Good Performance Metrics for Cloud Service Brokers

John O'Loughlin and Lee Gillam
Department of Computer Science
University of Surrey
Guildford, England
{John.oloughlin,l.gillam}@surrey.ac.uk

***Abstract*—The large number of Cloud Infrastructure Providers offering virtual machines in a variety sizes, but without transparent performance descriptions, makes price/performance comparisons difficult. This presents opportunities for Cloud Service Brokers (CSBs). A CSB could offer transparent price/performance comparisons, or performance discovery. This paper explores the kinds of performance measures such CSBs should use. Using three quite different benchmarks, povray, bzip2 and STREAM, we show a 17%, 77% and a 340% increase in performance, respectively, from worst to best in an existing large Cloud. Based on these results, we propose a discovery service for best price/performance for use in aggregation of heterogeneous resources.**

Keywords- Cloud Computing; performance;brokers;metrics

I. INTRODUCTION

In Infrastructure Clouds, users can obtain a wide range of compute resources, predominant amongst which are Virtual Machines (VMs). There are a large (and increasing) number of Cloud providers, all selling a variety of VM types. These VM types are typically defined in terms of amounts of resources provided: number of vCPUs, amount of RAM and an amount of local storage. The lack of transparent performance descriptions for instance types, and the large performance variation of instances of the same type that we [1] and others [14] have previously reported, makes finding the best price/performance a daunting task. It is, therefore, not surprising that Cloud Service Brokers (CSBs), which can act as intermediaries between Cloud users and Cloud providers, are gaining attention. There is a clear opportunity for such CSBs to address the price/performance issue.

For a CSB to address price/performance, they must first clarify how performance is defined, measured and, especially, useful to Cloud users. Provider-specific performance measures, such as Amazon's EC2 Compute Unit (ECU), are of limited value to a user since they do not always correlate well to the performance of the application they wish to run [1].

In this paper, we explore the kinds of performance measures that could be useful to users, and therefore beneficial to profit-seeking CSBs. CSBs will need to know the deliverable performance of various Cloud resources with respect to the applications user wish to run. However, large Clouds are inevitably heterogeneous and resource availability may be unpredictable. And so it is vital to find strategies for obtaining the best resources when it is not possibly simply to request the best.

The rest of this paper is structured as follows: In section II, we explore the types of services a CSB could offer and show how they can help users with the daunting task of finding comparable instances across providers. In section III, we explore performance measurement and identify metrics that are appropriate for Cloud users. In section IV, we explain and present the performance results obtained from approximately 300 m1.small instances on EC2. We show how performance is dependent on both the CPU model backing the instance **and** the application. We use these results to propose a performance discovery service, initially for Amazon EC2; which we explore in detail in section V. In section VI, we present conclusions and future work.

II. OPPORTUNITIES FOR CSBS

Resources in Infrastructure Clouds should be available on demand and with the ability to obtain or release more and/or bigger and smaller VMs in order to scale use as required. Additionally, the user should not have to worry about the provider's ability to meet this demand. Such capability, together with a 'pay for what you use' charging model, makes Infrastructure Clouds particularly attractive for handling workload spikes or ad-hoc computing tasks. Presently, if they hope to obtain best performance for price, each Cloud user needs to understand both the resource requirements of their application and the capabilities and pricing structures offered by each Infrastructure provider. Clearly there are costs and risks associated to such determination, and likely much repetition of effort across users.

At minimum, a CSB could reduce the repetition and associated costs of determining best performance for price by having transparent information about likely performance. Gartner [2] identifies three kinds of service that brokers can provide: ***intermediation***, ***aggregation*** and ***arbitrage***, and each of these kinds of service could benefit from such determination.

An ***intermediation*** service enhances existing services to add value to them. A potential performance service exists

 ISBN: 978-1-61208-338-4

simply in being able to find instances whose performance is within a particular range.

In financial markets, ***arbitrage*** refers to the practice of simultaneously buying and selling the same resources in different markets and exploiting the price difference. CSBs could exploit performance variation by reselling better performing instances at a higher price and worse performing instances at a lower price.

CSBs wishing to ***aggregate*** resources across Clouds to find truly optimal performance would need first to understand the deliverable performance of each them, and then to be able to exploit the variations across them.

Such opportunities readily exist: simply finding comparable instance types between providers is not always straightforward. For example, both Amazon's Elastic Compute Cloud (EC2) First Generation Standard Instances [3] and Microsoft's Azure Standard Instances [4] start with a very similar 'small' instance type: 1 vCPU with 1.7GB and 1.75GB of RAM respectively. After this they diverge, with the EC2 extra large instance type having 4vCPU and 15GB RAM whilst the Azure extra large has 8vCPU with 14GB RAM.

Comparing expected performance is not always possible either. EC2 offers a compute rating for their instance types, in the form of the EC2 Compute Unit (ECU), but Microsoft does not do so for Azure. Therefore, price/performance comparison between the respective small instance types requires the customer to conduct benchmarking experiments of both types. Comparing a larger range of instances types, across multiple providers, will quickly become expensive, and quite possibly prohibitively so, for all except the largest users. In addition, as we show later in this paper, it would be a mistake to consider just a few instances in even a single provider as necessarily representative.

For EC2, Amazon define the ECU in terms of equivalent performance to a reference machine, '…an early 2006 Xeon' [5]. How the equivalence is established is not explained. Following EC2 we find both the Google Compute Engine Unit (GCEU) [6] and the HP Cloud Compute Unit (HPCCU) [7] defined in terms of reference machines. It is unclear how these metrics relate to established performance metrics; such as program execution time. Clearly there are multiple opportunities for the CSB in both understanding and providing price/performance information for instance types in terms of performance metrics that are useful to customers. And in the next section we explore what those metrics might be.

III. COMPUTE PERFORMANCE METRICS AND MEASUREMENTS

The question of how compute performance should be defined and measured is surprisingly contentious. There is no commonly accepted unit of work on a computer – and therefore no accepted definition of either how fast a computer is or how much work has been done per unit of time. Some performance metrics involve physical characteristics of the machine, such as CPU clock rates or Theoretical Peak Performance (TPP). Such approaches tend to have common failings: (1) they are only valid when comparing machines of the same micro-architecture (2) they tend to correlate poorly to actual application performance.

Defining machine performance in terms of application performance also leads to some difficulties: should we use actual applications (that are in common use) or applications which are, in some sense, typical of a class of applications? Should we use one application or a suite of applications? If we use a suite of applications how do we best summarise them? We do not discuss these important questions in further detail here but we do note that the trend is for actual applications and not kernels or micro applications.

There are, however, certain characteristics a good performance metric should have [8]. Four of the more important characteristics are:

- A metric is *linear* if, when its value changes by a given ratio, the actual performance (as measured by application performance) of the machine changes by the same ratio. For example, the Amazon ECU to be linear we might expect a 2 ECU machine to run a CPU bound application in half the time of a 1 ECU machine.
- A metric is *reliable* if, whenever it indicates that machine A should outperform machine B, it does.
- A metric has *repeatability* if the same value (within an error bound) is obtained every time we take a measurement.
- Finally, a good metric should be *easy to measure*.

Our previous work [1] has shown that the ECU may be *reliable* but is neither *linear* nor *repeatable*, unless a large error bound is considered. It is also not easy (or indeed possible) to measure since it is defined in terms of equivalent performance to a reference machine without defining how the equivalence is established or what approaches are used to construct it.

A. *'Bad Metrics'*

The following commonly found metrics all fail on at least one of the above characteristics: CPU clock rate, Theoretical Peak Performance, the maximum number of instructions a CPU could in theory execute per second, (TPP), Millions of Instructions per Second (MIPS), BogoMIPS and Floating Point Operations per Second (FLOPS). We discuss these metrics further below:

Clock Rate: A number of providers [12] express expected performance of their instances in terms of a clock rate but do not specify the CPU model. Clock rate is generally not a reliable indicator of application performance. The Pentium 4 range, for example, had higher clock rates than the Pentium 3 models but without a corresponding increase in application performance due to a significant increase in the depth of the CPU pipeline.

TPP: TPP for a multi-core CPU is calculated as the number of cores multiplied by the number of execution units multiplied by the clock rate. It serves only as an upper bound on performance, and assumes that the CPU pipeline is full at all times. However, due to pipeline stalls, branch

 ISBN: 978-1-61208-338-4

mis-predictions and memory hierarchy latency, application performance may well differ significantly from the one predicted by the TPP. This is of course an issue with peak metrics in general – they are often unobtainable.

MIPS: MIPS is calculated as the instruction count for an application divided by execution time*10^6. A MIPS rating cannot be used to compare the performance of CPUs with different instruction sets, since the same program may compile to a different total number of instructions. For example, we cannot readily compare a RISC machine to a CISC machine. MIPS, along with FLOPS, suffers from being a throughput (or rate) metric and yet the unit of work being done (execute an instruction or floating point instruction) is not constant. For example, memory access instructions will take longer to execute than instructions which operate on data present in CPU registers. Whilst MIPS are not commonly used by providers, they are the default performance measure available in the well-known Cloud simulation toolkit Cloudsim.

BogoMIPS: BogoMIPS stands for Bogus MIPS and is defined by the number of NOOP (no operation) operations a machine performs per second. It is used in the early stages of Linux Kernel boot process as a calibration tool and was not intended as a performance metric. In spite of this, some have claimed [10] that a machine's BogoMIPS can be related to its performance - without offering any supporting evidence for such a claim.

FLOPS: Similar to MIPS, FLOPS uses an inconsistent unit of work – the FLOP. Different FLOPS may take different amounts of time to execute depending on what they do. However, peak GigaFLOPS (GFLOPS), as measured by High Performance Linpack, is still the measure used to rank systems in the well-known top 500 HPC list. Some supercomputing centres are now moving away from peak performance to sustained performance of applications their users will run. It should also be noted that expressing performance in terms of FLOPS will not inform a user of a system how well an application that contains no floating point operations will run, and so the measurement is domain specific.

B. 'Good Metrics'

The metrics above are generally defined in terms of machine characteristics; better performance metrics tend to be defined in terms of application performance. We discuss some of these below.

Program Execution Time: This metric is defined by the elapsed wall clock time from program start to finish. Some authors [9] consider this to be the only meaningful and informative metric and suggest that any other metric may be misleading. *Performance* is defined as the inverse of execution time and so faster execution times give higher performance scores.

Throughput: Throughput (CPU bandwidth) is defined as the number of units of work per unit time (usually per second) the CPU can perform. For consistency, the unit of work should be well defined and remain constant. As discussed there is no commonly accepted definition of unit of work, and so this becomes workload dependent.

Work done in a fixed time: In the program execution time metric, the amount of work done is fixed and wall clock time is the variable of interest. In [11], the authors argue that some systems, such as HPC, are purchased in order to allow more work to be done in the same amount of time when compared to older systems. By 'more work' they tend to mean solving a larger problem, not just an increase in throughput. This could be, for example, running a Monte Carlo Simulation at a much greater number of iterations to produce smaller error bounds on estimates.

Response Time: The above metrics are suitable for batch jobs. For interactive applications or websites, response time (also known as application latency) is a good metric. It has been shown [16] that higher response times lead to lower user satisfaction.

In general, the good metrics relate to application performance and not machine characteristics. Given this, ratings that relate equivalent performance to specified physical machines, as currently favoured by large Cloud providers, are unsatisfactory for most purposes. And, as we will show, are also not particularly meaningful even for comparing virtual machines in the same Cloud (provider). Cloud Service Brokerages, then, would add good value by selecting good performance metrics that can clearly relate to the applications that their customers wish to run.

IV. EXPERIMENTS ON EC2

In this section we address a question that we believe will be of interest to a typical Cloud user: Given a number of workloads, where can I obtain best performance for them? Here, we explore this question in one Region of Amazon's EC2 (US-East), which reveals several insights into performance variability.

A. Experimental Setup and Results

We consider the following 3 workloads:

1. A bzip2 compression on an Ubuntu 10.04 desktop ISO.
2. A povray ray trace on the benchmark.pov file.
3. STREAM memory bandwidth benchmark using the triad kernel.

Both bzip2 (albeit with different input files) and povray are part of the Standard Performance Evaluation Corporation (SPEC) CPU benchmark suite [15]. They measure different aspects of the CPU: bzip2 primarily uses integer arithmetic whilst povray makes heavy use of floating point operations.

We run into an immediate and interesting difficulty. The EC2 account we are using has access to just 4 of a possible 5 Availability Zones (AZs) [13] in US East: us-east-1b, us-east-1c, us-east-1d and us-east-1e. As we shall see, EC2 AZs have different performance characteristics from each other. It is therefore entirely possible that the AZ this account does not have access to, us-east-1a, provides better performance.

 ISBN: 978-1-61208-338-4

We ran approximately 300 m1.small instances. In each instance the workloads were run sequentially. For bzip2 and povray we recorded the **Programme Execution Time,** whilst STREAM reports memory bandwidth in MB/s. As the unit of work is *consistent*, STREAM is an example of a good throughput metric. In Table 1 below, we record the summary statistics (to the nearest second or MB/s):

TABLE I. SUMMARY STATISTICS

Workload	Mean	Max	Min	SD	CoV
Bzip2 (s)	528	745	421	78	0.15
Povray (s)	636	701	599	33	0.05
STREAM (MB/s)	2853	5860	1328	1239	0.43

We determine the CPU model backing an instance by examining the file /proc/cpuinfo. All of these instances were backed by one of the following Intel Xeon CPU models: E5430, E5-2650, E5645 and E5507. The CPU models found are the same as in our previous work. In Table II below, we present the statistics for each of the workloads broken down by CPU model.

TABLE II. SUMMARY STATISTICS BY CPU MODEL

Workload	Statistic	E5430	E5-2650	E5645	E5507
Bzip2	Mean(s)	439	468	507	621
	Max(s)	467	500	535	745
	Min(s)	421	451	490	567
	SD(s)	11	12	10	31
	CoV	0.025	0.026	0.02	0.05
Povray	Mean(s)	693	614	606	632
	Max(s)	701	624	628	650
	Min(s)	687	606	599	625
	SD(s)	3	5	7	5
	CoV	0.004	0.008	0.011	0.008
STREAM	Mean(MB/s)	1446	5294	3395	2348
	Max(MB/s)	1572	5860	4008	2448
	Min(MB/s)	1328	4935	2995	2078
	SD(MB/s)	66	191	287	104
	CoV	0.045	0.036	0.085	0.044

The coefficient of variation (CoV) is the ratio of the standard deviation relative to the mean and is useful for comparing the amount of variation between two data sets. The CoV here shows that the amount of variation for each workload is greater when considered across all CPU models than for a particular CPU model. For example, across all CPU models the CoV for the bzip2 workload is 0.15 whilst for the individual CPUs the largest CoV we find is 0.05, and the smallest CoV is 0.02, as found on the E5645. We interpret this as follows: There is approximately 8 times the amount of variation in bzip2 results considered across all CPU models than we find on the E5645. We also note that the E5507 has twice the variation as found on the other models. We have similar findings for the povray and STREAM workloads.

From the results we see that performance for all workloads depends on the CPU model backing the instance. As such, we can order the CPUs by how they perform the task. For bzip2 we have (from best to worst): E5430, E5-2650, E5645 and E5507. Interestingly, the orderings for both povray and STREAM are different, for example, for STREAM the ordering would be: E5-2650, E5645, E5507 and E5430. This shows that it is not possible to identify a 'best' CPU for all workloads, providing an opportunity for a CSB to identify which CPUs models provide best performance for specific applications.

These results suggest that the E5-2650 is the most versatile CPU, for these three workloads - it is the second best performing CPU model for both the bzip2 and povray tests and the best for STREAM. In Fig.1, Fig.2, and Fig.3 below we present histograms of the results, broken down by CPU model, which show this more clearly.

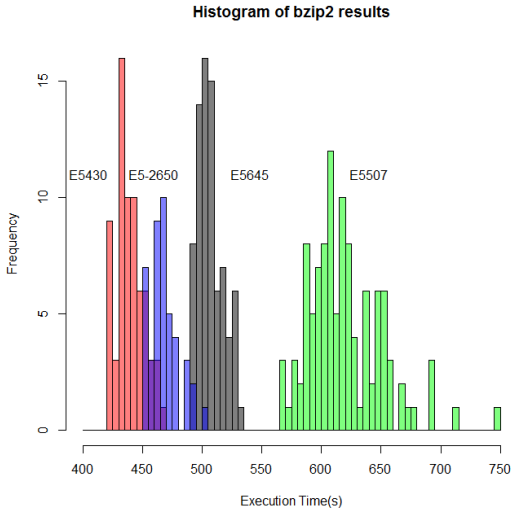

Figure 1. Bizp2 Execution Time(s)

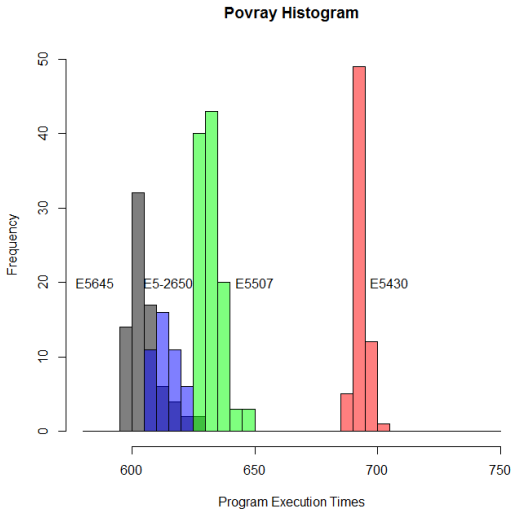

Figure 2. Povray Execution Time(s)

Figure 3. Triad Memory Bandwidth

We can use these results to identify which CPU model is most suitable for a particular workload. The question we now consider is this: How would I obtain instances with this CPU model backing it when I can't specify it in the request? In Table III below, we record the percentages of the CPU models we found in the 4 AZs (note that, due to Amazon's structuring within EC2, different users may well have different mappings):

TABLE III. CPU MODEL DISTRIBUTION BY AZ

	E5430	E5-2650	E5645	E5507
us-east-1b	25%	0	40%	35%
us-east-1c	27%	0	23%	50%
us-east-1d	30%	0	36%	34%
us-east-1e	0%	88%	12%	0%

Considering STREAM, we know that the E5-2650 is the best performing (from Table II), and from Table III, we see that we should choose us-east-1e when running this workload. For povray it is slightly more complicated. The E5645 is, on average, the best for this task but the E5-2650 gives very similar performance. Whilst it may be tempting to run instances in us-east-1b for this workload, we note a high percentage of the E5430, the worst performing CPU for the task. A safer option may still be us-east-1e.

B. Work Done and Price/Performance Considerations

Although performance information for users is useful, arguably more important for them is price/performance – more so when considering systems or workloads requiring multiple instances or multiple instance types. For the bzip2 and povray workloads we consider one completed compression or image rendering as a unit of work. For each CPU model we can calculate the average number of units of work per hour that can be performed, and from this we deduce a price per unit of work. The instance price per hour for an m1.small in the US East Region (as of 01/14) is $0.06. The cost of an instance is the number of wall clock hours elapsed since the instance was launched, with partial hours charged as full hours. For example, an instance started at 20:50 and terminated at 22:05 will be charged for **3** hours. In Table IV below, we record the partial Units of Work (UoW) per hour (which we call the completion rate) together with the price per UoW.

TABLE IV. UNITS OF WORK

Workload	**E5430 ($/UoW)**	**E5-2650 ($/UoW)**	**E5645 ($/UoW)**	**E5507 ($/UoW)**
Bzip2	8.2 0.0073	7.7 0.0078	7.1 0.0085	5.8 0.001
Povray	5.2 0.0112	5.9 0.0101	5.9 0.0101	5.7 0.0103

Table IV uses mean application execution times, and also includes partial work done. Whilst in some cases this may be useful, in general it is not clear if a user would be interested in partial completion of work. Instead, we can consider a simpler question, for example: What are the best and worse prices for completing my work using m1.small instances? One such piece of work might be 'compress 10 desktop ISO images using bzip2'.

From Table 1, we calculate the best and worst completion rate as 8.6 and 4.8 (min E5430 and max E5507). Based on EC2 wall clock hours, 10 units will see the user charged for at least 2 hours on the E5430 and at least 3 hours on E5507: best price would be $0.12 (potentially $0.18 depending on job start timing) and worst price is $0.18 (potentially $0.24). So, assuming start on the hour, we may see a 50% increase in cost for the same work. And yet, if we compare the actual execution times - 421s to 745s - we find a 77% increase. However, to complete 50 units of work requires 6 hours on the E5430 and 11 hours on the E5507, with respective costs being at least: $0.36 and $0.66, an 83% increase in the cost.

V. EC2 PERFORMANCE DISCOVERY SERVICES

Based on the foregoing, we would envisage one performance service which could be used for EC2 as follows: A user requests performance information for a workload on a range of instance types. We assume the workload would be representative of an application the user wishes to run. For example, a user may have determined on their local systems that their application requires high memory bandwidth, so STREAM workloads, as described in section IV, would be of interest. In general, these workloads could be either well known benchmarks, a dwarf kernel [17] or a self-produced effort. How well the workload and the application correlate is the responsibility of the user not the broker at this point.

The broker will then determine the performance ordering of the CPU models associated to the class. So for example, a user requesting a povray run against the standard benchmark input file on m1.small instances would have the following returned to them: E5645, E5-2650, E5507 and E5430.

In this first step we have identified which CPU models give best price/performance. Next, and based on historical data, the broker will inform the user in which AZ they are **most likely** to find the better performing CPU models, as

 ISBN: 978-1-61208-338-4

discussed in section IV sub-section B. The historical data could also be used to provide estimates of the probability of actually obtaining a given model.

Additional services can easily be imagined, such as obtaining instances with given CPU models on behalf of the user. Further, as we have demonstrated, performance variation is greatest amongst different CPU models, and exists in instances backed by the same model. In this case the variation could be a function of resource contention, or simply variation in quality of other system parts, and so is a run time property. Finding best performing instances at run time is another potential performance service.

VI. CONCLUSIONS AND FUTURE WORK

There is an overwhelming variety of instance types on offer currently, in various Infrastructure Clouds, and a lack of transparent performance information. This make choosing instances types that offer best price/performance for given applications difficult. This would seem to be a clear opportunity for CSBs to add performance related services on top of existing Cloud offers. To be successful, we would argue that the notions of performance must be in-line with ones that are relevant to the applications that users wish to run. As discussed in section III, these are most likely to involve one or more of execution times, throughput and work done.

We have shown that performance of an instance depends on both the CPU model backing the instance and the application. And so determining best price/performance requires knowledge of both, as well as an ability to predict where the 'best' CPU models for the application can be found. Based on our work here, and on previous results, we proposed a performance discovery service. As an example, we showed that for the best memory bandwidth performance for m1.small instances (our EC2 account), make requests to us-east-1e.

In future work, we wish to explore these ideas further, and in particular to tackle the problem of performance monitoring with respect to the 'good' metrics described here. This is needed for CSBs who would need to offer Service Level Agreements (SLAs) with performance guarantees for instances obtained on behalf of users. In the same way that providers describe performance in terms of machine characteristics, we find most performance monitoring focuses on system metrics. It is unclear how system metrics relate to the *good* performance metrics as described here, and we hope to address suitable performance monitoring also.

REFERENCES

[1] J. O'Loughlin and L. Gillam, "Towards performance prediction for Public Infrastructure Clouds: an EC2 case study," Proc. IEEE Fifth International Conference on Cloud Computing Technology and Science (CloudCom 2013), Dec2013, pp. 475-480.

[2] "IT Glossary," www.gartner.com. [Online]. Available: http://www.gartner.com/it-glossary/cloud-services-brokerage-csb [Accessed: 27th January 2014].

[3] "Amazon EC2 Instance," aws.amazon.com. [Online]. Available: http://aws.amazon.com/ec2/instance-types/ [Accessed: 27th January 2014].

[4] "Virtual Machine Pricing Details," www.windowsazure.com. [Online]. Available: http://www.windowsazure.com/en-us/pricing/details/virtual-machines/ [Accessed: 27th January 2014].

[5] Amazon EC2 FAQs," aws.amazon.com. [Online]. Available: http://aws.amazon.com/ec2/faqs/#What_is_an_EC2_Compute_Unit_and_why_did_you_introduce_it [Accessed:27th January 2014].

[6] "Google Cloud Platform," cloud.google.com. [Online]. Available: https://cloud.google.com/pricing/compute-engine [Accessed:27th January 2014].

[7] "HP Cloud Pricing," www.hpcloud.com. [Online]. Available: https://www.hpcloud.com/pricing [Accessed:27th January 2014].

[8] D. Lilja, Measuring Computer Performance, New York, Cambridge University Press, 2000.

[9] J. Hennessy and D. Patterson, Computer Architecture a Quantitative Approach, 5th Ed. Waltham, Elsevier, 2012.

[10] I. Goiri, F. Julii, J. Fito, M. Macias and J. Guitart, "Supporting CPU-based guarantees in Cloud SLAs via resource level QoS metrics", Future Generation Computer Systems,Vol 28, pp. 1295-1302, 2012.

[11] Q. Snell and J. Gustafson, "A new way to measure computer performance", Proc. Hawaii International Conference on Systems Science, pp.392-401, 1995.

[12] "CloudLayer Computing", softlayer.com. [Online]. Avaialble: http://www.softlayer.com/cloudlayer/computing [Acessed: 27th January 2014].

[13] Global Infrastructure," aws.amazon.com. [Online]. Available: http://aws.amazon.com/about-aws/globalinfrastructure/ [Accessed: 27th January 2014].

[14] M Armbrust et al, "Above the clouds: a Berkely view of cloud computing". Technical Report EECS-2008-28, EECS Department, University of California, Berkeley.

[15] "SPEC CPU2006," www.spec.org. [Online]. Available: http://www.spec.org/cpu2006/ [Accessed: 27th January 2014].

[16] J. Hoxmeier and C. DiCesare, "System response time and and user satisfatcion: an experimental study of browser based applications", Proc. Of the Association of Information Systems Americas Conference, Long Beach California, pp.140-145, 2000.

[17] Dwarf-Mine," view.eecs.berkeley.edu/wiki. [Online]. Available: http://view.eecs.berkeley.edu/wiki/Dwarfs [Accessed: 27th January 2014]

 ISBN: 978-1-61208-338-4