Performance Benchmarking of Infrastructure-as-a-Service (IaaS) Clouds with Cloud WorkBench

Tutorial Paper

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

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

ABSTRACT

The continuing growth of the cloud computing market has led to an unprecedented diversity of cloud services with different performance characteristics. To support service selection, researchers and practitioners conduct cloud performance benchmarking by measuring and objectively comparing the performance of different providers and configurations (e.g., instance types in different data center regions). In this tutorial, we demonstrate how to write performance tests for IaaS clouds using the Web-based benchmarking tool Cloud WorkBench (CWB). We will motivate and introduce benchmarking of IaaS cloud in general, demonstrate the execution of a simple benchmark in a public cloud environment, summarize the CWB tool architecture, and interactively develop and deploy a more advanced benchmark together with the participants.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**; • **Software and its engineering → Software performance**.

KEYWORDS

Cloud Computing, Performance, Benchmarking

ACM Reference Format:
Joel Scheuner and Philipp Leitner. 2019. Performance Benchmarking of Infrastructure-as-a-Service (IaaS) Clouds with Cloud WorkBench: Tutorial Paper. In *Tenth ACM/SPEC International Conference on Performance Engineering Companion (ICPE '19 Companion), April 7–11, 2019, Mumbai, India.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3302541.3310294

TUTORIAL SUMMARY

- *Presenters:* Joel Scheuner and Philipp Leitner
- *Duration:* half-day (3 hours)
- *Past Versions:* none
- *Requirements:* Ideally 2 beamers (for live demo plus supporting material), but the tutorial can also be held with one. Reliable wifi/internet connection in the room (for connecting to cloud providers and downloading dependencies).
- *ICPE Relevance:* Performance measurement and benchmarking are two core ICPE topics as per the Call for Papers. Benchmarking cloud systems is an up-and-coming research topic in the community.
- *Rough Tutorial Outline:*

(1) IaaS cloud benchmarking (~20min): We present a general introduction into the area, with background, motivation, and selected results.
(2) My first benchmark with CWB (~30min): We demonstrate how to modify an existing simple benchmark and execute it in a public IaaS cloud using the CWB web interface. Tutorial participants are invited to follow these steps in their web browser, and the presenters support the participants to do so.
(3) CWB Architecture Overview (~20min): We present a short overview of the basic architecture of CWB from a tool user perspective.
(4) Building an advanced benchmark with CWB (~90min): In the second half of the tutorial, we introduce how to design and implement a more advanced benchmark from scratch, using Chef[a] cookbooks to install dependencies and configure hooks to control the lifecycle of a benchmark. Interested tutorial participants will be given credentials and support to design, build, and execute such a benchmark from scratch using a template.

\+ Extra topics: If time permits or questions arise, we will prepare additional topics, such as debugging, configuring multi-machine benchmarks, or integration testing benchmark setups.

[a] https://www.chef.io/chef

ICPE '19 Companion, April 7–11, 2019, Mumbai, India

ACM ISBN 978-1-4503-6286-3/19/04…$15.00
https://doi.org/10.1145/3302541.3310294

1 INTRODUCTION

Cloud computing has become the standard way of deploying computing instrastructure in many domains and has largely replaced privately owning computational resources such as server racks. In the Infrastructure-as-a-Service (IaaS) [1] service model [14], 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). VMs are typically available in different configurations or sizes also known as instance types, machine types, or flavors. This diversity ranges from tiny-sized VMs with a fractional CPU core and 0.6 GB RAM (*e.g.*, *f1-micro*) to super-sized VMs with 128 CPU cores and 3904 GB RAM (*e.g.*, *x1e.32xlarge*).

Given the large service diversity, selecting an appropriate VM configuration for an application is a non-trivial challenge. One example for the rapid growth of service diversity can be found in the 12 years (2006-2018) release history of Amazon Elastic Compute (EC2) instance types[1], which covers 153 different VM types categorized into five specialization families (*i.e.*, general purpose, compute-, memory-, storage-, GPU-optimized). Furthermore, other providers introduced tailorable processor and memory specifications such as the custom machine types[2] from Google. 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 quantified tediously.

Cloud benchmarking is the field of research dedicated to objectively measuring and comparing the differences in performance between the various cloud services. A large body of literature [5, 8, 13, 15–17, 20] 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.

2 CWB BENCHMARKING APPROACH

This section summarizes how performance experiments in IaaS cloud are typically conducted and then explains how an experimenter uses CWB to measure IaaS cloud performance in real cloud environments.

2.1 Basic IaaS Benchmarking Approach

Figure 1 shows a very simplified view of an IaaS experiment. A benchmark manager acts as coordinating entity to acquire the instances that are to be benchmarked via the provider API, and to provision (configure) them. Once the setup is finished, it starts the execution of a benchmark within the instance, which returns the metrics (i.e., results) of its execution, and finally destroys the instances once the benchmark is completed. Cloud environments typically deliver fairly unpredictable performance and therefore this execution lifecycle is repeated until a sample size with the desired statistical confidence is achieved. We refer to literature for the more detailed view [4] or a more generic architecture for IaaS cloud benchmarking [9].

[1]https://aws.amazon.com/blogs/aws/ec2-instance-history
[2]https://cloud.google.com/custom-machine-types
[3]https://www.cloudorado.com/cloud_providers_comparison.jsp

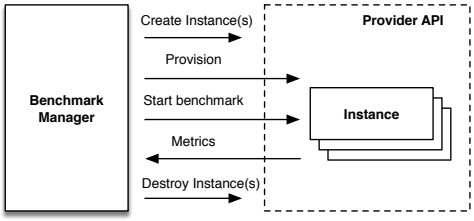

Figure 1: Basic IaaS Benchmarking Approach

2.2 Benchmarking with CWB

The following five steps explain how an experimenter can use CWB to conduct a cloud benchmarking experiment:

(1) The experimenter writes a benchmark configuration as an Infrastructure-as-Code (IaC) [7] script using tools such as Chef[4], Puppet[5], or Ansible[6]. The main idea is to express the entire configuration process in code and automate it in a repeatable and portable way for scaling out to many VMs and multiple cloud providers. This configuration process includes installing dependencies (*e.g.*, benchmark tools such as sysbench[7]), setting up configuration files or simulation data, and defining execution hooks for starting the measurements, submitting metrics, and notifying the termination of the execution. CWB uses Chef cookbooks[8] (*i.e.*, a collection of configuration scripts) for this purpose so that one can leverage existing packages from the Chef Supermarket community platform[9]. We published a collection of parametrizeable benchmarks[10], including a generic *cli-benchmark*, such that this step can be skipped for simple benchmarks.

(2) The benchmark definition declares IaaS resources, and parametrizes the benchmark configuration from step 1. The resource declaration section specifies the type of IaaS resources (*e.g.*, instance type, data center region, disk type, base image) and the authentication settings (*e.g.*, API credentials, SSH login keys) for a specific provider. The benchmark parametrization section selects one or multiple benchmark configurations and optionally adjusts the default configuration via key-value pairs (*e.g.*, duration of testing, number of repetitions). CWB uses Vagrant[11] and its domain specific language (DSL) for Vagrantfiles to implement resource declaration and benchmark parametrization. This makes it easy to extend CWB with further cloud providers by leveraging the ecosystem of 30+ existing Vagrant provider plugins. In addition, CWB provides resource declaration defaults and abstracts the authentication setup using a centralized configuration such that it is easy to setup an array of benchmark variations across multiple providers with minimal configuration effort.

[4]https://www.chef.io/chef
[5]https://puppet.com
[6]https://www.ansible.com
[7]https://github.com/akopytov/sysbench
[8]https://docs.chef.io/cookbooks.html
[9]https://supermarket.chef.io/
[10]https://github.com/sealuzh/cwb-benchmarks
[11]https://www.vagrantup.com

(3) The execution of a benchmark definition can be manually triggered via the CWB web interface or automated by attaching a periodic schedule written as crontab[12] expression. CWB automates the entire benchmarking lifecycle such that no human intervention is required.
(4) The metrics of a completed experiment can be downloaded as a comma-separated values (CSV) file.
(5) The analysis and archiving of results is up to the experimenter and not supported by CWB.

3 CLOUD WORKBENCH ARCHITECTURE

This section summarizes the overall architecture and explains the lifecyle of a single benchmark execution.

3.1 System Overview

Figure 2 summarizes the components and interactions involved in executing a benchmark with CWB. The *experimenter* defines benchmarks via the provisioning service and the CWB web interface. The *CWB server*, a standard 3-tier web application, provides the web interface, implements the business logic leveraging dependencies, and stores its data (*i.e.*, benchmark definitions and results) in a relational database. The *CWB server* acquires and releases cloud resources, such as cloud VMs (*i.e.*, the system under test), using a *provider API* via the provider plugins abstraction. A small *CWB client library* eases the interaction between the *cloud VMs* and the *CWB server*. This library is automatically installed together with the benchmark configuration retrieved from the provisioning service. All components interact with each other over REST [6] services to foster loose coupling and reusability, with one exeception where the *CWB Server* communicates with the *cloud VMs* over the standard Linux utilities *ssh* and *rsync* for reasons of simplicity.

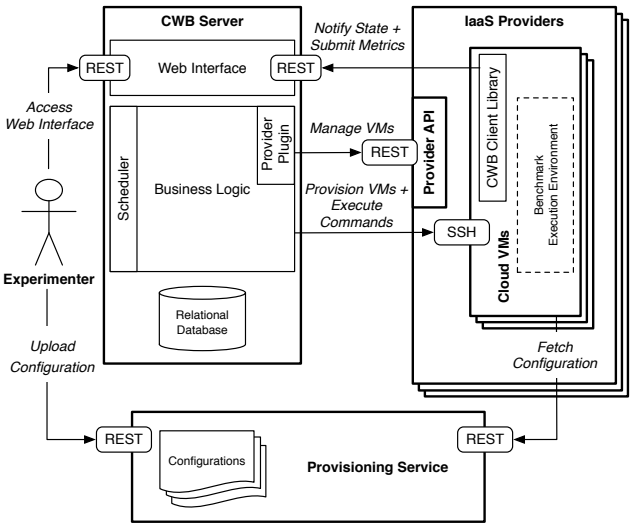

Figure 2: Architecture Overview

[12]https://linux.die.net/man/5/crontab

3.2 Execution Lifecyle

Figure 3 illustrates the interactions when a new benchmark execution is triggered by the experimenter or the scheduler. For simplicity, we focus on a successful execution and omit various failure cases. Firstly, the *CWB server* acquires cloud resources, such as cloud VMs,

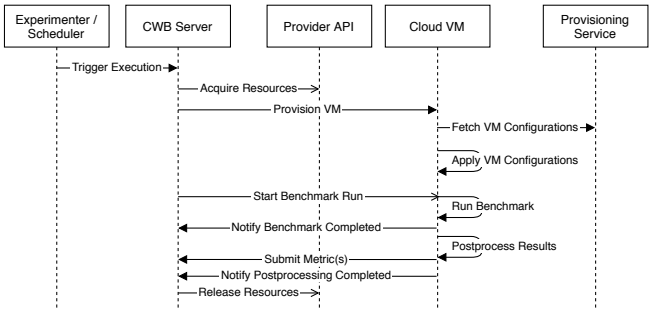

Figure 3: Execution Lifecycle

dedicated block storage, or dynamically mapped IP addresses. As soon as the *cloud VM* is accessible via a remote shell connection, the *CWB server* orchestrates the provisioning of the *cloud VM*, which fetches its role-depenent configuration from the provisioning service. Once all *cloud VMs* are provisioned (*e.g.*, installed all software for benchmarking), the *CWB server* asynchronously invokes the benchmark execution hook in the *cloud VM*. Once the actual benchmark workload is completed, the benchmark should notify this state update to the CWB server via the client library. The benchmark results are then post-processed, which typically involves textual result extraction, and submitted to the CWB server as individual metrics or as a collection of metrics via a CSV file. After completed work, the *cloud VM* notifies the state update to the *CWB server* to trigger the release of all resources.

4 CLOUD BENCHMARKING TOOLS

This section summarizes other tools for cloud benchmarking and highlights the differences to CWB.

CloudBench [19] supports complex benchmarks with dynamic scale-out workloads and evolved into the *CBTOOL*[13] for rapid cloud experimentation and analysis. Another actively maintained tool that comes with a collection of pre-configured benchmarks is Google's *PerfKit Benchmarker*[14], which comes with an optional dashboard for performance analysis (*i.e.*, PerfKit Explorer). Previous academic work introduced [11] and extended [10] Expertus, a code generation-based shell scripting approach, and introduced [2] and extended [3] Cloud Crawler, a declarative approach defining a custom YAML-based DSL.

In comparison to these tools, CWB builds upon a strong IaC core to define benchmarks based on standard tooling. The integration of configuration management and abstraction of provider APIs makes it easy to define variations of benchmarks across multiple providers. CWB provides a web interface to configure benchmark variations and schedule periodic executions.

[13]https://github.com/ibmcb/cbtool
[14]https://github.com/GoogleCloudPlatform/PerfKitBenchmarker

5 EXAMPLE STUDIES USING CWB

We now briefly present a selection of our own studies as examples of research that is enabled by CWB.

- In TOIT'16 [13], we have quantitatively validated 15 hypotheses of cloud performance formulated through a thorough literature review. CWB enabled us to easily scale up data collection, ultimately collecting close to 55.000 data points for four cloud providers and multiple regions in a fully automated manner.
- In CCGrid'17 [4], we used CWB to benchmark a realistic, multi-tier Web application. CWB was used to set up, in a fully automated and reproducible manner, web servers, databases, and a cluster of JMeter-based load generators.
- In CLOUD'18 [18], CWB was utilized to collect microbenchmarking (CPU, IO, etc.) information about cloud instances, which was then used to estimate the application performance of different types of cloud applications or services.
- In EMSE'19 [12], we showed that CWB can also be used for studies in other sub-disciplines (software engineering in that case). We used CWB to provision software peformance tests in Java and Go across different cloud providers, enabling us to easily test and evaluate them in different environments.

These are only examples of studies that can profit from using the approach and tooling introduced in this tutorial. Essentially, while CWB has originally been built to foster performance evaluation of IaaS Infrastructure, we have found that the same tooling can also be used to conduct various other types of performance experiments.

6 CONCLUSION

This tutorial paper addresses tools for benchmarking IaaS clouds. We motivate the importance of cloud benchmarking, demonstrate the execution of a simple IaaS benchmark in a public cloud provider using the CWB web interface, and develop a more advanced benchmark with configuration management integration.

ACKNOWLEDGMENTS

This work was supported by the Wallenberg Autonomous Systems Program (WASP). We would like to thank the Ericsson Research Data Center (ERDC) for providing hardware resources for this tutorial.

REFERENCES

[1] 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 UCB/EECS-2009-28. EECS Department, University of California, Berkeley. 25 pages. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

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

[3] M. Cunha, N. C. Mendonça, and A. Sampaio. 2017. Cloud Crawler: a declarative performance evaluation environment for infrastructure-as-a-service clouds. *Concurrency and Computation: Practice and Experience* 29, 1 (2017), e3825. https://doi.org/10.1002/cpe.3825 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.3825

[4] Christian Davatz, Christian Inzinger, Joel Scheuner, and Philipp Leitner. 2017. An Approach and Case Study of Cloud Instance Type Selection for Multi-Tier Web Applications. In *17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. 534–543.

[5] 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 Third ACM Symposium on Cloud Computing (SoCC '12)*. 20:1–20:14. https://doi.org/10.1145/2391229.2391249

[6] Roy T. Fielding and Richard N. Taylor. 2000. *Architectural styles and the design of network-based software architectures*. Ph.D. Dissertation. University of California, Irvine Irvine, USA.

[7] Michael Hüttermann. 2012. *Infrastructure as Code*. Apress. 35–156 pages. https://doi.org/10.1007/978-1-4302-4570-4_9

[8] 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

[9] Alexandru Iosup, Radu Prodan, and Dick Epema. 2014. IaaS Cloud Benchmarking: Approaches, Challenges, and Experience. In *Cloud Computing for Data-Intensive Applications*, Xiaolin Li and Judy Qiu (Eds.). Springer New York, 83–104. https://doi.org/10.1007/978-1-4939-1905-5_4

[10] Deepal Jayasinghe, Josh Kimball, Siddharth Choudhary, Tao Zhu, and Calton Pu. 2013. An automated approach to create, store, and analyze large-scale experimental data in clouds. In *14th IEEE International Conference on Information Reuse and Integration (IRI)*. 357–364. https://doi.org/10.1109/IRI.2013.6642493

[11] 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 International Conference on Cloud Computing (CLOUD)*. 115–122. https://doi.org/10.1109/CLOUD.2012.98

[12] Christoph Laaber, Joel Scheuner, and Philipp Leitner. 2019. Performance testing in the cloud. How bad is it really? *Empirical Software Engineering* (2019). https://doi.org/10.1007/s10664-019-09681-1 To appear. Preprint http://t.uzh.ch/T4.

[13] Philipp Leitner and Jürgen Cito. 2016. Patterns in the Chaos — A Study of Performance Variation and Predictability in Public IaaS Clouds. *ACM Trans. Internet Technol.* 16, 3 (April 2016), 15:1–15:23. https://doi.org/10.1145/2885497

[14] Peter Mell and Timothy Grance. 2011. *The NIST Definition of Cloud Computing*. Technical Report 800-145. National Institute of Standards and Technology (NIST). http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[15] 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

[16] 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

[17] 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

[18] J. Scheuner and P. Leitner. 2018. Estimating Cloud Application Performance Based on Micro-Benchmark Profiling. In *IEEE 11th International Conference on Cloud Computing (CLOUD)*. 90–97. https://doi.org/10.1109/CLOUD.2018.00019

[19] 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

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