Performance Evaluation Methodology for Cloud Computing using Data Envelopment Analysis

Leonardo Menezes de Souza
Universidade Estadual do Ceará (UECE)
Fortaleza/CE - Brazil
Email: leonardo@insert.uece.br

Marcial Porto Fernandez
Universidade Estadual do Ceará (UECE)
Fortaleza/CE - Brazil
Email: marcial@larces.uece.br

Abstract—**Cloud Computing is a new distributed computing model based on the Internet infrastructure. The computational power, infrastructure, applications, and even collaborative content distribution is provided to users through the Cloud as a service, anywhere, anytime. The adoption of Cloud Computing systems in recent years is remarkable, and it is gradually gaining more visibility. The resource elasticity with the cost reduction has been increasing the adoption of cloud computing among organizations. Thus, critical analysis inherent to cloud's physical characteristics must be performed to ensure consistent system deployment. Some applications demand more computer resources, other requests more storage or network resource. Therefore, it is necessary to propose an approach to performance measurement of Cloud Computing platforms considering the effective resource performance, such as processing rate, memory buffer refresh rate, disk I/O transfer rate, and the network latency. It is difficult to discover the amount of resources are important to a particular application. This work proposes a performance evaluation methodology considering the importance of each resource in a specific application. The evaluation is calculated using two benchmark suites: High-Performance Computing Challenge (HPCC) and Phoronix Test Suite (PTS). To define the weight for each resource, the Data Envelopment Analysis (DEA) methodology is used. The methodology is tested in a simple application evaluation, and the results are analyzed.**

Keywords*–*Cloud Computing; Performance evaluation; Methodology.

I. INTRODUCTION

The cloud computing infrastructure meets several workload requirements simultaneously, which of these are originated from Virtual Machine (VM). The evaluation addressed in this work is focused on criticality and performance on the cloud platform virtualized resources. Such evaluation is required because the performance of virtualized resources is not transparent to the network management, even when using a software monitor. Thus, it is demanded a methodology which allows to quantify the performance according to the platform particularity, using it to performance periodic measurements and to assure the promised available and reducing malfunctioning risks.

In this work, we propose a generic methodology to assess the performance of a cloud computing infrastructure; standardizing the method and covering a wide range of systems. Such methodology will serve any cloud computing structure, since it is oriented to the resources' performance. The assessment must consider the influence of each resource on the overall system performance. Then it is determined which of these resources has greater relevance to the system, aiding in deciding which infrastructure model will provide the best consumption efficiency to users, developers and managers.

We consider the average performance of the hardware and network critical points, such as processing, memory buffer refresh rate, storage Input/Output (I/O) and network latency. We used two benchmarking suites to evaluate these important points: High Performance Computing Challenge (HPCC) and Phoronix Test Suite (PTS).

HPCC uses real computing kernels, allowing variable inputs and runtimes according to system capacity [1]. It consists of seven benchmarks responsible for each critical component individual analysis according to its specificity.

The PTS [2] is the basic tool of the *Cloud Harmony* [3] website, which analyzes public cloud systems all over the world. It consists of over 130 system analysis tests, which were selected by its effective handling and compatibility of results, with higher stability and likelihood when compared to benchmarks with the same goal.

From the results obtained in both benchmark suites, we analyze it using Data Envelopment Analysis (DEA), which will assign weights according to each resource's relevance in the infrastructure; then transcribe a formulation considering each resource's average performance in each deployed VM instance. The formulation considers the overhead attached to each evaluated resource, culminating in its real performance representation. The proposal was validated in a experiment done in a Datacenter running a typical Web application.

The rest of the paper is structured as follows. In Section II, we present some related work, and Section III introduces the proposed performance evaluation methodology. Section IV shows the results and Section V concludes the paper and suggests future work.

II. RELATED WORK

Ostermann [4] and Iosup [5] created a virtual platform using the Amazon Elastic Compute Cloud (EC2) [6] instances. In this scenario, the infrastructure is shared by many independent tasks, and the benchmarks will run over the Multi-Job Multi-Instance (MJMI) sample workloads. It was noticeable two main performance characteristics: the workload makespan stability, and the resource's aquisition/liberation overhead.

The performance of several cloud computing platforms, e.g., Amazon EC2, Mosso, ElasticHost and GoGrid, were suitable to using the HPCC benchmark suite. It was noticeable that cloud computing is a viable alternative to short deadline applications, because it presents low and stable response time. It brings a much smaller delay for any cloud model when compared to scientific environment, meeting effectively to the stability, scalability, low overhead and response time criteria. The contribution of these works stands for the methodology and the metrics evaluation, besides the pioneering idea of analyzing the performance of cloud computing systems [5].

Benchmark's references for performance verification and infrastructure limitations were made in [7]. The benchmarks were classified in three categories according to the moment of the infrastructure (deployment, individual or cluster). All of them brings a sense of loss carried by virtualization. In this work, it was executed simulations to assess the Central Processing Unit (CPU)/Random Access Memory (RAM), storage I/O and network usage metrics. It was verified that CPU usage tests have a little overhead introduced by virtualization. The I/O tests show performance gain caused by virtualization. Such fact possibly occurs because virtualization creates a new cache level, improving the I/O performance. On the other hand, there are components, which execute I/O functions that are affected by large cache, reducing performance and becoming the cache useless. It is difficult to predict the performance behavior in a specific I/O task.

The increasing complexity and dynamics in deployment of virtualized servers are highlighted in Huber [8]. The increasing of complexity is given by gradual introduction of virtual resources, and by the gap left by logical and physical resource allocation. The dynamics increasing is given by lack of direct control over hardware and by the complex iterations between workloads and applications. Results of experimentations using benchmarks presented that performance overhead rates to CPU virtualization is around 5%. Likewise, the performance overhead to memory (RAM), networks and storage I/O virtualizations reach 40%, 30% and 25%, respectively.

Different from cited works, this paper presents a proposal to evaluate a cloud computing system considering the application demand. Although it is possible to use HPCC or PTS metrics and calculate an index weighted by parameters based in operator experience, the results are not precise. Our proposal uses DEA methodology to define the relevance of each parameter and calculate a unique value to compare against other cloud providers.

III. A Methodology to Evaluate the Performance of a Cloud Computing System

Amazon Elastic Compute Cloud (Amazon EC2) is a service provided by Amazon cloud computing platform. The users can access the platform by the Amazon Web Services (AWS) interface. Amazon's offer the Amazon Machine Image in order to create a Virtual Machine (VM), which is called an *instance*, containing user's software. A user can create, deploy, and stop server instances as needed. They pay the service by the amount of hours of active server instance it used.

In each Amazon's VM, or VM instance, works as a virtual private server. To facilitate for user to choose the amount of resources they would buy, Amazon defines a set of instance size based on Elastic Compute Units. Each instance type offers different quantity of memory, CPU cores, storage and network bandwidth. The Amazon's pre-defined VM types used in this work are shown in Table I.

First, we deploy VMs based on the model provided by Amazon EC2 [6]. The overall performance of the resources is not used, since virtualization generates communication overhead in the resource management. After the allocation of resources in need, we installed the benchmark suites to run the tests.

TABLE I. Amazon EC2 Virtual Machine Model [6].

MVs	ECUs(Cores)	RAM[GB])	Arq[bit]	Disk[GB]
m1.small	1 (1)	1,7	32	160
c1.medium	5 (2)	1,7	32	350
m1.large	4 (2)	15	64	850
m1.xlarge	8 (4)	15	64	1690
c1.xlarge	20 (8)	7	64	1690

According to Jain [9], the confidence interval only applies to large samples, which must be considered from 30 (thirty) iterations. Therefore, we ran the experiments for each resource of each VM instance at least thirty times, ensuring the achievement of a satisfactory confidence interval (95%). Then, we can state that each benchmark will follow this mandatory recommendation to achieve an effective confidence interval. After the tests, we calculate the mean and the confidence interval of the obtained results, presenting a high reliability level.

In order to ponder the performed experiments, we opted for the DEA methodology; using the BCC model output-oriented (BCC-O), which involves an alternative principle to extract information from a population of results. Then, we determine the weights inherent to the VMs and the resources analyzed. We used the results of each benchmark iteration in each VM as an input, achieving the weights for each benchmark. Finally, we apply this procedure in the formulation which will be detailed later.

In short, we analyze a cloud performance simulating the behavior of applications by running benchmarks. We did an efficiency analysis from the achieved results, assigning weights to each one of them. Then, we proposed a formulation which showed the consumption ratio of each platform resource, considering the associated overhead. The execution order of activities is shown in Figure 1.

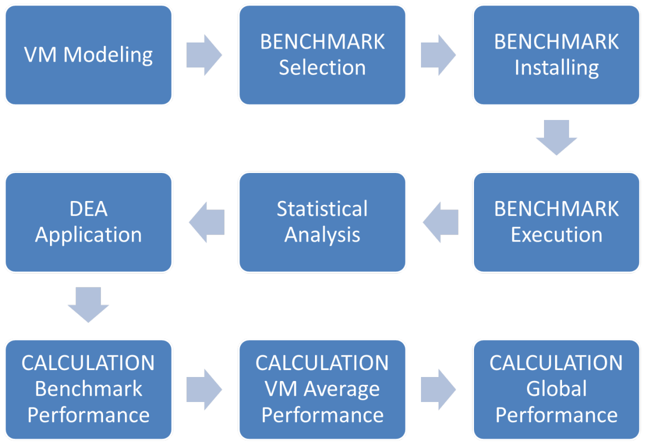

Figure 1. Cloud Computing performance evaluation methodology flowchart.

A. Benchmarks

In this work, we use two benchmark suites, the HPCC [1] and PTS [2]), which will measure the performance of critical points in a cloud computing system. These benchmarks require the Message Passing Interface (MPI) [10] and Basic Linear Algebra Subprogram (BLAS) [11] library's availability to run the tests. The benchmarks from HPCC suite ran both in local and online environments and has shown favorable results to its utilization. Then, the benchmark results showed independence and adaptability within the cloud nodes.

The HPCC benchmark suite comprises seven different tests that will stress the system hardware critical points such as is presented as follows:

- High-Performance Linpack (HPL) [12] uses 64-bit double precision arithmetics in distributed memory computers to measure the floating point rate of execution for solving matrices through random dense linear equations systems.
- Double-precision General Matrix Multiply (DGEMM) [13] simulates multiple floating point executions, stressing the process through double-preciosion matrix multiplication.
- PTRANS [14] has several kernels where pairs of processors communicate with each other simultaneously, testing the network total communication capability. It transposes parallel matrices and multiplies dense ones, apllying interleaving techniques.
- Fast Fourier Transform (FFT) [15] measures the floating point rate through unidimensional double-precision discrete Fourier transforms (DFT) in arrays of complex numbers.
- STREAM [16] measures the memory bandwidth that supports the processor communication (in GB/s). It also measures the performance of four long-vector operations. The array is defined to be larger than the cache of the machine which is running the tests, privileging the memory buffer updates through interdependence between memory and processor.
- Random Access [17] measures the performance of random memory (main) and access memory (cache) buffer updates in multiprocessor systems. The results are given in Giga Updates Per Second (GUPS), calculated by updated memory location identification in one second. This update consists in a Read-Modification-Write (RMW) operation controlled by memory buffer and the processor.
- Effective Bandwidth Benchmark (b_{eff}) [18] measures the bandwidth efficiency (effective) through estimated latency time for processing, transmission and reception of a standard message. The message size will depend on the quotient between memory-processor ratio and 128.

Beyond the HPCC, we also used another benchmark suite to run the remaining tests and enable a bigger coverage of evaluated resources. The PTS suite comprises more than 130 system analysis tests. We have selected the benchmarks to be part of this experiment according to its importance within the benchmarking set, minimizing inconsistencies and improving our sample space. Finally, we achieve the three most adaptive benchmarks which will be presented as follows:

- Loopback Transmission Control Protocol (TCP) Network Performance [19] is a simple Peer-to-Peer (P2P) connectivity simulation which measures the network adapter performance in a loopback test through the TCP performance. This test is improved on this benchmark to transmit 10GB via loopback.
- RAM Speed SMP [20] measures the performance of the interaction between cache and main memories in a multiprocessor system. It allocates some memory space and starts a write-read process using 1Kb data blocks until the array limit, checking the memory subsystem speed.
- PostMark [21] creates a large pool of little files constantly updating just to measure de workload transaction rate, simulating a big Internet e-mail server. The creation, deletion, read, and attaching transactions have minimum and maximum sizes between 5Kb and 512Kb. PostMark executes 25.000 transactions with 500 files simultaneously, and after the transactions, the files are deleted, producing statistics relating its contiguous deletion.

In short, we present all benchmarks used in this work and its basic characteristics on the Table II.

TABLE II. BENCHMARKS CHARACTERISTICS.

RESOURCE	BENCHMARK	UNIT
CPU	HPL DGEMM PTRANS	GFLOPs
	FFT	GB/s
MEM	STREAM RAM Speed SMP	GB/s
	Random Access	GUPS
STO	PostMark	Transactions/s
NET	b_{eff}	μs
	Loopback TCP	s

B. Resources Overhead

Simplifying the organization of the resources' performance analysis in a cloud computing system, we can split them into two requirement groups: CPU and I/O resources. Performance studies utilizing general benchmarks show that the overhead due to CPU virtualization reach 5% as was mentioned before at Section II. The host hypervisor directly controlling the hardware and managing the actual operational system, showing low overhead.

Virtualization also imposes I/O overhead, concerning memory, networks and storage. Cloud applications have specific requirements, according to their main goal. In this way, the network is critical to every single cloud application because it determines the speed with which each remaining I/O resource will work. In other words, the network must provide capability, availability, and efficiency enough to allocate resources without compromising delays.

The online content storage is just one of the most popular features of cloud computing systems. Its performance is so much dependent on memory buffer updates rate as regarding the processing rate that feeds the buffer. These two active functions affect significantly the storage services on the cloud.

Lastly, but not less important, memory is the most required resource on a cloud computing system. In distributed systems, it is considered a critical issue, because it works along with processing in the updates of running applications, user requirements, and in the data read/write coming through network adapter or storage component. So, many functions overload the resource, representing the biggest bottleneck in whole cloud computing infrastructure.

TABLE III. VIRTUALIZATION OVERHEADS [8].

RESOURCE		OVERHEAD (%)
I/O	Memory	40
	Network	30
	Storage	25
CPU	Processing	5

Each hardware resource available in the cloud computing infrastructure possesses a unique utilization quota regarding its own functioning. However, they feature interdependencies between to each other. Table III shows the overhead portions to each resource analyzed in this work. Then, we address weights based on the significance of each resource in a cloud computing infrastructure using the DEA methodology.

C. DEA Methodology

The DEA methodology is a linear programming mathematical technique which consists of a multicriteria decision support, analyzing multiple inputs and outputs simultaneously. In this way, the DEA is capable of modeling real-world problems meeting the efficiency analysis [22].

This methodology provides comparative efficiency analysis from complex organizations obtained by its unit performance revelation so that its reference is obtained by the observation of best practices. The organizations once under DEA analyses are called Decision Making Unit (DMU)s and must utilize common resources to produce the same results. With this, will be defined efficient DMUs (those which produce maximum outputs by inputs) and the inefficient ones. The first ones are located on the efficiency frontier while the later ones under that same frontier.

In this work, we chose one model among all DEA methodology models, which is the Multipliers BCC-O model. The output orientation was chosen because of the input variables (VM instances) are fixed. The main goal is to obtain the best benchmarks' performance executed on the VMs, then we intend to obtain the larger amount of outputs by inputs. By the way, the DEA methodology was applied to parametrize the benchmarks results calculated for each resource in all VM instances.

The required terms to the weighting on the proposed formulation are generated by the BCC-O model. This mathematical model consists of the calculation of the input (VM resources) and output (benchmarks results) variables weights. In the model objective function we minimize the input weighted sum (product from input value by its respective weight) subjected to four restrictions, presented on the formulation shown in (1).

Running the model shown earlier in a linear programming solver, we can get the weight sum equal to 1, showed in (1b). The restriction of the inequality (1c) will be performed for each one of the 1500 total iterations from running instances. This model allows weights to be chosen for each DMU (VM iteractions) in a way that suits it better. The calculated weights must be greater than or equal to zero as it is shown on inequalities (1e) and (1f). The efficiency ratios of each DMU is calculated by the objective function too. Thus, the number of models to be solved is equal to the number of problem DMU.

In order to achieve the best performance of the resulting benchmarks (outputs) ran on the five VMs showed in Table I. The weights are obtained by a weighted average according to the significance of each test on the system. The greater values will have the higher weights. We consider each one of the ten benchmarks executed ran, at least, 30 times for each one of the five VMs used in this experiment, accounting for 1500 iterations. Each one of these had its respective weight calculated by DEA, then we ran a solver (BCC-O) to calculate the inputs and outputs weighted sum obeying the methodology constraints.

$$\text{Minimize} \quad ef(0) = \sum_{i=1}^{m} v_i X_{i0} + v \tag{1a}$$

$$\text{Subject to} \sum_{j=1}^{S} u_j Y_{j0} = 1 \tag{1b}$$

$$\sum_{j=1}^{S} u_j Y_{jk} - \sum_{i=1}^{m} v_i X_{ik} - v \leq 0 \tag{1c}$$

$$k = 1 \ldots n \tag{1d}$$

$$u_j \geq 0, \forall j \tag{1e}$$

$$v_i \geq 0, \forall i \tag{1f}$$

Where: $v \in \Re$, v unrestricted
u_j = output j weight
v_i = input i weight
$k \in \{1 \ldots n\}$ DMUs
$j \in \{1 \ldots s\}$ outputs of DMUs
$i \in \{1 \ldots m\}$ inputs of DMUs
Y_{jk} = output j value of DMU k
X_{ik} = input i value of DMU k

Concerning the constraints, first of all, the outputs' weighted sum must be equal to one, setting a parameter for assigning weights in each VM. The inputs and outputs' weights must be greater than or equal to zero. Lastly, the subtraction between the inputs and outputs' weighted sums and the scale factor, must be lower than or equal to zero. The scale factor will not be considered because it will just determine if the production feedback is increasing, decreasing or constant to a set of inputs and products. This way, weights are the factors considered on the formulation.

D. Formulation

In a cloud computing system, the required resources are allocated automatically according to user needs. All of them have a standard overhead and significance variable level according to hosted application guidance. To analyze the system performance, we used a mathematical formulation that provides evidence from utilization levels measured, and from the iteractions among resources. The DEA was used to define the weights of Performance Index.

We must consider that benchmark execution will simulate an application that overloads the assessed resource. Then, we adopted PI_{R_G} as the Resource Global Performance Index, whose variable will assume the resulting value from the product between RPI_R (Resource Real Performance Index) and the API_{R_j} (Average Performance Index by Resource in each VM Instance), as shown in (2).

$$PI_{R_G} = RPI_R \times API_{R_j} \tag{2}$$

The term RPI_R is the result from the subtraction between the maximum theoretical performance (100%) and the overhead associated to each running resource, shown on the Table III. The relation is shown in (3).

$$RPI_R = (100\% - Ov_R\%) \tag{3}$$

The term API_{R_J} is calculated by the mean of each BPI_{R_j} (Benchmark Performance Index by Resource in each Instance), as it is shown in (4). BPI_{R_j} is calculated by the product sum between weights (U_{iR_j}) obtained from DEA methodology for benchmarks (i) by resource (R) in each instance (j). The term n_j stands for the amount of VMs where benchmarks were hosted. In this case, five VMs were implemented to run the tests based on the Amazon EC2 infrastructure.

$$API_{R_j} = BPI_{R_j} \div n_j \tag{4}$$

The results (X_{iR_j}) obtained from benchmarks (i), by resource (R) in each instance (j), as shown in (5), where p is the number of benchmarks and q is the number of instances. The X_{iR_j} is normalized related to maximum theoretical performance in order to permit an index independent from benchmark units, e.g., GB/s, GFLOPS, Sec.

$$BPI_{R_j} = \sum_{\substack{1\le i\le p\\ 1\le j\le q}} (U_{iR_j} \times X_{iR_j}) \tag{5}$$

The benchmark suites were set up to simulate each resource behavior in a cloud computing infrastructure. We will calculate the (BPI_{R_j}) Benchmarks Performance Index to each resource (R) in each instance (j), considering each benchmark running to its respective resource, and after that we calculated the mean for each resource, obtaining the API_{R_j} dividing each BPI_{R_j} by the number of VM instances n_j. In following formulation, CPU means computing resource, MEM means memory, STO means storage resource and NET means network resource.

$$\begin{aligned}
BPI_{CPU_j} &= (U_{HPL} \times X_{HPL}) + (U_{DGEMM} \times X_{DGEMM})\\
&\quad + (U_{FFT} \times X_{FFT}) + (U_{PTRANS} \times X_{PTRANS})\\
BPI_{MEM_j} &= (U_{STREAM} \times X_{STREAM}) + (U_{RA} \times X_{RA})\\
&\quad + (U_{RSMP} \times X_{RSMP})\\
BPI_{STO_j} &= (U_{BB} \times X_{BB}) + (U_{PM} \times X_{PM})\\
BPI_{NET_j} &= (U_{BE} \times X_{BE}) + (U_{LTCP} \times X_{LTCP})
\end{aligned}$$

$$\begin{aligned}
API_{CPU_j} &= \sum BPI_{CPU_j} \div n_j\\
API_{MEM_j} &= \sum BPI_{MEM_j} \div n_j\\
API_{STO_j} &= \sum BPI_{STO_j} \div n_j\\
API_{NET_j} &= \sum BPI_{NET_j} \div n_j
\end{aligned}$$

The next step consists in solving the global performance expression:

$$\begin{aligned}
PI_{CPU_G} &= RPI_{CPU} \times API_{CPU_j}\\
PI_{MEM_G} &= RPI_{MEM} \times API_{MEM_j}\\
PI_{STO_G} &= RPI_{STO} \times API_{STO_j}\\
PI_{NET_G} &= RPI_{NET} \times API_{NET_j}
\end{aligned}$$

IV. Results and Discussion

All the results are based on the initial set of benchmarks showed in Section III. As we could see in Table I, we created a homogeneous environment from 1 to 21 cores based on five Amazon EC2 instances, where we run the benchmarks which will evaluate the performance on the cloud environment.

The hardware used was a Dell Power Edge M1000e enclosure with six blades powered by Intel Xeon x5660 2.8 GHz processor and 128 GB 1333 MHz DDR3 RAM. All blades have 146 GB SAS HDs. The storage was a Dell Compellent with six 600 GB SAS disk and six 2.0 TB NL-SAS disk. The OS was the Linux Ubuntu 12.04 over VMWare ESXi 5.0.0 hypervisor.

The application chose was an XAMPP 1.8.1 Web server [23]. After running each benchmark, we generate Table IV which shows the efficiency index of each experiment related to maximum theoretical performance. The normalization is necessary to compare different units from benchmarks. Then, we calculated its efficiency percentage to use it on the proposed formulation.

In order to consider the results from the benchmark experiments, we used DEA methodology through BCC-O model (output-oriented). Beyond the efficiency index calculation, we calculate the output variable weights (benchmark results). In this way, we minimize the inputs weighted sum dividing it by the outputs' weighted sum of the benchmark at hand. After that, we ran a BCC-O solver to address weights to each benchmark, considering each VM instance according to its influence in the obtained results shown in Table IV. Table V shows the weights calculated by the BCC-O solver that will influence the performance of each resource attached to each benchmark in each VM.

The benchmark results were shown in Table IV and the efficiency index were calculated by DEA methodology (BCC-O) in Table V. Applying these results on (5), its two factors will assume values for benchmark performance to each resource in each instance (X_{iRj}), considering the DEA assigned weight to each benchmark result (U_{iRj}). We can observe the more the resource is used, greater is the weight assigned to it.

We can see in Figure 2 the network performance is clearly greater than the rest, and the memory is the only resource that has an index relatively close. These resources are the most affected ones by the overhead issue, justifying their bottleneck condition. The Figure 3 shows the relevance of each instance through benchmark execution. The c1 instances have very similar performances because they both have a processor/memory ratio which allows achieving quite satisfying performance levels.

From these results we verified, the memory and network performances are the most relevant to a cloud computing system. These two resources, when well balanced, leverage the cloud computing infrastructure managing workloads, reaffirming its bottleneck condition. In this way, this proposal gives more information regarding resource performance relevance in application when comparing to the work of Huber [8].

TABLE IV. BENCHMARK RESULT FOR EACH VM (X_{iRj}) RELATED TO MAXIMUM THEORETICAL PERFORMANCE.

BENCHMARKS		m1.small	c1.medium	m1.large	m1.xlarge	c1.xlarge
CPU	HPL	4.64%	11.27%	14.84%	24.81%	27.51%
	DGEMM	1.15%	13.27%	4.30%	8.54%	11.08%
	FFT	0.94%	3.62%	2.49%	4.52%	4.59%
	PTRANS	6.83%	27.86%	14.71%	39.63%	38.52%
MEM	RAMSpeed SMP/Integer	22.01%	28.38%	25.7%	30.4%	30.77%
	RAMSpeed SMP/Float	24.46%	28.96%	26.30%	27.13%	31.36
	STREAM	19.53%	28.27%	44.02%	37.53%	41.36%
	RandomAccess	0.41%	9.82%	3.73%	17.3%	17.6%
NET	b_{eff}	98.2%	99.9%	98.8%	99.5%	99.4%
	Loopback TCP	0.58%	62.07%	92.65%	94.34%	96.02%
STO	PostMark	3.75%	4.42%	13.99%	13.00%	14.26%

TABLE V. WEIGHTS ADDRESSED TO RESOURCES TO EACH VM (U_{iRj}).

BENCHMARKS		m1.small	c1.medium	m1.large	m1.xlarge	c1.xlarge
CPU	HPL	0.77	0.13	0.66	0.28	0.51
	DGEMM	0.88	0.42	0.23	0.29	0.20
	FFT	0.003	0.58	0.25	0.5	0.58
	PTRANS	0.15	0.32	0.07	0.33	0.43
MEM	RAMSpeed SMP/Integer	0.38	0.78	0.17	0.42	0.37
	RAMSpeed SMP/Float	0.46	0.96	0.3	0.68	0.65
	STREAM	0.14	0.18	0.67	0.33	0.61
	RandomAccess	0.91	0.93	0.37	0.73	0.56
NET	b_{eff}	0.42	0.59	0.48	0.55	0.43
	Loopback TCP	0.24	0.43	0.33	0.28	0.48
STO	PostMark	0.57	0.24	0.19	0.42	0.62

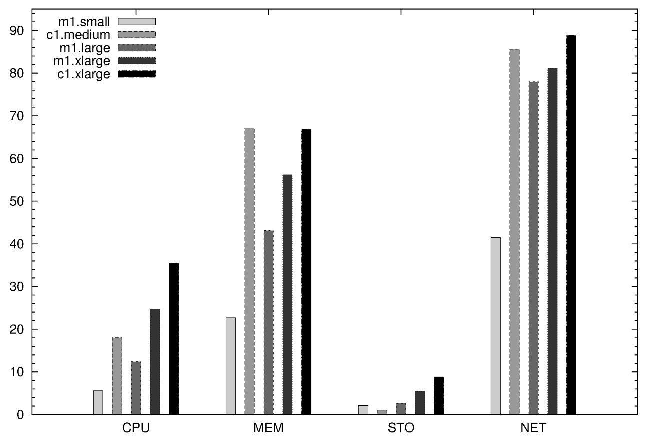

Figure 2. Benchmark Performance by Resource.

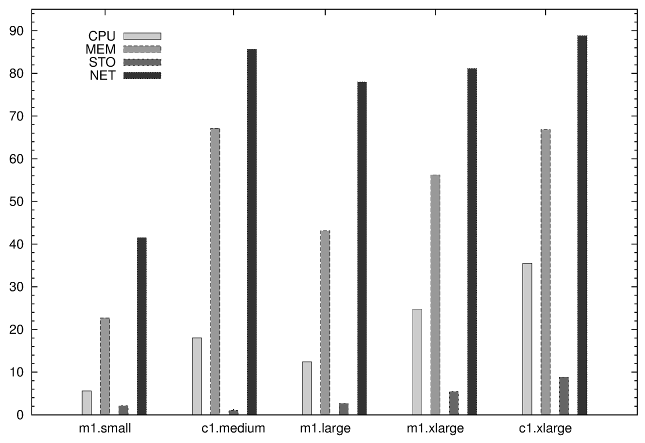

Figure 3. Benchmark Performance by Instance.

V. CONCLUSION AND FUTURE WORK

In this work, we could observe that the benchmarks had met the simulation needs very well, overloading the resources efficiently, returning real-world results. The DEA methodology helped us to analyze the efficiency of each experiment, providing an efficiency index (weight) to benchmarks in each instance implemented, for each resource evaluated. Finally, the proposed formulation highlighted the impact of resource's overhead on the global performance evaluation.

Then, we concluded that, in a generic Web application, the memory and network resource performance is the most relevant to a cloud computing system, and for this reason, they are considered the bottlenecks. We confirmed that the resource performance evaluated here is directly proportional to the overhead execution rates, assigned in [8].

Since develop an application to be hosted on a cloud environment to measure its resource consumption rate, or its behavior during a VM migration process, until configure the benchmarks in a more aggressive way, generating more data blocks. We should, then, pay attention to cloud computing system constant evolution to make possible the use of the approach proposed in this work.

REFERENCES

[1] J. Dongarra and P. Luszczek, "HPCC High Performance Computing Challenge," Last accessed, Mar 2015. [Online]. Available: http://icl.eecs.utk.edu/hpcc

[2] M. Larabel and M. Tippett, "Phoronix Test Suite," Last accessed, Mar 2015. [Online]. Available: http://www.phoronix-test-suite.com

[3] J. Read, "Cloud Harmony: Benchmarking the Cloud," Last accessed, Mar 2015. [Online]. Available: http://www.cloudharmony.com

[4] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "An early performance analysis of EC2 cloud computing

services for scientific computing," Cloud Computing, 2010, pp. 115–131.

[5] A. Iosup, S. Ostermann, M. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 6, 2011, pp. 931–945.

[6] Amazon, "Amazon Elastic Compute Cloud EC2," Last accessed, Mar 2015. [Online]. Available: http://aws.amazon.com/ec2

[7] N. Cardoso, "Virtual clusters sustained by cloud computing infrastructures," Master's thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, Dec 2011.

[8] N. Huber, M. von Quast, M. Hauck, and S. Kounev, "Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments," in 1st International Conference on Cloud Computing and Services Science, 2011, pp. 7–9.

[9] R. Jain, The Art of Computer Systems Performance Analysis. John Wiley & Sons, 2008.

[10] J. Dongarra, R. Hempel, T. Hey, and D. Walker, "The Message Passing Interface (MPI) Standard," Last accessed, Mar 2015. [Online]. Available: https://mcs.anl.gov/research/projects/mpi

[11] C. Lawnson, R. Hanson, D. Kincaid, and F. Krogh, "Basic Linear Algebra Subprograms for FORTRAN Usage," ACM Transactions on Mathematical Software (TOMS), vol. 5, no. 3, 1979, pp. 308–323.

[12] A. Petitet, R. Whaley, J. Dongarra, and A. Cleary, "HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers," Last accessed, Mar 2015. [Online]. Available: http://netlib.org/benchmark/hpl/

[13] J. Dongarra, I. Duff, J. Croz, and S. Hammarling, "Subroutine DGEMM," Last accessed, Mar 2015. [Online]. Available: http://www.netlib.org/blas/dgemm.f

[14] T. Hey, J. Dongarra, and H. R., "Parkbench Matrix Kernel Benchmarks," Last accessed, Mar 2015. [Online]. Available: http://www.netlib.org/parkbench/html/matrix-kernels.html

[15] M. Frigo and S. Johnson, "benchFFT," Last accessed, Mar 2015. [Online]. Available: http://www.fftw.org/benchfft/

[16] J. McCalpin, "STREAM: Sustainable Memory Bandwidth in High Performance Computers," Last accessed, Mar 2015. [Online]. Available: http://www.cs.virginia.edu/stream/

[17] D. Koester and B. Lucas, "Random Access," Last accessed, Mar 2015. [Online]. Available: http://icl.cs.utk.edu/projectsfiles/hpcc/RandomAccess/

[18] R. Rabenseifner and G. Schulz, "Effective Bandwidth Benchmark," Last accessed, Mar 2015. [Online]. Available: https://fs.hlrs.de/projects/par/mpi//b_eff/

[19] M. Larabel and M. Tippett, "Loopback TCP Network Performance," Last accessed, Mar 2015. [Online]. Available: http://openbenchmarking.org/test/pts/network-loopback

[20] R. Hollander and P. Bolotoff, "RAMspeed," Last accessed, Mar 2015. [Online]. Available: http://alasir.com/software/ramspeed/

[21] J. Katcher, "PostMark: A New File System Benchmark," Last accessed, Mar 2015. [Online]. Available: http://www.netapp.com/technology/level3/3022.html

[22] W. W. Cooper, L. M. Seiford, and K. Tone, "Data envelopment analysis: A comprehensive text with models, applications, references and dea-solver software. second editions," Springer, ISBN, vol. 387452818, 2007, p. 490.

[23] K. Seidler and K. Vogelgesang, "XAMPP Distribution Apache + MySQL + PHP + Perl," Last accessed, Mar 2015. [Online]. Available: https://www.apachefriends.org