Initial Experiments with Duet Benchmarking: Performance Testing Interference in the Cloud

Lubomír Bulej Vojtěch Horký Petr Tůma
Department of Distributed and Dependable Systems
Charles University, Faculty of Mathematics and Physics
Malostranské náměstí 25, 118 00 Prague 1, Czech Republic
{bulej|horky|tuma}@d3s.mff.cuni.cz

***Abstract*—Accurate performance testing may require many measurements and therefore many machines to execute on. When many machines are needed, the cloud offers a tempting solution, however, measurements conducted in the cloud are generally considered unstable. In the context of comparing performance of two workloads, we propose a measurement procedure that improves accuracy by executing the workloads concurrently and using the measurements to filter outside interference. Depending on the platform used, experiments show average accuracy improvement ranging from 114% to 683% over sequential measurements on workloads running the ScalaBench suite with the Graal compiler.**

I. Introduction

The increasing use of agile development practices brings more emphasis on continuous software testing, especially when relying on continuous integration and deployment (CI/CD). Ideally, this testing should focus not only on functional correctness, but also on performance, in particular to prevent unintended performance degradation between commits [3].

Particular to performance testing is the inherent trade-off between accuracy and test execution time. This is a problem when automating performance test execution and evaluation, which is necessary for CI/CD. While long execution time provides better results due to averaging of noise, it is expensive both in terms of time and computing resources, and may easily become impractical if the speed of development exceeds the performance testing capacity. Shorter test execution time may produce false alarms or lose sensitivity – both highly undesirable outcomes.

Existing strategies to alleviate this problem include testing only subsets of software versions and bisecting when a performance change has been detected [9], or identifying changes that are likely to impact performance [16]. Orthogonal to these strategies, performance testing can also be effectively parallelized, especially if there are multiple tests to be executed with each version, because these can be built and executed independently. However, neither test reduction nor test parallelization entirely removes the problem of infrastructure capacity limits.

In this context, an obvious question to ask is whether performance testing can be offloaded to the cloud, just like other CI/CD tasks. However, the answer is not clear – the cloud is known to provide affordable computing capacity, but not necessarily the performance stability required for performance testing. In fact, performance measurements in the cloud are known to be noisy, in part due to overheads associated with virtualized execution, in part due to interference from neighbor workloads.

Our work contributes a novel performance measurement procedure, dubbed duet measurement, which improves measurement accuracy in virtualized environment and thus makes performance testing in the cloud more effective. We evaluate the approach on three different instance types in three different zones of the Amazon Elastic Cloud and on a reference bare metal infrastructure. For the selected performance testing workloads and depending on the platform used, the accuracy improvement expressed as reduction in mean confidence interval width over common sequential measurements ranges from 114% to 683%. While the approach is general, we demonstrate its use on specific performance tests used with the open source Graal compiler project.

In the rest of the paper, Section II provides background on performance testing and cloud specific challenges, Section III provides description of the approach, and Section IV presents and discusses results of experimental evaluation. We review related work in Section V and conclude the paper in Section VI

II. Background

From the many forms of performance testing, we focus on the task of detecting performance changes between two versions of a software project. A common approach is to use a benchmark workload to exercise both versions of the software project, and to measure and compare the workload execution times. To accommodate the variability inherent to the observations, the comparison relies on statistical hypothesis testing.

Different sources of variability can influence the observed execution times at different granularities, and the performance testing procedure must ensure that significant sources of variability are sufficiently represented in the measured data. Benchmarks therefore repeatedly execute the same task and measure the time of each *iteration*. This captures variability caused by factors such as scheduling decisions, processor caches, or background load. In addition, benchmarks are executed repeatedly to obtain sequences of iteration times from multiple benchmark *runs*. This captures variability caused by

This is the authors' version of the paper: *L. Bulej, V. Horký, P. Tůma: Initial Experiments with Duet Benchmarking: Performance Testing Interference in the Cloud, in Proc. 27th IEEE Intl. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 249-255, 2019.*
The final published version can be found at http://doi.org/10.1109/MASCOTS.2019.00035

factors such as memory mapping or decisions of the managed platform (e.g. just-in-time compilations or garbage collections), which change between runs but not within a single run.

To avoid excess measurement variability, benchmarks are typically executed on dedicated machines configured to disable disruptive features such as advanced power management. This requires degree of control over the experimental platform that is often not available in the cloud. Furthermore, cloud providers offer abstract virtual machine types that can run on different types of physical hosts [14], resulting in different execution times even for the same code. Cloud virtual machines can also suffer from performance interference of neighbor workloads, which the virtualization technology cannot entirely eliminate. The same holds for continuous integration solutions executing in the cloud, such as Travis [22] or GitLab Runner [7].

In summary, measurements from different virtual machines are incomparable, and measurements from a single virtual machine may be heavily influenced by performance interference. Such data is of little use to a performance testing procedure relying on data coming from the controlled environment of a private infrastructure, we therefore have to design a procedure that takes the specifics of cloud into account.

III. Duet Measurements

Current best practice for performance measurements in the cloud uses sequential measurements with randomized interleaving of workloads [1]. This practice is based on an experiment model where the execution environment suffers from external performance interference, such as neighbor workloads. When the evaluated workloads are measured in random order, the external performance interference impacts each workload with equal probability. A long enough execution should therefore avoid possible bias due to interference [1].

Our approach takes this idea further – we aim to obtain simultaneous measurements of execution time for both evaluated workloads. To obtain such paired measurements, the benchmark programs that implement the test workloads are executed in parallel inside a virtual machine with two virtual cores, with each of the workloads restricted to one virtual core. In addition, the benchmark runs and the task iterations are synchronized using a shared-memory barrier so that they always start at the same time. We call this procedure duet measurements.

A. Synchronized Interference

Crucial difference between the randomized interleaving of workloads and the duet measurements is the synchronized character of interference. With the randomized interleaving of workloads, external performance interference impacts the measured workloads independently and individually. With a long enough execution, the interference should eventually impact all the measured workloads similarly, avoiding possible bias but still increasing variance across measurements.

With duet measurements, both measured workloads execute in parallel in one virtual machine, with synchronized benchmark runs and task iterations. Any external performance interference that impacts the virtual machine as a whole is therefore encountered simultaneously in both workloads. Where the randomized interleaving of workloads eliminates systematic bias only across long enough execution (such that any external performance interference has enough opportunities to hit all workloads equally), the duet measurements prevent such bias already with individual measurements.

Additionally, the duet measurement procedure naturally provides paired measurements, that is, measurements collected on the two workloads at the same time. For some types of external performance interference – such as linear slowdown due to resource contention – this may help separate the variance due to the interference from the variance inherent to the measured workloads and therefore further improve accuracy.

B. Impact Symmetry

While the duet measurement procedure makes sure any external performance interference is encountered simultaneously in both workloads, this is only useful if the actual impact of the interference on the workload performance is similar in both workloads. This very much depends on both the nature of the interference and the configuration of the execution environment, which is often proprietary or outside experiment control – for example, both the Amazon Elastic Cloud [2] and the Google Compute Engine [8] originate from hypervisor technologies that use weighted fair share processor scheduling by default, but neither documents the actual scheduling strategy used in their commercial services.

To avoid basing our case for impact symmetry on a complex arrangement of ever changing technical details, we look instead at the practical implications that a lack of symmetry would have. When employed for detecting performance degradation between commits, the duet measurement procedure executes two similar workloads bound to two virtual cores of the same virtual machine. If a cloud platform were to exhibit systematic performance difference between the two virtual cores with duet measurements, it would likely exhibit similarly unwarranted performance difference in many common concurrent workloads. Such behavior would be considered a bug and likely remedied.

As the flip side of the same argument, the duet measurements are less likely to work when comparing dissimilar workloads – for example, should one of the workloads be strongly processor bound and the other strongly I/O bound, external processor sharing interference would be more likely to impact the former, and external I/O sharing interference the latter. This is less likely to happen when comparing neighboring commits of the same software project, as we do in our use case.

C. Mutual Workload Interference

Compared to common sequential measurements, where each measured workload executes in isolation, the duet measurement procedure adds the potential for the two measured workloads to interfere with each other. This is the case especially when the two virtual cores used by the workloads map to two hardware threads of the same physical processor core. Such virtual cores would compete for the shared execution units of the core subject

This is the authors' version of the paper: *L. Bulej, V. Horký, P. Tůma: Initial Experiments with Duet Benchmarking: Performance Testing Interference in the Cloud, in Proc. 27th IEEE Intl. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 249-255, 2019.*
The final published version can be found at `http://doi.org/10.1109/MASCOTS.2019.00035`

to the processor scheduling policy, which generally aims to maximize the execution unit use [11].

The use of two hardware threads of the same physical processor core is explicitly documented for some platforms, such as the Amazon Elastic Cloud [2]. Other platforms, such as the Google Compute Engine [8], do not document the mapping.

Given the recently reported security issues related to microarchitectural data sampling, it is likely that hardware threads of the same physical processor core will be mapped to the same virtual machine on most cloud platforms. A configuration with virtual cores mapped to different physical cores can be obtained for example by renting larger virtual machine instances and disabling some cores, but the disabled cores are included in virtual machine cost and this solution therefore does not appear economical.

Regardless of the actual mapping (and possible interference through additional shared resources such as the memory subsystem), the workload symmetry argument again suggests any systematic difference in performance is unlikely in practice.

D. Computing Relative Performance

The duet measurements are naturally paired, we can therefore compute ratios of the paired task execution times, producing samples that describe relative performance of the two evaluated workloads. We assume that any noise due to performance interference will manifest as constant speedup or slowdown factor in each pair of samples, the ratio of the samples then filters out the correlated noise and reduces the total variance.

Our goal is comparing performance of two workloads for the purpose of performance testing, we therefore use duet measurements to derive a confidence interval for the ratio of task execution times. We use a Monte Carlo procedure based on standard bootstrap confidence interval computation [10], specifically:

- For an experiment with R runs of I iterations each, we denote $x_{r,i}$ and $y_{r,i}$ the task execution times measured in iteration $i \in 1 \dots I$ of run $r \in 1 \dots R$.
- In the duet measurement procedure, for each r and i the values of $x_{r,i}$ and $y_{r,i}$ are paired, we can therefore compute speedup from x to y as $s_{r,i} = x_{r,i}/y_{r,i}$.
- We aggregate speedup across iterations in a run by computing the geometric mean, $\forall r \in 1 \dots R : gms_r = \sqrt[I]{s_{r,1} \cdot s_{r,2} \dots s_{r,I}}$.
- We further aggregate speedup across runs in an experiment by computing another geometric mean, $gms = \sqrt[R]{gms_1 \cdot gms_2 \dots gms_R}$. The gms value represents our estimate for the ratio of task execution times.
- We use non parametric bootstrap to estimate the percentile confidence interval for gms, drawing with replacement from $gms_\bullet$ and computing gms^* as Monte Carlo estimates for gms.

When the confidence interval for the ratio of task execution times straddles 1.0, we consider the observed performance of the two workloads equal, otherwise we report a performance difference. We note that this procedure is similar to the practice described in [3], where a confidence interval for the difference in means is constructed using bootstrap. Following [3] further, we can also define a procedure that uses A/A measurements to learn the distribution of gms in a situation with no performance difference, and applies this knowledge to decide on performance changes in A/B measurements with small number of samples.

IV. Experimental Evaluation

To evaluate how our duet measurements impact the performance evaluation accuracy, we perform a series of A/A measurements, where the evaluated workloads are equal and any reported difference therefore directly reflects the accuracy. To save space, we compare the measurement procedures across multiple workloads and instances by comparing 99% confidence intervals for the ratios of means. The confidence intervals for the duet measurements are computed using the procedure in Section III, the confidence intervals for the sequential measurements are common bootstrap confidence intervals such as in [3].

A direct comparison of confidence intervals is hindered by the fact that intervals for duet measurements concern ratios of means (centered around 1.0 for A/A measurements), but intervals for sequential measurements typically concern differences of means (centered around 0.0 for A/A measurements). We therefore convert both types of confidence intervals to a value expressing their width relative to mean performance – for a ratio of means interval (r_{lo}, r_{hi}) we report $r_{hi} - r_{lo}$, and for a difference of means interval (d_{lo}, d_{hi}) we report $(d_{hi} - d_{lo})/\mu$, where μ is the sample mean computed from all samples (in A/A measurements all samples concern the same workload and can therefore be averaged).

A. Detailed Configuration

We run the cloud experiments on the Amazon Elastic Cloud platform in three different zones (us-east-1, us-east-2, us-west-2) and on three different instance types that were the smallest general purpose computing instances with two virtual cores and sufficient memory – t3.medium (two virtual cores based on Intel Xeon Platinum 8000, reported 2.5 GHz 20% burstable, 4 GB RAM), m5.large (two virtual cores based on Intel Xeon Platinum 8000, reported 3.1 GHz, 8 GB RAM), and m5a.large (two virtual cores based on AMD EPYC 7000, reported 2.7 GHz, 8 GB RAM). The instances were rented in spot mode (0.012 USD/h for t3, 0.020 USD/h to 0.034 USD/h for m5), running Amazon Linux 2.0.20190115.

For bare metal measurements that are to represent the most stable baseline, we have used multiple Intel Xeon E3-1230 v6 machines (four cores, 3.5 GHz, 32 GB RAM) with disabled hardware threads and power management features, running Fedora Linux 27 with kernel 4.15.6.

The workloads use benchmarks from ScalaBench 0.1.0 [20], with the harness adopted to report accurate timing and support duet execution, running with selected versions of the open source Graal compiler and the HotSpot JVM. We use 28 workloads that were identified as workloads with potential

This is the authors' version of the paper: *L. Bulej, V. Horký, P. Tůma: Initial Experiments with Duet Benchmarking: Performance Testing Interference in the Cloud, in Proc. 27th IEEE Intl. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 249-255, 2019.*
The final published version can be found at http://doi.org/10.1109/MASCOTS.2019.00035

performance regressions during Graal development, the workloads are listed in Table I.

All experiments bind each benchmark to a single randomly chosen processor core. To minimize startup artifacts, the JVM was run with fixed heap size (1.5 GB for Amazon t3 instances, 3.5 GB for m5 instances and bare metal) and disabled garbage collector ergonomics. Garbage collection was forced between iterations. Each workload was run at least 30 times (minor differences exist due to failures and restarts), with 20 iterations inside each run. The first 15 iterations were discarded to avoid measurements taken before the compilation of the hottest methods, however, it was not our ambition to guarantee steady state measurements – the diversity of the configurations means we would have to rely on runtime steady state detection, which would introduce additional variability between runs. In computations, we always consider mean performance from 30 runs with 5 warm iterations each. We employ outlier filtering with winsorization, replacing at most one observation in a run with its nearest neighbor when that observation is further than 20% away from the min-max range of the remaining observations. Our bootstrap computations use 10000 replicates.

B. A/A Testing

Our A/A measurement results are in Table I. The columns list triplets of relative confidence interval widths – the first number is the width computed as described in Section III, the second number is computed in the same way except for randomly assigning runs into pairs, and the third number is the width from sequential measurements computed as described in [3]. The best width of each triplet is shown in boldface. For reference, the accuracy achieved with bare metal measurements is listed alongside cloud measurements. For the cloud measurements, the table shows that in 79% of the cases, our duet measurements yield better accuracy than sequential measurements, the opposite is true for 21% of the cases, and the computation with random workload pairs never works best.

By comparing the first and the second number in each column, we evaluate how the duet measurements eliminate the correlated performance interference. By randomizing which runs form pairs, we make it it unlikely that both workloads experience the same interference and thus eliminate the advantage of the method while preserving other aspects of the experiment. For further illustration, we plot the ratio of the second to the first number for the cloud measurements in Figure 1. The accuracy improvement ranges mostly between 1 and 5. Any accuracy decrease is necessarily a result of an accidental correlation and therefore rare – it can also be taken to indicate the degree of ucertainty introduced by the limited size of our experimental sample.

By comparing the first and the third number in each column, we evaluate the overall difference in accuracy between the duet measurements and the sequential measurements. The difference in accuracy includes not only the positive contribution of the duet measurements to filtering interference, but also the potentially negative impact of running the workloads in pairs. We plot the ratio of the third to the first number for the cloud

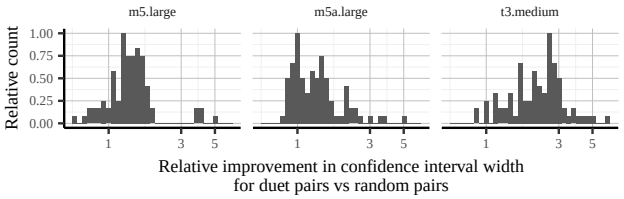

Figure 1. Distribution of accuracy improvement relative to random pairs (ratio of (1) to (2) in triplets from Table I). Higher is better, 1.0 for no improvement. Note logarithmic x axis.

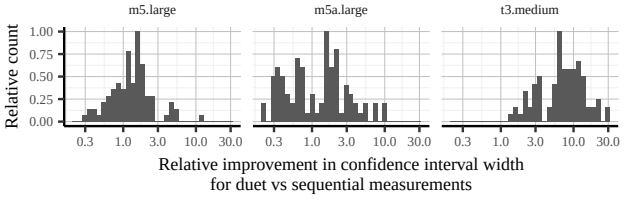

Figure 2. Distribution of accuracy improvement relative to sequential measurements (ratio of (1) to (3) in triplets from Table I). Higher is better, 1.0 for no improvement. Note logarithmic x axis.

measurements in Figure 2. The accuracy improvement is most pronounced for the t3.medium instances, with a geometric average of 683%. The t3.medium instances use burstable processor scheduling and therefore introduce significant variability into sequential measurements. The m5.large and m5a.large instances exhibit an average improvement of 130% and 114% respectively.

C. Discussion

Our experiments show that the duet measurements typically yield better accuracy than sequential measurements. As a notable exception, most measurements of the actors benchmark have better accuracy with sequential measurements – of the 18 cloud configurations in Table I where the sequential measurements are more accurate than the duet measurements, 13 execute the actors benchmark. The actors benchmark differs from the other benchmarks in that it internally uses a very high number of threads that dispatch relatively small tasks, it is therefore much more sensitive to scheduling anomalies, which exhibit themselves as outliers in the measurements. Interestingly, for some instance types such outliers exist in the duet measurements, but not in the sequential measurements, with obvious impact on accuracy. Because the same outliers and the same loss of accuracy are also apparent in the bare metal measurements, we do not consider this problem to be specific to the duet measurements.

The improvement in measurement accuracy of duet measurements over sequential measurements, shown in Figure 2, averages at 216%. One way to interpret this improvement is by looking at the measurements costs associated with reaching specific accuracy – the mean confidence intervals tend to

This is the authors' version of the paper: *L. Bulej, V. Horký, P. Tůma: Initial Experiments with Duet Benchmarking: Performance Testing Interference in the Cloud, in Proc. 27th IEEE Intl. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 249-255, 2019.*
The final published version can be found at `http://doi.org/10.1109/MASCOTS.2019.00035`

Table I
RELATIVE 99% CONFIDENCE INTERVAL WIDTHS IN % FOR A/A MEASUREMENTS.

Benchmark	Commit	Timestamp	bare metal	m5.large	m5a.large	t3.medium
actors	35cc2e8d	2016-12-13 13:17	**9.5** : 9.8 : 34	4.9 : 4.5 : **3.7**	15 : 15 : **9.3**	**3** : 7.1 : 19
actors	26c07924	2016-12-13 16:27	7.4 : **7** : 22	4.1 : 4.9 : **3.8**	17 : 17 : **9**	**2** : 6.8 : 14
actors	e1a85465	2017-02-10 09:02	7.5 : 7.5 : **2.7**	**3** : 3.1 : 3.2	**10** : 13 : 11	**2.4** : 5.9 : 14
actors	5b8cebee	2017-02-10 16:31	**9.3** : 9.6 : 25	4.8 : 5.5 : **4.5**	**11** : 14 : 11	**4** : 9.8 : 18
actors	11ec1deb	2017-04-21 15:58	**6.4** : 6.5 : 22	**2.1** : 2.9 : 2.5	12 : 15 : **7.7**	**3.8** : 6.2 : 15
actors	01a039cb	2017-04-24 19:34	5.9 : 7.6 : **4**	3.2 : 2.9 : **2.6**	16 : 17 : **8.1**	**3.1** : 8.8 : 21
actors	aa0c8c38	2017-05-18 10:16	7.9 : 7.7 : **6.2**	**2** : 2.9 : 13	12 : 12 : **5.2**	**2.7** : 5.6 : 25
actors	14133843	2017-05-19 11:59	10 : 10 : **4**	**2.8** : 3.3 : 3	13 : 13 : **5.5**	**2.3** : 6.4 : 22
actors	8d598327	2017-05-22 11:11	5.2 : 4.9 : **4**	**3.2** : 4 : 5.5	11 : 13 : **6.4**	**1.9** : 5.2 : 19
actors	645c8d28	2017-05-22 15:06	10 : 10 : **6.4**	4.3 : 4.3 : **2.9**	13 : 14 : **10**	**3** : 8.4 : 15
apparat	2a72cadd	2016-09-28 13:32	9.2 : 10 : **8.5**	**4.9** : 7.8 : 13	**5.1** : 12 : 17	**4.4** : 9.3 : 17
apparat	e1fdea82	2016-09-29 08:15	6.1 : 6.4 : **4.3**	8.5 : 15 : **7.2**	16 : 24 : **13**	**3.7** : 12 : 21
factorie	8479c086	2016-09-16 19:21	5.5 : 5.6 : **5.4**	**3.8** : 5.4 : 6.1	**3.8** : 9.7 : 9.8	**2.5** : 5.9 : 15
factorie	624c7823	2016-09-20 07:43	6.3 : 6.6 : **5.6**	**4.3** : 6.2 : 6.7	**4.7** : 10 : 9.4	**3.2** : 7.7 : 22
factorie	12841706	2018-03-05 19:34	3.4 : 3.6 : **3.1**	**2.2** : 3.8 : 3.4	**2.4** : 11 : 12	**1.5** : 4.5 : 19
factorie	d1098293	2018-03-05 21:03	3.6 : 3.8 : **2.9**	**2.3** : 3.5 : 3.2	**2.9** : 12 : 12	**1.3** : 5.4 : 17
factorie	6b9b1e38	2018-03-19 21:56	5.9 : 5.4 : **5**	**3.3** : 5.2 : 4.9	**4.4** : 7.7 : 8.4	**1.9** : 6 : 18
factorie	a188baff	2018-03-19 22:55	5.5 : 5.7 : **5.3**	**3.1** : 5.2 : 4.6	**4.3** : 7.9 : 9	**2** : 6.5 : 18
kiama	8f0b0417	2018-10-24 04:35	9 : 7.5 : **4.6**	7 : 8.8 : **6.7**	**8.9** : 12 : 11	**6.8** : 9.3 : 17
kiama	e7ed8a09	2018-10-24 11:40	11 : 8.8 : **4.3**	8.3 : 9.2 : **6.4**	12 : 13 : **11**	**8.2** : 9.6 : 14
lusearch	f04190bf	2017-11-22 18:09	2.1 : **1.9** : 2.3	**1.5** : 3.3 : 4	**1.5** : 9 : 13	**1.3** : 4.1 : 16
lusearch	841ffaef	2017-11-23 12:33	2.8 : **2.7** : 3.3	**1.4** : 3.6 : 4.4	**1.4** : 9.3 : 12	**1.3** : 4.6 : 13
pmd	c9525825	2018-10-31 21:24	4.9 : **4.6** : 4.7	**3.4** : 4.7 : 6.3	**2.7** : 8 : 9.5	**2.8** : 6.1 : 28
pmd	72255e2e	2018-11-01 12:59	8.8 : 9.2 : **7.3**	**6.2** : 7.8 : 7.3	**7.1** : 11 : 9.3	**5.3** : 9.2 : 31
scalaxb	dd0bae32	2019-01-17 21:40	5.5 : 5.6 : **4.6**	**1.6** : 3.5 : 3.1	**2.2** : 9.6 : 10	**1.3** : 3.3 : 17
scalaxb	034380de	2019-01-18 20:50	4.1 : **3.8** : 4.3	**1.5** : 3.5 : 3.6	**3.4** : 9.9 : 11	**1.4** : 2.6 : 18
tmt	4f77905b	2018-12-13 12:42	0.74 : **0.71** : 0.82	**0.52** : 2 : 2.7	**1.1** : 4.9 : 6.9	**0.79** : 3.5 : 19
tmt	a066d033	2018-12-13 14:32	0.92 : **0.81** : 0.86	**0.48** : 1.9 : 2.7	**0.84** : 4.6 : 7.8	**0.71** : 3.6 : 14

The benchmark, commit and timestamp columns identify the workload used. The remaining columns identify the platform used for the measurement (bare metal or one of three cloud instance types). In those columns, each triplet gives relative confidence interval widths obtained (1) when using the duet method, (2) when applying the duet method computation on randomly shuffled runs, and (3) when using standard sequential measurements. Lower is better, the best interval is bold.

shrink with the square root of the number of samples,[1] an average improvement of around 216% can therefore roughly correspond to an average four fold reduction in the volume of measurements collected.

Looking at the threats to external validity, we should start with stating that the duet measurement procedure hinges on the assumption of performance interference impacting the pair workloads equally. We believe this is more likely to happen with processor bound workloads, where the scheduling disciplines tend to emphasize fairness, rather than with I/O bound workloads, where queueing disciplines may prefer efficient execution over fair resource distribution. Our experimental evaluation used mostly processor bound workloads and our conclusions may not extend to I/O bound workloads.

Existing work points out that cloud performance characteristics can vary significantly across provides and platforms, our conclusions are therefore potentially restricted to the m5.large, m5a.large and t3.medium instances of the Amazon Elastic Cloud platform. In particular, we have seen that when the virtual cores provided by the platform are served by two hardware threads of the same host processor core, the synchronized performance interference that the duet measurements target is more likely to occur than in other configurations.

The duet measurement procedure requires concurrent execution of the pair workloads. With certain workloads, this may be difficult to achieve. We can, for example, imagine a workload that alternates between computing and accessing an exclusive resource. Two such workloads might form a convoy on the exclusive resource and therefore never execute their computing phases concurrently, weakening the duet measurement assumptions. In fact, even the processor bound workloads from our evaluation did not always execute concurrently – we have synchronized at the start of each iteration, for pair workloads with different iteration times this means that the workload with longer iterations executes alone for some time.

Finally, our confidence interval computation assumes that performance interference has a multiplicative character, in other words, we expect it to slow down or speed up both workloads by the same multiplicative factor. This seems to be a reasonable assumption for similar processor bound workloads, but is not something that is guaranteed in general.

Threats to internal validity are related in particular to our choice of the cloud platform, the instance types, and the workloads. Given their mostly black box character, we cannot rule out that some of the effects we observe are due to internal

[1] Asymptotically, this dependency holds due to the Central Limit Theorem, however, we are referring more to the empirical observations at small sample counts, where our experience suggests roughly the same behavior.

This is the authors' version of the paper: *L. Bulej, V. Horký, P. Tůma: Initial Experiments with Duet Benchmarking: Performance Testing Interference in the Cloud, in Proc. 27th IEEE Intl. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 249-255, 2019.*
The final published version can be found at http://doi.org/10.1109/MASCOTS.2019.00035

mechanisms we do not analyze. If suspected, this eventuality can be remedied by broadening the experiment scope.

V. Related Work

Directly relevant to our work is the paper by Laaber et al. [13], which investigates the accuracy achievable in the cloud with standard performance testing methods, that is, when executing the evaluated workloads one after another with randomization as recommended by [1]. Laaber et al. demonstrate that when using the standard confidence interval overlap test with 95% confidence intervals for the mean, A/A testing needs fairly high experiment repetition counts (20 instances, 5 runs per instance) to reduce the false alarm rate below 5%. The authors conclude that for most of their workloads, "small slowdowns (less than 5%) cannot reliably be detected in the cloud, at least not with the maximum number of instances (they) tested (20)" [13].

As much as the results can be compared across different workloads and measurement methodologies, the findings of Laaber et al. are in line with our observations. For A/A tests performed on sequential measurements, our bootstrap based 99% confidence interval construction yields median confidence interval widths of 7%, with the most accurate width as little as 1% and the least accurate width as much as 43%. This may be considered compatible with the best accuracy reported in [13], which aims for confidence level of 5%. Finally, our duet measurements further improve accuracy by removing correlated interference.

The work of Abedi and Brecht [1] shows how the ordering of trials can impact the experiment conclusions. Utilizing A/A testing, the authors show that possible regularity in performance interference can be incorrectly interpreted as actual difference in performance between alternatives. Randomized ordering of trials is proposed as a remedy. Our duet measurements address the same problem from a different angle – where the randomized ordering of trials makes sure performance interference impacts both evaluated workloads equally in a statistical sense, across multiple trials, duet measurements make sure both evaluated workloads are impacted equally in each individual trial. Given the black box nature of public cloud, we naturally cannot rule out performance interference that would systematically impact different processors of the same virtual machine in different manner. To address this eventuality, we randomize the assignment of workloads to processors, which is our analogy to the randomized ordering of trials.

In broader sense, our work is connected to research on cloud performance characteristics. A study by Leitner and Cito from 2016 [14] collects previously published observations on cloud performance and tests these observations with experiments. Especially relevant to our work are their conclusions on the performance stability of individual instances – this is shown to depend on the workload, with I/O bound workload performance being sensitive to noisy neighbors, and processor bound workload performance depending mostly on actual allocated hardware. Short term performance stability of individual instances is shown to be poor for some configurations with I/O bound workloads, and good for most configurations with processor bound workloads, except for the burstable instances, where the performance stability is always poor.

Among studies that show significant performance variability in the cloud, many attribute that variability mostly to hardware heterogeneity. Cerotti et al. [4] investigate the effects of hardware heterogeneity on instance performance, showing that instances of the same type can be backed by different processor types. In some of their experiments, the authors use the DaCapo benchmarks, and report that the difference between the slowest and the fastest processor type can impact the benchmark performance by 20% to 30%. Farley et al. [6] also examine the effects of hardware heterogeneity. For Amazon public cloud, different processor types are shown to differ in performance by as much as 280%. Differences of around 15% are observed among different instances with the same processor types, similar differences are observed for the same instance across time. Ou et al. [17] report similar findings. For Amazon public cloud and performance differences between instances of the same type, processor performance variability ranges between 10% and 20% and memory performance variability reaches as much as 270%. Other studies that concern various aspects of cloud performance variability include [19], [12], [15], [5], [18], [21]. Often, the purpose of the studies is to work towards efficient strategies of cloud resource allocation.

Overall, performance variability in public cloud is an accepted fact, but the actual numbers observed in individual studies can rarely be compared directly due to differences in experimental settings. In our experiments, we have observed very little processor heterogeneity, and are mostly concerned with variability in time. If this were not the case, strategies to reduce processor heterogeneity in allocated instances can be utilized during testing.

VI. Conclusion

Our work presented a novel performance measurement procedure that improves (sometimes significantly) the achievable measurement accuracy for CI/CD related performance testing activities in the cloud. On a selection of mostly processor bound workloads running Java benchmarks with a modern just-in-time compiler, we achieve on average 216% more narrow 99% confidence intervals for the mean performance difference. This improvement can help with more effective use of cloud infrastructure for performance testing purposes.

Additionally, our results include computations that characterize the baseline measurement accuracy that can be achieved on a particular virtual machine instance. This can be used to determine, for a desired level of accuracy, whether performance testing on a particular virtual machine instance can be performed in a cost-efficient manner.

This work was partially supported by the ECSEL Joint Undertaking (JU) under grant agreement No 783162.

This is the authors' version of the paper: *L. Bulej, V. Horký, P. Tůma: Initial Experiments with Duet Benchmarking: Performance Testing Interference in the Cloud, in Proc. 27th IEEE Intl. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 249-255, 2019.*
The final published version can be found at `http://doi.org/10.1109/MASCOTS.2019.00035`

References

[1] Abedi, A., Brecht, T.: Conducting Repeatable Experiments in Highly Variable Cloud Computing Environments. In: Proc. ICPE. pp. 287–292. ACM, New York, NY, USA (2017)

[2] Amazon, Inc.: Amazon Elastic Cloud Platform User Guide. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide (2019)

[3] Bulej, L., Bureš, T., Horký, V., Kotrč, J., Marek, L., Trojánek, T., Tůma, P.: Unit Testing Performance with Stochastic Performance Logic. Automated Software Engineering pp. 1–49 (2016)

[4] Cerotti, D., Gribaudo, M., Piazzolla, P., Serazzi, G.: Flexible CPU Provisioning in Clouds: A New Source of Performance Unpredictability. In: Proc. QEST. pp. 230–237 (2012)

[5] Ericson, J., Mohammadian, M., Santana, F.: Analysis of Performance Variability in Public Cloud Computing. In: Proc. IRI. pp. 308–314 (2017)

[6] Farley, B., Juels, A., Varadarajan, V., Ristenpart, T., Bowers, K.D., Swift, M.M.: More for Your Money: Exploiting Performance Heterogeneity in Public Clouds. In: Proc. SoCC. pp. 20:1–20:14. ACM, New York, NY, USA (2012)

[7] GitLab Inc.: GitLab Runner. https://about.gitlab.com (2019)

[8] Google, Inc.: Google Compute Engine Documentation. https://cloud.google.com/compute/docs (2019)

[9] Heger, C., Happe, J., Farahbod, R.: Automated Root Cause Isolation of Performance Regressions During Software Development. In: Proc. ICPE. pp. 27–38. ACM, New York, NY, USA (2013)

[10] Hesterberg, T.: What Teachers Should Know about the Bootstrap: Resampling in the Undergraduate Statistics Curriculum. arXiv:1411.5279 [stat] (2014)

[11] Intel, Inc.: Intel IA-64 and IA-32 Architectures Software Developer Manual. Number 325462-069US. (2019)

[12] Iosup, A., Yigitbasi, N., Epema, D.: On the Performance Variability of Production Cloud Services. In: Proc. CCGRID. pp. 104–113 (2011)

[13] Laaber, C., Scheuner, J., Leitner, P.: Software Microbenchmarking in the Cloud. How Bad is it Really? Empirical Software Engineering (Apr 2019)

[14] Leitner, P., Cito, J.: Patterns in the Chaos—A Study of Performance Variation and Predictability in Public IaaS Clouds. ACM Trans. Internet Technol. **16**(3), 15:1–15:23 (2016)

[15] Lenk, A., Menzel, M., Lipsky, J., Tai, S., Offermann, P.: What Are You Paying For? Performance Benchmarking for Infrastructure-as-a-Service Offerings. In: Proc. CLOUD. pp. 484–491 (2011)

[16] Oliveira, A.B.D., Fischmeister, S., Diwan, A., Hauswirth, M., Sweeney, P.F.: Perphecy: Performance Regression Test Selection Made Simple but Effective. In: Proc. ICST. pp. 103–113 (2017)

[17] Ou, Z., Zhuang, H., Lukyanenko, A., Nurminen, J.K., Hui, P., Mazalov, V., Ylä-Jääski, A.: Is the Same Instance Type Created Equal? Exploiting Heterogeneity of Public Clouds. IEEE Trans. on Cloud Computing **1**(2), 201–214 (2013)

[18] Ristov, S., Mathá, R., Prodan, R.: Analysing the Performance Instability Correlation with Various Workflow and Cloud Parameters. In: Proc. PDP. pp. 446–453 (2017)

[19] Schad, J., Dittrich, J., Quiané-Ruiz, J.A.: Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. Proc. VLDB Endow. **3**(1-2), 460–471 (2010)

[20] Sewe, A., Mezini, M., Sarimbekov, A., Binder, W.: Da Capo Con Scala: Design and Analysis of a Scala Benchmark Suite for the Java Virtual Machine. In: Proc. OOPSLA. pp. 657–676. ACM, New York, NY, USA (2011)

[21] Shankar, S., Acken, J.M., Sehgal, N.K.: Measuring Performance Variability in the Clouds. IETE Technical Review **35**(6), 656–660 (2018)

[22] Travis CI, GmbH: Travis CI. https://travis-ci.com (2019)

This is the authors' version of the paper: *L. Bulej, V. Horký, P. Tůma: Initial Experiments with Duet Benchmarking: Performance Testing Interference in the Cloud, in Proc. 27th IEEE Intl. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 249-255, 2019.*
The final published version can be found at `http://doi.org/10.1109/MASCOTS.2019.00035`