget.com/news/1514547/Cloud-computing-benchmarks-on-the-rise.

Cantrell, D.W. (2012). Pythagorean Means. Retrieved May 26, 2013, from http://mathworld.wolfram.com/PythagoreanMeans.html.

Carlyle, A.G., Harrell, S.L. and Smith, P.M. (2010). Cost-effective HPC: The Community or the Cloud, Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2010), IEEE Computer Society, Indianapolis, USA, pp. 169–176.

Chen, J., Wang, C., Zhou, B.B., Sun, L., Lee, Y.C. and Zomaya, A.Y. (2011). Tradeoffs between Profit and Customer Satisfaction for Service Provisioning in the Cloud. Proceedings of the 20th international symposium on High performance distributed computing (HPDC 2011), ACM Press, San Jose, USA, pp. 229–238.

Dejun, J., Pierre, G. and Chi, C.-H. (2009). EC2 Performance Analysis for Resource Provisioning of Service-Oriented Applications, Proceedings of the 2009 International Conference on Service-Oriented Computing (ICSOC/ServiceWave 2009), Springer-Verlag, Stockholm, Sweden, pp. 197–207.

Evangelinos, C. and Hill, C.N. (2008). Cloud Computing for Parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-ocean Climate Models on Amazon’s EC2, Proceedings of the 1st Workshop on Cloud Computing and Its Applications (CCA 2008), Chicago, IL, USA, pp. 1–6.

Ferdman, M., Adileh, A., Kocherber, O., Velos, S., Alisafaei, M., Jevdijc, D., Kaynak, C., Popescu, A.D., Ailamaki, A. and Falsafi, B. (2012). Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware, Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2012), ACM Press, London, UK, pp. 37–48.

He, Q., Zhou, S., Kohler, B., Duffy, D. and McGlynn, T. (2010). Case Study for Running HPC Applications in Public Clouds. Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC2010), ACM Press, Chicago, Illinois, pp. 395–401.

Iosup, A., Ostermann, S., Yigitbasi, N., Prodan, R., Fahringer, T. and Epema, D.H.J. (2011). Performance Analysis of Cloud Computing Services for Many-tasks Scientific Computing, IEEE Transactions on Parallel Distributed Systems, 22(6), pp. 931–945.

Islam, S., Lee, K., Fekete, A. and Liu, A. (2012). How a Consumer can Measure Elasticity for Cloud Platforms, Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE 2012), ACM Press, Boston USA, pp. 85–96.

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

Jackson, K.R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H.J. and Wright, N.J. (2010). Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud, Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2010), IEEE Computer Society, Indianapolis, USA, pp. 159–168.

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

Kossmann, D. and Kraska, T. (2010). Data Management in the Cloud: Promises, State-of-the-art, and Open Questions, Datenbank Spektrum, 10(3), pp. 121–129.

Le Boudec, J.-Y. (2010). Performance Evaluation of Computer and Communication Systems, EPFL Press, Lausanne, Switzerland.

Lenk, A., Menzel, M., Lipsky, J., Tai, S. and Offermann, P. (2011). What are you Paying for? Performance Benchmarking for Infrastructure-as-a-Service Offerings, Proceedings of the 4th IEEE International Conference on Cloud Computing (CLOUD 2011), IEEE Computer Society, Washington DC, USA, pp. 484–491.

Li, Z., O’Brien, L., Zhang, H. and Cai, R. (2012a). A Factor Framework for Experimental Design for Performance Evaluation of Commercial Cloud Services, Proceedings of the 4nd IEEE International
Conference on Cloud Computing Technology and Science (CloudCom 2012), IEEE Computer Society, Taipei, Taiwan, pp. 759–764.

Authors

Zheng Li received his Degree of M.E. by Research from the University of New South Wales (UNSW). He is now a PhD student at the School of Computer Science at the Australian National University (ANU), and a graduate researcher with the Software Systems Research Group (SSRG) at National ICT Australia (NICTA). He is the author of more than 15 journal and conference publications. His research interests include empirical software engineering, software cost/effort estimation, machine learning, Web service composition, and Cloud computing.

Liam O’Brien has over 23 years experience in research and development in software engineering. He is a Software and Applications Architect with Geoscience Australia and was previously Chief Software Architect with CSIRO and a Principal Researcher at NICTA’s e-Government Initiative. He is also a Member-at-Large of the Service Science Society Australia which he co-founded in 2010. He has previously worked as a researcher with Lero (Ireland), Carnegie Mellon University’s Software Engineering Institute (USA), CSIRO (Australia) and the University of Limerick (Ireland). His main areas of research include enterprise architecture, software architecture, SOA, service science, software reuse, software modernisation, and cloud computing. He holds a BSc and PhD from the University of Limerick, Ireland. He is a member of the IEEE and IEEE Computer Society.

He Zhang is a professor at State Key Laboratory of Novel Software Technology at Software Institute at Nanjing University. Before joining Nanjing University, he was a Research Scientist with National ICT Australia (NICTA), and a Conjoint Lecturer at University of New South Wales (UNSW). He received his Ph.D from UNSW. He joined academia after 7 years in industry, developing software systems in the areas of aerospace and complex data management. He has published 70+ peer-reviewed research papers in international journals, conferences, and workshops. His current research areas include software process modeling and simulation, process assessment and improvement, software engineering for embedded systems, empirical software engineering, and service-oriented computing.