A Cloud Benchmark Suite Combining Micro and Applications Benchmarks

Joel Scheuner
Chalmers | University of Gothenburg
Software Engineering Division
Gothenburg, Sweden
scheuner@chalmers.se

Philipp Leitner
Chalmers | University of Gothenburg
Software Engineering Division
Gothenburg, Sweden
philipp.leitner@chalmers.se

ABSTRACT

Micro and application performance benchmarks are commonly used to guide cloud service selection. However, they are often considered in isolation in a hardly reproducible setup with a flawed execution strategy. This paper presents a new execution methodology that combines micro and application benchmarks into a benchmark suite called RMIT Combined, integrates this suite into an automated cloud benchmarking environment, and implements a repeatable execution strategy. Additionally, a newly crafted Web serving benchmark called WPBench with three different load scenarios is contributed. A case study in the Amazon EC2 cloud demonstrates that choosing a cost-efficient instance type can deliver up to 40% better performance with 40% lower costs at the same time for the Web serving benchmark WPBench. Contrary to prior research, our findings reveal that network performance does not vary relevantly anymore. Our results also show that choosing a modern type of virtualization can improve disk utilization up to 10% for I/O-heavy workloads.

CCS CONCEPTS

• **Software and its engineering → Cloud computing**; **Software performance**;

KEYWORDS

cloud computing, benchmarking, performance,
micro benchmark, application benchmark, Web application

ACM Reference Format:
Joel Scheuner and Philipp Leitner. 2018. A Cloud Benchmark Suite Combining Micro and Applications Benchmarks. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering Companion , April 9–13, 2018, Berlin, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3185768.3186286

1 INTRODUCTION

In the cloud computing service model Infrastruce-as-a-Service (IaaS) [2], computing resources, such as CPU processing time, disk space, or networking capabilities, can be acquired and released as self-service via an Application Programming Interface (API) prevalently in the form of Virtual Machines (VMs). Such VMs are typically available in different configurations or sizes also known as instance types, machine types, or flavors. This diversity ranges from tiny-sized VM with less than 1 (shared) CPU core and 1GB RAM (*e.g.*, *f1-micro*[1]) to super-sized VMs with 128 CPU cores and 1952 GB RAM (*e.g.*, *x1.32xlarge*[2]).

Given the large service diversity, selecting an appropriate VM configuration for an application is a non-trivial challenge. While functional properties can be compared by studying provider information or using tools such as Cloudorado[3], non-functional properties, such as performance, need to be measured tediously. The field of research called cloud benchmarking is devoted to objectively measuring and comparing the differences in performance between the various cloud services. A large body of literature [9, 13, 17, 19, 20, 22] reports performance measurements for different workloads at the very resource-specific (*e.g.*, CPU integer operations) and artificial micro-level or at the domain-specific (*e.g.*, Web serving) and real-world application-level.

Existing literature largely focuses on either application benchmarks or micro benchmarks in isolation. Researchers propose new cloud-specific application benchmarks [10, 21] and evaluate their performance [4, 8, 15] in cloud environments. Such application benchmarks tend to require an elaborate setup, run over a long time, and deliver polysemous results with multiple metrics. Therefore, researchers often choose micro benchmarks, which are typically easy to install, quick to run, and clear to interpret as single metrics. However, there is a lack of combining micro and application benchmarks to obtain insights into the relevancy of micro benchmarks to assess application performance.

The goal of this paper is to present a new cloud benchmarking methodology that combines micro and application benchmarks into a benchmark suite that automates execution in cloud environments and implements a repeatable execution strategy. This paper makes the following three contributions: 1) It provides an automated benchmark that combines single-instance and multi-instance micro and application benchmarks. 2) It extends the Web-based cloud benchmark manager Cloud WorkBench (CWB) [24, 25] with a modular benchmark plugin and execution coordinator system. 3) It presents a newly crafted Web serving application benchmark with three different load scenarios.

ICPE '18, April 9–13, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5629-9/18/04…$15.00
https://doi.org/10.1145/3185768.3186286

[1] https://cloud.google.com/compute/docs/machine-types
[2] https://aws.amazon.com/ec2/instance-types/x1/
[3] https://www.cloudorado.com/cloud_providers_comparison.jsp

2 RELATED WORK

Micro benchmarking aims at measuring cloud service performance for individual resources such as CPU, I/O, memory, and network. Initial studies [31] were extended in scope and led to some of the most important contributions in this field [13, 19]. Assessing and comparing the performance of cloud services has also become a business and companies such as CloudHarmony[4] or Cloud Spectator[5] offer comparison services and publish their own price-performance analysis reports [28].

One of the earliest efforts geared towards representative workloads for the cloud comprises the Cloudstone benchmark [27], which proposes a new interaction-heavy Web 2.0 workload. CloudSuite [10] contributes an entire collection of scale-out workloads, which were incrementally[6] (v3.0 as of Jan, 2018) improved [21] and the *SPEC Cloud^{TM} IaaS 2016* [5] is specifically aimed to measure IaaS cloud performance. The YCSB suite [6] maintains a large collection of scale-out workloads for database systems. Several conceptual contributions [3, 11] suggest ideas and guidelines on how to design and implement application benchmarks for cloud environments.

Abedi and Brecht [1] reveal considerable flaws in the methodology used by many performance studies conducted in cloud environments. Simulations with performance traces from previous benchmarking experiments [22] have shown that inappropriate ordering of benchmark executions "could lead to erroneous conclusions" [1]. The *Single Trial* approach, where every benchmark is executed only once, neglects intra-instance variability. The *Multiple Consecutive Trials (MCT)* approach, where every benchmark is repeated N times before proceeding with the next benchmark, fails to take environmental changes into account. The *Multiple Interleaved Trials (MIT)* approach, where in a first round every benchmark is executed once followed by N repetitions of this first round, ignores periodic patterns that could cause performance deviations for particular repetitions. Therefore, the authors recommend the use of the *Randomized Multiple Interleaved Trials (RMIT)* approach for fair comparison of competing alternatives. The RMIT approach is a variation of the MIT approach where the benchmark order within the individual rounds is randomized instead of kept constant.

Cloud WorkBench (CWB) [24, 25] is a Web-based cloud benchmark manager, which schedules and executes benchmarks without manual interaction. It fosters the definition of configurable and reusable CWB benchmarks that are entirely defined by means of code by leveraging Infrastructure-as-Code (IaC). Therefore, CWB benchmarks are portable across cloud providers and their regions with minimal effort. Other tools focus on scale-out workloads (CloudBench [26]), templated code generation [16], declarative DSL [7], and community benchmark collection[7].

3 BENCHMARKING METHODOLOGY

Based on Cloud Benchmarking guidelines [3, 11, 14, 30], relevant benchmarks that cover different cloud resources and application domains were selected, designed, and integrated into the CWB [25] execution environment. The execution of these benchmarks is then automated following the RMIT methodology to conduct repeatable experiments.

3.1 Architecture

Figure 1 illustrates the high-level architecture of this cloud benchmarking methodology and lists the selected benchmarks. The *Benchmark Manager* coordinates the entire lifecyle of all benchmark executions. Its *Scheduler* component triggers new executions and its *Cloud Manager* component abstracts the cloud Provider API, Cloud VM provisioning, and communication with the Cloud VM. *Cloud VMs* are acquired via the *Provider API*. Within the cloud VM, the *Chef Client* controls the VM provisioning and the *CWB Client* steers the execution of the entire benchmark suite. The Chef Client fetches the provisioning configuration for the Cloud VM from the *Provisioning Service* and applies it to install and configure all *Micro* and *App* benchmarks. The CWB Client directs the execution order and handles communication with the Benchmark Manager such as submitting result metrics via a REST API. Multi-VM benchmarks, such as iperf and WPBench, submit their tasks to the *Load Generator*, which generates the specified task workload from another dedicated cloud VM.

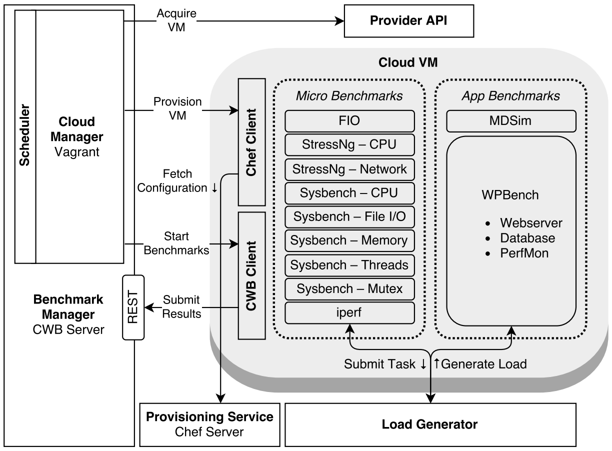

Figure 1: Architecture Overview

3.2 Cloud WorkBench Extensions

CWB was extended to modularly define benchmark plugins and combine them into a collection of benchmarks called benchmark suite. These extensions are then leveraged to package micro and application benchmarks into a combined CWB benchmark and implement a remote load generator to support multi-instance benchmarks.

[4] https://cloudharmony.com/
[5] http://cloudspectator.com/
[6] http://cloudsuite.ch/
[7] https://github.com/GoogleCloudPlatform/PerfKitBenchmarker

Benchmark plugins can now be defined more concisely and individual functionality can be unit-tested using Rspec[8], and smoke-tested locally using the *cwb* command line utility. Such a benchmark plugin typically executes a benchmark command, extracts some metrics of interests from the result, and submits these metrics to the CWB server. Benchmark suites were introduced to CWB to control the execution order of a collection of CWB benchmark plugins. In addition, cross-cutting concerns can be handled in such benchmark suites such as logging execution progress, notifying the CWB server, handling execution errors, or reporting metadata. The *rmit-combined*[9] benchmark developed in this paper bundles all micro and application benchmarks as benchmark plugins and provides a custom benchmark suite that implements the RMIT execution methodology. For auditability, the benchmark suite reports metadata for every execution about the instance (*e.g.*, CPU model name), the system (*e.g.*, gcc compiler version), the benchmarks (*e.g.*, version number), and the execution order (*i.e.*, randomized RMIT schedule).

A decoupled load generator can run arbitrary workload against a cloud VM using CWB tasks that follow the guidelines for CWB benchmark plugins. This allows to define load generating benchmark plugins for multi-machine benchmarks (*e.g.*, iperf and WPBench) and run their workloads (*i.e.*, TCP network test and JMeter test plan) against a cloud VM. The load generator is also available as self-deploying open source software[10].

3.3 Benchmarks

This section summarizes the selected micro benchmarks and presents the newly crafted Web serving application benchmark called *WPBench*.

3.3.1 Micro Benchmarks. The selection of micro benchmarks aims for broad-resource coverage and specific-resource testing while trying to minimize redundancy and execution time. To obtain an extensive instance profile, the selected micro benchmarks cover resources in the domains computation, I/O, network, and memory. Within each category, micro benchmarks were selected to specifically test different aspects. For example, the I/O domain is divided into low-level disk I/O and higher-level file I/O. Each of these subdomains can be further divided based on operation type (*e.g.*, sequential/random and read/write) or operation size (*e.g.*, 4k/8k block size). Given the large space of micro benchmarks, benchmark selection tries to avoid very similar benchmarks that are expected to deliver redundant information and also attempts to tune execution time under the premise that still meaningful results are delivered. Consequently, exceedingly long-running benchmarks without suitable tuning options had to be discarded. An additional practical criterion was to favor benchmarks from the same benchmarking tool where suitable to avoid unnecessary installation overhead. Table 1 lists the micro benchmark tools and their version numbers used in this paper. For detailed reproduction description and benchmark configuration rationales, we refer to Scheuner [23].

Table 1: Micro Benchmark Tools

Benchmark Tool	Version	Source
FIO Tester	2.1.10	[11]
iperf	2.0.5 (pthreads)	[12]
StressNg	0.07.27	[13]
Sysbench	0.4.12	[14]

3.3.2 Application Benchmarks. The Molecular Dynamics Simulation (MDSim) benchmark serves as a representative for scientific computing applications when benchmarking cloud instances [29]. An MDSim performs step-wise evolution of moving particles in a three-dimensional space according to the physical laws considering particle positions and velocities.

The Wordpress benchmark called *WPBench* was designed and implemented to serve as a representative for Web serving applications. WPBench runs different JMeter[15] load scenarios against a Wordpress[16] server and measures typical metrics such as response time and throughput. Wordpress was chosen because it is the most popular CMS software (60% market share) used by 29.3% of the top 10 million websites (as of January 2018) according to the Web technology surveys from W3Techs[17]. It has also been used for benchmarking cloud VMs [4].

Figure 2 illustrates the interaction design of WPBench. On the asynchronous *Start server* call, the Wordpress server starts the Web server, the corresponding database, and a performance monitoring agent. The *Submit JMeter task* message contains the JMeter test plan for all three load scenarios. These scenarios create detailed log files which are analyzed to summarize each test scenario. While the log files remain on the load generator for more detailed analysis, the metric summary is submitted to the CWB server. Afterwards, the Wordpress server notifies of test completion and CWB WPBench stops the server to prevent interference with subsequent benchmarks.

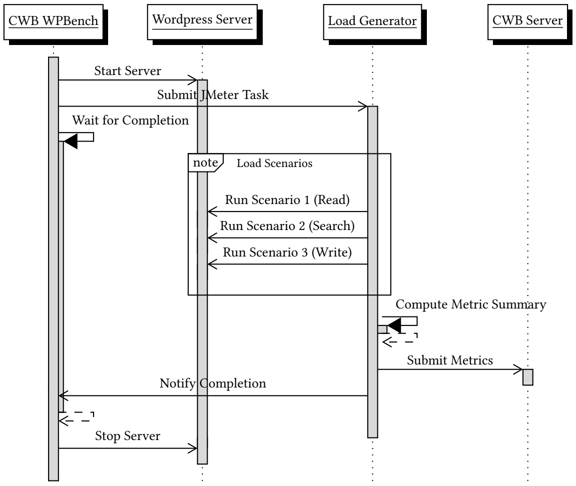

Figure 2: WPBench Execution

[8] http://rspec.info/

[9] https://github.com/sealuzh/cwb-benchmarks/blob/master/rmit-combined/

[10] https://github.com/joe4dev/load-generator

[15] http://jmeter.apache.org/

[16] https://wordpress.org/

[17] https://w3techs.com/technologies

WPBench is able to automatically install and setup Wordpress including all of its dependencies to achieve portability across different platforms and cloud providers as encouraged by CWB [25]. Further, it generates a test data set that provides sample content such as users and posts including images, comments, and tags. The three different load scenarios of WPBench aim to simulate short read, search, and write Web browsing sessions. To accurately capture representative Web browser scenarios, a JMeter proxy recorded these real Firefox browsing sessions. In iterative refinement, these captured traces were generalized, organized, and enriched with additional configurations into a JMeter test plan. This test plan is configured to run in a step-wise growing load pattern as visualized in Figure 3. Following the active benchmarking methodology proposed in [12], several resources of the cloud VM are monitored at system-level during test plan execution. The monitored metrics cover memory, disk I/O, network I/O, TCP connections, and three different CPU utilization indicators. Three CPU utilization metrics (*i.e.*, combined, idle, steal) were used to attribute for CPU throttling as discussed in [18] because cloud VM are often artificially throttled by the VM hypervisor and thus do not get all CPU cycles. A distributed master-slave testing mode was implemented to support powerful instance types where one single load generator is unable to generate sufficient workload.

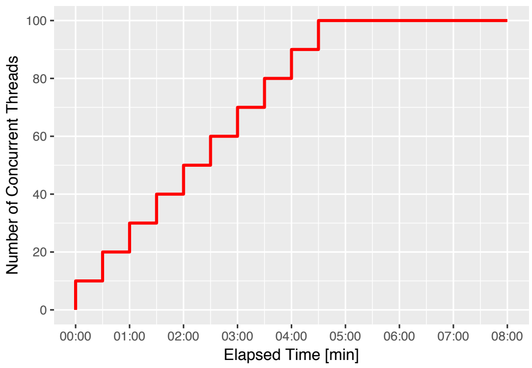

Figure 3: WPBench Load Pattern

4 CASE STUDY

Using the methodology from the previous section, a benchmarking data set was collected for the Amazon EC2 cloud provider.

4.1 Setup

All configurations build upon the officially maintained Ubuntu 14.04 LTS images[18] and have attached the general purpose storage type *gp2*, which AWS recommends for most workloads[19].

Table 2 lists the specifications for the EC2 instance types in this study. It includes all 11 available (as of April 2017) non-bursting instance types with memory size below 15 GB, except for c1.medium which consistently failed during experimentation for an unknown reason. This RAM threshold was chosen to keep experimentation cost at a reasonable level because the I/O workload grows substantially with increasing RAM size. The regions *eu-west-1* (Ireland) and *us-east-1* (N. Virginia) were chosen to compare the results with prior work [17]. Each configuration is scheduled to execute once every 3 hours (*i.e.*, 8 times per day) and runs 3 iterations. Every iteration takes between 45 and 70 minutes depending on the instance type. This corresponds to almost continuous execution on a rolling basis (*i.e.*, a new instance is acquired once the previous instance is released) between 4 to 8 days for two low-tier, two medium-tier, and one large-tier instance type. The number of executions are at least 58 for *m3.medium* (eu) and *m3.large* (eu) and at least 33 for *m1.small* (eu/us) and *m3.medium* (us). At least one execution was run for the remaining instance types. In total, 62952 measurements were collected over 244 executions between April and May 2017.

Table 2: EC2 Instance Type Specifications[20]

Instance Type	vCPU	RAM	Network	Cost*
m1.small	1	1.7	Low	0.047
m1.medium	1	3.75	Moderate	0.095
m3.medium	1	3.75	Moderate	0.073
m1.large	2	7.5	Moderate	0.190
m3.large	2	7.5	Moderate	0.146
m4.large	2	8.0	Moderate	0.111
c3.large	2	3.75	Moderate	0.120
c4.large	2	3.75	Moderate	0.113
c3.xlarge	4	7.5	Moderate	0.239
c4.xlarge	4	7.5	High	0.226
c1.xlarge	8	7	High	0.592

*USD/h for Linux On-Demand in eu-west-1 (2017-05-19)

4.2 Results

Figure 4 illustrates the performance of the WPBench read scenario in relation to the cost for each instance type. The instance types on the Pareto front towards the bottom-left corner deliver the best performance in terms of lowest response time per cost. The response time for each instance types is obtained from the average over three iterations on the same instance. A prestudy has shown that such few samples are sufficient to achieve a typical confidence interval of at least 95%.

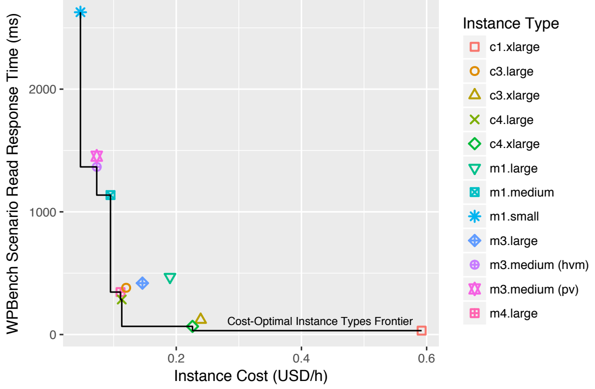

Figure 4: WPBench Reponse Time Cost Frontier

[18] https://cloud-images.ubuntu.com/locator/ec2/
[19] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html
[20] http://www.ec2instances.info/
https://aws.amazon.com/ec2/instance-types/
https://aws.amazon.com/ec2/previous-generation/

Figure 5 shows the single thread intra-cloud TCP network bandwidth (*i.e.*, between two VMs in the same data center) over the period of 8 days for *m3.large* and *m3.medium (hvm)* and 4 days for *m1.small*. During this period, the iperf benchmark was run roughly every hour (individual repetitions depend on RMIT schedule and instance speed) and on a new instance after three iterations. The Relative Standard Deviation (RSD) is below 5% for all three scenarios given its formal definition as $RSD = 100 \cdot \frac{\sigma_m}{\overline{m}}$ where σ_m is the absolute standard deviation and $\overline{m}$ is the arithmetic mean of the metric m.

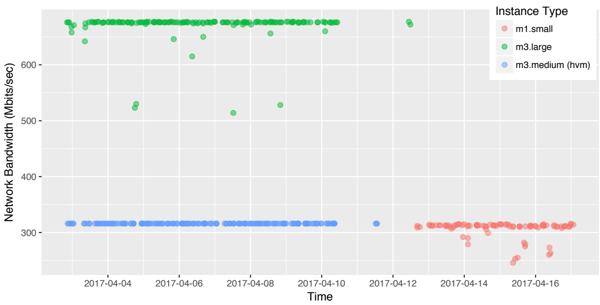

Figure 5: Network Bandwidth over Time

Figure 6 depicts the disk utilization during the execution of two FIO disk I/O benchmarks. Disk utilization is a measure of I/O efficiency where 100% utilization means that the disk is fully saturated and 50% would imply that the disk is idling half of the time. Further, the graph also annotates the type of Linux hardware virtualization, either Para-Virtualization (PV) with guest OS extensions or Hardware-assisted Virtual Machine (HVM) with host OS extensions for privileged instructions. These results for each instance type are again based on the average of three iterations on the same instance.

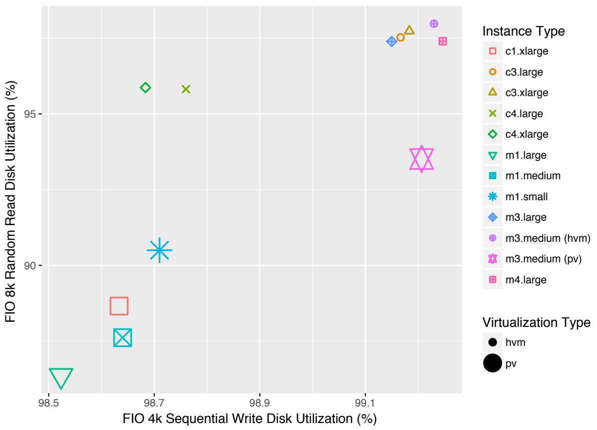

Figure 6: Disk Utilization during I/O Benchmark

4.3 Discussion

The results reveal that choosing an optimal instance type for this type of Web serving application (*i.e.*, WPBench) can have a big impact on the cost efficiency. For example, *c4.large* delivers 25% - 40% better performance for a 5% - 40% cheaper price compared to *c3.large*, *m3.large*, and *m1.large*. Further the two extreme examples of *m1.small* and *c1.large* demonstrate that the cheapest price comes a performance penalty by orders of magnitude and achieving the best performance is comparatively expensive. To exemplify, *m1.small* cost 35% less than its next pricier instance type *m3.medium* but delivers 80% worse performance (+1200ms response time). Similarly, *c1.xlarge* costs 2.6 times more compared to *c1.xlarge* but only delivers about 2.1 times the performance (-35ms response time).

Most surprisingly, intra-cloud network performance achieved almost perfect stability, which contrasts the 25% RSD observed several years ago between March and April in 2012 [9]. Presumably, AWS fundamentally changed their approach to intra-cloud networking and might perform customer-based placement optimizations using strategies such as placement groups. Generally, this exemplifies the trend from delivering performance at best effort towards specifically designed performance levels.

Visual interpretation reveals two clusters based on the virtualization type of the underlying instance type. The bottom-left corner contains the older generation instance types with PV virtualization and the top-right corner contains new HVM virtualized instance types. While the less than 1% difference for the sequential write workload is almost neglectable, the random read workload seems to benefit from HVM virtualization with 8% - 10% better utilization rates.

5 CONCLUSION

This paper presents a cloud benchmarking methodology that combines single-instance and multi-instance micro and application benchmarks into a benchmark suite called *RMIT Combined* following an execution methodology for repeatable benchmarking in the cloud. As part of the benchmark suite, a Web serving benchmark called *WPBench* with three different load scenarios is contributed. A case study within the Amazon EC2 cloud has been conducted where more than 60000 measurements have been collected over 244 benchmark executions. The results reveal that choosing an optimal instance type for serving Web content with WPBench can have a big impact on the cost efficiency as more efficient instance types deliver up to 40% better performance for up to 40% lower costs at the same time. Contrary to previous research, our results reveal that network performance between instances within the same cloud does *not* vary relevantly anymore. Further, newer generation instance types with HVM virtualization can improve the disk utilization for I/O-heavy workload up to 10%.

Future work will emphasize the relationship between micro and application benchmarks by evaluating the usefulness of micro benchmarks to estimate application performance. To address the threat to external validity, other cloud providers, larger instance types, and additional application domains should be studied. Another avenue for future research is an extension to scale-out workloads with distributed application components.

ACKNOWLEDGMENTS

This work was supported by the Wallenberg Autonomous Systems Program (WASP).

REFERENCES

[1] Ali Abedi and Tim Brecht. 2017. Conducting Repeatable Experiments in Highly Variable Cloud Computing Environments. In *8th ACM/SPEC International Conference on Performance Engineering (ICPE)*. 287–292. https://doi.org/10.1145/3030207.3030229

[2] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, and Matei Zaharia. 2009. *Above the Clouds: A Berkeley View of Cloud Computing*. Technical Report. EECS Dep., Univ. of California, Berkeley. 25 pages.

[3] Carsten Binnig, Donald Kossmann, Tim Kraska, and Simon Loesing. 2009. How is the Weather Tomorrow?: Towards a Benchmark for the Cloud. In *Proceedings of the 2nd International Workshop on Testing Database Systems (DBTest)*. ETH Zurich, ACM, New York, NY, USA, Article 9, 6 pages. https://doi.org/10.1145/1594156.1594168

[4] A. H. Borhani, P. Leitner, B. S. Lee, X. Li, and T. Hung. 2014. WPress: An Application-Driven Performance Benchmark for Cloud-Based Virtual Machines. In *2014 IEEE 18th International Enterprise Distributed Object Computing Conference*. 101–109. https://doi.org/10.1109/EDOC.2014.23

[5] The SPEC Consortium. 2016. SPEC Cloud™ IaaS 2016 Benchmark. (2016). http://spec.org/cloud_iaas2016/

[6] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking Cloud Serving Systems with YCSB. In *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC)*. 143–154. https://doi.org/10.1145/1807128.1807152

[7] Matheus Cunha, Nabor Mendonça, and Américo Sampaio. 2013. A Declarative Environment for Automatic Performance Evaluation in IaaS Clouds. In *6th IEEE International Conference on Cloud Computing (CLOUD)*. 285–292. https://doi.org/10.1109/CLOUD.2013.12

[8] Jiang Dejun, Guillaume Pierre, and Chi-Hung Chi. 2010. *EC2 Performance Analysis for Resource Provisioning of Service-Oriented Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, 197–207. https://doi.org/10.1007/978-3-642-16132-2_19

[9] Benjamin Farley, Ari Juels, Venkatanathan Varadarajan, Thomas Ristenpart, Kevin D. Bowers, and Michael M. Swift. 2012. More for Your Money: Exploiting Performance Heterogeneity in Public Clouds. In *Proceedings of the 3rd ACM Symposium on Cloud Computing (SoCC '12)*. Article 20, 14 pages. https://doi.org/10.1145/2391229.2391249

[10] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. 2012. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 37–48. https://doi.org/10.1145/2150976.2150982

[11] Enno Folkerts, Alexander Alexandrov, Kai Sachs, Alexandru Iosup, Volker Markl, and Cafer Tosun. 2013. Benchmarking in the Cloud: What It Should, Can, and Cannot Be. In *Selected Topics in Performance Evaluation and Benchmarking*. Vol. 7755. Springer, 173–188. https://doi.org/10.1007/978-3-642-36727-4_12

[12] Brendan Gregg. 2013. *Systems Performance: Enterprise and the Cloud*. Prentice Hall.

[13] Alexandru Iosup, Simon Ostermann, Nezih Yigitbasi, Radu Prodan, Thomas Fahringer, and Dick Epema. 2011. Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. *IEEE Transactions on Parallel and Distributed Systems* 22, 6 (June 2011), 931–945. https://doi.org/10.1109/TPDS.2011.66

[14] Alexandru Iosup, Radu Prodan, and Dick Epema. 2014. *IaaS Cloud Benchmarking: Approaches, Challenges, and Experience*. Springer, 83–104. https://doi.org/10.1007/978-1-4939-1905-5_4

[15] Alexandru Iosup, Nezih Yigitbasi, and Dick Epema. 2011. On the Performance Variability of Production Cloud Services. In *11th IEEE/ACM Int. Symp. on CCGrid*. 104–113. https://doi.org/10.1109/CCGrid.2011.22

[16] Deepal Jayasinghe, Galen Swint, Simon Malkowski, Jack Li, Qingyang Wang, Junhee Park, and Calton Pu. 2012. Expertus: A Generator Approach to Automate Performance Testing in IaaS Clouds. In *5th IEEE Int. Conf. on Cloud Computing (CLOUD)*. 115–122. https://doi.org/10.1109/CLOUD.2012.98

[17] Philipp Leitner and Jürgen Cito. 2016. Patterns in the Chaos — A Study of Performance Variation and Predictability in Public IaaS Clouds. *ACM Transactions on Internet Technology (TOIT)* 16, 3, Article 15 (April 2016), 23 pages. https://doi.org/10.1145/2885497

[18] Philipp Leitner and Joel Scheuner. 2015. Bursting With Possibilities – an Empirical Study of Credit-Based Bursting Cloud Instance Types. In *8th IEEE/ACM International Conferrence on Utility and Cloud Computing (UCC)*. https://doi.org/10.1109/UCC.2015.39

[19] Simon Ostermann, Alexandria Iosup, Nezih Yigitbasi, Radu Prodan, Thomas Fahringer, and Dick Epema. 2009. A performance analysis of EC2 cloud computing services for scientific computing. In *Cloud Computing*. Vol. 34. Springer, 115–131. https://doi.org/10.1007/978-3-642-12636-9_9

[20] Z. Ou, H. Zhuang, A. Lukyanenko, J. K. Nurminen, P. Hui, V. Mazalov, and A. Ylä-Jääski. 2013. Is the Same Instance Type Created Equal? Exploiting Heterogeneity of Public Clouds. *IEEE Transactions on Cloud Computing* 1, 2 (July 2013), 201–214. https://doi.org/10.1109/TCC.2013.12

[21] Tapti Palit, Yongming Shen, and Michael Ferdman. 2016. Demystifying Cloud Benchmarking. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 122–132. https://doi.org/10.1109/ISPASS.2016.7482080

[22] Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. 2010. Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. *Proceedings of the VLDB Endowment* 3, 1 (Sept. 2010), 460–471. https://doi.org/10.14778/1920841.1920902

[23] Joel Scheuner. 2017. *Cloud Benchmarking – Estimating Cloud Application Performance Based on Micro Benchmark Profiling*. Master's thesis. University of Zurich. http://www.merlin.uzh.ch/publication/show/15364

[24] Joel Scheuner, Jürgen Cito, Philipp Leitner, and Harald Gall. 2015. Cloud WorkBench: Benchmarking IaaS Providers based on Infrastructure-as-Code. In *Proceedings of the 24th International World Wide Web Conference (WWW) - Demo Track*. https://doi.org/10.1145/2740908.2742833

[25] Joel Scheuner, Philipp Leitner, Jürgen Cito, and Harald Gall. 2014. Cloud WorkBench - Infrastructure-as-Code Based Cloud Benchmarking. In *Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. https://doi.org/10.1145/2740908.2742833

[26] M. Silva, M.R. Hines, D. Gallo, Qi Liu, Kyung Dong Ryu, and D. Da Silva. 2013. CloudBench: Experiment Automation for Cloud Environments. In *IEEE International Conference on Cloud Engineering (IC2E)*. 302–311. https://doi.org/10.1109/IC2E.2013.33

[27] Will Sobel, Shanti Subramanyam, Akara Sucharitakul, Jimmy Nguyen, Hubert Wong, Arthur Klepchukov, Sheetal Patil, Armando Fox, and David Patterson. 2008. Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0. (2008).

[28] Cloud Spectator. 2017. *Price-Performance Analysis of the Top 10 Public IaaS Vendors*. Technical Report. Cloud Spectator.

[29] B. Varghese, O. Akgun, I. Miguel, L. Thai, and A. Barker. 2017. Cloud Benchmarking For Maximising Performance of Scientific Applications. *IEEE Transactions on Cloud Computing* PP, 99 (2017), 1–1. https://doi.org/10.1109/TCC.2016.2603476

[30] V. Vedam and J. Vemulapati. 2012. Demystifying Cloud Benchmarking Paradigm - An in Depth View. In *36th IEEE Computer Software and Applications Conference (COMPSAC)*. 416–421. https://doi.org/10.1109/COMPSAC.2012.61

[31] Edward Walker. 2008. Benchmarking Amazon EC2 for High-Performance Scientific Computing. *Usenix Login* 33, 5 (October 2008), 18–23.