POSTER: Precise Capacity Planning for Database Public Clouds

Ningxin Zheng*, Quan Chen*, Yong Yang‡, Jin Li*, Wenli Zheng*, Minyi Guo*
*Department of Computer Science and Engineering, Shanghai Jiao Tong University
‡Alibaba Cloud
{ningxinzheng,lijin}@sjtu.edu.cn, {chen-quan,zheng-wl,guo-my}@cs.sjtu.edu.cn, zhiche.yy@alibaba-inc.com

I. Introduction

Platform-as-a-service (PaaS) is a type of Cloud computing in which a service provider delivers a platform to tenants. Within the PaaS category, the fastest-growing segment is the database platform as a service (dbPaaS). Tenants rent Cloud instances to ensure the good performance of their database workloads empirically. However, the empirical method often lead to the excessive purchase of resources. Prior work [1] has shown that more than 90% of Cloud applications apply for $5\times$ more resources than their actual demands. Capacity planning that identifies the smallest resource specification (number of cores, size of memory space) required by an application while its performance requirement can be satisfied is profitable for both tenants and Cloud providers.

There are several challenges in achieving the above goals. First of all, Cloud providers have to identify appropriate resource specification for a database workload online quickly, because tenants would not provide their workloads for offline profiling due to privacy reason. Second, only hardware event statistics and system-level indexes are available to plan the capacity, because workloads are in black boxes for Cloud providers. We propose a runtime system named **URSA** to address the above challenges.

II. Design of URSA

While planning capacity, URSA first predicts the performance scaling surface (as shown in Figure 1) of the target workload based on the hardware events and system-level indexes. Then, URSA plans capacity according to the predicted performance scaling surface.

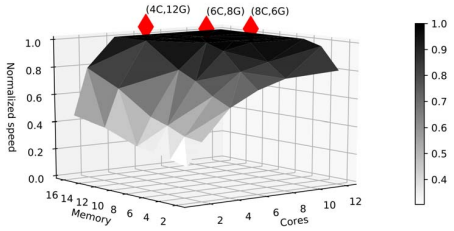

Fig. 1: Performance scaling surface of workload w.

A. Philosophy in URSA

The insight of URSA is that some system-level indexes and performance event statistics of a workload, such as instruction per cycle (IPC), can reflect its scale characteristic. IPC can be used to determine whether a workload is CPU-bound or memory-bound. The performance of the CPU-bound workload is positively related to the number of allocated cores. Therefore, if the indexes and performance event statistics of two workloads are close, then these two workloads tend to have similar scale characteristics. Although URSA is proposed for real system dbPaaS Clouds that only host databases (e.g., Alibaba RDS), it can be generalized for Clouds that host general applications as long as more training samples are collected from them to capture their scaling characteristics.

*Quan Chen is the corresponding author of this paper.

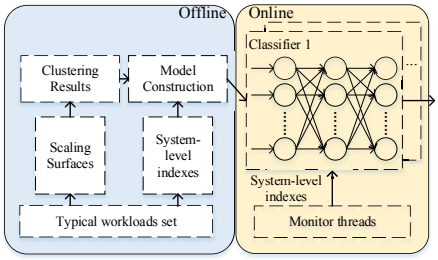

Fig. 2: Design of URSA.

B. Identifying Correlated Features

We use Lasso regression [2] to identify the features related to the workload's performance. Accurately, URSA first collects various hardware counter events and system-level indexes from the representative workloads. Then, URSA uses Lasso to filter out the most relevant system-level indexes and hardware counters of the workload's performance. The filtered system-level indexes and hardware counter events are used as input feature of the online classifier of URSA.

C. URSA Construction

Figure 2 shows the design of URSA. First, URSA collects the scaling surfaces and the selected indexes of a set of representative workloads in a specified configuration region offline. Second, URSA clusters a set of scaling patterns according to the collected scaling surfaces. To build the training set of the online classifier, URSA aggregates the system indexes and event statistics of the applications that have the same $ClusterID$. Specifically, the training set is built as $\langle SystemIndexes, ClusterID\rangle$ where $ClusterID$ is the clustering result of the application's scaling surface. $SystemIndexes$ is the average values of the performance statistics of the workload for each time period (such as 10 seconds) during the execution. Each cluster's center scaling surface is the mean vector of the other scaling surfaces and is used as the representative scaling surface of this cluster for specification searching. Finally, URSA uses $SystemIndexes$

978-1-7281-3613-4/19/$31.00 ©2019 IEEE
DOI 10.1109/PACT.2019.00043

as input, $ClusterID$ as output to train the classification model.

URSA has one or more base configurations, and each base configuration corresponds to a classifier. The $SystemIndexes$ used to train the classifier are collected under the corresponding base configuration. For the scenario in which the workloads' resource configuration cannot be changed during the capacity planning, URSA can train a classifier for each configuration in the configuration region. Then, URSA selects the current configuration of the target workload as the base configuration and uses the corresponding classifier to predict its performance scaling surface. In this case, URSA can perform capacity planning without customers' perception. It is worth noting that training a classifier for each base configuration will not introduce more data collection overhead, because the *System Indexes* can be collected at the same time while collecting the scaling surface.

When URSA performs capacity planning for an online workload, the monitor thread collects the system-level indexes and hardware counter statistics of the workload under the base configuration and passes them to the trained classifier. The classifier classifies the workload into a scaling surface cluster according to the collected indexes. URSA returns the representative scaling surface of the cluster as the predicted scaling surface of the target workload.

III. Evaluation of URSA

To simulate real-system database workloads, we generate 55 database workloads using two widely-used workload generators: Sysbench [3], and OLTP-Bench [4]. The 55 workloads have different read-write ratios, compute densities, and database operating transactions, thus simulate a spectrum of real-system workloads. We evaluate URSA using these 55 workloads in the configuration region: [1 core-12 cores, 2GB-16GB]. The workloads set is randomly divided into a training set containing 44 workloads and a validation set containing 11 workloads. We use the transaction per second (TPS) as the performance indicator of the workloads. Specifically, the K-means is used to cluster the scaling surfaces and the Multilayer Perceptron is used as the classifier in URSA. Equation 1 defines the error between the predicted scaling surface and the actual scaling surface of the target workload. In this equation, N_{conf} is the number of resource configuration in the configuration region, $Speedup_i$ is the predicted speedup of the configuration *i* relative to the base configuration, and the $Speedup'_i$ represents the actual speedup of the configuration *i* relative to the base configuration.

$$Err_{surface} = \sum_{i=1}^{N_{conf}} |\frac{Speedup_i}{Speedup'_i} - 1|/N_{conf} \qquad (1)$$

A. Prediction Accuracy

Figure 3 shows the prediction errors of the workload in validation set when the *(6C, 8G)* is selected as the base configuration and the number of clusters in K-means is 20. Observed from Figure 3, the maximum prediction error of validation workloads equals to 5.88%. Thus, the capacity planner can accurately predict the scaling surface of the workload based on the hardware counter events and the system-level indexes.

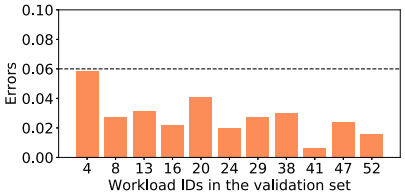

Fig. 3: Prediction errors of the capacity planner.

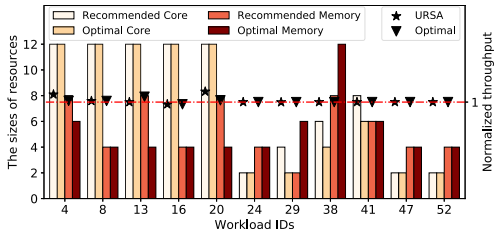

Fig. 4: Specifications recommended by URSA and the optimal specifications for reducing rent cost.

B. Reducing Rent Cost

In this experiment, URSA recommends a smaller specification that has the same performance with the current specification. To emulate the scenario that tenants over-rent resources for ensuring high performance, we assume the origin resource specification of each workload in the validation set is (12C,16G) that consists of 12 cores and 16GB memory.

Figure 4 shows the specification recommended by URSA and the optimal (i.e., the smallest) specification identified by searching through all the possible resource specifications. If there are multiple local optimal specifications, the specification with the least number of cores is selected as the optimal configuration. Observed from the figure, URSA identifies the optimal resource specifications for 5 out of 11 requests. For the other requests, the specifications recommended by URSA only has 1 more core and/or 4GB memory. For the 11 workloads, the specifications recommended by URSA reduces 43.6% cores and 65.5% memory compared with the original specifications without degrading the original performance, and uses 7.7% more cores and 3.4% more memory than the optimal specifications. Meanwhile, Figure 4 also shows that the specifications recommended by URSA do not hurt the performance of all the workloads in the validation test.

IV. Acknowledgement

This work is partially sponsored by the National R&D Program of China (No. 2018YFB1004800), the National Natural Science Foundation of China (NSFC) (61602301,61632017,61702328).

References

[1] C. Delimitrou and C. Kozyrakis, "Quasar: resource-efficient and qos-aware cluster management," *ACM SIGPLAN Notices*, vol. 49, no. 4, pp. 127–144, 2014.

[2] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[3] A. Kopytov, "Sysbench: a system performance benchmark," *http://sysbench. sourceforge. net/*, 2004.

[4] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux, "Oltp-bench: An extensible testbed for benchmarking relational databases," *Proceedings of the VLDB Endowment*, vol. 7, no. 4, pp. 277–288, 2013.