A Novel Approach for Performance Characterization of IaaS Clouds

Sandor Acs[1 2], Nemeth Zsolt[1], and Mark Gergely[1]

[1] Computer and Automation Research Institute, Hungarian Academy of Sciences, Hungary
acs.sandor@sztaki.mta.hu
[2] Obuda University, Hungary

Abstract. Infrastructure-as-a-Service (IaaS) clouds are widely used today, however there are no standardized or commonly used performance evaluation methods and metrics that can be used to compare the services of the different providers. Performance evaluation tools and benchmarks are able to grasp some aspects or details of performance but for various reasons are not capable to characterize cloud performance. Our aim is to collect these elementary or primitive facets of performance and derive high-level aggregated and qualitative performance characterization semantically far above the output of tools and benchmarks. We designed and implemented a framework that collects low-level raw performance data (in terms of CPU, disk, memory and network) of cloud providers based on standard benchmark tools and these data are aggregated and evaluated using a hierarchical fuzzy system. In this process performance characteristics are associated with symbolic values and fuzzy inference is applied to produce the normalized qualitative comparable and readable performance metrics. In this paper, we discuss the issues of cloud performance analysis, present the concept and implementation of our method, illustrate the proposed solution by comparing –in terms of performance– the general purpose medium instance type of the Amazon EC2 cloud (in Ireland) and the standard instance type of the OpenNebula installation at MTA SZTAKI.

Keywords: cloud computing, performance evaluation, fuzzy sets, hierarchical fuzzy inference

1 Introduction

Cloud computing, from a user's perspective is a contract: the provider offers a service of negotiated quality and the consumer pays a negotiated fee. Just like in any other commercial scenario, the consumer is curious if the value vs price ratio is right and acceptable. Cloud vendors provide multi-tenant infrastructures and generally do not disclose the technical details of the services (e.g., overprovisioning rate) that makes it difficult for customers to anticipate cloud performance. Furthermore, cloud providers often use different terminologies for resource allocation. Albeit, Service Level Agreements (SLAs) declare the guaranteed level of

services, these are typically just the lower limits that neither express what the consumer really gets nor make the services comparable to other providers. From the consumers' point of view the expected "performance" would be interesting so that it enables the comparison to other providers and to the costs. Albeit in this work we discuss "performance", it is worth mentioning that characterizing a service involves many other aspects that are quite related to performance such as Quality of Service (QoS), Quality of Experinece (QoE), in lesser extent robustness, fault tolerance, trust and many others.

Performance analysis of parallel computing environments has been studied extensively in the past (e.g., [9] [15] [10] [13] [2] just to mention a few.). Novel distributed paradigms obsolete parallel performance analysis models and new approaches and tools for characterizing large-scale inhomogeneous and dynamic distributed systems are required [4][11]. The advent of cloud computing introduced new features that rendered performance evaluation largely unsolved and target of intensive research.

The challenges of cloud performance analysis largely stem from virtualization, the complete separation of the physical and virtual entities. All former performance evaluation approaches were focusing on the physical infrastructure and the physical performance profile of resources. Virtualization, a characteristic and inherent feature of clouds however, introduces another dimension of complexity: the measured and evaluated performance, i.e. what a consumer gets, are *not of a physical machine*. This requires new approaches and potential, new definitions for performance evaluation for the following reasons. (i) Service providers may split or merge physical resources to accommodate virtual machines. Hence, performance measured at the physical level does not characterize the performance of virtual machines. (ii) Service providers may offer different instance types. Thus, performance of virtual machines is loosely coupled to physical machines: it is a characteristics of a hypothetical (volatile) infrastructure and potentially not characteristic to the entire or physical infrastructure. (iii) Multi-tenancy adds a large uncertainty factor. (iv) Due to all these reasons, for the time being, there are no widely accepted performance analysis techniques for cloud infrastructures. Performance itself is multi-dimensional, composed of many facets, performance metrics are not standardized, not comparable, sometimes even hard to interpret and cannot be measured at the physical level.

Our work is aimed at establishing a framework that enables the performance characterization of IaaS providers so that services of different providers or instance types of the same provider became comparable by metrics that are (i) symbolic for easy interpretation (ii) aggregated to cut down dimensions and data volume (iii) comparable to each other. Furthermore, our method is (iv) especially tailored for the virtualized machines. We apply logic and fuzzy inference to create the abstract, symbolic performance characterizations from raw performance data. The result is a readable, abstract yet precise and comparable description of virtual machines.

As it was stated we try to characterize and compare *cloud services* from performance point of view. This is not identical to the notion of performance

analysis that typically refers to the *performance of an application*. On the other hand, it is not benchmarking either. Benchmarking is a comparison of the *performance of the infrastructure as a whole* to an established, industry-leading reference point (such as the TOP500 list) where performance tests are carried out in a standardised way in a closed environment (the number and specifications of the hardware elements are known and static) under controlled (preferably non-existent) loads and eliminating all intrusive effects. Hence, informally we call our approach as *performance characterization* as opposed to, and differentiate from, benchmarking or performance evaluation.

Furthermore, benchmarking is a questionable technique (as has been for grid computing [11]) due to the fact that performance figures cannot be representative to the *entire cloud*, just for some *services* — potentially a very small subset of the cloud — *actually utilized* in the experiment and virtualization adds another factor to unreproducible experiments. Hence, in our view, the traditional notion of benchmarking physical resources is not applicable for the entire cloud. On the other hand we do use benchmark tools but not for measuring the infrastructure rather, for providing raw data on the performance of a VM instance. Benchmarking in this setting is part of the process but not the process itself.

In the followings, in Section 2 the related work is intorduced, in Section 3 the concept, design principles and technical details of our proposed performance characterization method are presented. Test cases in Section 4 give a practical validation of the approach and Section 5 concludes the paper.

2 Related work

An overview of grid performance analysis and its tools [4] already surveyed the difficulties of measuring any dynamic, heterogeneous computing infrastructure and pointed out that well-established methods of (parallel) benchmarking are not applicable where virtualization is present.

The primary goal of the CloudHarmony [1] is to make cloud services comparable, therefore they provide objective, independent performance comparisons between different cloud providers. Using these data, customers can quickly compare providers and have reasonable expectations for cloud performance. However, CloudHarmony can only provide quantitative performance data in a raw form produced by benchmark tools and can not present refined qualitative information created from processed benchmark results. As performance data are produced by multiple benchmark tools, they quite often contain discrepancies, contradictions or simply not easily interpretable by a human.

Garg et al. [3] also recognize the difficulty for customers to select service providers. In their paper, they propose a mechanism and a framework to measure the quality and prioritize Cloud services. This solution evaluates Cloud offerings and rank them based on their ability to meet the users Quality of Service (QoS) requirements. This work addresses slightly different aspects of characterizing a service than our work: less emphasis on the performance and more on the quality.

The notion of fuzzy inference for resource modeling appears in [18] in a different setting. The work is aimed at realizing a two-level resource management system with autonomic resource allocation. To this end, local and global controllers are using fuzzy logic to deal with the complexity of the virtualized data center and the uncertainties of the dynamically changing workloads. Virtual containers are treated as black boxes and their behaviour is modeled without any a priori knowledge using fuzzy logic. Ultimately, workload and related resource needs to meet QoS criteria are learned by the system. Albeit strongly related to our work, in this setting the fuzzy inference is used for resource control and not for performance characterisation — performance data are simply input to the mechanism.

Another aspect of fuzzy resource management appears in [17], namely fulfilling performance guarantees in the presence of interference of requests (especially, for non-partitionable resources) of co-hosted VMs. The aim of the fuzzy model is to detect the performance coupling of co-hosted VMs using a fuzzy a multi-input-multi-output model. The model quantifies the contention of competing resources and this information is used for VM placement and resource allocation. This approach is also similar to ours but analyses performance in a broader sense: capturing the relationship between resource allocations and the performance of the hosted applications.

The work presented in [8] applies the same mechanism to a different problem: trust and trust management but defines performance as a component of trust. Other factors are financial and agility and each such factor has many sub-metrics. The similarity to our work appears in unifying the diverse quantities into a single metrics of trust. They also propose a hierarchical (2-stage) fuzzy inference framework. This work differs mainly in the scope and the level of details of performance characterisation.

A.Vanitha et al., [6] investigate the cloud infrastructures as test bed environments for software developments. They presented a similar notion of fuzzy logic for performance evaluation. The most important difference to our work is in the fuzzy inference mechanism: they apply multidimensional inference whereas we propose hierarchical one. Their model uses a few input parameters only and they do not take the CPU performance into account in the procedure. This presumably could be the consequence of the complexity of multidimensional inference. Our solution is aimed at eliminating this obstacle by a hierarchy and hence, it can provide a generic framework for performance evaluation in cloud environments.

3 A Novel Approach to Cloud Performance characterization

3.1 Principles

The theoretical and technical difficulties of performance characterization of a complex infrastructure were presented in Section 1. In this section we narrow

the scope to processing, presenting and interpreting the performance metrics. We assume, raw performance data are gathered by some monitoring and/or benchmarking tool. Still, this data set is inappropriate due to its large dimensionality, incomparable and incompatible data types and lack of any structure. Common approaches apply statistical methods, noise filtering, feature extraction and similar numerical procedures to reduce the information to the most essential details and get readable and comparable performance figures. Performance characterisation has many dimensions and these are not independent. For instance, comparing two CPUs is possible but comparing two CPUs so that the memory is also taken into consideration is surprisingly complex: neither the CPU nor the memory speeds determine the performance but their interaction via subtle details in the actual application. These correlations are present in practically all dimensions of performance yet, their exact formulation is extremely hard analytically.

Fuzzy techniques have a vast range of features and potential application fields. We focus on their ability to transform quantitative information into qualitative one so that the resulted data is concise, readable, interpretable and comparable. Fuzzy techniques are based on the negation of the basic principle in set theory as 'a certain element is either element of a set or not'. Instead, fuzzy set theory assumes a metric, how much, or in what degree a certain element belongs to a set. This metric called *membership function* ranges between 0 (not element of the set) and 1 (element of the set). In such a way uncertain values, subjective measures can be captured and handled in a mathematical framework.

Fuzzy logic is a many-valued logic based on the fuzzy sets where logic variables have values between 0 (false) and 1 (true). It allows reasoning on uncertain or partial information where different degrees of 'true' is possible [20]. Fuzzy values may also be assigned symbolic or linguistic tags resembling intuitive classification. A fuzzy inference is a method where fuzzificated (values assigned to fuzzy sets) input variables are mapped onto output variables and the result is defuzzificated.

Recall the example above, a CPU of architecture A and frequency f_1 with memory of size M_1 and bandwidth b_1 is hardly comparable numerically to a CPU of architecture B and frequency f_2 with memory of size M_2 and bandwidth b_2; none of the numerical comparisons would yield a definite answer. However, after fuzzification – transforming the values into fuzzy sets – this question is reduced to a more comprehensible form of comparing an 'upper mediocre' processor with 'large and fast' memory to a 'lower top' processor with 'small and very fast' memory. The relationship between these sets can be precisely described by fuzzy rules resulting a similarly readable and easily comparable result.

A fundamental problem in a fuzzy inference system is that the number of rules increases exponentially with the system variables involved. The hierarchical fuzzy systems (HFS) [19] [7] have the advantage that the total number of rules is greatly reduced by a hierarchical structure, linear with the number of input variables [12]. A HFS divides the inference into stages so that a subset of input variables produce intermediate results and these results are taken as

inputs in subsequent stages whereas, the intermediate results may also possess interpretable meaning.

The core of our concept is to build a hierarchical fuzzy system so that the stages of the hierarchy correspond to certain aspects of performance. We consider CPU, memory, disk and network as main determining factors. In an experimental setup these were captured by 157 parameters. In a flat fuzzy inference system the number of corresponding fuzzy rules would be in the magnitude of n^{157} where n is proportional to the number of fuzzy sets (i.e., granularity of rules, how finely the sets are described). We categorized the parameters according to the four main aspects and established sub-categories within each (cf. Figure 2.) In such a way input parameters to an inference stage do not exceed 7 and thus, the overall number of rules in the system is bounded by $c * n^7$ where c is the number of inference stages.

3.2 Framework Design

Figure 1 provides an architectural overview and presents the components of the proposed system. At the lowest level of hierarchy input data called "raw performance data" are produced by benchmarking probes. These probes are realised as virtual machines (VM instances) and executed on some cloud resources, involving steps of authentication, deployment, and VM control. On one hand the system core (depicted as Core & Valuator) provides a part of these essential functions. The Image Repository and the IaaS client/API interface (top right) are responsible for storing the disk image for the probes and handling (deploying, launching and stopping) VM instances. The disk image contains a preconfigured Phoronix Test Suite (PTS) application (a probe) for realizing the raw measurement procedures on the infrastructure. Images are deployed on target clouds (Cloud#1, Cloud#2, etc.) prepared and run as VM instances, called PerfVMs in the followings. PerfVMs execute the appropriately configured benchmark suite and push the raw results into the central object store. The Valuator part of the Core & Valuator component realises database handling as well as evaluating fuzzy results (to be described in details in Section 3.3).

3.3 Details of Valuator

The Valuator realizes the essential functionality of the performance characterization as it aggregates raw benchmark data and associates symbolic performance values with the IaaS clouds. It provides the fuzzy inferences systems and stores the results in a database. The corresponding fuzzy rules are described in a standardized control language [5].

Figure 2 presents the proposed three layered HFS. In the first layer (L0), the raw benchmark results produced by the probes of PerfVM are taken as inputs. These are already processed by the probes so that erroneous measurements and noise are filtered out and the deviation of the results are within limits. Data are grouped by benchmark tools such as compilation performance, database performance, disk write performance, numeric performance, etc. A fuzzy inference

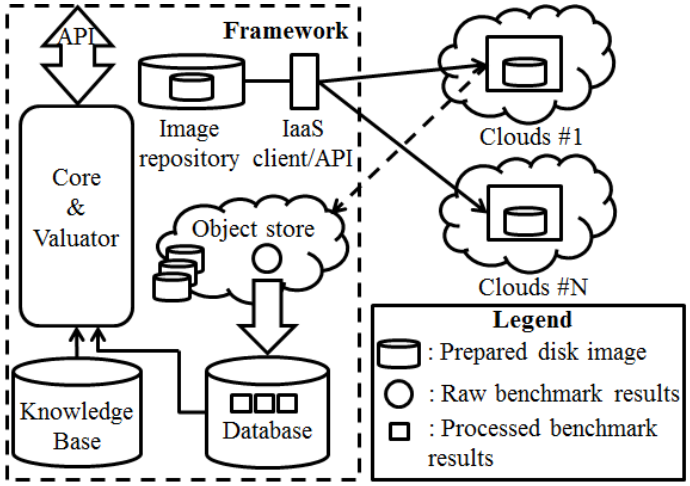

Fig. 1: Framework components

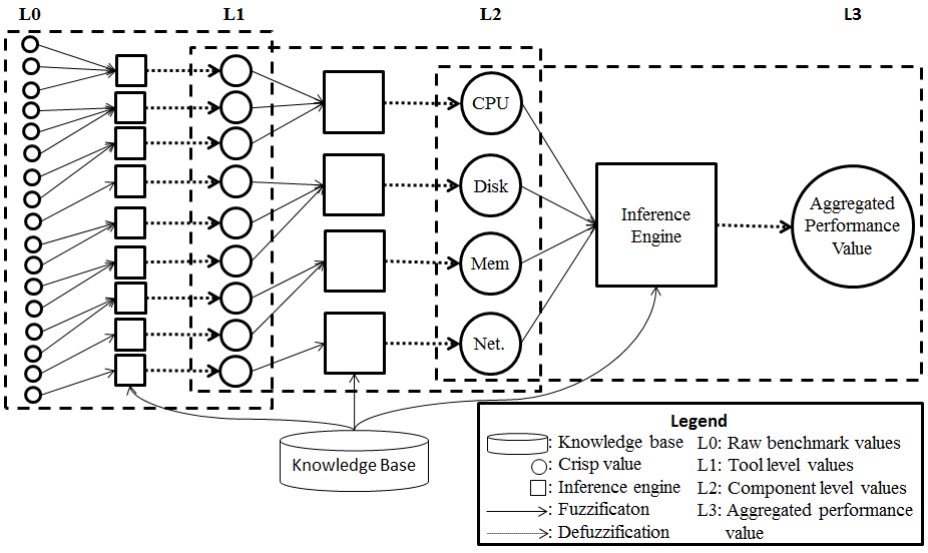

Fig. 2: Hierarchical fuzzy system

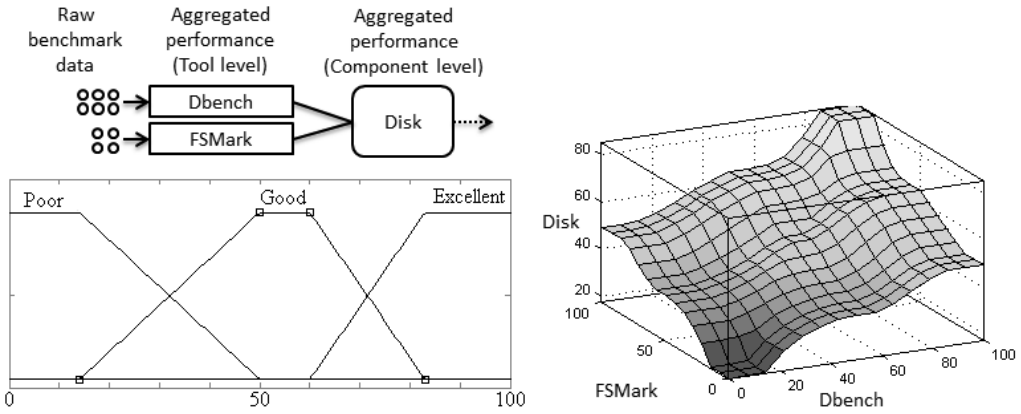

Fig. 3: Example

produces a single output per input groups. The outputs of the first layer are the inputs of the second layer (L1) where these values are grouped again by probe types and a fuzzy inference is initiated on these inputs. For instance, first level inputs are the execution times of three CPU benchmark tool. As an output, a single fuzzy metric is produced that represents the CPU performance. Similar tests are carried out for disk, memory and network. From the numerous raw data items four outputs are generated representing each categories.

The last layer groups the component level values and provides a single value (L3) that we consider characteristic to the cloud infrastructure in scope.

A key aspect is how fuzzy rules are constructed. The knowledge base contains the membership functions, rules and reference benchmark values. These reference values are used to establish the fuzzy sets and are empiric as a common practice for designing and tuning fuzzy logic based systems and services. References were established on a local IaaS cloud system running production services and also used for developing and testing new products so the workload of the cloud is diverse in different time periods (e.g., it is usually more utilized in the daytime). Therefore, the infrastructure was measured during a month and the reference values were calculated as averages of the benchmark results.

All fuzzy rules and membership functions are based on the same template. The weighted rules with the membership functions, presented in Figure 3, provide an appropriate characteristic, because they make the system insensitive against the peak results and they penalize the low performance.

3.4 Implementation

For implementing the prototype of the framework, we solely relied on open source components. The PerfVM uses Debian GNU/Linux operating system and PTS for producing benchmark data from the target clouds. These data are stored in central storage that is accessible via a Simple Storage Service (S3) compatible

interface. The proposed framework is implemented in Python [16] and it uses the freely available draft version of Fuzzy Control Language (FCL) and pyfuzzy (a Python fuzzy library package). The FCLs describes (i) the requested input variables and their fuzzy membership functions; (ii) the output variables; (iii) the defuzzification method and rules. The system uses the Central of Gravity (CoG) defuzzification method. The fuzzy inference results are stored in a MySQL database. For the evaluation, the Fuzzy Toolbox of Matlab [14] was used.

4 Proof of Concept

In this section, we demonstrate the effectiveness of our proposed performance characterization method by explaining a part of the whole procedure, the disk performance process as an example. Experiments were carried out on raw performance data collected from the SZTAKI Cloud and from the European region of Amazon EC2. The experiment is a hypothetic performance (wrt. disk I/O) comparison of the two services running on the general purpose m1.medium instance type. We present the benchmark results both representing the 'common approaches' and the characterization produced by our method and contrast the two. They are examined in terms of correctness, comparability and readability.

Figure 3 shows our example, the disk evaluation sub-process composed of a two layered HFS.

DBench and fs_mark produce the raw benchmark data (L0). The tool uses I/O patterns that are similar to what a particular application performs. It can simulate concurrent clients in order to predict the robustness and I/O throughput of the underlying storage system. The fs_mark tool can test synchronous write workloads with different running options such as number of files, file size, directory depth or number of used threads (for instance the third column in Table 1, where the test wrote 5000 files through 4 threads with 1 MB size per file) that makes it adequate for benchmarking I/O performance. Both tools provide reliable information about the I/O system of the tested machines as presented in Tables 1 and 2. Recall our aim as it was put forward in Section 1: customers are curious if a service meet the expectations, if the performance of two services can be compared objectively. If one examines the figures in Tables 1 and 2, no clear conclusion can be drawn. For instance, raw performance figures in Table 1 suggest that SZTAKI Cloud is superior to Amazon EC2 yet, it is impossible to trace *how much* it is better (differences are not proportional); *in what measure* it exceeds the limits declared in SLAs. Roughly the same applies to measurements presented in Table 2. Hence, it is a difficult to infer the performance characteristic of clouds solely from the raw benchmark data at level L0. On the other hand, performance metrics at level L1 produced by our method (denoted as *Calculated value* in Tables 1 and 2) are a result of fuzzy inference, normalisation and defuzzification.

Understanding and analysing raw benchmark data requires domain knowledge whereas the calculated values are easily comprehensible: instead of a vector of metrics, a single aggregated value between 0 and 100 represents the charac-

terization. Hence, comparison is straightforward. Reference values (e.g., limits) can also be transformed into the [0-100] scale that makes it possible to compare the performance wrt. SLA minimums. In this particular example the reference value was set to 50. Furthermore, these performance metrics produced by a fuzzy calculation can be easily transformed to symbolic, easy-to-read values for human interpretation such "medium performance", "high performance", "upper medium performance" and similar tags in arbitrary details and resolution. Accordingly, if reference values are introduced into the system, one may compare symbolicaly as "above the reference point", "close to the reference point", etc. At the next level of hierarchy these calculated values at level L1 are taken as inputs and values for level 2 (component level in this example) are produced in a similar way, cf. Table 3 for the summary of L1 and L2 values.

Important to notice the properties of the raw and calculated performance metrics as compared in Table 3. While benchmark tools typicaly generate outputs by simply averaging the measurements, HFS is a more elaborate calculation that is able to highlight or dampen (reward or penalize) certain aspects or details of the performance characteristics. Observing the results and the generated level L2 system output surface, presented in Figure 3, it can be seen that the HFS or its rules cannot be substituted by any linear approximation schemes.

In this particular experiment SZTAKI Cloud performed better than the reference system because both of the DBench and fs_mark resulted a score above than 50. In case of Amazon EC2, the DBench performance was better than the reference value, however the fs_mark was below. The introduced example and its results demonstrated that the fuzzy inference method generated a performance characterization that enabled the straightforward comparison or classification of services based on their performances and the created HFS meets the requirements set in Section 1.

5 Conclusion and future work

In this paper, we discussed the difficulties of the performance characterisation of IaaS clouds that originate mainly from virtualization. There is an evident need for consumers to compare the cloud services in terms of "performance" on the other hand, it is not easy to capture the notion of "cloud performance"

Table 1: fs_mark results

Args [Options]	1000 Files, 1MB size	5000 Files, 1MB Size, 4 Threads	4000 Files, 32 Sub Dirs, 1MB Size	1000 Files, 1MB Size, No Sync / FSync	Calculated value [0-100]
SZTAKI [Files/s]	58.55	93.87	68.33	132.63	62.55
Amazon [Files/s]	38.87	49.73	40.83	119.07	43.60

Table 2: Dbench results

Arguments [Client(s)]	1	6	12	48	128	256	Calculated value [0-100]
SZTAKI [MB/s]	113.385	225.64	242.08	220.02	185.53	134.38	72.16
Amazon [MB/s]	80.14	174.37	166.09	176.74	177.51	119.48	62.06

Table 3: Aggregated results

Tool	DBench	fs_mark	Calculated value
SZTAKI [0-100]	72.16	62.55	62.98
Amazon [0-100]	62.06	43.60	47.05

and the conventional performance tools such as benchmarks deliver large sets of numeric data that are not necessarily consistent and hard to analyse or compare. We analysed the reasons and background of this issue. Our work is aimed at establishing a framework for normalized, comparable and readable performance analysis of IaaS providers so that services of the different providers become easily characterized.

Our method also builds on benchmark tools at the low level but performance data are processed in a hierarchical fuzzy systems. The fuzzy framework allows to transform multi-dimensional numeric (quantitative) values into symbolic (qualitative) metrics of lesser dimensionality. This transformation is based on fuzzy inference governed by fuzzy rules. The large number of variables may lead to unacceptable exponential complexity of rules. We alleviated this issue by a hierarchical fuzzy inference system that both reduces the complexity of a single inference stage and also classifies performance variables so that meaningful performance characterization can be established at different levels and different details of the system.

Finally, we evaluated the prototype by comparing the Amazon EC2 and SZTAKI IaaS clouds that confirmed the applicability of the framework. In the future, we plan to improve our framework by refining FCLs for more comprehensive evaluations. Moreover, we plan to extend the framework with the capability of assessing other aspects of performance (see Sections 1 and 2) such as Service Level Agreement (SLA) violations of IaaS providers.

Acknowledgment

The authors would like to thank KMR-12-2012-0055 – "Cloud accreditations service" for its financial support.

References

1. Cloudharmony.com. `http://cloudharmony.com`, June 2014.
2. L. A. Crowl. How to measure, present, and compare parallel performance. *IEEE Parallel & Distributed Technology: Systems & Technology*, 2(1):9–25, 1994.
3. S. Garg, S. Versteeg, and R. Buyya. Smicloud: A framework for comparing and ranking cloud services. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 210–218, 2011.
4. M. Gerndt, B. Mohr, and J. L. Träff. Evaluating openmp performance analysis tools with the apart test suite. In M. Danelutto, M. Vanneschi, and D. Laforenza, editors, *Euro-Par*, volume 3149 of *Lecture Notes in Computer Science*, pages 155–162. Springer, 2004.
5. International Electrotechnical Commission, Programmable controllers - Fuzzy control programming, IEC 61131-7, 2000.
6. A. V. Katherine and K. Alagarsamy. Article: A fuzzy mathematical model for peformance testing in cloud computing using user defined parameters. *International Journal of Software Engineering and Applications*, 4(4):27–39, July 2013.
7. M.-L. Lee, H.-Y. Chung, and F.-M. Yu. Modeling of hierarchical fuzzy systems. *Fuzzy sets and systems*, 138(2):343–361, 2003.
8. S. M, V. L.j, K. Sangeeta, and G. K. Patra. Article: Estimating trust value for cloud service providers using fuzzy logic. *International Journal of Computer Applications*, 48(19):28–34, June 2012. Published by Foundation of Computer Science, New York, USA.
9. B. P. Miller, M. D. Callaghan, J. M. Cargille, J. K. Hollingsworth, R. B. Irvin, K. L. Karavanic, K. Kunchithapadam, and T. Newhall. The paradyn parallel performance measurement tool. *Computer*, 28(11):37–46, 1995.
10. R. Nelson, D. Towsley, and A. N. Tantawi. Performance analysis of parallel processing systems. *Software Engineering, IEEE Transactions on*, 14(4):532–540, 1988.
11. Z. Németh, G. Gombás, and Z. Balaton. Performance evaluation on grids: directions, issues, and open problems. In *Parallel, Distributed and Network-Based Processing, 2004. Proceedings. 12th Euromicro Conference on*, pages 290–297. IEEE, 2004.
12. G. Raju, J. Zhou, and R. A. Kisner. Hierarchical fuzzy control. *International journal of control*, 54(5):1201–1216, 1991.
13. S. S. Shende and A. D. Malony. The tau parallel performance system. *International Journal of High Performance Computing Applications*, 20(2):287–311, 2006.
14. S. Sivanandam. *Introduction to fuzzy logic using MATLAB*. Springer, 2007.
15. H.-L. Truong and T. Fahringer. Scalea: A performance analysis tool for distributed and parallel programs. In *Euro-Par 2002 Parallel Processing*, pages 75–85. Springer, 2002.
16. G. Van Rossum et al. Python programming language. In *USENIX Annual Technical Conference*, 2007.
17. L. Wang, J. Xu, and M. Zhao. Modeling vm performance interference with fuzzy mimo model. In *7th International Workshop on Feedback Computing (Feedback-Computing, co-held with ICAC2012)*, 2012.
18. J. Xu, M. Zhao, J. A. B. Fortes, R. Carpenter, and M. S. Yousif. On the use of fuzzy modeling in virtualized data center management. In *ICAC*, page 25, 2007.
19. R. R. Yager. On the construction of hierarchical fuzzy systems models. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 28(1):55–66, 1998.
20. L. A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.