Investigations of Micro-Benchmarks for Performance Profiling in Multi-Tenant Clouds

Hamidreza Moradi[1], Wei Wang[2], and Dakai Zhu[2]

[1] The University of Mississippi Medical Center
hmoradi@umc.edu
[2] The University of Texas at San Antonio
{wei.wang, dakai.zhu}@utsa.edu

Abstract. With the significant increase in the adoption of cloud computing, modeling and predicting the performance of the applications executing in clouds have gained interests from cloud researchers and users. However, shared resource contention caused by multi-tenancy makes the performance modeling and prediction a challenging task. In this paper, we designed and evaluated a comprehensive set of micro-benchmarks to probe and profile the impacts of contention of key resources. Various machine-learning models have been exploited to model and predict cloud application performance with the profiling information of these micro-benchmarks. The results of our extensive experiments showed that the micro-benchmarks can effectively probe the contention levels of the resources, enabling accurate performance prediction models for cloud applications with only 6.8% error on average. Moreover, to reduce profiling overhead, the results showed that the duration of 0.4 seconds execution for each micro-benchmark can achieve relatively accurate prediction even with only the CPU micro-benchmark.

Keywords: Machine Learning, Deep Learning, Cloud Computing, Micro-benchmarks, Performance Modeling

1 Introduction

The adoption of cloud computing by many organizations necessitates the performance prediction of cloud applications [2, 26]. Accurate performance predictions can help cloud users (e.g. organizations) to make insightful decisions in choosing the proper types of Virtual Machines (VM), design better auto-scaling/scheduling policies and choose the more-suitable cloud provider [6, 25, 27]. However, predicting the performance of cloud applications is a challenging task due to the interference (i.e., resource contention) introduced to the system by other cloud tenants. The difficulty is mainly due to the high variation of this interference, which fluctuates with the VMs (and their running applications) co-located in the same physical server. As shown in prior work, a VM and its application may experience different levels of performance degradation depending on the severity of the contention in various resources, including CPU, Memory bandwidth/latency, I/O, L1, L2 and L3 cache [11].

A cloud application's performance degradation varies considerably depending on the severity of the contention. To accurately model and predict the cloud application

performance under resource contention, reliable profiling of the severity of the contention in different resources is required. Yet, architecture-level profiling from the readings of hardware performance monitoring units (PMU), which was extensively used in prior work [11, 21], is not accessible by ordinary cloud users. Additionally, OS-level resource utilization, such as CPU utilization and memory usage, cannot provide reliable profiling of this contention.

To address this problem, we proposed to employ micro-benchmarks to profile the severity of the resource contention. Several prior studies have used micro-benchmarks to estimate the performance of certain resources in the cloud, such as the CPU speed and storage bandwidth [5, 28]. In [28], Leitner et al. used a set of 23 micro-benchmarks to model the performance of only two cloud running applications on different instances. Likewise, Baughman et al. used actual application executions with different input data sizes to build the performance model of an application [5]. Here, both studies only used the micro-benchmarks to profile the average performance of a particular hardware resource, instead of the contention severity of the resource at run-time. However, to enable more efficient resource management and job scheduling, it may be necessary to predict the performance of a cloud application at the run-time by considering the in-situ severity of resource contention.

Given that the above studies focused on the average performance of cloud resources, they were unsuitable to predict the run-time performance of a cloud applications. In our previous works [22, 23, 24], to address the existing limitation, we have designed frameworks to model and predict the in-situ performance of cloud applications. To collect contention information and corresponding application's performance, the frameworks run three specifically devised micro-benchmarks (CPU, memory, and disk I/O) followed by the actual execution of the application. Then collected contention information and corresponding in-place performance of the application is used for offline or online performance modeling and prediction. Note that studied frameworks have not analyzed the effects of different profiling durations and different micro-benchmarks as feature sets, which is essential to minimize the run-time overhead of profiling and to evaluate the corresponding prediction accuracy.

Therefore, in this paper, we exploited our recently studied *uPredict* framework to accurately model any correlation between the source and intensity of contention and application's performance [22]. Moreover, a thorough investigation has been conducted to evaluate the impacts of the source of contention and profiling duration on model's prediction accuracy. We designed and implemented a comprehensive set of micro-benchmarks based on the lmBench3 open-source benchmark suite [20] to probe and profile the resource contention experienced by a given VM in multi-tenant cloud environment. With the profiled information from such micro-benchmarks, various performance prediction models for cloud applications could be built, as illustrated in our previous studies [22, 23, 24]. For different sources of contention in various resources in a target VM, we developed seven micro-benchmarks to probe CPU, Memory Latency/Bandwidth, L1, L2, and L3 cache. Following the same principle in the *uPredict* framework, the developed micro-benchmarks were evaluated on their effects on the prediction accuracy of different prediction models. Specifically, we considered both a private and the Chameleon clouds [8] in our experiments. Moreover, nine benchmark ap-

plications from PARSEC [3], NAS Parallel Benchmarks (NPB) [4] and CloudSuite [13] were employed. For prediction models, we considered various machine learning techniques, including Support Vector Regression (SVR), Lasso Regression, and Neural Network (NN) [14, 15, 31].

Our experiment results showed that the profiling information from the micro-benchmarks can lead to quite accurate performance prediction models. In particular, with feed-forward neural networks, the prediction error of the considered models was only 6.8% on average when all micro-benchmarks with each running for the longest profiling duration (i.e., 3 seconds) were considered. With the objective of reducing profiling overhead, we also evaluated the effects of profile duration of micro-benchmarks as well as different subsets of micro-benchmarks, where the results showed that the duration of 0.4 seconds for each micro-benchmark is good enough to achieve accurate performance prediction. In addition, when the micro-benchmark that probes CPU resources is utilized as the only feature, the prediction errors increase only slightly to 8.5% on average. The main contributions of this work include:

1. The design and implementation of a comprehensive set of micro-benchmarks to probe the severity of the resource contention in the clouds for various types of resources, including the CPU, Memory bandwidth/latency, I/O, L1, L2 and L3 caches.
2. By considering various machine learning based performance prediction models, extensive experiments have been conducted for representative benchmark applications running on both a private and Chameleon clouds to evaluate the effects of different aspects of micro-benchmarks (including profiling duration and subsets of features) on the prediction accuracy.
3. With the objective of reducing profiling overhead, our evaluation results indicated that the profiling duration of 0.4 seconds per micro-benchmark could lead to quite accurate prediction results, which is inline with the existing results regarding to the minimal profiling duration. Moreover, the best subset of features from micro-benchmarks was also identified.

The rest of this paper is organised as follows. Section 2 presents the closely related works on performance modeling and prediction of cloud applications. Detailed implementations of micro-benchmarks, feature selection, and profiling duration are discussed in Section 3. Section 4 discusses the experimental setup and evaluation results. Finally, Section 5 concludes the paper.

2 Related Works

To avoid long tail latencies and improve the quality of resource provisioning on the cloud many researches have investigated modeling and prediction of the incoming requests to cloud providers for better collocation [9, 19, 36]. In [36], Yu et al. use Neural Networks (NN) and existing pool of tasks to cluster workloads and learn the common features among them. Later, for any submitted job, its initial workload patterns and parameters can be used in conjunction with an existing NN model to find an appropriated cluster that the job belong to. This predicted information about the job characteristics

will be used to make scheduling decisions that prevent later overhead of migrating or scaling up the instance. However, a pool of incoming requests with known workloads' characteristics are required for these category of researches which may not always accessible, since users not always disclose the detail of their running workload.

Another category of researches have focused on the application specific performance modeling (ex. Hadoop and Spark workloads) of cloud running applications [7, 33, 34]. These researches leverage the existing additional information about the job execution in the framework. In [34], authors used Directed Acyclic Graph (DAG) information of a spark task for performance modeling and prediction, or in [33] sequential stages in spark job execution have been used as a progress indicator for modeling and predictions. Although these approaches were successful, additional information about task execution does not always exist for all cloud running applications which make it impossible to adopt.

Several studies have considered micro-benchmarks for performance modeling and prediction of cloud applications [5, 28, 35]. Although these are the closest researches to our approach, they are all focused on predicting the average performance of the applications running on different types of instances. Therefore, these studies were not designed for, and thus could not be used to, determine the run-time performance of a cloud application. Such run-time performance information can usually benefit cloud task/request scheduling algorithms. Additionally, these studies lacked in-depth analysis on how to reduce the profiling overhead, including profiling duration and the subset of feature (i.e., required micro-benchmarks).

3 Performance Profiling with Micro-Benchmarks

Micro-benchmarking for performance evaluation in an isolated environment have a long history and is considered a foundational tool [18, 30, 32]. However, using micro-benchmarks in the cloud environments has its own complexities to adopt due to hardware sharing among multiple cloud tenants. Hardware sharing will result in multiple VMs competing for the same resources which will slowdown the executions, and any changes in the collocated VMs or their running applications will greatly impact the performance of other cloud tenants. Moreover, each cloud running application has its own sensitivity to the intensity of the interference introduced to the system by other cloud tenant. Such sensitivity of applications can be modeled through machine learning algorithms and be used for performance predictions of future executions. Yet, the effect of source and intensity of interference, profiling duration of the micro-benchmarks, and machine learning algorithm on prediction accuracy needs to be thoroughly evaluated.

Given that the key resources that affect single-VM cloud applications are CPU, memory and disks (i.e., I/O operations), we designed and implemented a comprehensive set of micro-benchmarks based on the lmBench3 open-source benchmark suite [20]. We would like to point out that, although there are many micro-benchmarks have been studied [5, 28], none of them exactly fits our needs to perform in-situ profiling of contention of a given resource. In what follows we will first explain the detailed micro-benchmarks design, followed with discussions on profiling duration, feature sets, and machine learning models.

3.1 Micro-benchmarks

CPUs: To evaluate the contention level of available CPUs, a multi-threaded micro-benchmark is implemented. Number of thread are equal to the number available CPUs, and each thread in the micro-benchmark will increment an in-register counter for a specific predefined duration. Here, an in-register counter will ensure that the micro-benchmark execution is heavily depend on the number of CPU cycles allocated to the VM. At the end of execution the sum of counter values for all the threads will be considered as an overall progress of the micro-benchmark (c_{cpu}). Any increase in the counter value indicated more CPU cycles dedicate to the VM, and decrease in the counter can be translated to increase in the contention level and decrease in the dedicated CPU cycles.

Cache: Similarly, three micro-benchmarks are implemented with each micro-benchmark evaluating the L1, L2 and L3 cache contention level. These three multi-threaded micro-benchmarks each will access an array of size 256KB, 2MB and 20MB respectively, with the stride size of 128. The number of accesses for each micro-benchmark will be considered as a progress indicator (c_{l1}, c_{l2}, and c_{l3}). Fast progress can be inspected with the increase in number of accesses in a fix period of time, and vice versa. The sizes of the arrays are devised with consideration of cache sizes of each cache level per thread, so that by the first few accesses all the elements brought to the corresponding cache level. The retention of data in the cache could result in high access rates. However, introduced contention may push some of the data out of the cache, resulting in a decreased access rate and necessity of re-fetching the data from the memory or lower cache levels.

Memory Bandwidth: To measure the available memory bandwidth a 2GB array of data will be accessed with the stride size of 128 and threads equal to the number of available CPUs. Due to the large size of the array each access will cause an off-core memory access. Similar to the previous micro-benchmarks, number of accesses will be considered as an indicator of experienced contention in memory bandwidth(c_{mem}). As explained earlier, increase in the number of accesses will be an indicator of increase in the memory bandwidth and decrease in the contention level.

Memory Latency: Micro-benchmark is also devised to profile the latency of system memory to evaluate its effect on the applications. With this idea in mind, the micro-benchmark accesses an array of 2GB with a stride size of 128 with only one thread to do a pointer-chasing. In pointer-chasing, each accessed memory location will contain the address of next memory location which should be accessed. One thread have been considered since theoretically is enough to measure the memory access latency and prevent inter-thread contention. The number of memory accesses ($c_{latency}$) will be considered as an indicator of memory latency. Here, larger latency will result in lower number of accesses, and vice versa.

Disk IO: To evaluate the IO contention a micro-benchmark is implemented that will access 256MB of data with 4 threads and page size of 4KB. Four thread have been considered since its enough to fully utilize the IO bandwidth. The number of IO accesses will be considered as an indicator of experienced memory contention (c_{disk}), which translates to higher access rate for lower contention level. During these operation the OS data caching will be disable. The 256 MB of data have been considered regrading

the maximum IO throughput in the specified short period of the time, for faster disks with larger cache size the data size can be increase to prevent data access from the disk cache.

3.2 Profiling Duration

When micro-benchmarks are used to probe and profile the resource contention in a target VM, how closely such profiling information can reflect the actual resource contention that cloud applications may experience will directly depend on the duration of profiling. Idealistically, longer profiling duration can capture variations of short-time changes in contention of cloud environment and lead to more accurate performance prediction, which however, may increase the profiling overhead. On the other hand, shorter profiling duration can reduce overhead but could lead to loss of the accuracy. Therefore, there is a trade-off between the prediction accuracy of the considered models and the profiling duration.

To support fine-grained evaluation regarding to the profiling duration, we have devised the micro-benchmarks to support sub-second executions. Once the profiling duration is determined, the Micro-benchmarks will be executed sequentially right before a target application to obtain the in-situ resource contention information, where the profiling output of each micro-benchmark will be a feature used by the considered performance prediction models. The effects of profiling duration of the micro-benchmarks on the prediction accuracy of the model will be presented in Sec. 3.2.

3.3 Feature Sets

Each micro-benchmark can provide us the contention of a particular resource experienced by the target application. However, not all applications are sensitive to contention of all resources. Hence, the effects of micro-benchmarks as features on modeling and prediction accuracy need to be thoroughly evaluated. For instance, a subset of more features can be selected with the goal of achieving higher prediction accuracy, or a subset of fewer features can be selected to reduce the profiling overhead. The combinations of different features (i.e., micro-benchmarks) and their effects on prediction accuracy of the considered models have been extensively evaluated, and the results will be reported in Sec. 4.3.

3.4 Predictive Models and Performance Prediction

In this work, we considered three representative predictive models: 2D-polynomial with Lasso Regression, Support Vector Regression (SVR) and the feed-forward neural network (NN) models. The obtain the training data for these models, the developed micro-benchmarks and a target cloud application are executed sequentially and repeatedly, where the profiling output of k micro-benchmarks will be used as features $< p^1_{feature}, \cdots, p^k_{feature} >$ associated with the measured application performance (t_{app}). These information will be provided as the input to a given predictive model to find the out the model parameters for function f as presented in Equation (1).

$$t_{app} = f(< p^1_{feature}, \cdots, p^k_{feature} >) \quad (1)$$

Once the model parameters were obtained for a given model, the function f can then be used to predict the application performance at run-time by utilizing the in-situ profiled resource contention information with the micro-benchmarks. Note that, the number and type of inputs for predictive model depends on selected features when building models, which should be the same for all the models to achieve fair comparison. The prediction accuracy for the considered predictive models are presented in Sec. 4.3.

4 Evaluations and Discussions

We have performed extensive experiments to evaluate the effects of different micro-benchmarks on the prediction accuracy for the considered three predictive models. In this section, we will first present our experimental setups. Then, we show that when the profiling information of all micro-benchmarks (i.e., all features) with extended profiling duration were utilized, the predictive models can provide quite accurate performance prediction. Finally, the effects profiling duration and feature selection (i.e., when different micro-benchmarks were utilized) on the accuracy of predictive models are evaluated.

4.1 Experiments Setup

Representative Benchmark Applications: We considered a total of nine benchmark applications from PARSEC [3], NAS Parallel Benchmarks (NPB) [4] and CloudSuite [13]. It includes *canneal*, *streamcluster*, and *swaptions* with native inputs from PARSEC, *ep*, *sp* and *lu* with class c input size from NPB, and *In-Memory Analytic*, *Graph Analytic*, and *Web Search* from CloudSuite. For *Graph analytics* and *Web Search* benchmarks the only default data inputs, and for the *In-Memory Analytic* the largest data input have been used. These benchmark applications have been selected to represent a wide range of cloud applications. PARSEC benchmarks are selected to represent batch processing jobs, NPB benchmarks as HPC applications and the ones from CloudSuite are selected to represent business applications. For all the selected benchmarks, sixteen threads were specified as the execution parameter.

Clouds and VM Configurations: Two different types of clouds have been considered: a private cloud and the Chameleon cloud. For the private cloud, OpenStack Ocata was installed on a server with two Intel Xeon E5-2630 processors (for a total of 16 cores) and 128GB memory. Each VM will utilize 16 vCPUs and 16 GB memory to be able to run all the specified benchmarks. To introduce resource contention to the system, up to three background VMs with the same configuration will be lunched. For each randomly created interference configuration at run-time, 100 data points will be collected and then the interference configuration will change. This process will repeat for 10 iterations to collect total of 1000 data points. The interference VMs will be chosen to randomly execute CPU, Memory and I/O intensive applications from iBench [10] and FIO [17]. For Chameleon, which is a scientific public cloud computing infrastructure [8], the VM instance that has the closest configuration as our private cloud experiment is *m1.xxxlarge*,

which has 16 vCPU, 32GB of RAM and 160GB HDD drive. For all the experiments, Ubuntu Server 16.04 as the OS for the VMs has been used.

Data Collection: We considered seven different profiling duration (0.2, 0.4, 0.6, 0.8, 1, 2, and 3 seconds) for each designed micro-benchmark. For a given profiling duration, all micro-benchmarks will be executed for that amount of time prior to the execution of the target application. In this way, the profiling information from the micro-benchmarks can provide us the current contention level experienced by the application performance executed next. The nine benchmark applications have been executed individually to collect the required performance data. For each application, 1,000 data points have been collected, which runs over the course of 2 month on the private cloud and 45 days on Chameleon.

For each benchmark application, 60% of the collected data points are randomly selected to train the model, 20% used for validation and model's hyperparameter optimization, and finally, the remaining 20% have been used for testing.

Model Implementation: The considered predictive models have been implemented by utilizing the existing open-source libraries. Here, Support Vector Regression (SVR) and Lasso Regression from scikit-learn version 0.19.2 [29] have been used. The NN model was implemented using TensorFlow version 1.12 [1]. Due to sensitivity of NN to the hyper-parameters, hyper-parameter optimization library - HyperOpt version 0.1.1 [16] - has been employed, where 100 iterations were used to find the best NN structure [12] for modeling and prediction from the provided search space. The resulting model are denoted as *NN*.

4.2 Prediction Accuracy of Predictive Models

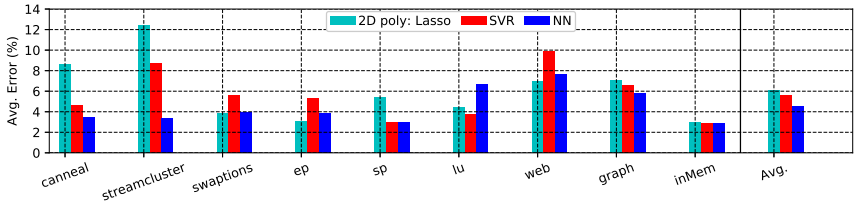


Fig. 1: Prediction errors of different models on the local cloud with all micro-benchmarks and 3-second profiling duration.

First, by considering all features from the profiling information for all micro-benchmarks with the longest duration (i.e., 3 seconds) for each micro-benchmark, Figures 1 and 2 give the prediction errors (i.e., reverse of accuracy) of the considered predictive models in the local and Chameleon clouds, respectively. Here, we can see that, the prediction errors of all three predictive models for the considered applications are relatively low, with the average error being under 6% for the local clouds and 10% for the Chameleon cloud.

The 2-degree polynomial Lasso model usually have lower accuracy due to their inability to represent more complex relationships between the profiled resource con-

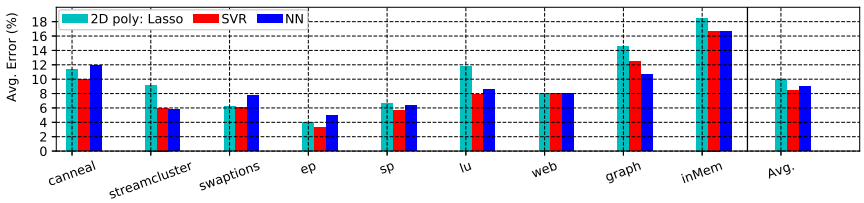


Fig. 2: Prediction errors of different models on Chameleon with all micro-benchmarks and 3-second profiling duration.

tention (features) and the execution time (labels) of the applications. On the other hand, SVR model can be used to model non-polynomial relationships between the features and labels, which result in smaller prediction errors as shown in the figures. For comparison, the NN model generally achieves slightly better prediction accuracy (i.e., smaller errors) for the applications running on the local cloud, which has rather high interference. The average error of the NN model in our private cloud is only 4.5%. On the Chameleon cloud, the prediction error for the NN model has the average of 9.1%, which is almost the same as SVR. Given that the expected interference on Chameleon cloud is rather light, it indicates that SVR can have relatively better performance for low-interference cloud.

Moreover, for the NN model, the maximum prediction error is only 7.6% (for Web Search) on the private cloud and 16.6% (for In-Memory Analytic) on the Chameleon cloud. These results indicate that the profiled information from our designed micro-benchmarks can indeed provide accurate estimation on the resource contention, and thus enable the predictive models achieve accurate performance prediction on both local and public clouds. In what follows, due to space limitation, we will present only the effects of feature selection and profiling duration on the prediction accuracy of the NN model.

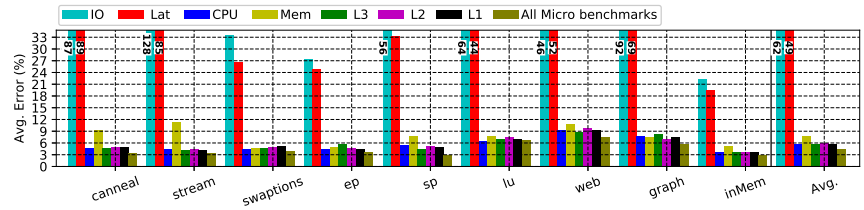


Fig. 3: Prediction errors of the NN model for different single feature on the private cloud.

4.3 Feature Selections with Micro-benchmarks

Although using all seven micro-benchmarks as the features can provide high prediction accuracy, the profiling overhead with all micro-benchmarks could be rather high (that is, $3 \times 7 = 21 seconds$ for each execution of an application). Therefore, it would be necessary to whether utilizing the profiling data from fewer micro-benchmarks (i.e., fewer features) could still allow the predictive models get similar prediction accuracy.

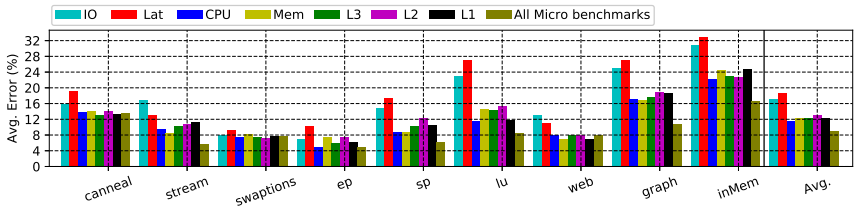
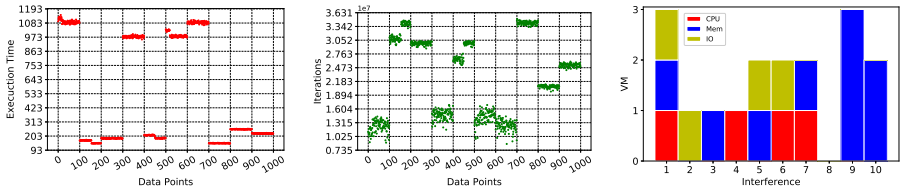


Fig. 4: Prediction errors of the NN model for different single feature on the Chameleon cloud.

Intuitively, it is possible for fewer micro-benchmarks to provide necessary resource contention information for accurate models, since not all cloud applications will be affected by the contention of every resource. For example, a CPU-intensive application is less likely to be affected by the disk contention. Moreover, although the micro-benchmark is designed to probe the contention of one resource, it may still be able to detect the contention at other related resources (as shown later in this section).



a. Exec. times of StreamCluster; b. Output of CPU micro-benchmark; c.Background VMs.

Fig. 5: The execution times of Streamcluster vs. profiled CPU contention on the private cloud.

To find out what will be the best subset with the smallest number of features, we have evaluated different variations of the feature sets with one to four features. Figure 3 shows the prediction error of the NN model when only one feature (from one micro-benchmark) is utilized for the applications on the local cloud. Here, the prediction errors of the NN model with all features being considered are also shown in the figure for comparison. From the results, we can see that, when only the CPU micro-benchmark's profiling is used as the feature, the average prediction error for the NN model is 5.6%, which was very close to the average error of 4.5% when all features are considered. However, not all the single-feature set can lead to the same prediction accuracy. In particular, we can see that, the NN model with the feature from the disk I/O or memory latency micro-benchmarks can result in extremely high prediction errors (more than 50% for most applications). These high prediction errors suggest that it is crucial to select the right subset of features from the micro-benchmarks when building the NN model, especially when reduced number of features are considered. Actually, the considered applications are not solely disk-intensive or memory-latency-sensitive. Therefore, only using these two micro-benchmarks could not provide accurate NN models.

To investigate the underlying reasons of why only using the CPU micro-benchmark can provide accuracy results, we further examined how the profiling results of the CPU micro-benchmark fluctuated with the background contending VMs in our private cloud. Fig. 5a gives the execution times of *Streamcluster* on the private cloud over the period of the experiments, along with the CPU micro-benchmark profiling results (Fig. 5b) and the contending VMs introduced in the background (Fig. 5c). As the results show, the CPU micro-benchmark's profiling results fluctuated along with the background VMs, even when the background VMs included only memory- and disk-contending VMs. These results showed the the CPU micro-benchmarks could also probe the contention at the memory and disk I/O resources, which allows it to provide relatively accurate models by itself. Further analysis showed that, although CPU micro-benchmark was only designed to probe CPU contention, its underlying VM also requires memory and disk to operate. Therefore, the VM that executed the CPU micro-benchmark was also affected by the memory and disk contention, where the profiling data from the CPU micro-benchmark includes contention information for the memory and disks. Note that, when all-features from all micro-benchmarks are utilized, the NN model delivers the best prediction accuracy (i.e., the lowest errors), which indicates that including the profiling information of the disk I/O and memory is still valuable compared to relying on the CPU micro-benchmark as an indirect indicator of the contention at the memory and disk. Similar results have been observed on the Chameleon cloud and were omitted due to lack of space.

Feature(s)	Micro-benchmarks	Error
1	CPU	5.6%
2	CPU - Cache	5.3%
3	CPU - Cache - IO	4.8%
4	CPU - Cache - IO - Mem	5.1%
5	All	4.5%

a. The local cloud.

Feature(s)	Micro-benchmarks	Error
1	CPU	11.5%
2	Mem - IO	9.6%
3	CPU - Mem - IO	9.0%
4	CPU - Mem - IO - Latency	8.6%
5	All	9.1%

b. The Chameleon cloud.

Table 1: The effects of different number of features on prediction errors of the NN model.

When the features from the profiling information of other micro-benchmarks were included with the CPU feature, TABLE 1 show the average prediction errors of the NN model on the local and Chameleon clouds, respectively. First, from TABLE 1a, when the number of features increases from 1 to 4 (with the selected features as shown in the table), the average prediction errors can be reduced from 5.6% to 5.1%. Similar results can be seen in TABLE 1b for the Chameleon cloud. The error reduction shows that, although CPU micro-benchmark could provide some indirect profiling information regarding to the contention of other resources, including the features from the direct profiling of these resources with their respective micro-benchmarks can be more accurate. Nonetheless, using the profiling data from more resources increases the

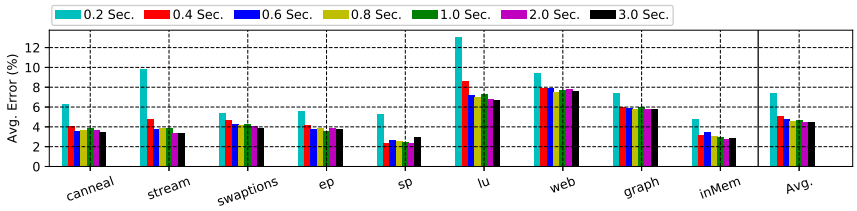


Fig. 6: Prediction errors of NN with different profiling duration on our private cloud.

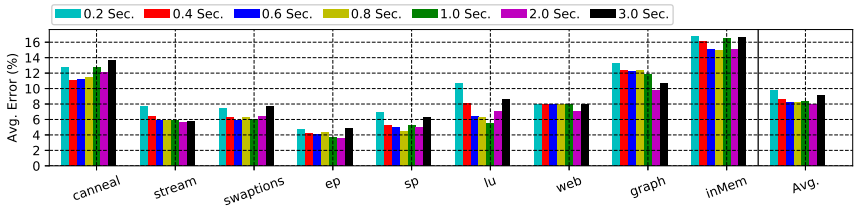


Fig. 7: Prediction errors of NN with different profiling duration on the Chameleon cloud.

profiling overhead, and cloud users should carefully balance this overhead with their desired prediction accuracy.

4.4 Profiling Duration

Another important factor in profiling overhead is the duration that each micro-benchmark is executed. In the previous experiments, each micro-benchmark was executed for 3 seconds. Although quite accurate performance prediction can be obtained for the considered predictive model with such long profiling duration for each micro-benchmark, it would be necessary to find out what is the smallest duration that can still provide accurate information regarding to resource contention. Here, we have evaluated the prediction accuracy of the NN model when the profiling duration of each micro-benchmark being 0.2, 0.4, 0.6, 0.8, 1, 2, and 3 seconds, and the results are shown in Figures 6 and 7 for the private and Chameleon clouds, respectively.

For the private cloud, although increasing the profiling duration could reduce the prediction errors, the additional benefits of using more than 0.4 seconds diminish quickly. Here, with 3-second profiling duration, the prediction error of the NN model for the considered application reduces less than 1% compared to the case of 0.4 seconds. However, further shorten the profiling duration could lead large increases in predication errors for some applications. On the Chameleon cloud, increasing the profiling duration did not always reduce the prediction errors, and the differences are rather small. Nonetheless, as long as the profiling duration was larger than 0.4 seconds, the average prediction errors has the difference of less than 1%. Therefore, for the NN model and the considered application, we can say that the best profiling duration will be 0.4 seconds.

5 Conclusions

We investigated the usage of micro-benchmarks in profiling based performance prediction for cloud applications in multi-tenant clouds. Several micro-benchmarks have been designed and developed for the key resources in VMs. Our evaluation results show that, the micro-benchmarks can accurately profile resource contention in the VM, which allows accurate performance prediction using different predictive models. The best accuracy can be achieved when all features are used. However the CPU micro-benchmark can provide acceptable results for performance modeling. Finally, the profiling duration of 0.4 seconds for the micro-benchmarks can obtain accurate information about the contention levels of different resources for performance modeling and prediction.

Bibliography

[1] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] A. O. Ayodele, J. Rao, and T. E. Boult. Performance measurement and interference profiling in multi-tenant clouds. In *IEEE Int'l Conf. on Cloud Computing (CLOUD)*, 2015.

[3] Christian B. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.

[4] D. H. Bailey. Nas parallel benchmarks. In *Encyclopedia of Parallel Comput.*, pages 1254–1259. Springer, 2011.

[5] M. Baughman, R. Chard, L. Ward, J. Pitt, K. Chard, and I. Foster. Profiling and predicting application performance on the cloud. In *IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*, 2018.

[6] R. Begam, H. Moradi, W. Wang, and D. Zhu. Flexible vm provisioning for time-sensitive applications with multiple execution options. In *Proc. of the IEEE Int'l Conf. on Cloud Comput. (CLOUD)*, 2018.

[7] A. Castiglione, M. Gribaudo, M. Iacono, and F. Palmieri. Modeling performances of concurrent big data applications. *Software: Practice and Experience*, 45(8):1127–1144, 2015.

[8] Chameleon Cloud. https://https://chameleoncloud.org.

[9] Jeffrey Dean and Luiz André Barroso. The tail at scale. *Communications of the ACM*, 56(2):74–80, 2013.

[10] C. Delimitrou and C. Kozyrakis. iBench: Quantifying interference for datacenter applications. In *IEEE Int'l Symp. on Workload Characterization (IISWC)*, 2013.

[11] C. Delimitrou and C. Kozyrakis. Quasar: Resource-efficient and QoS-aware Cluster Management. In *Proc. of Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2014.

[12] N. Ebadi, M. Jozani, K. R. Choo, and P. Rad. A memory network information retrieval model for identification of news misinformation. *IEEE Transactions on Big Data*, 2021.

[13] M. Ferdman et al. Clearing the clouds: A study of emerging scale-out workloads on modern hardware. *Proc. of Int'l Conf. on Architectural Support for Programming Languages and Operating Syst. (ASPLOS)*, 2012.

[14] S. R. Gunn et al. Support vector machines for classification and regression. *ISIS technical report*, 14(1):5–16, 1998.

[15] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural Networks for Perception*, pages 65–93. Elsevier, 1992.
[16] HyperOpt. https://github.com/hyperopt/hyperopt.
[17] J. Axboe. https://github.com/axboe/fio.
[18] M. H. Jamal, G. Mustafa, A. Waheed, and W. Mahmood. An extensible infrastructure for benchmarking multi-core processors based systems. In *Inte'l Symp. on Performance Evaluation of Computer Telecommunication Systems*, 2009.
[19] J. Kumar and A. K. Singh. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems*, 81:41 – 52, 2018.
[20] LMbench. http://www.bitmover.com/lmbench/.
[21] J. Mars, L. Tang, R. Hundt, K. Skadron, and M. L. Soffa. Bubble-Up: Increasing Utilization in Modern Warehouse Scale Computers via Sensible Co-locations. In *Annual IEEE/ACM Int'l Symposium on Microarchitecture (MICRO)*, 2011.
[22] H. Moradi, W. Wang, A. Fernandez, and D. Zhu. uPredict: A user-level profiler-based predictive framework for single vm applications in multi-tenant clouds. In *Proc. of the IEEE Int'l Conf. on Cloud Engineering (IC2E)*, 2020.
[23] H. Moradi, W. Wang, and D. Zhu. Adaptive performance modeling and prediction of applications in multi-tenant clouds. In *Proc. of the IEEE Int'l Conf. on High Performance Comput. and Communications (HPCC)*, 2019.
[24] H. Moradi, W. Wang, and D. Zhu. Online performance modeling and prediction for single-vm applications in multi-tenant clouds. *IEEE Transactions on Cloud Computing (TCC)*, 1:1, 2021.
[25] M. Safaei Pour, A. Mangino, K. Friday, M. Rathbun, E. Bou-Harb, F. Iqbal, S. Samtani, J. Crichigno, and N. Ghani. On data-driven curation, learning, and analysis for inferring evolving internet-of-things (IoT) botnets in the wild. *Computers & Security*, page 101707, 2019.
[26] A. Sahba and J. J. Prevost. Hypercube based clusters in cloud computing. In *World Automation Congress (WAC)*, 2016.
[27] R. Sahba. A brief study of software defined networking for cloud computing. In *World Automation Congress (WAC)*, 2018.
[28] J. Scheuner and P. Leitner. Estimating Cloud Application Performance Based on Micro-Benchmark Profiling. In *Int'l Conf. on Cloud Comput. (CLOUD)*, 2018.
[29] Scikit-learn. http://scikit-learn.org/stable/.
[30] P. Shivam, V. Marupadi, J. S. Chase, T. Subramaniam, and S. Babu. Cutting corners: Workbench automation for server benchmarking. In *USENIX Annual Technical Conference*, 2008.
[31] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
[32] J. Vera, F. J. Cazorla, A. Pajuelo, O. J. Santana, E. Fernandez, and M. Valero. Measuring the performance of multithreaded processors. In *SPEC Benchmark Workshop*, 2007.
[33] K. Wang and M. M. H. Khan. Performance prediction for apache spark platform. In *2015 IEEE International Conference on High Performance Computing and Communications (HPCC)*, 2015.
[34] F. Xu, H. Zheng, H. Jiang, W. Shao, H. Liu, and Z. Zhou. Cost-effective cloud server provisioning for predictable performance of big data analytics. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 30(5):1036–1051, May 2019.
[35] N. J. Yadwadkar, B. Hariharan, J. E. Gonzalez, B. Smith, and R. H. Katz. Selecting the best vm across multiple public clouds: A data-driven performance modeling approach. In *Proc. of the Symp. on Cloud Computing (SoCC)*, 2017.