Same Same, but Different: A Descriptive Intra-IaaS Differentiation

Yehia Elkhatib, Faiza Samreen, Gordon S Blair
MetaLab, School of Computing and Communications, Lancaster University, UK
Email: {i.lastname}@lancaster.ac.uk

***Abstract*—Users of cloud computing are overwhelmed with choice, even within the services offered by one provider. As such, many users select cloud services based on description alone. In this quantitative study, we investigate the services of 2 of major IaaS providers. We use 2 representative applications to obtain longitudinal observations over 7 days of the week and over different times of the day, totalling over 14,000 executions. We give evidence of significant variations of performance offered within IaaS services, calling for data-driven brokers that are able to offer automated and adaptive decision making processes with means for incorporating expressive user constraints.**

***Keywords*-Cloud computing; IaaS; Cloud brokers**

I. Introduction

The cloud is a transformative computing paradigm that has touched almost every application in the modern world. The cloud computing market is a fierce one with high competition between enormous multinational technology companies such as Google, Amazon and IBM, and relatively smaller and more specialised companies such as Flexiant and DigitalOcean. There are well documented differences between such cloud service providers (CSPs), most notably in terms of pricing schemes and hardware heterogeneity. This gives impetus for the development of inter-CSP brokers, an active area of research (*cf.* [1], [2], [3], [4]).

However, there is also need for work on *intra*-CSP decision support. On the surface of it, the services offered by any single CSP might seem straight forward as they are classified under easily identifiable tags such as *general-purpose*, *high-memory*, and *cpu-optimised*. As such, intuition would dictate that a customer simply needs to select a class that matches their application type and then select an instance from that class that falls within their budget.

Nonetheless, selection is not as easy as it looks. First, most CSPs offer a bewildering range of IaaS services in the form of different instance types under some variants of the aforementioned classes. A quick survey of the major CSPs demonstrates this as depicted in Fig. 1. CSPs such as Amazon and Microsoft offer a total of 57 and 67 instance types, respectively. One CSP not represented in the plot is IBM SoftLayer, which allows its customers to create custom instances using parameter sliders including number of cores (between 1 and 56), memory (between 1GB and 242GB), and storage (between 25GB and 100GB), as well as other settings. In total, this offers a range of 768 possible setting permutations! Ultimately, this leads to customer frustration [5], [6] and suboptimal selection of instances [7].

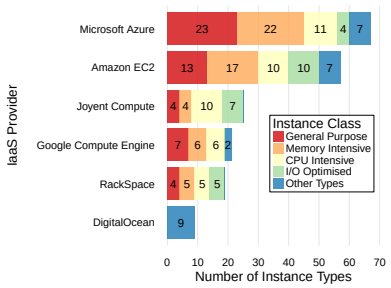


Figure 1: The number of Linux-based instance types offered by major IaaS vendors, as of July 2017.

Second, and more importantly for our purposes in this paper, other works [8], [9], [10] have indicated that instances of a single CSP are not necessarily equally cost effective. In other words, IaaS customers do not get more performance the more money they pay, even within instances of the same class (*e.g.* high-memory) of a single CSP.

In this paper, we investigate variation between intra-CSP instance types in detail. For this, we employ intensive benchmarking using 2 representative real-world applications, running them 14,300+ times over different times and days of the week. We use this methodology to analyze the intra-CSP performance of 2 major IaaS CSPs: Amazon Elastic Compute Cloud (EC2) and Google Compute Engine (GCE).

Our philosophy here is pragmatic. We use applications that are representative and open source, thus allowing others to produce follow-up comparative studies (as rightly called for in [9]). More importantly, we take the user's perspective, observing application performance as a user would.

II. Related Work

Cloud computing literature is crowded with efforts attempting to optimise service selection using a number of different approaches. One prevalent approach relies on the 'book value' attributed to offered services. Examples of this approach come from both academia (*cf.* [11]) and business (*cf.* [12]).

However, the above approach is flawed as there is sufficient evidence that the book values published by CSPs are at best only indicative, but neither representative nor reliable. An early study [13] looked into variances between a few EC2 instance types using standard benchmarks and noted that there are no "best performing instance type". Other studies have identified variances within one provider [13], [14], [15], [16], [17], and over a span of several days [18], [9], different times of the year [19], and different regions of a single IaaS provider [20], [9]. Recent works [8], [9] indicated that cloud instance performance is difficult to foresee based on the information the CSP offers and, thus, selecting the optimal instance is a non-trivial decision.

Therefore, some have tried to gain better understanding of the potential performance of cloud instances using benchmarking suites and various modelling techniques; *e.g.* [21], [17], [22]. Others use profile-based methods (*cf.* [23], [24], [25]) or application-specific performance models (*cf.* [26]).

However, little attention is given to variation in cost-effectiveness over similar instances or of the performance of a single instance type. Consequently, our knowledge of the IaaS instance search space is still constrained in dimensionality. This is what we address in our study.

III. Methodology

The objectives of our experimental strategy are to:

- Ascertain if instance classification and description is indeed helpful for instance selection or not.
- Identify the degree of variation in the performance of a single instance type.
- Uncover differences in cost effectiveness between instance types of a single CSP.

A. Experimental strategy

Our overall strategy is to study the variation in the performance of running a uniform application workload over different instance types. In order to collect enough data points to identify any potential performance variance, we repeated the workload with a 10 minute delay between each pair of runs. All application parameters and input were kept constant between application runs in order to reduce dimensionality. We also ran the applications over different times of the day and over all days of one week to control for temporal variances such as diurnal patterns.

Our main performance metric is the time an application workload takes to execute. In addition, we also use a suite of Linux system performance monitoring tools (namely `vmstat`, `glances`, and `sysstat`) for continuously monitoring VM resource utilisation.

B. Cloud infrastructures

We identified our target infrastructures as Amazon EC2 and Google GCE as two of the major players in the IaaS market. From each, we examined a subset of their instance types that seem suitable for running each of our applications. Note that there hardly was a straightforward answer to "which instance type is best for running this application", which is the point of running this study. As such, we ended up with a set from each provider. These are summarised in Tables I–II, all running 64-bit Ubuntu 14.04.

Table I: Computational specifications of EC2 instances.

Series	Instance Type	vCPU	ECU	RAM (GiB)	Storage (GB)	Price ($/h)
T2 (General Purpose)	`t2.small`	1	Var.	2	20	0.026
	`t2.medium`	2	Var.	4	20	0.052
M3 (General Purpose)	`m3.medium`	1	3	3.75	4(S)	0.070
	`m3.large`	2	6.5	7.5	32(S)	0.140
	`m3.xlarge`	4	13	15	32(S)	0.280
C4 (Compute Optimised)	`c4.large`	2	8	3.75	20	0.116
	`c4.xlarge`	4	16	7.5	20	0.232
	`c3.xlarge`	4	14	7.5	32(S)	0.239

Table II: Computational specifications of GCE instances.

Series	Instance Type	vCPU	GCEU	RAM (GB)	Storage (GB)	Price ($/h)
Standard Type	`n1-standard-1`	1	2.75	3.75	16	0.036
	`n1-standard-2`	2	5.5	7.5	16	0.071
	`n1-standard-4`	4	11	15	16	0.142
High Mem.	`n1-highmem-2`	2	5.5	13	16	0.106
High CPU	`n1-highcpu-2`	2	5.5	1.8	16	0.056
	`n1-highcpu-4`	4	11	3.6	16	0.118
	`n1-highcpu-8`	8	2.2	7.2	16	0.215

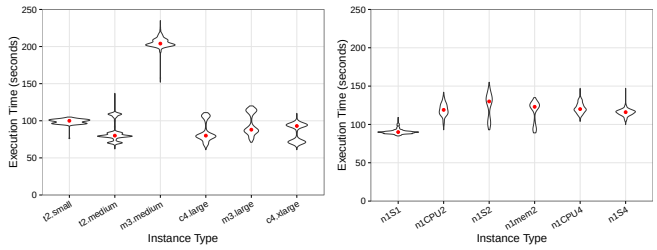
Only on-demand instances are used; they have no long-term commitments and are charged on a pay-as-you-go basis at an hourly rate. All instances are located in western Europe zones. The 'Price' column refers to the hourly charge for running a VM of the referenced instance type.

It is important to note the units the two providers use to describe their respective instance types. Both give the number of virtual cores assigned to a VM (*i.e.* 'vCPU'). They both give an indicative amount of CPU capacity, but they each use their own opaque unit: EC2 uses 'ECU' while GCE uses 'GCEU'. Amazon does not advise how an EC2 Compute Unit (ECU) relates to physical processing speed; it only assures that it is a standard unit across its different IaaS offerings. Google compute engine unit (GCEU)is an abstraction of compute resources where, according to Google, 2.75 GCEUs represent the minimum power of one logical core. In either case, there is no clear indication how they relate to physical processing speed.

C. Use cases

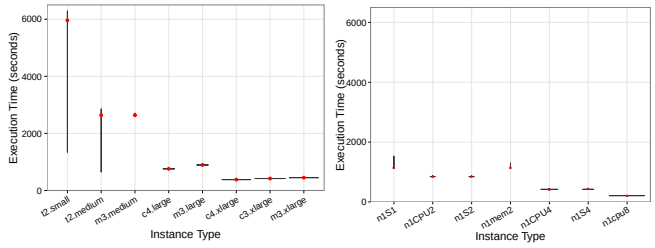
We use 2 applications representative of different architectures in relation to intensity of memory and CPU usage:

VARD: is a tool to detect and tag spelling variations in historical text [27], and is is a pre-processor to a wide range of corpus linguistic tools. It runs as a single-threaded Java application that holds an in-memory representation of the full text and dictionaries that are constantly being updated as



(a) EC2 (b) GCE

Figure 2: The overall distribution of VARD execution times.



(a) EC2 (b) GCE

Figure 3: The overall distribution of smallpt execution times.

text is being processed. VARD is representative of repetitive memory-intensive applications used for other uses such as business transaction processing, and document analysis.

smallpt: is a C++ application for simulating multiple light sources and how their illumination reflects off different objects in a 3D space. smallpt is a multi-threaded OpenMP based CPU-intensive application. For our benchmarks, we select a box scene that is constructed out of 9 very large overlapping spheres. The image is computed using equations that solve the rendering equation. Monte Carlo path tracing is used with Russian roulette for path termination.

IV. Results

A. Overall distributions

The distributions of application execution times are displayed using violin plots (Fig. 2–3), where red dots mark medians and instance types are sorted increasingly by cost from the left. We directly observe some interesting patterns.

First, running smallpt is generally much more predictable than VARD in terms of how long it will take to execute. Quartiles are narrow for smallpt over most instances types. We ascribe this to smallpt being a CPU intensive application, as memory is typically under higher contention from other guest VMs than CPU resources are. This is validated when we inspect the only exceptions: the cheapest 2 EC2 instances, which exhibit large uncertainty. This is easily explained considering EC2's CPU Credits scheme[1] that is only offered on the `T2` instance series. Under this scheme, customers collect credits for idle instances that they can later spend for full CPU utilisation.

Second, VARD execution times exhibit rather wide ranges. Moreover, many VARD execution times follow bimodal distributions, which are quite pronounced on EC2 instances. In comparison, GCE instances are again more predictable with distributions that are closer to being uni- instead of bimodal.

Finally, and perhaps most peculiarly, instance types clearly break the common intuition of "you get what you pay for". There are multiple examples of this. One surprising case is that of EC2 `m3.medium`[2] that consistently performs badly for VARD (a memory intensive application!) as well as smallpt. Another example is `n1S1` outperforming all other GCE instance types for running VARD.

A general observation from the above confirms the complexity of selecting an IaaS instance. Choosing one purely based on its computational specifications, the performance it promises, or how much budget is available is an assured recipe for uncertainty that may result in very low performance. It is worth stressing that this uncertainty is experienced with 2 market-leading IaaS providers.

B. What could be done in X hours?

Due to the observed variation, we identify two scenarios that we use going forward. These are labelled *Best Case* and *Worst Case*, which correspond to the lower and upper quartiles, respectively, as observed in the overall distributions.

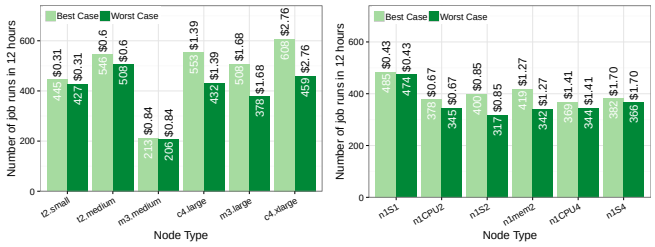
We focus first on the number of times each application could be executed in a certain amount of time (we chose 12 hours) as a proxy for performance for applications requiring repetitive or sequential execution. See Fig. 4–5.

The first impression is a reinforcement of the notion that "you get what you pay for" does not always hold. This is signified by plots that do not follow the trend of increasing number of executable jobs for more expensive instances. This is especially noticeable for VARD on both EC2 and GCE. On EC2, one could run just as many jobs on the cheapest instance type (`t2.small`) as on the most expensive one (`c4.xlarge`). For GCE, it is actually more effective to use the cheapest instance (n1S1) than any other. With smallpt, the trend is closer to what is expected with some minor deviations. For instance, `c4.xlarge` is able to run more jobs than the slightly more expensive `c3.xlarge` and `m3.xlarge` instances on EC2.

To further examine such counter-intuitive cost-effectiveness, we calculated the cost of running the maximum number of jobs per instance type, indicated above each plotted bar. The costs reveal further implications to instance selection: for relatively short, repetitive workloads like VARD and smallpt, smaller instances are

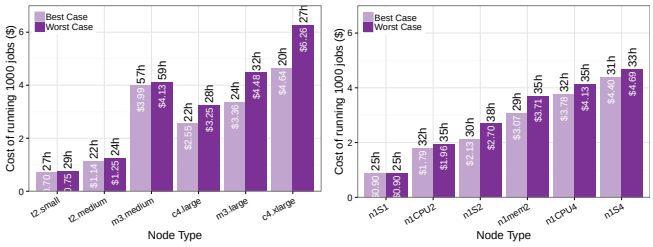
[1] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/t2-instances.html#t2-instances-cpu-credits

[2] Since carrying out our work, AWS has discontinued the `m3.medium` EC2 instance type.



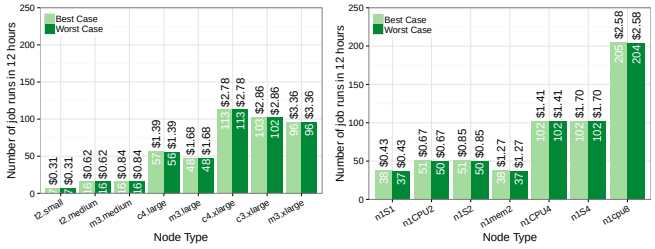
(a) EC2 (b) GCE

Figure 4: The maximum number of VARD jobs in 12 hrs.



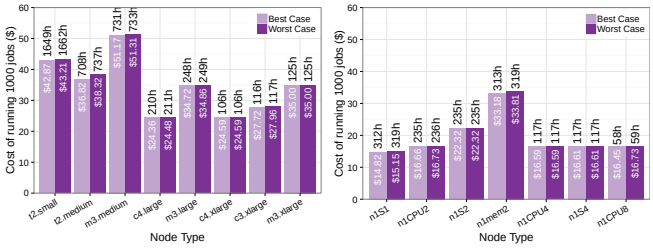
(a) EC2 (b) GCE

Figure 6: The cost of running 1000 VARD jobs.



(a) EC2 (b) GCE

Figure 5: The maximum number of smallpt jobs in 12 hrs.



(a) EC2 (b) GCE

Figure 7: The cost of running 1000 smallpt jobs.

extremely more cost effective than those equipped with higher computational specifications. For example, one could run between 427 and 445 VARD jobs on EC2 `t2.small` for just $0.31 as opposed to 459–608 jobs for almost 9 times as much ($2.76) on `c4.xlarge`. Another observation is that VARD performance on higher spec instance types is less certain than on cheaper ones. This was exhibited by the distributions in §IV-A, but is made more evident now with only the interquartile range affecting the best and worst case scenarios. Coupled with the low cost-effectiveness of such instance types, higher spec instance types pose higher risk and cost with relatively less reward.

C. How much (and how long) for Y jobs?

We now try to unravel cost-effectiveness and variation therein from the perspective of users who need to execute a certain number of jobs. This is a decisive constraint for many applications where the number of sub-jobs is synonymous with work rate, aiming to process a certain amount of data points or to reduce uncertainty to a desirable level, etc. For this, we apply the Best and Worst case scenarios to the average costs of running 1,000 jobs. We assume that each submitted job takes the same amount of time. The results are plotted in Fig. 6–7 where we also indicate the amount of time needed for executing 1,000 jobs above each bar.

With VARD on EC2, the cheapest instance `t2.small` is the most cost-effective: only $0.75 to run 1,000. This cost rises between 6 and 8 folds for the most expensive instance. The amount of time reveals interesting facts as well. One can spend 27-29 hours to run 1,000 jobs on `t2.small` as opposed to 20-27 hours on `c4.xlarge` or 24-32 hours on `m3.large`. In effect, the user would pay much more for an uncertain reduction in execution time. `t2.medium` is the most balanced in terms of cost and execution time: $1.25 for 22-24 hours, *i.e.* 3-5 times cheaper than the expensive nodes and with a fairly certain and acceptable execution time.

For VARD on GCE, it is both time and cost effective to use the cheapest instance type `n1S1`, which can finish 1,000 job runs for just $0.90 in exactly 25 hours. All other instance types are more expensive in time and cost.

For smallpt, the trend of predictable performance is observed as seen in §IV-B with some additional observations. `c4.xlarge` provides the best cost:hour ratio compared to other EC2 instances, able to run 1,000 jobs for the same cost as with `c4.large`, but in nearly half the time. The same trend is noticed in `m3.xlarge` and `m3.large`.

Interestingly, the cost of smallpt jobs on GCE are almost equal on all instances except `n1S2` and `n1mem2`. In terms of time, `n1CPU8` (the most expensive per hour) takes only 58-59 hours which is less than half of the time needed on the next fastest instance type (`n1CPU4` and `n1S4`). This is a clear example illustrating that the cheapest instance is not necessarily the most cost- or time-effective.

smallpt also draws a stark contrast between the two CSPs. GCE seems to outshine EC2 for executing smallpt jobs. Comparing `n1S2`, the second least cost- and time-effective GCE instance, to its EC2 counterparts: it is of equivalent performance and cost to `c4.large` but much cheaper than `m3.large`. Furthermore, general purpose GCE instance types extremely outperform the EC2 counterparts.

V. Discussion

We now reflect on the presented results and distil a number of learned lessons regarding the current state of IaaS.

Lesson 1: IaaS instances are not always as advertised: An overarching outcome from our results is that selecting an instance type based solely on what its virtual hardware specifications are is an error-prone decision making strategy. Instances might offer different specifications but perform very similarly (*e.g.* GCE's `n1S4` and `n1CPU4`) and vice versa (*e.g.* EC2 `m3.large` and GCE `n1S2`). Performance variation within each instance and different pricing levels further exacerbates the decision process.

This is a very significant outcome as it undermines a large host of previous works that allocate cloud resources based only on clock speeds and RAM capacities. Our results demonstrate quite clearly that such model-based scheduling is, at best, naïve and sub-optimal. Attempts to draw up a rule set to formalise instance selection would fail to account for inexplicable deviances such as those observed with EC2 `m3.medium`. This is perhaps something that only the CSPs can explicate with any confidence. Instead, deeper understanding of how different instances really perform is needed. There is already some work based on live benchmarking (*e.g.* [24]) and subsequent machine learning (ML)-based allocation (*e.g.* [28], [8]), and we call for future research to develop further in this direction.

Lesson 2: Superficial application profiling is insufficient: The other side of selecting instances based on their specifications is restricting an application to a certain type / class of instances based on its stereotypical profile. We have observed how a memory-intensive application performs very poorly on a number of high-memory instances, and instead performs rather well on cpu-optimised ones.

This further confounds users wanting to optimise their cloud workloads as it essentially expands the selection space. Accordingly, this gives more impetus to use automated methods (such as those using ML) for the exploration of a wider search space of CSPs and their instance types. Such methods need to be adaptive to be able to tailor selection for each particular workload and its sensitivities such as those triggered by change in input parameters and data [29], [30].

Lesson 3: Decisions are heavily driven by user constraints: This seemingly obvious principle is surprisingly absent from many related works that reduce cloud deployment to a simple scheduling optimisation problem. We observed how the decision to run one application (say VARD) on a single IaaS (say EC2) would change completely based on what the user's constraints are. In this particular case, they could maximise the chance of running as many jobs as possible during a certain period of time by choosing `t2.medium` (under the worst case scenario), but if they were on a budget then they would choose `t2.small`. If, instead, they wanted this to be done as soon as possible and budget was not a restriction, `c4.xlarge` would be best. These are only a basic set of constraints; there would be other sets of functional and non-functional constraints that would heavily influence the decision making process. As such, any automated process needs to allow users to express such constraints and different combinations thereof, and be able to take said constraints in consideration when forming an instance selection method.

Lesson 4: Comparing across providers is complicated: There are significant differences between instances of different CSPs even if they appear similar on the surface. It is very difficult for users to compare across providers, especially with added uncertainty of variable instance performance.

Both EC2 and GCE use their own flavours of well known hypervisors: Xen in the case of EC2, and KVM for GCE. From the end user perspective, these hypervisors are black boxes. The details of parallel workload on virtual machines, resource allocation algorithms, and how virtual cores are pinned to physical cores are some of the key details that are not (and probably will never be) provided by these and other IaaS CSPs to users. Consequently, IaaS users cannot perceive any collocation or interference effect on their running application. This further laments the need for automated adaptive selection processes.

A related side note: we observed EC2 performance to be more variable than GCE's. This fluctuation might be attributed to EC2's underlying hypervisor technology, Xen, which others have observed variability with [31].

Lesson 5: All is not lost: Pascal's quote holds true here: *"It is not certain that everything is uncertain"*. Despite all the observed variability, there still remains a fair degree of certainty that is only clear once we detach ourselves from the instances' 'book value'. This, of course, comes at a cost and requires automation to achieve and also to detect. As such, there is an opportunity here to build adaptive and customisable brokers to provide such knowledge [3], [4].

VI. Conclusion

We carried out extensive experiments on 2 market-leading IaaS providers, EC2 and GCE, identifying variances in instance performance and cost-effectiveness. Our results indicate that instance selection incurs a considerable degree of uncertainty as performance does not necessarily match declared computational specifications. In addition, matching general application profile with instance types is suboptimal. This is especially true for running memory-intensive applications, and is more discernible in EC2.

Nonetheless, we still found a fair degree of confidence in instance performance albeit over large execution samples. Coupled with the sheer number of instances and their varying configurations and pricing, the search space for an optimal instance for a given application becomes substantial. However, predictability of optimal cloud instance selection can only be achieved through automated and adaptive search of such space.

ACKNOWLEDGMENT

We thank AWS for their Education Research Grant Award. The work was partly funded by the Adaptive Brokerage for the Cloud (ABC) project, UK EPSRC grant EP/R010889/1.

REFERENCES

[1] B. Javed, P. Bloodsworth, R. U. Rasool, K. Munir, and O. Rana, "Cloud market maker: An automated dynamic pricing marketplace for cloud users," *Future Gener. Comput. Syst.*, vol. 54, pp. 52 – 67, 2016.

[2] A. Quarati, A. Clematis, and D. DAgostino, "Delivering cloud services with QoS requirements: Business opportunities, architectural solutions and energy-saving aspects," *Future Gener. Comput. Syst.*, vol. 55, pp. 403 – 427, 2016.

[3] Y. Elkhatib, "Mapping Cross-Cloud Systems: Challenges and Opportunities," in *HotCloud*, 2016, pp. 77–83.

[4] A. Elhabbash, F. Samreen, J. Hadley, and Y. Elkhatib, "Cloud brokerage: A systematic survey," *Comput. Surv.*, vol. 51, no. 6, pp. 119:1–119:28, 2019.

[5] J. McKendrick, "What cloud computing customers want: Clarity, simplicity, support," 2014. [Online]. Available: https://www.forbes.com/sites/joemckendrick/2014/07/19/what-cloud-computing-customers-want-clarity-simplicity-support/

[6] Cloud Standards Coordination (Phase 2), "Cloud computing users needs - analysis, conclusions and recommendations from a public survey," ETSI, Special Report 003 381 V2.1.1, 2016. [Online]. Available: http://csc.etsi.org/phase2/UserNeeds.html

[7] C. Kilcioglu, J. M. Rao, A. Kannan, and R. P. McAfee, "Usage patterns and the economics of the public cloud," in *WWW*, 2017, pp. 83–91.

[8] F. Samreen, Y. Elkhatib, M. Rowe, and G. S. Blair, "Daleel: Simplifying cloud instance selection using machine learning," in *NOMS*, 2016, pp. 557–563.

[9] P. Leitner and J. Cito, "Patterns in the chaos — a study of performance variation and predictability in public IaaS clouds," *Trans. Internet Techn.*, vol. 16, no. 3, 2016.

[10] N. Ghrada, M. F. Zhani, and Y. Elkhatib, "Price and performance of cloud-hosted virtual network functions: Analysis and future challenges," in *PVE-SDN*, 2018.

[11] D. Lin, A. C. Squicciarini, V. N. Dondapati, and S. Sundareswaran, "A cloud brokerage architecture for efficient cloud service selection," *Trans. Serv. Comput.*, vol. 12, no. 1, pp. 144–157, 2019.

[12] M. Lang, M. Wiesche, and H. Krcmar, "What are the most important criteria for cloud service provider selection? a delphi study," in *ECIS*, 2016.

[13] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of EC2 cloud computing services for scientific computing," in *CloudComp*, 2009, pp. 115–131.

[14] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high performance computing applications on the AWS cloud," in *CloudCom*, 2010, pp. 159–168.

[15] Z. Ou, H. Zhuang, J. K. Nurminen, A. Ylä-Jääski, and P. Hui, "Exploiting hardware heterogeneity within the same instance type of Amazon EC2," in *HotCloud*, 2012.

[16] Z. Li, L. O'Brien, R. Ranjan, and M. Zhang, "Early observations on performance of google compute engine for scientific computing," in *CloudCom*, 2013.

[17] K. Hwang, X. Bai, Y. Shi, M. Li, W. G. Chen, and Y. Wu, "Cloud performance modeling with benchmark evaluation of elastic scaling strategies," *Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 130–143, 2016.

[18] Z. Li, L. OBrien, and H. Zhang, "CEEM: A practical methodology for cloud services evaluation," in *World Congress on Services*, 2013, pp. 44–51.

[19] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: Observing, analyzing, and reducing variance," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 460–471, 2010.

[20] J. L. Lucas-Simarro, I. n. S. Aniceto, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, *A Cloud Broker Architecture for Multicloud Environments*. John Wiley & Sons, Inc., 2013, pp. 359–376.

[21] S. C. Phillips, V. Engen, and J. Papay, "Snow white clouds and the seven dwarfs," in *CloudCom*, 2011, pp. 738–745.

[22] J. Scheuner, P. Leitner, J. Cito, and H. Gall, "Cloud work bench – infrastructure-as-code based cloud benchmarking," in *CloudCom*, 2014, pp. 246–253.

[23] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing public cloud providers," in *IMC*, 2010, pp. 1–14.

[24] B. Varghese, O. Akgun, I. Miguel, L. Thai, and A. Barker, "Cloud benchmarking for performance," in *CloudCom*, 2014, pp. 535–540.

[25] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, "Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics," in *NSDI*, 2017, pp. 469–482.

[26] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient performance prediction for large-scale advanced analytics," in *NSDI*, 2016, pp. 363–378.

[27] A. Baron and P. Rayson, "VARD2: A tool for dealing with spelling variation in historical corpora," in *Postgraduate conference in corpus linguistics*, 2008.

[28] R. C. Chiang, J. Hwang, H. H. Huang, and T. Wood, "Matrix: Achieving predictable virtual machine performance in the clouds," in *ICAC*, 2014, pp. 45–56.

[29] L. M. Vaquero, F. Cuadrado, Y. Elkhatib, J. Bernal-Bernabe, S. N. Srirama, and M. F. Zhani, "Research challenges in nextgen service orchestration," *Future Gener. Comput. Syst.*, vol. 90, pp. 20 – 38, 2019.

[30] B. Varghese, P. Leitner, S. Ray, K. Chard, A. Barker, Y. Elkhatib, H. Herry, C.-H. Hong, J. Singer, F. P. Tso, E. Yoneki, and M. F. Zhani, "Cloud futurology," *IEEE Computer*, 2019.

[31] A. J. Younge, R. Henschel, J. T. Brown, G. von Laszewski, J. Qiu, and G. C. Fox, "Analysis of virtualization technologies for high performance computing environments," in *IEEE Cloud*, 2011, pp. 9–16.