Proactive Workload Prediction in Clouds via Machine Learning

Ji Qi
3030058142
jqi@cs.hku.hk

Tianxiang Shen
3030058776
txshen2@cs.hku.hk

Xusheng Chen
3035028520
xschen@cs.hku.hk

1 Introduction

With the prevalence of cloud computing, more and more companies choose to deploy their service in the cloud. One key advantages of deploying service on the cloud over deploying on-premise cluster is that it can be scale-in and scale-out easily [1]. For instance, a web service can deploy more servers (scale-out) during daytime to balance the loads and to handle more clients; the service can shutdown some of the server (scale-in) in midnight.

It is a challenging problem to making these scaling decisions in practice. Users' workload trend and distribution is unknown and may contain spikes. Failing to handle these workload spikes may make server applications to be overloaded, or even crashed. In 2018, Amazon failed to predict the huge amount of transactions in its Prime Day sale, which caused one of its storage crashed and the service to be unavailable for an hour [2].

Existing web development typically uses a passive scaling method [4, 5, 6], where a server program does a scale-out operation if the workload exceed certain thresholds. However, this approach has two major drawbacks. First, this threshold-based method is sufficient for gradually-increased workloads but cannot handle workload spikes. If the workload increases suddenly, it needs some time before a scaling-out operation is completed. During that time, the service's Service Level Objective (SLO) is breached. Moreover, to avoid the application from crashing and losing data, this method needs to reject part of the user requests [7], causing bad user experience.

Second, different from stateless applications (e.g., a web server) that can scale out easily, scaling out a stateful application (e.g., a database or a key-value store) takes a long time because scaling-out a stateful application involves expensive data replications. Letting a highly loaded or overloaded stateful application to does a scale-out operation will exacerbate its situation and greatly increase the time for finishing the scale-out operation [3].

Therefore, an active method that can predict workload spikes and scaling out in advance is highly desirable. Machine learning is a promising techniques to solve this problem. In this project, we are going to analyze the existing workload prediction algorithms and frameworks, compare their performance, and discuss their applicability in practice. Specifically, we will use online open source workload traces to train our model and evaluate its performance. Moreover, we are also considering combing workload-trace based model with sentiment analysis because some workload spikes may come from a social event. For instance, the aforementioned Amazon outrage was caused by the launching of a big sale.

The remaining of the proposal is organized as follows. §2 describes the problem and the methods we are going to investigate. §3 presents the ways to evaluate the system performance.

2 Methodology for Workload Prediction

This section presents our methodology for realizing the cluster workload prediction. There are two approaches to allocate the correct mount of resources dynamically. The first approach is reactive approach, where thresholds are defined in advance, and resources are increased or decreased when the thresholds are reached. Another approach is proactive approach. The key insight of this approach is that the workload changes are time-dependent, so the resource re-allocation can be triggered before the occurrence of load fluctuates. This approach can achieve better performance but requires the system to monitor and predict the workloads.

In this study, we use machine learning to determine how many requests will come in the near future. The proposed system is consist of three components: *workload collector*, *predictor* and *resource manager*. In the beginning, the historical data set is fed into the *predictor*, which trains the model based on these data. When the system starts running, the *predictor* gives an estimation of system workload with certain time-interval and *resource manager* will allocate suitable resources according to the prediction. The time interval is assigned depending on the specific application requirements. During system operation, *workload collector* collects the actual number of requests in each time-interval and feeds this data to the predictor to adjust model on-the-fly.

2.1 Predictor

For a database service, it is very difficult to accurately predict the traffic. Usually people use an empirical approach to subjectively judge the workload of the next stage, but such an approach is not accurate enough, and once prediction fails, it can cause immeasurable adverse effects. For example, if traffic forecasts for Tmall[1] Double 11 activities fails, a shortage of server resources can cause significant economic losses. This problem becomes increasingly serious, and there is an urgent need for a solution to predict the workload more accurately.

In order to achieve better accuracy, we use machine learning (ML) methods to do prediction. Classic ML tasks include classification and regression. The goal of classification is that, give an object with parameters, using a model (e.g. Support Vector Machine) to determine the category of the object. The result is an exact value (e.g. Group ID). Whereas the outcome of regression tasks is a series of consecutive numbers, such as the predicted age of an adult. *Logistic Regression*, *Support Vector Machines* (SVM) and *Random Forest* are three classic ML models, which have good performance on ML tasks. We will test the performance of these three models on traffic prediction problems. Next we will briefly introduce these three models. Further detailed methodology will be discussed in §3.

Logistic Regression (LR) is essentially a linear regression, except that a layer of function mapping is added to the feature-to-result mapping, that is, the features are first linearly summed, and then the most hypothetical function is used to predict using the function g(z). g(z) can map continuous values to 0 and 1.Its advantage is that the prediction outcome can be mapped between 0 and 1, so that the results are very clear. Also, it can be applied to both continuity and categorical independent variables.

Support Vector Machines (SVM) is a two-class model. Its purpose is to find a hyperplane to segment the sample. The principle of segmentation is to maximize the interval and finally transform it into a convex quadratic programming problem.The name "support vector" comes from the basic theory of the model: in the case of linear separability, an instance of the sample point closest to the separated hyperplane in the sample points of the training data set is called a support vector. SVM is essentially a nonlinear classifier, and its learning strategy is to maximize the interval.There are already some mature SVM tool kits available for use, such as Matlab's SVM toolbox, LIBSVM or SciKit Learn under the Python framework.

Random Forest (RF)RF combines the advantages of Bagging and the Decison Tree. Bagging has the feature of reducing the variance of different gt variances. This is because Bagging uses the form of voting to combine all the gts to achieve averaging, thus reducing variance. The Decision Tree has the feature of increasing the variance of different gt variances. Bagging can reduce variance, and Decision Tree can increase variance. RF combines the advantages of both and therefore performs well.

2.2 Optimization

To improve the prediction accuracy, two optimization methods: bootstrap aggregation (bagging) and boosting will be used. In bagging, several predictors are trained in parallel; their results are aggregated to the final prediction. Bagging can give substantial gains in accuracy. However, multiple

[1]Tmall is a subsidiary of Alibaba, a famous e-commerce company in China.

predictors may have low accuracy for the same subset of data. We are going to use boosting to solve this problem. In this method, predictors are trained iteratively. In each iteration, data with poor accuracy will be assigned with higher weight.

3 Evaluation

We are going to evaluate the system using cluster tracing data from a real production environment. The data set contains data over about a month-long period. The data for the first three weeks are used for model training. The fourth week's data are used for evaluation.

In the simulation, requests are sent to the system, and the algorithm gives the predicted number of requests at the end of this time interval. Resources are deployed before the start of the next time interval. The duration of a time interval should be longer than the deployment time.

The number of requests for each time-interval is collected and outputted with different confidence ranges. The accuracy of the prediction will be evaluated using several metrics.

References

[1] S. Figone. Scaling in the Cloud – When You Grow, Cloud Grows.

[2] E. Kim. Internal documents show how Amazon scrambled to fix Prime Day glitches, 2018.

[3] C. Kulkarni, A. Kesavan, T. Zhang, R. Ricci, and R. Stutsman. Rocksteady: Fast migration for low-latency in-memory storage. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 390–405. ACM, 2017.

[4] M. Mao, J. Li, and M. Humphrey. Cloud auto-scaling with deadline and budget constraints. In *2010 11th IEEE/ACM International Conference on Grid Computing*, pages 41–48. IEEE, 2010.

[5] A. W. Servie. AWS Auto Scaling.

[6] B. Tardif. Scaling Up and Scaling Out in Windows Azure Web Sites.

[7] H. Zhou, M. Chen, Q. Lin, Y. Wang, X. She, S. Liu, R. Gu, B. C. Ooi, and J. Yang. Overload control for scaling wechat microservices. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 149–161. ACM, 2018.