A CPU Contention Predictor for Business-Critical Workloads in Cloud Datacenters

Vincent van Beek
Delft University of Technology
vincent.vanbeek@solvinity.com

Giorgos Oikonomou
Delft University of Technology

Alexandru Iosup
VU Amsterdam
a.iosup@vu.nl

***Abstract*—Resource contention is one of the major problems in cloud datacenters. Many types of resource contention occur, with important impact on the performance and sometimes even the reliability of applications running in cloud datacenters. Cloud applications run together on the same physical machines with different workloads resulting in non-synchronized accesses to the shared resources. This leads to cases where co-hosted applications are contending for the common resources and not receiving the demanded resource amounts. In this work, we investigate the contention in CPU resources, as CPU is allowed to be over-committed by typical SLAs. We propose a CPU-contention predictor for the demanding business-critical workloads, which require low resource contention to deliver the required performance to customers. Our predictor is based on a set of regression models and metrics which we evaluate extensively. We tune the predictor with data collected from a real-world cloud operation spanning multiple datacenters and servicing business-critical workloads.**

***Index Terms*—CPU contention, Resource contention, business-critical workloads**

I. Introduction

Attracted by the dual promise of infrastructure efficiency [1] and widespread uptake [2], large organizations are increasingly using public- and/or private-cloud resources to run their large-scale *business-critical workloads* (BCWs) [3]. Although the promises are enticing, hosting BCWs is relatively new, and raises many resource management and scheduling challenges. The efficient operation of cloud datacenters requires the development of resource managers and scheduling policies that can cope with BCWs. Real world experiments are risky, time consuming, and expensive. Because datacenter operators are averse to any of these properties and because emulation is not possible at datacenter scale, simulation is often the only remaining approach to conduct datacenter-scale research [4]. For simulators to be accurate, they need to be able to take into account complex conditions that appear only at the scale of datacenters, such as widespread resource contention. Addressing this need, in this work we propose a CPU contention predictor that can be used in datacenter simulation.

The use of virtual machines (VMs) is the de facto standard in modern datacenters for hosting BCWs. When multiple VMs are co-hosted on the same physical machine, any of the VMs may experience transient performance degradation, caused by the contention for resources between multiple VMs.

To address this problem during scheduling, we must be able to anticipate contention in datacenter simulators. We focus in this work on CPU contention, which is likely to occur because SLAs BCWs allow for CPU resources to be over-committed (unlike, for example, memory resources, for which typical SLAs require it to be guaranteed as non-shared).

In this work we design a CPU-contention predictor that can use data available at runtime. Ours is the first predictor able to address CPU-contention in VMs hosted in clouds, efficiently. Our main contribution for using a CPU-contention predictor in datacenter simulation is three fold:

1. Release of a workload trace with contention data for BCWs (Section II): Cloud applications may suffer from performance degradation when the consolidated VMs contend constantly for the same resources. We release a one month workload trace of 1800 VMs which contains detailed contention data.

2. Design of a representative predictor for contention in real environments running business-critical workloads (Section III): To investigate the risks of SLA violations due to the decrease of VM performance, we need to measure the performance levels of VMs in simulation. We design a process via which we develop and test a CPU contention predictor that can predict performance degradation for VMs.

3. Experimental evaluation of a predictor for contention in real environments running business-critical workloads (Section IV): We evaluate and tune our predictor with real-world workload traces collected in production datacenters. We show that our predictor is accurate and can be used in simulation.

II. Resource Contention in Cloud Datacenters

There are many types of resource contention that occur in cloud datacenters. Applications deployed in cloud datacenters run together resulting in non-synchronized accesses to the shared resources. This leads to cases where co-hosted applications are contending for the common resources and are not receiving the demanded resource amounts.

Research in the field of identifying the reasons of resource contention [5],[6] focus on CPU and memory unavailability. In this work, we investigate the contention in CPU resources as CPU is allowed to be over-committed by typical SLAs for BCWs.

A. CPU-Contention Metrics

There are a couple of contention metrics engineers use to monitor contention in cloud datacenters [7]. All of these can be directly retrieved by the Performance Monitor even for short time durations. Such metrics are the *CPU-Ready* (ms and %), the *CPU Contention* (%) and the *CPU co-stop* (ms) metrics. There are also other metrics which can be used indirectly for measuring contention in the system by exploiting the relations between the metrics. One example of this is the relation of *CPU-run* and *CPU I/O wait* indicating the time duration a CPU core spends on running and on waiting for I/O calls respectively.

B. CPU-Ready Metric

In this work, we use the CPU-Ready as our contention indicator which measures the time a VM is ready to run but waits to be scheduled to run on the physical CPU [7]. The ready time increases when the scheduler is busy handling many waiting VMs which in turn have different provisions of cpu-cores and diverse resource demands. The CPU-Ready metric has become the *de facto standard* for engineers to measure contention problems. Thus we consider it as a good indicator to the problem of resource contention.

In Figure 1 we depict the contention durations of VMs in our workload trace (see Section II-C). To the best of our knowledge, this is the first work presenting contention periods of a production workload (we will release the trace via the Grid Workload Archive [8] after acceptance of the article). The contention durations are calculated by measuring the consecutive timestamps of VMs for which the *CPU Ready* metric is above 10%. We observe a heavy-tailed distribution with few VMs having max contention durations of at least 1 hour (13%) while the mean duration of at least 1 hour is attributed to even fewer VMs (4.2%). The outliers in the figure depict a small number of VMs whose contention durations reach up to more than 2 weeks.

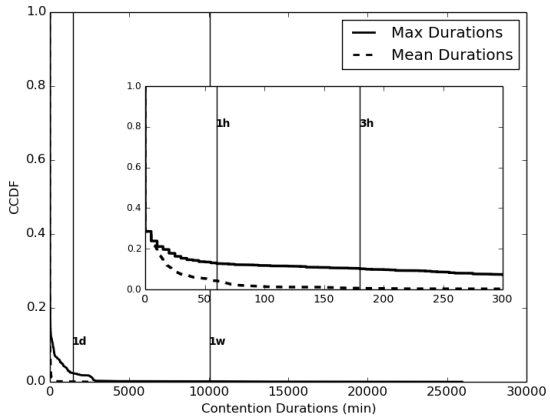


Fig. 1: Complementary cumulative distribution function (CCDF) of contention durations (min) in business-critical workloads.

C. VM Workload Trace from a Cloud Datacenter

We retrieve a trace of BCWs running in cloud datacenters. The trace contains a month worth of time serires data for 1800 VMs from 12 production clusters in cloud datacenters owned and operated by Solvinity (a Dutch cloud provider). In previous work we present a first characterization of business-critical workloads which are significantly different from scientific workloads and from the analytics workloads common at large web-scale companies (e.g., Facebook, Google, Tabao) [3]. To support the reproducibility of our work the trace will be made public after acceptance of the article via the Grid Workload Archive [8].

III. Selection Method for Predictors

We follow the modeling process described in [9] which can select a suitable model among various regressions models. The process is split into three steps as shown in Figure 2.

Firstly, we find correlated metrics to the CPU-Ready metric from a set of monitored metrics (raw metrics - Step 1). The correlated metrics are candidates for independent variables of the predictor (regressors-predictor input) and the CPU Ready is the dependent variable of the predictor (predictor's output). Secondly, we select the correlated metrics from the first step and try out combinations of the selected metrics with different regression models to evaluate their accuracy (Step 2). In the end, we choose the regression model and the combination of regressors that together give the best accuracy of predicted values considering also the predictor runtime (Step 3). The values which the model predicts are the CPU-Ready values of VM v at time t since every VM experiences the contention according to individual characteristics.

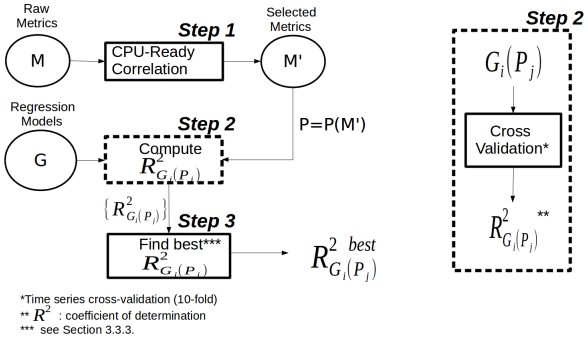


Fig. 2: Selection process. The Ellipses indicate sets of elements while the rectangles indicate a process.

A. Correlation-based Selection (Step)

We collect multiple operational metrics and find their correlation with the CPU-Ready metric (Step 1 in Figure 2). The metrics that have the highest correlations according to some boundaries are selected for the second step of the process.

There are hundreds of operational metrics that can be monitored and be candidates of our selection. The monitored

Regression Model	Description
Linear Regression (*LR*)	Fit data points to a line by minimizing least squares distance
Lasso	Least squares Model with regularization [1] using L1 norm
Ridge	Least squares Model with regularization using L2 norm
Elastic Net	Least squares Model with regularization using L1,L2 norm
Lars	Stepwise regression using forward selection
Polynomial Regression	n-th degree of polynomial regression
k-NN	Prediction according to the properties of k nearest neighbors
Gradient Boosting (*GB*)	Regression using decision trees with n estimators by applying loss function

TABLE I: Description of regression models.

metrics represent different levels of operations in VMs and in clusters starting from hardware performance counters of CPU cores up to cluster resource utilization. The research in [6] and in [9] investigates many metrics for their purposes but we focus only on metrics showing the resource utilization of VMs and of clusters preserving the concept of *opaque* VMs. As a result, we end up with a highly accurate predictor even if we start with only a few raw metrics (25 metrics).

We monitor the set M (see Figure 2) of metrics which consists of 25 performance metrics of VMs and of clusters. The performance metrics monitor all the four resource types we investigate in this work (CPU, memory, storage, network) and include metrics showing the resource utilization (e.g. read/write KBs in network, used MBs in memory) and latency of VMs (e.g., read/write latency in storage, cpuWait in CPU). reported in traces

a) Correlation in time-series: The VM traces (see II-C) comprise of all the traces of the M metrics including the time-series of CPU-Ready. For each VM, we evaluate the correlations between the VM's CPU-Ready and the VM's metrics in the set M. The VM correlations regarding metric M_j and CPU-Ready are aggregated first by cluster (a VM is hosted by one cluster) and then by all the clusters reported in the traces. In the end, for each metric $M_j \in M$, we have an aggregated correlation value depicting the correlation of the metric with CPU-Ready as well as the p-value of the correlation. The p-value presents the validity of the correlation whether the correlation value is wrongly as high as it is calculated. The mean correlation coefficient of the correlation values of VMs and clusters is calculated using z-transformation [10] since the correlation coefficients are not additive quantities (ordinal scale type) to compute their average directly. To allow aggregating of the p-values, we double the mean of the individual p-values [11].

b) Different correlation coefficients: To calculate the correlations, we use two correlation coefficients and we present the result of both correlations. The two coefficients, *Pearson* and *Spearman*, reflect different behaviors of data. Pearson's correlation (r) measures the linear relationship of data but it is susceptible to outliers. On the other hand, Spearman's correlation (ρ) measures a monotonic relationship of the variables because it applies ranks to values and is more robust to outliers.

We use two different correlation types because, in Step 2, we investigate multiple regression models some of whom work better with linear data and other regress data accurately even with non-linear relationships among regressors.

In the end of the correlation step, we choose the set $M' \subseteq M$ which includes the metrics with the highest correlation(r/ρ) with CPU-Ready and the lowest p-values ($p < 0.05$).

B. Model Accuracy (Step 2)

We investigate multiple regression models to end up with the best model for predicting the CPU-Contention in the system. Step 2 in Figure 2 shows the process we follow to measure the accuracy of regression models given a set of input metrics (regressors).

As set of possible regressors for a regression model, we use the output of step 1 (set M') and it contains the highly-correlated metrics with CPU-Ready among the raw metrics of set M. For each regression model in set G of regression models, we try out every combination of metrics in M'. The combination of metrics may comprise of one up to the total number of available metrics $|M'|$, therefore we define the power set $P = P(M')$ containing all the possible subsets of set M' as the input in Step 2.

Every regression model $G_i(P_j)$, where G_i is a regression model in G using as regressors the subset $P_j \in P$, is tested with cross-validation to measure the accuracy of the regression model. After calculating the accuracy of every possible combination $G_i(P_j)$, we select the best model for the use in a datacenter simulator (Step 3).

a) Cross-validation: Cross-validation is a technique for model validation and is used to estimate how accurate a predictive model will perform in practice [12]. A predictor may work well with specific input datasets but may not generalize the same to an independent dataset. Therefore, by changing the input multiple times cross-validation reduces this uncertainty of a falsely good predictor. The technique splits the input dataset of the predictor into k folds of which some are used as the training dataset while the remaining folds are used as the testing dataset. Because we work with time-series predictions, we use a special version of cross-validation (*time-series cross-validation*) in which the folds are ordered by time, thus a fold used in training dataset must not include future data

compared to data in the testing dataset. The accuracy of the predictor after cross-validating the model is the mean of k accuracy's calculated by the k rounds of cross-validation.

b) Regression Models: We investigate multiple regression models from both supervised and the unsupervised learning methods. In supervised learning, we feed the predictor with desired output values to adjust the model's internal states while in the unsupervised method, the model tries to identify the desired output without any exterior information. Additionally, we try to investigate regression models working well either with linearly or with non-linearly correlated data as mentioned in Section III-A. We use the following models: *Linear Regression* (**LR**), *Curvlinear Regression*, *k-Nearest Neighbors regression* (**k-NN**) and *Gradient Boosting regression* (**GB**).

The regression models along with some descriptions about the regression approach of the models are presented in Table I. *Linear Regression* is the typical approach to model relationships on data by minimizing the error between the training data and the produced line. *Lasso* ,*Ridge* and *Elastic Net* work in a similar as *Linear Regression* but they penalize the errors with cost functions derived from the norms of $L1$,$L2$ and both norms respectively. To differentiate the models, we use multiple solvers for *Ridge* for finding the best fitting line in the shortest run-time. *Lars* is an approach which adds available regressors to the model only if there is a significant improvement in the accuracy, thus resulting in a model with a few meaningful regressors. *Polynomial Regression*, or curvlinear regression, fits the data using polynomial functions which can explain nonlinear relations of data while *k-NN* clusters the training data according to the regressors values and reveals trends of the values of the independent variable (CPU Ready in our case). Finally, *Gradient Boosting* regression uses cost functions l such as least-squares or quantiles (Q) and builds (iteratively) decision trees sized n that indicate which output values reduce the costs.

c) $\mathbf{R^2}$ *- Coefficient of determination:* The coefficient of determination (R^2) [13] is a metric used to assess the accuracy of a predictor model. It evaluates the proportion of the variance in the predictor output (independent variable) that is predictable from the input (regressors). For usable models have the range of the metric is $[0, 1]$, the higher the better, although it can have negative values if the selected regressors are meaningless (our interest is in the non-negative range). We can use this metric to measure each predictor's accuracy (see $G_i(P_j)$ before) in Step 2. The metric limitation is that R^2 is monotone increasing, meaning that always increases when we add extra regressors to the model. We address this issue in Section III-C.

At the end of Step 2, we have a list of all models $G_i(P_j)$ (see Figure 2) with their respective R^2 scores. This list is the input of Step 3 of our selection process.

C. Predictor Selection (Step 3)

In the final step, Step 3 in Figure 2, we select the most promising predictor for estimating CPU contention. We make our selection among the list of models $G_i(P_j)$ considering three factors: a) The accuracy of the model depicted by the R^2 score, b) the runtime of the regression model and c) the number of regressors in the model.

Term	Description
d_v^t	Demanded CPU of VM v at time t
d_c^t	Demanded CPU of cluster c hosting VM v at time t
n_v^t	Number of virtual CPU cores of VM v at time t
r_v^t	CPU Ready of VM v at time t
r_v^{t-1}	CPU Ready of VM v at time $t-1$
r_v^{t-2}	CPU Ready of VM v at time $t-2$
$\overline{r_c^{t-1}}$	Mean CPU Ready of VMs in cluster c hosting VM v at time $t-1$

TABLE II: Terminology in the selection method for CPU-contention predictor.

The first two factors (a) and (b) are easily addressed by sorting the results in an increasing order for the R^2 scores of the models and in a decreasing order for the runtime of the models. As for the third factor, we leverage the method in [9] where we add a new regressor in the model only if it improves the R^2 score more than a given threshold (R^2_{impr}). A metric contributing to the predictor's accuracy less than R^2_{impr} will not be considered as an additional regressor.

The results of our selection method are presented in Section IV where we show the best accuracy for every type of regression model we use.

IV. Results for Selecting a CPU-contention predictor for BCWs

In this section, we present the results for the selection of a CPU-contention predictor described in Section III. The selected predictor will be used to estimate performance degradation of BCWs in datacenter simulation in future work. The predictor estimates the CPU Ready levels of every running VM in the system at time t (r_v^t).

A. Results of Step 1

As a result of this step, we have a set of 6 selected metrics (set M' in Figure 2) having correlation with the CPU Ready metric. We set the correlation value between a raw metric and the CPU Ready metric to be higher than $r/\rho >= 0.4$ and the p-value less than $p < 0.05$ to select a raw metric as a candidate regressor of the predictor. Table II summarizes the terminology used for the metrics of set M'.

The six selected metrics are: a) the demanded CPU of a VM at time t (d_v^t), b) the demanded CPU of the cluster hosting a VM at time t (d_c^t), c) the number of provisioned virtual cores of a VM at time t (n_v^t), d) the CPU Ready of a VM at time $t-1$ (r_v^{t-1}), e) the CPU Ready of a VM at time $t-2$ (r_v^{t-2})

[1]Penalizing bad fitting of data, mitigating data overfitting

and f) the mean CPU Ready of VMs in the cluster hosting a VM at time $t-1$ ($\overline{r_c^{t-1}}$).

In Figure 3, we present the results for the selected metrics. We show the correlation coefficients between each selected metric and the CPU Ready r_v^t. Next to the correlation values for every cluster, we show the mean correlation coefficient combining the results of all clusters, this is used to select a metric for step 2. In Figure 3a we present the auto-correlation of CPU Ready having been calculated by aggregating the autocorrelations of all the VMs in the trace. The trace has a monitoring period of 5 minutes and we present the lags in hour units. We see that for a very short time period, there is a significant auto-correlation for CPU Ready. For this reason, we select the last two entries, r_v^{t-1} ($corr = 0.85$) and r_v^{t-2} ($corr = 0.79$), as our candidate regressors in step 2.

In Figures 3b-3e we show the correlations of the other selected metrics which will be the input of the process in step 2. We present the Pearson and the Spearman correlation coefficients for every cluster as well as the mean correlation values. The differences of the correlation values among the values of the same correlation type are small implying stable correlation levels between the metrics. The only metric that shows irregular correlation behavior between clusters is the number of provisioned virtual cores of a VM and its CPU Ready values (Figure 3d). However, we observe that even though the correlations of clusters differentiate a lot, the mean correlation is high enough to select the number of vCPUs n_v^t as a possible regressor for step 2.

B. Results of Steps 2

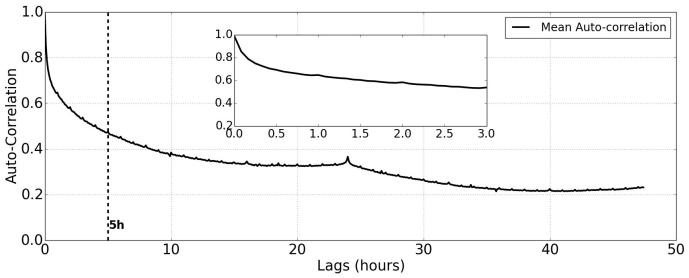
At step 2, we try every combination of the six selected metrics described in Section IV-A with each regression model in Table I. We test every pair of regression model and set of metrics with every (sliding) time window of 5 hours in the trace and we calculate the accuracy of the pair by aggregating the accuracy's of the time windows. The reason we split the trace into time windows is that the CPU contention levels for VMs is a temporary effect (we also see it in Figure 3a where the values are related to values of the recent past-5h data) and a predictor should be trained only by recent contention levels.

In Figure 4 we present the best (highest) accuracy's that every regression model achieved in step 2. With the additional requirement that 3 or more metrics are used as regressors.

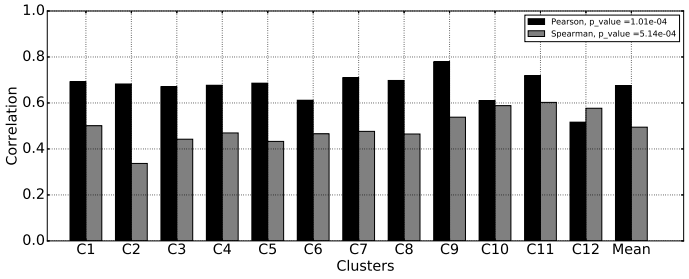
Figure 4 depicts for each regression model the average of R^2 accuracy's of the time windows along with the standard deviation of those accuracy's. We see that *Gradient Boosting* (GB) achieves the best accuracy (0.89) from all the other regression models while we have three models in the second place (*LR =Lars =Ridge =*0.86). The *Polynomial* model achieves impractical accuracy (negative R^2) with high standard deviation thus we exclude it from the selection process of Step 2.

C. Results of Step 3

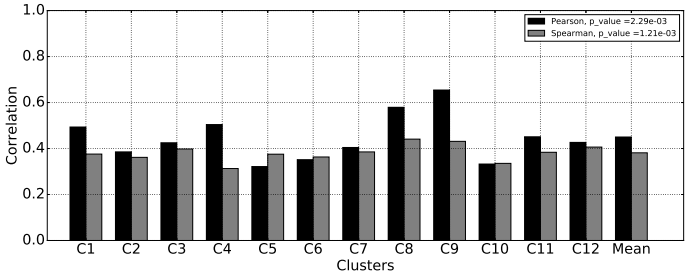
In step 3, we take the results shown in Figure 4 and we select the most suitable pair of model and set of metrics for



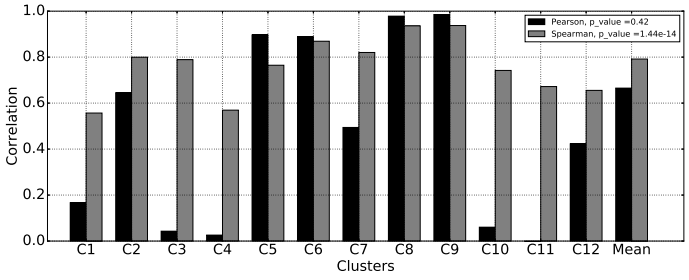
(a) CPU Ready auto-correlation.



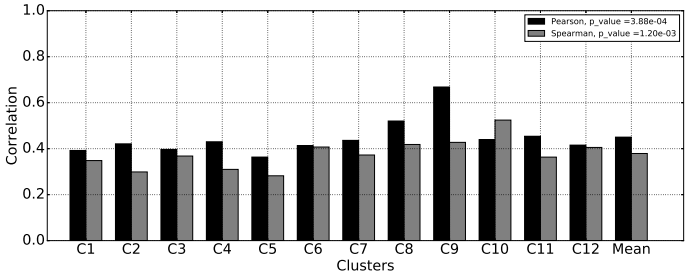
(b) d_v^t - r_v^t correlation.



(c) d_c^t - r_v^t correlation.



(d) n_v^t - r_v^t correlation.



(e) $\overline{r_c^{t-1}}$ - r_v^t correlation.

Fig. 3: Correlations of the selected metrics in Step 1.

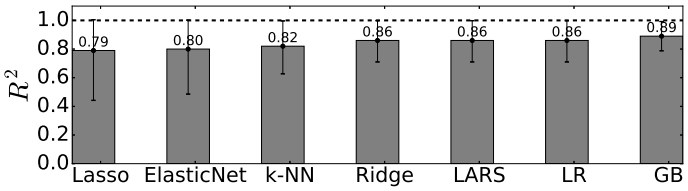


Fig. 4: Accuracy of seven regression models.

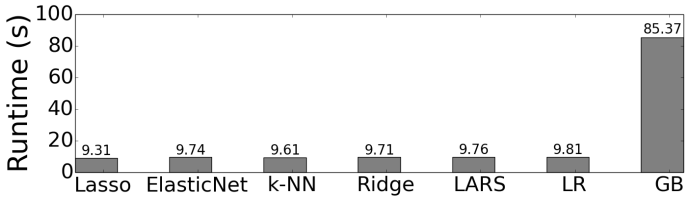


Fig. 5: Runtimes of the seven regression models.

our case as described in Section III-C. In Figure 5, we present the run-times of the models-set of metrics that are depicted in Figure 4.

The run-time values are the averages of the run-times of all time windows explained in the previous section. The run-time of *Gradient Boosting* is significantly higher than of the second-best models with difference larger than 800%. This increase of run-time is crucial for our simulations as the predictor is used at every monitoring event to estimate the current contention for each running VM. Therefore, we choose **Linear Regression** as our regression model which is the simplest among the other second-best models and introduces a lower computation overhead for the predictor run-time than that of *Gradient Boosting*. As for the number of metrics/regressors of the linear regression model, we end up with the 5 out of 6 selected metrics because the remaining metric increases the model accuracy less than R^2_{impr} ($LR^5 = 0.862, LR^6 = 0.865$). In the end we choose as CPU contention predictor the model presented in Equation 1.

$$G_i(P_j)^t_v = \mathbf{LR}(d^t_v, d^t_c, n^t_v, r^{t-1}_v, \overline{r^{t-1}_c}) \quad (1)$$

D. Verification of the model accuracy

To verify our chosen regression model, we run our datacenter simulator using the CPU-Contention predictor and compare the results with the real-world values collected by the system monitor. Figure 6 depicts this comparison by presenting the R^2 score for the percentage of data being tested ($\#data > 19$ million values). The results indicate a highly accurate predictor of CPU-Contention for 99% of the values (0.93), albeit a lower accuracy for the total number of values (0.61).

V. Conclusion

In this work, we address our research question about selecting an efficient predictor for CPU contention among VMs running business-critical workloads in cloud datacenters. We present a method through which we pick metrics correlated to CPU contention and use combinations of those metrics

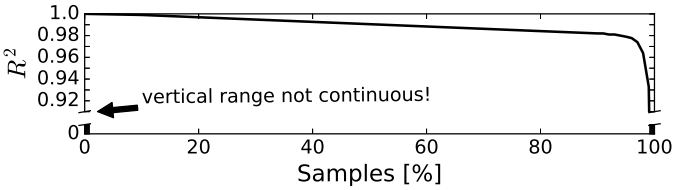


Fig. 6: Accuracy of the tuned predictor.

to investigate the predictive accuracy of multiple regression models. The method we propose could be used for selecting a predictor of a different contention metric than CPU Ready or even of a performance metric with different purpose.

Selecting an efficient predictor does not only require accuracy measurements, because the prediction process might be time-inefficient. Thus, we also consider the computation overhead of the predictions in our selection process.

For the future, we will test the method against other workloads and setups. Further, we will incorporate the CPU Contention predictor presented in this work in datacenter scheduling research in future work.

References

[1] IDC, "Worldwide and regional public it cloud services: 2013-2017 forecast," IDC Tech Report. [Online] Available: www.idc.com/getdoc.jsp?containerId=251730, 2013.

[2] European Commission, "Uptake of cloud in europe," Final Report. Digital Agenda for Europe report. Publications Office of the European Union, Luxembourg, 2014.

[3] S. Shen, V. van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *CCGRID*, 2015.

[4] V. van Beek, J. Donkervliet, T. Hegeman, S. Hugtenburg, and A. Iosup, "Self-expressive management of business-critical workloads in virtualized datacenters," *IEEE Computer*, vol. 48, no. 7, pp. 46–54, 2015.

[5] X. Ren, S. Lee, R. Eigenmann, and S. Bagchi, "Prediction of resource availability in fine-grained cycle sharing systems empirical evaluation," *Journal of Grid Computing*, vol. 5, no. 2, pp. 173–195, 2007.

[6] S. Blagodurov, S. Zhuravlev, and A. Fedorova, "Contention-aware scheduling on multicore systems," *ACM Transactions on Computer Systems (TOCS)*, vol. 28, no. 4, p. 8, 2010.

[7] VMware, "vSphere 6 documentation: Monitoring and performance," Official support resources. [Online]. Available: https://www.vmware.com/support/pubs/vsphere-esxi-vcenter-server-6-pubs.html

[8] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. J. Epema, "The grid workloads archive," *FGCS*, vol. 24, no. 7, pp. 672–686, 2008.

[9] P. Xiong, C. Pu, X. Zhu, and R. Griffith, "vPerfGuard: an automated model-driven framework for application performance diagnosis in consolidated cloud environments," in *ICPE*. ACM, 2013, pp. 271–282.

[10] D. M. Corey, W. P. Dunlap, and M. J. Burke, "Averaging correlations: Expected values and bias in combined pearson rs and fisher's z transformations," *The Journal of General Psychology*, vol. 125, no. 3, pp. 245–261, 1998.

[11] V. Vovk, "Combining p-values via averaging," *arXiv preprint arXiv:1212.4966*, 2012.

[12] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation," in *Encyclopedia of database systems*. Springer, 2009, pp. 532–538.

[13] N. R. Draper, H. Smith, and E. Pownell, *Applied regression analysis*. Wiley New York, 1966, vol. 3.