Estimating Cloud Application Performance Based on Micro-Benchmark Profiling

Estimating Cloud Application Performance Based on Micro-Benchmark Profiling

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

Philipp Leitner
Software Engineering Division
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. To support service selection, micro-benchmarks are commonly used to identify the best performing cloud service. However, it remains unclear how relevant these synthetic micro-benchmarks are for gaining insights into the performance of real-world applications.**

Therefore, this paper develops a cloud benchmarking methodology that uses micro-benchmarks to profile applications and subsequently predicts how an application performs on a wide range of cloud services. A study with a real cloud provider (Amazon EC2) has been conducted to quantitatively evaluate the estimation model with 38 metrics from 23 micro-benchmarks and 2 applications from different domains. The results reveal remarkably low variability in cloud service performance and show that selected micro-benchmarks can estimate the duration of a scientific computing application with a relative error of less than 10% and the response time of a Web serving application with a relative error between 10% and 20%. In conclusion, this paper emphasizes the importance of cloud benchmarking by substantiating the suitability of micro-benchmarks for estimating application performance in comparison to common baselines but also highlights that only selected micro-benchmarks are relevant to estimate the performance of a particular application.

I. Introduction

In Infrastructure-as-a-Service (IaaS) [1], 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 less than 1 (shared) CPU core and 1 GB RAM (*e.g.*, *f1-micro*) to super-sized VMs with 128 CPU cores and 1952 GB RAM (*e.g.*, *x1.32xlarge*).

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[1], non-functional properties, such as performance, need to be quantified tediously. Previously promoted provider-defined performance metrics, such as Amazon's Elastic Compute Unit (ECU), have been quietly discontinued in favor of specifying the number of vCPUs and the type of processor as customary in on-premise data centers[2]. Moreover, with the increasing specialization of instance types (e.g., compute-, memory-, I/O-optimized), it appears apparent that resource costs are insufficient to derive application performance. An alternative way of dynamically assessing cloud resource performance is cloud benchmarking. This field of research is dedicated to objectively measuring and comparing the differences in performance between the various cloud services. A large body of literature [2]–[8] 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 [9], [10] and evaluate their performance [11]–[13] in cloud environments. Extensive studies have been conducted to collect micro-benchmark measurements for many different VM configurations [2]–[4], [14]. However, it remains unclear how relevant these artificial benchmarks are to gain insights into the performance of real-world applications.

The goal of this paper is to investigate the suitability of micro-benchmarks for estimating cloud application performance across different instance types. A prestudy in line with previous work on cloud benchmarking [2], [3] quantifies the performance variability for equally configured services (i.e., how variable do repeatedly acquired instances of the same instance type perform) because high variability could favor (if correlated) or hamper (if random) meaningful estimates and low variability could facilitate estimation across instance types. Following the prestudy, our main research addresses two questions:

- RQ1 How accurate can a set of micro-benchmarks estimate application performance?
- RQ2 Which subset of micro-benchmarks estimates application performance most accurately?

In order to answer these questions, a cloud benchmarking study has been designed, implemented, and conducted using Amazon's Elastic Compute Cloud (EC2) as an example of a

[1] https://www.cloudorado.com/cloud_providers_comparison.jsp

[2] http://blogs.gartner.com/kyle-hilgendorf/2014/04/16/aws-moves-from-ecu-to-vcpu/

real public cloud computing environment. We benchmarked 11 instance types and collected 38 metrics for 23 micro-benchmarks from nine micro-benchmark suites (e.g., StressNg or iperf) and two example application benchmarks (a scientific computing application and a Web application implemented via WordPress). This suite of micro- and application benchmarks [15] has been automated using the Web-based cloud benchmark manager Cloud WorkBench (CWB) [16], [17]. The prestudy reveals remarkably low variability in performance for different instances of the same instance type in comparision to prior research, which indicates that the studied cloud provider fundamentally changed its performance model. Our main results show that selected micro-benchmarks can estimate the duration of a scientific computing application with a relative error of less than 10% and the response time of a Web serving application with a relative error between 10% and 20%. We further show that choosing an appropriate micro-benchmark to estimate application performance on previously unseen instance types can vastly outperform commonly used baselines such as the number of vCPUs [18], provider-defined performance metrics [19], or resource costs. However, our study also indicates that the appropriate micro-benchmark choice can be very sensitive to configuration parameters and micro-benchmarks testing the same resource cannot necessarily be used interchangeably.

II. Related Work

Studies extensively analyzed the stability of performance delivered by cloud providers. One of the first large-scale studies to address variability in a cloud environment collected hourly measurements for over one month and revealed large variability around 20% for CPU, I/O, and network performance [5]. Other studies also observed high variability for instances of the same type [12], [20] and identified hardware heterogeneity [6], [7], [21] as the major cause for varying CPU performance [22] beyond CPU sharing and noise due to multi-tenancy [23], [24]. Further studies over time [2], [25] have shown that performance variability remained relevant, in particular for smaller instance types.

Micro-benchmarking aims at measuring cloud service performance for individual resources such as CPU, I/O, memory, and network. Initial studies [8] were extended in scope and led to some of the most important contributions in this field [3], [4]. Assessing and comparing the performance of cloud services has also become a business and companies such as CloudHarmony[3] or Cloud Spectator[4] offer comparison services and publish their own analysis reports [26].

One of the earliest efforts geared towards more modern workloads for the cloud comprises the Cloudstone benchmark [27], which proposes a new interaction-heavy Web 2.0 workload. CloudSuite [9] contributes an entire collection of scale-out workloads, which were incrementally (v3.0 as of Jan, 2018) improved [10] and the *SPEC Cloud™ IaaS 2016* [28] is specifically aimed to measure IaaS cloud performance. The YCSB suite [29] maintains a large collection of scale-out workloads for database systems. Several conceptual contributions [30], [31] suggest ideas and guidelines on how to design and implement application benchmarks for cloud environments.

Application profiling aims to capture the performance behavior of applications on different platforms using system-level monitoring tools [32], [33] or program similarity [34]. Application performance prediction is most closely related to the work in this paper. CloudProphet [35] collects resource traces of on-premise Web applications and replays them in cloud environments to accurately predict application performance for cloud instance types. CloudProphet focuses on accurate predictions for few instance types whereas this paper provides rough estimates for many different instance types. Hence, these approaches are complementary and could be combined to achieve broad instance type coverage and leverage CloudProphet to reduce the sampling effort, which is required to train the model in this paper. The CherryPick [18] system guides cloud configuration choices and iteratively refines runtime and cost predictions for distributed big data analytic jobs using a Bayesian Optimization model. In comparison, our work covers other application domains and requires less initial training samples. Stewart and Shen [36] contribute a performance model to predict the throughput and response time of multi-component online services by combining queuing models with system-level resource monitoring. Their evaluation is limited to 3 different server types while we focus on a broader range of 11 cloud instance types.

III. Methodology

Based on existing guidelines for cloud benchmarking [30], [31], [37], [38], we selected and designed relevant benchmarks that cover different cloud resources and application domains, as well as integrated them into the Cloud WorkBench (CWB) [16] execution environment. Details regarding this integration are out of scope here, and can be found in an accompanying publication [15]. These automated benchmarks are then repeatedly executed in a cloud environment and performance measurements are collected from all these benchmark executions. The raw performance data is pre-processed and prepared for the main analyses guided by the previously introduced RQs.

Figure 1 illustrates the high-level architecture of this cloud benchmarking methodology and lists the selected benchmarks. The *Benchmark Manager* coordinates the entire lifecycle of all benchmark executions. Its *Scheduler* component triggers new executions and its *Cloud Manager* component abstracts the cloud Provider APIs, Cloud VM provisioning, and communication with the Cloud VM. Via the *Provider API*, *Cloud VMs*, which represent the System Under Test (SUT), are acquired. 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

[3]https://cloudharmony.com/

[4]http://cloudspectator.com/

provisioning configuration for the Cloud VM from the *Provisioning Service* and applies it to install and configure all *Micro* and *Application (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 WordpressBench (WPBench), submit their tasks to the *Load Generator*, which generates the specified task workload from another dedicated cloud VM.

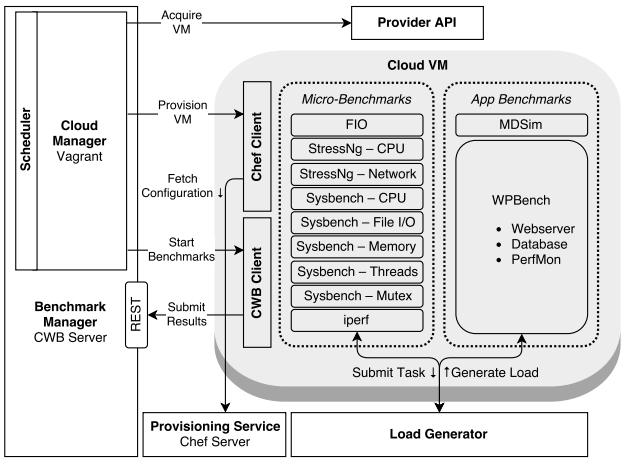

Figure 1. Architecture Overview

Figure 2 depicts an example benchmark configuration within the CWB Web interface. The configuration includes provider-specific resources (*e.g.*, the geographic region), specifies an execution schedule (*i.e.*, run every 3 hours), and refers to the entire benchmark suite. The benchmark suite called *rmit-combined* bundles all micro- and application benchmarks and implements the Randomized Multiple Interleaved Trials (RMIT) execution methodology [39] for a fair comparison of competing alternatives by randomizing the benchmark order within individual rounds. The selection of micro-benchmarks is motivated by prior use in research and aims for broad-resource coverage in the domains computation, I/O, network, and memory but also specifically tests individual resources (e.g., dividing I/O into low-level disk I/O and higher-level file I/O with different operation types and sizes). The application benchmarks consist of a Molecular Dynamics Simulation (MDSim) from the scientific computing domain and a WordPress Benchmark (WPBench) from the Web serving domain. MDSim is similarly used as an example of a scientific computing application in previous work [14] and WordPress was chosen because it is the most popular Content Management System (CMS) software (60% market share) used by 30% of the top 10 million websites (as of March 2017 from W3Techs[5]) and it also has been used previously for benchmarking cloud VMs [13]. The WPBench implements three user scenarios, which capture short read, search, and write blogging browser sessions. For detailed reproduction description and benchmark design rationales, we refer to Scheuner [40]. Further, we provide the entire benchmark suite as open source software[6].

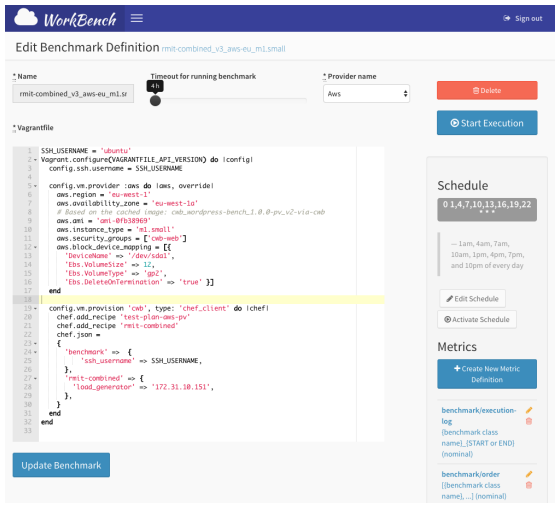

Figure 2. Benchmark Configuration

Data pre-processing transforms the raw performance data into an interim data set to facilitate further analysis by filtering (e.g., discarding failed executions), restructuring (e.g., pivoting), converting units (e.g., "Kb/sec" to "Mb/sec"), and replacing missing values. Appropriate replacement methods have been used in four cases where missing values occurred due to an adjustment in metric parsing and few transient failures. For detailed description and illustration, we again refer to Scheuner [40]. In addition, all scripts, as well as the raw, interim, and output data sets, are documented and freely available on Github[7].

IV. Benchmarking Data Set

Using the methodology from the previous section, a benchmarking data set was collected for the Amazon EC2 cloud provider. All configurations build upon the officially maintained Ubuntu 14.04 LTS images[8] and have attached the general purpose storage type *gp2*, which Amazon Web Services (AWS) recommends for most workloads.

Table I
EC2 Instance Type Specifications[9]

Instance Type	vCPU	ECU	RAM	Network	Cost (eu / us)*
m1.small	1	1	1.7	Low	0.047 / 0.044
m1.medium	1	2	3.75	Moderate	0.095 / 0.087
m3.medium	1	3	3.75	Moderate	0.073 / 0.175
m1.large	2	4	7.5	Moderate	0.190 / 0.175
m3.large	2	6.5	7.5	Moderate	0.146 / 0.133
m4.large	2	6.5	8.0	Moderate	0.111 / 0.100
c3.large	2	7	3.75	Moderate	0.120 / 0.105
c4.large	2	8	3.75	Moderate	0.113 / 0.100
c3.xlarge	4	14	7.5	Moderate	0.239 / 0.210
c4.xlarge	4	16	7.5	High	0.226 / 0.199
c1.xlarge	8	20	7	High	0.592 / 0.520

*in USD/h for Linux On-Demand as of 2017-05-19

[5]https://w3techs.com/technologies/overview/content_management/all

[6]https://github.com/sealuzh/cwb-benchmarks/tree/master/rmit-combined

[7]https://github.com/joe4dev/cwb-analysis

[8]https://cloud-images.ubuntu.com/locator/ec2/

Table I 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. Table I also provides the EC2 Compute Unit (ECU) specification, which Amazon used to promote as their own relative measure for CPU performance. An ECU is equivalent to the CPU power of a *m1.small* instance or a 1.0-1.2 GHz 2007 Opteron or Xeon processor type [4]. Amazon claims to conduct benchmarking to align the ECU measure with CPU power in particular regarding integer operations. However, AWS quietly discontinued this approach in 2014 and moved to a more traditional way, as customary in on-premise data centers, of specifying the number of vCPUs and the type of processor[10]. The ECU model is insufficient to describe the family of general purpose instance types that follow a formal model for burstable CPU performance [41]. These bursting instance types are not included in this study because their inherently varying performance impedes controlled benchmarking.

The regions *eu-west-1* (Ireland) and *us-east-1* (N. Virginia) were chosen to compare the results with prior work [2]. 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.

V. Variability for the same Instance Types

Before addressing our research questions, it is necessary to assess the performance variability for different instances of the same instance type. We compare the Relative Standard Deviations (RSDs) for 38 metrics against a 5% relevancy threshold following the definition of a large benchmarking study [2]. The RSD is formally defined as $\text{RSD} = 100 \cdot \frac{\sigma_m}{\overline{m}}$ where σ_m is the absolute standard deviation and $\overline{m}$ is the arithmetic mean of the metric m. The five selected configurations (*i.e.*, instance type and region) have 33 samples for the 38 metrics and focus on smaller instance types because prior work has shown that they tend to deliver less stable performance than larger instance types [2], [24], [25].

[9] http://www.ec2instances.info/

[10] http://blogs.gartner.com/kyle-hilgendorf/2014/04/16/aws-moves-from-ecu-to-vcpu/

1) Results: Figure 3 summarizes the variability in terms of RSD for each selected configuration and all benchmarks using violin plots with annotated mean values to attribute for its non-normal distribution. All medians are clearly below the 5% threshold and almost all means, denoted by the blue diamond, lie underneath this relevant variability threshold. Thus, we conclude that performance does not vary relevantly for the majority of benchmarks in all these tested configurations.

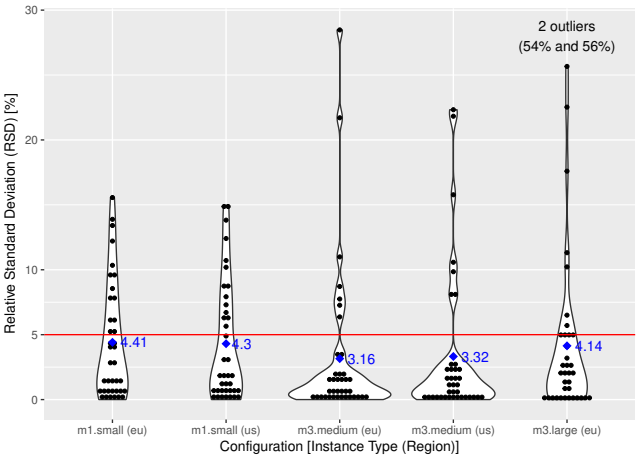

Figure 3. Variability per Configuration

2) Discussion: The observed performance stability is fairly surprising and reveals a shifts from previously reported large variability in performance (>10% RSD) between supposedly identical instances [5]–[7], [12], [20]–[24], [42] towards delivering much more stable performance. Concerning Amazon EC2, all these studies exclusively focused on instance types of the first generation[11] and more recent studies provide further evidence that newer generations perform remarkably more stable [2], [43], [44]. CPU performance became very predictable at levels below 0.3% RSD, which is largely due to diminishing relevancy of hardware heterogeneity in AWS (which is also reported by Leitner and Cito [2]). The migration from an HDD- to an SSD-backed file I/O storage reduced variability from 20%-100% [2] to below 1.5%-10%. Most surprisingly, network performance achieved almost perfect stability, which contrasts the 25% RSD from 2012 [6].

3) Implications: The results of this paper suggest that instance seeking and placement gaming approaches [6], [7], [21] are no longer worthwhile under current conditions in Amazon EC2. Furthermore, cloud benchmarking studies spent a lot of resources on obtaining relevant sample sizes to achieve statistically plausible results within typical confidence intervals (*i.e.*, 95% or 99%). Our results indicate that benchmarking efforts can be reduced considerably and even single samples can be sufficient to achieve the 99% confidence interval for highly stable categories such as CPU or intra-Availability Zone (AZ) network performance. This makes clouds more viable to collect a relevant amount of performance data for a broad

[11] https://aws.amazon.com/blogs/aws/ec2-instance-history/

range of instance types. Further, public clouds can now potentially be used even in use case such as software performance testing, where performance predictability is key [44].

VI. Results and Discussion

We now present, discuss, and summarize the results guided by the previously introduced research questions.

A. RQ1 – Estimation Accuracy

To estimate the application performance, a linear regression model is trained using 38 metrics from 23 micro-benchmarks and evaluated for two applications from different domains. Motivated by the findings from our prestudy, as discussed in Section V, sample filtering selects three iterations from the same execution for each instance type in the European data center. The boundary instance types (*i.e.*, the smallest and largest) are labeled as training data to capture the largest possible instance type diversity. A forward feature selection algorithm is combined with linear regression to automatically identify the best performing set of features (*i.e.*, metrics) regarding the relative error performance criterion, which is defined as $Relative\ Error(RE) = 100 \cdot \frac{\epsilon}{m}$ where ϵ is the absolute error and m is the actual measurement. Forward feature selection starts with an empty set of features and iteratively adds a previously unused feature. The candidate feature set is then used to build a linear regression model with the training data. This model is applied to the test data and the mean relative error is calculated between the predicted and actual application performance. In doing so, only features that yield the highest gain for the performance criterion (*i.e.*, minimize the relative error) are kept. This process is repeated until no additional feature can further improve the relative error. Finally, forward feature selection outputs a weighted feature list and various performance indicators such as the relative error or the Pearson correlation coefficient, also known as squared correlation or R^2.

1) Results: Table II reports the estimation accuracy in terms of the relative estimation error achieved by the best micro-benchmark predictor for WPBench and MDSim. For WPBench, all three scenarios (*i.e.*, read, search, write) are evaluated regarding their Response Time (RT). For the results of the additional throughput metric, we refer to [40]. The boundary instance types are labeled as training data (*i.e.*, *Train*) and the relative estimation error is listed per instance type. For each instance type, the averaged relative error over the three iterations indicates how far the estimated performance consistently lies above (*cf.*, +) or below (*cf.*,-) the actual performance. Wherever the actual values are spread on both sides of the regression line, the pipe (*cf.*, |) indicates the absolute error due to high variability between iterations. In summary, the mean Relative Error (RE) combined with the max RE indicates the fitness of cross-instance performance estimation. The *max RE* estimates the upper bound for the relative error assuming that the smallest instance performs worst and the largest instance performs best. This provides an orientation on how far the minimum and maximum of the application performance is spread. Hence, a high max RE implies that high accuracy (*i.e.*, low relative error) is harder to achieve. Conversely, a low max RE diminishes the significance of low relative errors because they are more likely to occur by chance.

Table II
Relative Estimation Errors [%]

Instance Type	WPBench Response Time			MDSim Duration
	Read	Search	Write	
m1.small	Train	Train	Train	Train
m3.medium (pv)	+6.9	+7.5	|21.5|	+5.4
m3.medium (hvm)	+14.7	+6.1	|42.6|	+5.8
m1.medium	+9.0	+9.0	|36.2|	-0.2
m3.large	-17.8	-25.6	|53.1|	-10.2
m1.large	+17.0	+17.5	|40.9|	-0.8
c3.large	-17.4	-26.1	|51.5|	-10.1
m4.large	-3.6	-12.8	|52.8|	-12.4
c4.large	-9.7	-18.4	|50.3|	-12.5
c3.xlarge	-26.3	-34.4	+32.8	-11.2
c4.xlarge	-2.2	-17.6	+26.1	-13.7
c1.xlarge	Train	Train	Train	Train
Mean RE	**12.5**	**17.5**	**40.8**	**8.2**
Max RE	2100	1810	140	600

2) Discussion: The most accurate estimates are achieved by MDSim and the read and search scenarios of WPBench as shown in Table II. Duration estimates for MDSim reach 8.2% accuracy for its duration values in the interval $[69.7, 491.7]$ seconds (*cf.*, max RE of 600%). The read and search scenarios of WPBench exhibit by far the largest spread in their response time distribution in the interval $[65.8, 1457.8]$. This spread is illustrated in Figure 4 for the read scenario and results in a maximum relative error of 2100%. Nevertheless, moderate relative errors of 12.5% and 17.5% are achieved on average. Furthermore, these linear regression models are statistically significant at the 0.001 level and thus support the assumption of low variability shown in RQ1.

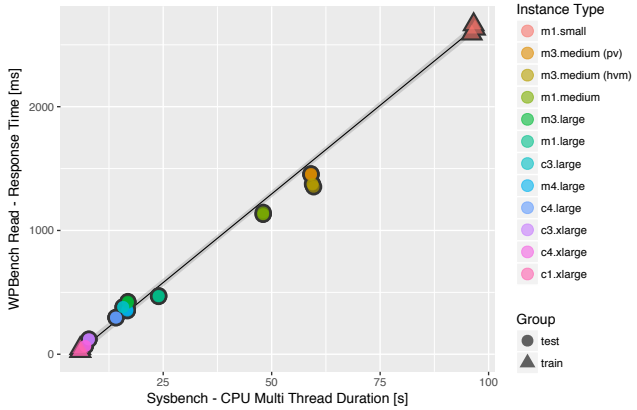

Figure 4. Linear Regression Model for WPBench Read – Response Time

The relative errors for the WPBench write scenario are generally high, particularly given the relatively low spread of their performance data. Additionally, even within the same instance type, application performance is overestimated and underestimated simultaneously and therefore provided as modulus value. Furthermore, their regression models are less significant at the 0.05 (response time) and the 0.1 (throughput) level,

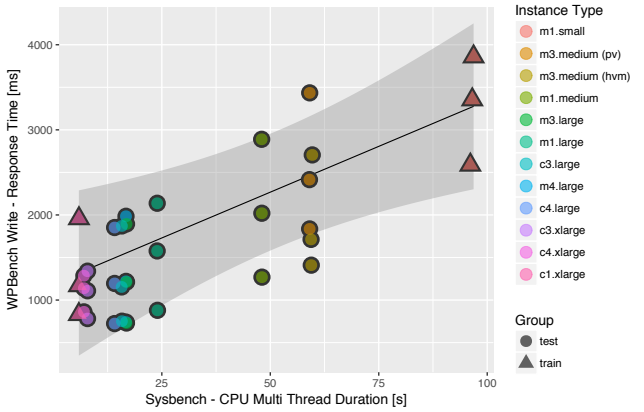

Figure 5. Linear Regression Model for WPBench Write – Response Time

which adds further evidence for the existence of performance variability between different iterations. This hypothesis is further investigated by performing statistical tests for the 5 instance types used in RQ1 with relevant sample sizes between 33 and 61 executions from the interim data set. A One-way ANOVA test [45] is performed upon the iteration column as its group attribute. The results confirm that both response time and throughput vary greatly (*i.e.*, particularly high f value) between the 3 different iterations with high significance (*i.e.*, $p < 0.001$) for all 5 tested instance types. ANOVA is an omnibus test and therefore only confirms a statistically significant difference between the iterations but does identify the specific iterations that differ statistically significant from each other. Therefore, we also conduct a Mann Whitney U-Test to demonstrate that even the differences between all pairs of iterations are statistically significant for all 5 instance types. The increasing performance between the individual iterations becomes apparent in the linear regression model shown in Figure 5. Notice that the statistical tests have also shown that apart from WPBench, none of the other benchmarks exhibit statistically significant differences between iterations.

3) Implications: The ability to estimate application performance with an acceptable accuracy highlights the usefulness of micro-benchmarks. It encourages to catalog cloud services based on their performance characteristics using generic micro-benchmarks and subsequently relate to the performance of individual applications by using at least two instance types at the boundary of the designated resource spectrum (i.e., the least and most capable resource) as training input for the estimation model. The remaining challenge is to identify a set of relevant micro-benchmarks for a given application.

B. RQ2 – Micro-Benchmark Selection

Following the same feature selection process as described for RQ1, the most important features (*i.e.*, micro-benchmark metrics) are identified and compared against three commonly used baselines.

1) Results: For the response time across all scenarios of WPBench and the duration of MDSim, forward feature selection included the Sysbench – CPU Multi-Thread micro-benchmark in the linear model. For the WPBench write scenario, two additional benchmarks were proposed with equal weights but rejected because their contribution to the model was statistically insignificant with p=0.393 for fio/8k-rand-read-latency and p=0.450 for fio/4k-seq-write-bandwidth. Similarly, for MDSim, the additional attribute fio/8k-rand-read-iops was discarded due to its p-value 0.0976 at the border of being insignificant. Due to limited space, we omit the throughput metric of the WPBench benchmark where the StressNg – Network Internet Control Message Protocol (ICMP) Ping was chosen as the best estimator.

Table III presents the best benchmark estimators and three instance type specification metrics serving as a baseline. For each estimator, the mean relative error with its range and the squared correlation R^2 are provided. R-squared, also known as the coefficient of determination, measures how well the data fits the regression line where 0% implies that the model captures no variability in the data and 100% implies that the model perfectly fits all data on the regression line. Finally, the max RE is provided analogous to its previous definition.

Table III
WPBench Response Time and MDSim Duration Estimators [%]

	WPBench			**MDSim**
Benchmark	Read RT	Search RT	Write RT	Duration
Sysbench – CPU Multi-Thread Duration				
RE±Range	12.5±7.1	17.5±8.7	40.8±34.9	8.2±4.7
R^2	99.2	98.9	42.5	99.8
Sysbench – CPU Single-Thread Duration				
RE±Range	454±520	411.72±451	41.7±20.8	232±163
R^2	85.1	83.8	38.7	87.3
Baseline				
vCPUs				
RE±Range	616±607	546±515	127±89.8	317±184
R^2	68.0	68.7	28.1	68.3
ECU				
RE±Range	359±219	319±185.13	100±79.2	206±95
R^2	64.6	64.7	27.3	65.6
Cost				
RE±Range	663±730	586±622	127.3±91.3	329.3±222
R^2	59.1	61.2	22.7	57.9
Max RE	2100	1810	140	600

For the response time of the WPBench read and search scenarios and the duration of MDSim presented in Table III, the multi-thread Sysbench – CPU benchmark serves as a good estimator. The almost perfect fit of the regression model (*i.e.*, $R^2 > 98.9$) together with low relative errors below 10% for MDSim and between 10% and 20% for the read and write scenarios of WPBench indicate that this multi-thread CPU benchmark can be a suitable estimator. Further, the vastly inferior results for the single-thread version of the same benchmark reveal that they cannot be used interchangeably. In addition, the improvements upon the vCPU, ECU, and hourly cost baselines are substantial. Although ECU is already ~50% more accurate than using the number of vCPUs, the Sysbench benchmark outperforms this baseline by factor 17 to 29 in terms of relative error. The CPU benchmark also fits the regression line considerably better with over 33%

improvement in squared correlation compared to the baseline metrics. Finally, the costs baseline performs worst in all metrics.

2) Discussion: While the results support the conjecture that these estimates could be meaningful for applications with a resource profile similar to micro-benchmarks, such as MDSim, the linear model also works surprisingly well for a more diverse application such as WPBench. The MDSim application is CPU-intense, potentially stresses the main memory, but does not involve I/O operations. Despite the fact that MDSim performs higher-level real-world computations compared to low-level artificial micro-benchmark workloads, such as iterating over meaningless loops, resource usage of MDSim is presumably very similar to a CPU micro-benchmark. Conversely, the WPBench application is much more heterogeneous and its resource footprint is not inherently obvious using various kind of system resources. Beyond CPU-driven request processing, WPBench receives requests and sends responses over the network, reads and writes from the file system, and requires the scheduler to switch between its various database or Web server processes. Therefore, it is not apparent whether micro-benchmarks are able to capture such a varying workload. Nevertheless, the results revealed that the linear model is able to assess application response time surprisingly well, prevalently with error rates below 20%.

3) Implications: Concurrency plays an important role when estimating the performance across instance types with a different number of vCPUs. The Sysbench – CPU single-thread versus multi-thread scenario revealed that micro-benchmarks need to match its estimation target application in terms of optimization for multi-core (*cf.*, multi-vCPUs) platforms. It also shows that CPU micro-benchmarks are suited to identify optimal instance types for workloads with a particular concurrency level (*e.g.*, single-threaded). Further, it emphasizes that benchmark parameters, such as the level of concurrency, can have a profound impact on results.

The baseline metrics vCPU, ECU, and cost are insufficient to estimate the performance of certain applications. The cost baseline is worst to estimate application performance and should never be used in isolation for estimating application performance. The number of vCPUs is only slightly better and fails to capture fundamental technological differences such as different CPU clock frequencies and thus exhibits large relative errors for many instance types. Although the ECU metric yields considerably better estimates than vCPU, its relative error is still unacceptably high above 100%. Therefore, ECUs could be used at most to obtain a very rough estimate if no other metric is available but application specific micro-benchmarks should be favored to obtain the most accurate application performance estimate.

VII. Conclusion

This paper investigated the relevancy of widely-used artificial micro-benchmarks to estimate real-world application performance. A cloud benchmarking methodology has been designed that combines single-instance and multi-instance micro- and application benchmarks. The methodology has been instantiated in a study with a market-leading cloud provider and a linear estimation model has been evaluated. Over 60000 measurements were collected to answer three research questions addressing performance variability, estimation accuracy, and micro-benchmark selection.

Contrary to previous research, our results reveal that performance between instances does *not* vary relevantly anymore. The low performance variability motivates inter-instance type performance estimation because only the sufficiency of small sample sizes makes such an approach practically viable. We show that selective micro-benchmarks are able to estimate certain application performance metrics with acceptable accuracy. A scientific computing application achieves relative error rates below 10% and the response time of a Web serving application is estimated with a relative error between 10% and 20%. Further, a single CPU benchmark was able to estimate the duration of a scientific computing application and the response time of a Web serving application most accurately. However, it has also been shown that benchmarks cannot necessarily be used interchangeably even if they test the same resource and benchmark parameters can have a profound impact.

This paper substantiates the suitability of micro-benchmarks for estimating application performance, but also highlights that only some micro-benchmarks are relevant for any particular application. It also emphasizes the importance of cloud benchmarking by showing that benchmark-based metrics can vastly improve estimation accuracy upon using instance type specification-based metrics. Further, this paper corroborates the dynamicity of cloud environments with indications that the tested cloud provider shifts from delivering best effort performance to specifically designed performance levels with high predictability.

Beyond covering further traditional instance types offered by well-known providers [46], such as Microsoft Azure, a particularly interesting extension examines individually tailorable instance types, such as offered by Century Link[12] or Google's Cloud Platform[13]. Another avenue for future research is the extension to scale-out workloads with distributed application components. Finally, cloud instance selection could become an integral part of scaling strategies combined with runtime performance data to train the estimation model instead of being perceived as wasted effort.

Acknowledgment

We are grateful to Adam Barker and Long Thai for their input to initial discussions regarding this research.

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, and M. Zaharia, "Above

[12] https://www.ctl.io/servers/#Features

[13] https://cloud.google.com/custom-machine-types/

the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley, Tech. Rep., Feb 2009.

[2] P. Leitner and J. Cito, "Patterns in the Chaos — A Study of Performance Variation and Predictability in Public IaaS Clouds," *ACM Trans. Internet Technol.*, vol. 16, no. 3, pp. 15:1–15:23, Apr. 2016.

[3] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 931–945, June 2011.

[4] 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 *Cloud Computing*. Springer, 2009, vol. 34, pp. 115–131.

[5] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance," *Proceedings of the VLDB Endowment*, vol. 3, no. 1, pp. 460–471, Sep. 2010.

[6] B. Farley, A. Juels, V. Varadarajan, T. Ristenpart, K. D. Bowers, and M. M. Swift, "More for Your Money: Exploiting Performance Heterogeneity in Public Clouds," in *Proc. of the 3rd ACM Symposium on Cloud Computing (SoCC '12)*, 2012, pp. 20:1–20:14.

[7] Z. Ou, H. Zhuang, A. Lukyanenko, J. K. Nurminen, P. Hui, V. Mazalov, and A. Ylä-Jääski, "Is the Same Instance Type Created Equal? Exploiting Heterogeneity of Public Clouds," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 201–214, July 2013.

[8] E. Walker, "Benchmarking Amazon EC2 for High-Performance Scientific Computing," *Usenix Login*, vol. 33, no. 5, pp. 18–23, October 2008.

[9] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," in *ASPLOS '12*, 2012, pp. 37–48.

[10] T. Palit, Y. Shen, and M. Ferdman, "Demystifying cloud benchmarking," in *IEEE ISPASS*, April 2016, pp. 122–132.

[11] A. Iosup, N. Yigitbasi, and D. Epema, "On the Performance Variability of Production Cloud Services," in *11th IEEE/ACM Int. Symp. on CCGrid*, May 2011, pp. 104–113.

[12] J. Dejun, G. Pierre, and C.-H. Chi, *EC2 Performance Analysis for Resource Provisioning of Service-Oriented Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 197–207.

[13] A. H. Borhani, P. Leitner, B. S. Lee, X. Li, and T. Hung, "Wpress: An application-driven performance benchmark for cloud-based virtual machines," in *2014 IEEE 18th International Enterprise Distributed Object Computing Conference*, Sept 2014, pp. 101–109.

[14] B. Varghese, O. Akgun, I. Miguel, L. Thai, and A. Barker, "Cloud Benchmarking For Maximising Performance of Scientific Applications," *IEEE Transactions on Cloud Computing*, no. 99, 2017.

[15] J. Scheuner and P. Leitner, "A cloud benchmark suite combining micro and applications benchmarks," in *4th International Workshop on Quality-Aware DevOps (QUDOS)*, In Press.

[16] J. Scheuner, P. Leitner, J. Cito, and H. Gall, "Cloud WorkBench - Infrastructure-as-Code Based Cloud Benchmarking," in *Proc. of the 6th IEEE Int. Conf. on CloudCom*, 2014.

[17] J. Scheuner, J. Cito, P. Leitner, and H. Gall, "Cloud WorkBench: Benchmarking IaaS Providers based on Infrastructure-as-Code," in *Proc. of the 24th Int. World Wide Web Conference (WWW) - Demo Track*, 2015.

[18] 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 *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017.

[19] J. O'Loughlin and L. Gillam, "Towards performance prediction for public infrastructure clouds: An ec2 case study," in *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 1, Dec 2013, pp. 475–480.

[20] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: Comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC '10)*, 2010, pp. 1–14.

[21] 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 *Proc. of the 4th USENIX Conference on HotCloud*, 2012.

[22] D. Cerotti, M. Gribaudo, P. Piazzolla, and G. Serazzi, "Flexible CPU Provisioning in Clouds: A New Source of Performance Unpredictability," in *9th Int. Conf. on Quantitative Evaluation of Systems*, Sept 2012, pp. 230–237.

[23] S. K. Barker and P. Shenoy, "Empirical evaluation of latency-sensitive application performance in the cloud," in *Proc. of the 1st ACM/SIGMM Conf. on Multimedia Systems (MMSys)*, 2010, pp. 35–46.

[24] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of amazon ec2 data center," in *Proc. IEEE INFOCOM*, March 2010, pp. 1–9.

[25] L. Kotthoff, "Reliability of computational experiments on virtualised hardware," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 26, no. 1, pp. 33–49, 2014.

[26] C. Spectator, "Price-Performance Analysis of the Top 10 Public IaaS Vendors," Cloud Spectator, Tech. Rep., 2017.

[27] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, A. Fox, and D. Patterson, "Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0," 2008.

[28] T. S. Consortium, "SPEC Cloud^{TM} IaaS 2016 Benchmark," 2016. [Online]. Available: http://spec.org/cloud_iaas2016/

[29] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking Cloud Serving Systems with YCSB," in *Proc. of the 1st ACM Symposium on Cloud Computing (SoCC)*, 2010, pp. 143–154.

[30] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing, "How is the weather tomorrow?: Towards a benchmark for the cloud," in *Proc. of the 2nd Int. Workshop on Testing Database Systems (DBTest)*. ETH Zurich, 2009, pp. 9:1–9:6.

[31] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun, "Benchmarking in the Cloud: What It Should, Can, and Cannot Be," in *Selected Topics in Performance Evaluation and Benchmarking*. Springer, 2013, vol. 7755, pp. 173–188.

[32] A. Evangelinou, M. Ciavotta, D. Ardagna, A. Kopaneli, G. Kousiouris, and T. Varvarigou, "Enterprise applications cloud rightsizing through a joint benchmarking and optimization approach," *Future Generation Computer Systems*, pp. –, 2016.

[33] M. Canuto, R. Bosch, M. Macias, and J. Guitart, "A methodology for full-system power modeling in heterogeneous data centers," in *Proceedings of the 9th International Conference on Utility and Cloud Computing (UCC '16)*, 2016, pp. 20–29.

[34] K. Hoste, A. Phansalkar, L. Eeckhout, A. Georges, L. K. John, and K. De Bosschere, "Performance prediction based on inherent program similarity," in *Proc. of the 15th Int. Conf. on Parallel Architectures and Compilation Techniques (PACT)*, 2006, pp. 114–122.

[35] A. Li, X. Zong, S. Kandula, X. Yang, and M. Zhang, "Cloudprophet: Towards application performance prediction in cloud," in *Proc. of the ACM/SIGCOMM Conf. (SIGCOMM)*, 2011, pp. 426–427.

[36] C. Stewart and K. Shen, "Performance Modeling and System Management for Multi-component Online Services," in *Proc. of the 2nd Conf. on Symposium on NSDI*, 2005, pp. 71–84.

[37] V. Vedam and J. Vemulapati, "Demystifying cloud benchmarking paradigm - an in depth view," in *36th IEEE Computer Software and Applications Conference (COMPSAC)*, July 2012, pp. 416–421.

[38] A. Iosup, R. Prodan, and D. Epema, *IaaS Cloud Benchmarking: Approaches, Challenges, and Experience*. Springer, 2014, pp. 83–104.

[39] A. Abedi and T. Brecht, "Conducting repeatable experiments in highly variable cloud computing environments," in *8th ACM/SPEC International Conference on Performance Engineering (ICPE)*, April 2017.

[40] J. Scheuner, "Cloud Benchmarking – Estimating Cloud Application Performance Based on Micro Benchmark Profiling," Master's thesis, University of Zurich, 2017. [Online]. Available: http://www.merlin.uzh.ch/publication/show/15364

[41] P. Leitner and J. Scheuner, "Bursting With Possibilities – an Empirical Study of Credit-Based Bursting Cloud Instance Types," in *8th IEEE/ACM Int. Conf. on Utility and Cloud Computing (UCC)*, 2015.

[42] Y. El-Khamra, H. Kim, S. Jha, and M. Parashar, "Exploring the Performance Fluctuations of HPC Workloads on Clouds," in *2nd IEEE Int. Conf. on CloudCom*, Nov 2010, pp. 383–387.

[43] C. Davatz, C. Inzinger, J. Scheuner, and P. Leitner, "An approach and case study of cloud instance type selection for multi-tier web applications," in *17th IEEE/ACM CCGrid*, 2017.

[44] C. Laaber, J. Scheuner, and P. Leitner, "Performance testing in the cloud. how bad is it really?" *PeerJ Preprints*, vol. 6, Jan. 2018. [Online]. Available: https://doi.org/10.7287/peerj.preprints.3507v1

[45] L. S. F. Barbara G. Tabachnick, *Using Multivariate Statistics*, 6th ed., ser. 6th Edition. Pearson, 2012.

[46] L. Leong, G. Petri, B. Gill, and M. Dorosh, "Magic Quadrant for Cloud Infrastructure as a Service, Worldwide," August 2016.