Prediction-based Resource Provision and Virtual Machines Placement in Cloud Data Center

Shengyu Du$^1$, Yufei Wu$^2$ and Changqing Yin$^1$

1 Tongji University, Shanghai, China
2 Shanghai Jiao Tong University, Shanghai, China

Abstract. With the rapid development of virtualization, cloud servers provide the power and flexibility that single servers struggle to provide. However, low utilization of physical machines and high energy consumption are the main concerns for cloud service providers. In this paper, we propose a predict-and-place framework to decrease the number of active physical machines in cloud centers. We analyze the historical VM (virtual machine) request records to predict the VM demands in the next several days and then use an offline VM placement strategy to keep the number of active servers as less as possible while satisfying the SLA (Service-Level Agreement) requirements. In the prediction process, we enhanced the classical Holt’s linear prediction method on account of inevitable outliers to increase the prediction accuracy. We conduct experiments to compare the prediction accuracy of Perceptron, classical Holt-Winters, and our refined, robust Holt-Winters method. The experiment based on real data shows the simple perceptron has an inferior result. Our improved prediction method reduces the MAPE (Mean Absolute Percentage Error) by about 50% compared with the classical one, and the predict-and-place framework decreases the average number of active physical machines effectively in the cloud data center.

1. Introduction

Cloud data center and virtualization are being a hot choice for enterprise and personal developer. However, over-provision (low-utilization) of physical machines is a public cost concern for the cloud provider. Simultaneously, customers prefer elastic compute service (ECS) to support their changing demand. In order to quickly respond to customer’s request and guarantee good performance at peak demand, data center usually activates redundant servers (physical machines) and makes them on standby. The standby machines should be as less as possible to reduce energy cost.

Norman et al. [1] studied a large number of traces from production servers. They classified the behaviors into three main categories and argued that the behaviors with strong variability and autocorrelation could benefit most from prediction. After analyzing the VM request records from a real cloud center, we also found the increase or decrease trend of VM numbers in the near future has a strong relationship with the recent demands. Therefore, the forecasting time series is the key to solve the problem. Hellerstein [2] decoupled periodic components of known length and then model the residual process using Auto Regressive. Guosheng Hu [3] used grid resources prediction based on Support Vector Regression to make scheduler manage the grid resources more effectively. Some of the exponential smoothing methods have occupied close to the top spots in the M3 forecasting competition rankings [4]. There is a variety of exponential smoothing algorithms such as double exponential smoothing [5] and Holt-Winters seasonal method [6]. However, classical Holt’s linear
method is supervised approach and performs poorly if there are outliers in the data. So how to handle with outliers is a primary step in many data-mining applications. Accurate prediction will provide better support for placement in the next specified time interval. VM placement is the process of selecting the most suitable PMs in cloud center to deploy VMs. Many existing works have proposed some solutions to the VM placement problem. For example, [7-8] model this problem as an offline bin-packing problem and solve it by heuristic algorithms. Optimal placement means hardware resources should have minimal fragmentation which implies a minimal number of PMs. Online bin-packing based on well-known heuristics can provide reasonable but not optimal results. In contrast, offline bin-packing assumes the arrival demands and finds an optimal result. However, as demands are changing over time, it is easy to violate SLA(Service-Level Agreement) requirement or lead to over-provision. If a service availability violates the SLO (Service-Level Object, which is measured by SLI (Service-Level Indicator)), the cloud operations need to react quickly to avoid it breaking SLA, otherwise, the company might need to refund money to customers. In this paper, we consider both request prediction and VMs placement to solve the low-utilization challenge. The main contribution includes,

- Robust Holt’s linear method with outliers which can decrease the MAPE error by about 50% in real dataset collected from the cloud center.
- An offline VM placement algorithm which considers both CPUs and main memory. We represent VM placement as CSP (Constraint Satisfaction Problem) and make use of mature solver to get a global optimum placement plan.
- We compare the prediction accuracy with the neural network and the traditional Holt’s linear method. The result shows our method can tolerate outliers and reach a higher forecast accuracy when using a real trace of VM demands in several months. The results show our algorithms can achieve both accurate prediction and higher utilization of PMs.

2. Model and concepts

2.1 Architecture

Figure 1 and Figure 2 show the architecture of our model and the predict-and-place algorithm. They present the resource management strategy for the cloud data center. The monitoring module collects the VM requests and stores it in the repository. By using these records, the prediction module forecasts the VM demands in the next interval. Then, the predicted values are used by the placement module to compute the mappings of VMs to PMs for the next interval.

![Figure 1. Architecture of model](image1)

![Figure 2. Predict and place algorithms](image2)
2.2 Assumption

2.2.1 VMs have fixed kinds of flavors. In this paper, we only consider the data center provides fixed-flavors VMs to customers. This is the most common way of service delivery for most cloud service providers. For example, Aliyun provides hundreds of VM flavors to choose. In practice, different flavors additionally have the difference in processor type, frequency and so on. To simplify, we assume that VMs belong to different flavors(types) mainly according to the kernel number of CPU and the size of memory.

2.2.2 VM recycle doesn’t happen in the prediction period (such as 7 days). In the cloud center, every request will be recorded, and the VM will not be recycled because of expiry in a short period. We make this hypothesis because most cloud providers offer only monthly or yearly payment such as Aliyun and Tencent Cloud.

3. Prediction

For each type of VM flavor, we analyze the historical request and build a predictor that predict the future request. The prediction used here is based on Holt’s linear trend method. We find that training data often have outliers(as shown in Figure 3). It is crucial to deal with outlier before applying Holt’s linear trend method. Corrections of outliers are much complex and can’t always have an accurate result. Therefore, an inaccurate outlier removal process may lead to bigger prediction accuracy. According to the above judgment, we choose the robust prediction method rather than deal with the outliers’ challenge.

3.1 Classical Holt’s Linear Trend Method

Classical Holt’s linear trend method involves a prediction equation and two smoothing equations (one for level and the other for the trend)

\[
\hat{y}_{t+h|t} = \ell_t + hb_t
\]

(1) is the prediction function, (2) is the level equation and (3) is the trend equation. \(\hat{y}_{t+h|t}\) means the predicted value at time \(t + h\), \(h\) is a time ahead from the last observation value. \(\ell_t\) denotes an estimate(smoothing) of the level at time \(t\), \(b_t\) denotes an estimate(smoothing) of the trend(slope) of the series at time \(t\). \(\alpha \in [0,1]\) is the smoothing parameter for the level and \(\beta \in [0,1]\) is the smoothing parameter for the trend.

As with simple exponential smoothing, the level equation shows that \(\ell_t\) is a weighted average of observation(historical) value \(y_t\) and the within-sample one-step-ahead prediction for time \(t\), given by \(\ell_{t-1} + b_{t-1}\). The trend equation shows that \(b_t\) is a weighted average of the estimated trend at time \(t\) based on \(\ell_t - \ell_{t-1}\) and \(b_{t-1}\), which is the previous estimate of the trend.
Therefore, the prediction equation is no longer flat as simple exponential smoothing but trending. We use a linear function of $h$ to predict. The $h$-step-ahead forecast is equal to the last estimated level plus $h$ times the last estimated trend value. In our method, for the best accuracy, the start of predicted time should be immediately the end of historical data. The error correction form of the level and the trend equations show the adjustments regarding the within-sample one-step prediction error:

\[
\ell_t = \ell_{t-1} + b_{t-1} + \alpha e_t \\
b_t = b_{t-1} + \alpha \beta e_t
\]

where

\[
e_t = y_t - (\ell_{t-1} + b_{t-1}) = y_t - \hat{y}_{t|t-1}
\]

3.2 Robust Holt’s Linear Trend Method

As a special case of KALMAN filter, we modify the Holt method to the robust version [9].

\[
\hat{y}_{t+h|t} = \ell_t + hb_t \\
\ell_t = \ell_{t-1} + b_{t-1} + \alpha s_{t-1}\psi(\hat{e}_t) \\
b_t = b_{t-1} + \alpha \beta s_{t-1}\psi(\hat{e}_t)
\]

where the robustifying function $\psi(\cdot)$ is defined as:

\[
\psi(x) = \begin{cases} 
  x, & |x| \leq u_{1-\frac{\nu}{2}} \\
  \text{sign}(x)u_{1-\frac{\nu}{2}}, & |x| > u_{1-\frac{\nu}{2}}
\end{cases}
\]

the recursive scale estimator $s_t \approx \sigma(y_{t+1} - \hat{y}_{t+1|h}) = \sigma(e_{t+1})$, and the normalized prediction error $\hat{e}_t$ is then estimated as $\hat{e}_t \approx e_{t+1}/s_t$. In order to evaluate the prediction accuracy, mean absolute percentage error (MAPE) is used:

\[
\text{MAPE} = \frac{1}{n} \sum_{t=1}^{n} \frac{|e_t|}{y_t} = \frac{\sum_{t=1}^{n} |y_t - \hat{y}_t|}{n}
\]

We denote that smoothing parameters $\alpha$ and $\beta$ is empirical value and should be specified by users. When $\alpha$ increases, the level $\ell$ has greater relationship with observation value $y$. Similarly, when $\beta$ increases, the trend $b$ has greater relationship with the difference of $\ell$ and less with smoothing value.

To select the best parameters with outliers, we implement the robust Holt’s linear in section 5.

4. Placement

Mapping many virtual machines into one single server is a common method to make good use of the hardware resources. In general case, VM request arrives without any notice. In that case, VM can be moved to a more suitable machine to reduce the number of active PMs with efficient migration techniques. However, many approaches have neglected the ensuing migration time. For example, while migrating a single Xen VM can be very efficient incurring only 6~26 seconds [10], it may not be possible to happen immediately. Also, migrating a VM to other PM may degrade the performance of those VMs in that PM for a while.

Therefore, with accurate prediction, we can focus on the problem of static VMs’ placement. We estimate the number of arrival requests in a period (several days) and compute the placement method having the minimum number of PMs with the fixed period. The placement objective is to maximize the average utilization of active PMs, hosting VMs subject to the constraints describing the set of VMs and their CPU and main memory requirements.

4.1 Constraints to avoid overloading

Each PM has a certain amount of main memory and numbers of CPU kernels, and each VM must require less amount of these resources than the remaining. These constraints must be satisfied otherwise the performance of VMs on those PM will be severely affected. For example, if every PM is a uniprocessor, then the configuration in Figure 4 is not viable because it includes two active VMs on a single PM.
The problem of finding a minimal, viable placement is reducible to the NP-Hard 2-Dimensional Bin Packing Problem \cite{11}, where the dimensions correspond to the amount of memory and number of CPUs.

4.2 Constraint Satisfaction Problems in VMs Placement

We use similar representation method as Entropy \cite{10} to express the VM placement problem as CSP. For each PM $p_i \in P$, the bit vector $H_i = < h_{i1}, \ldots, h_{ij}, \ldots, h_{ik} >$ denotes the set of VMs assigned to PM $p_i$. For example, $h_{ij}$ means PM $p_i$ is hosting the VM $v_j$. Let $R_p$ be the vector of CPU demand of each VM, $C_p$ be the vector of CPU capacity of each PM, $R_m$ be the vector of memory demand of each VM, and $C_m$ be the vector of memory capacity of each PM. Then, the constraints are shown as follows:

\begin{align}
R_p \cdot H_i & \leq C_p(i) \ \forall n_i \in N \\
R_m \cdot H_i & \leq C_m(i) \ \forall n_i \in N
\end{align}

Given these constraints, the objective function is defined as follows to minimize the value of $X$:

\begin{equation}
X = \sum_{i \in N} u_i, \ \text{where} \ u_i = \begin{cases} 
1, & \exists v_j \in V | h_{ij} = 1 \\
0, & \text{otherwise}
\end{cases}
\end{equation}

where $u_i$ is 1 if the PM $i$ hosts at least one VM, and 0 otherwise. We let $x_{mpp}$ denotes the solution. This is exactly a multiple knapsack problem and can be done by using a dynamic programming approach as in \cite{12}.

4.3 CSP Solver: Choco

Choco \cite{13} is a free open-source Java library dedicated to constraint programming. It is built on an event-based propagation mechanism with back trackable structures. Users only need to model their problem in a declarative way by stating the set of constraints.

Since the duration of collected records is almost four months and the flavor types are 15 (see section 5), data are a small amount, so we don’t use any optimization method. It will be included in our future work.

5. Evaluation

In this section, we first evaluate the prediction module on a real VM requests data set. Supervised and robust learning technique is used to find better parameters $\alpha$ and $\beta$. Secondly, we implement a placement module based on the predicted values. The results show the number of total active machines is reduced without violating SLA in most case comparing to static provision. The experimental results were obtained using a personal computer with Intel Core@2.7GHz and 16GB RAM.

5.1 Robust Holt’s Linear Prediction

In our experiment, we use a dataset of VM request series from January 2015 to May 2015 collected from a cloud data center provided by Huawei corporation. In this dataset, VMs have 15 flavors, and the PMs’ flavor is fixed. The dataset contains over 3000 records. The record format of data is defined in section 2.3 and the time granularity is the day.

In order to find the best $\alpha$ and $\beta$ to increase the prediction accuracy. Data is divided into the training set, validation set, and test set. We did an exhaustive search and choose those which had the least error (loss) in the validation set. The MAPE is computed in the test set to get the prediction accuracy.
As we can see in Figure 5, $\alpha$ and $\beta$ are the determinant of prediction accuracy. However, the optimization procedure cannot guarantee to reach the global minimum, Farnum showed that the response surface is not necessarily convex [14]. So, choose the best parameters is the most important to the model performance. We compare our robust Holt’s linear method with MSE’s error function method and Gelper’s [15] default parameters to show that our method has better performance with outliers and a good performance in general cases.

Before applying our prediction algorithm, we designed a 3-layers network with fully connected layers. We use the first five days (as input) to predict the future one day (as output). Hidden layer has 8 nodes to generate the latent pattern.

The result (Figure 6) shows the three layers Perceptron has a poor performance. The solid and dashed lines of the same color represent the real data and predicted value of the same flavor. It shows great deviation for the six flavors and thus proves the simple machine learning strategy is not feasible. Then we test two Holt’s Liner methods. As for 10th flavor (Figure 7), the request’s series had a long and steady zero up to the 45th day. For MSE estimation method, it chooses a small $\alpha$ and $\beta$ (0.05) because it leads to a small error in most data points of validation set. Errors caused by outliers are
amplified by the square operation. For robust estimation method, outliers (zero points) are weakened through absolute median error. So, larger $\alpha$ and $\beta$ are selected in this case. The predicted value will be smoothed and consistent with the general trend.

5.2 Placement

5.2.1 Reduction in total running PMs. Figure 9 compares active PMs daily required by the predict-and-place method and static allocation at a specified (1, 3, 5 respectively) standby (flavor-1). The black line represents the exact SLA requirement in 7 days. For static standby method, with the increase of standby machines, it provides better service quality guarantee (as shown in Figure 10[right], it decreases to 0 until 5 standbys). However, redundant standby machines obviously lead to the low-utilization challenge. With our predict-and-place method, the average number of standby machines is decreased (as shown in Figure 10[left], compared with 3-standby and 5-standby) and also have the 0-violation SLA.
5.2.2 Reduction in rate of SLA Violations. We compute the number of break SLA in 7 days (as shown in Figure 10). Although the predict-and-place method has a little higher average number of machines than 1-standby PM(s), it decreases the number of break SLA sharply than 1-standby PM (even the 3-standby PMs causes the SLA break once).

According to the number of SLA break and average active machines with an accurate prediction. The predict-and-placement achieve a good tradeoff between low utilization challenge and the strict SLA requirement.

![Comparison of average running machines and Comparison of SLA violations](image)

**Figure 10.** Comparison of average running machines and Comparison of SLA violations

6. Conclusion

In this paper, we proposed a predict-and-place framework to solve the physical machines’ low-utilization challenge. As for a prediction, robust Holt’s linear method is used to tolerate inevitable outliers in the meantime achieve accurate prediction. As for placement, we can use global optimum placement algorithm, which has less required PMs and higher average utilization of resources. Compared with previous methods, we provided the complete framework with robust prediction method and offline placement technology.

7. References

[1] Bobroff, N., Kochut, A., & Beaty, K. (2007). Dynamic Placement of Virtual Machines for Managing SLA Violations. *Ifip/ieee International Symposium on Integrated Network Management*.

[2] Shen, D., & Hellerstein, J. L. (2000). Predictive models for proactive network management: application to a production Web server.

[3] Hu, G., Hu, L., Song, J., Li, P., Che, X., & Li, H. (2010, June). Support vector regression and ant colony optimization for grid resources prediction. *In International Symposium on Neural Networks* (pp. 1-8). Springer, Berlin, Heidelberg.

[4] Andrawis, R. R., & Atiya, A. F. (2009). A new bayesian formulation for holt's exponential smoothing. *Journal of Forecasting*, 28(3), 218-234.

[5] Jr, L. V. (2003). Double Exponential Smoothing: An Alternative to Kalman Filter-Based Predictive Tracking. *Workshop on Virtual Environments*.

[6] Otexts.org. (2019). 7.3 Holt-Winters’ seasonal method | Forecasting: Principles and Practice. [online] Available at: https://www.otexts.org/fpp/7/5 [Accessed 18 Mar. 2019].

[7] Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5), 755-768.
[8] Chen, L., & Shen, H. (2014). Consolidating complementary VMs with spatial/temporal-awareness in cloud datacenters. *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. IEEE.

[9] Tomáš Hanzák, & Tomáš Cipra. (2011). Exponential smoothing for time series with outliers. *Kybernetika -Praha-, 2(2), 165-178.

[10] Hermenier, F., Lorca, X., Menaud, J. M., Muller, G., & Lawall, J. L. (2009). Entropy: a Consolidation Manager for Clusters. *Acm Sigplan/sigops International Conference on Virtual Execution Environments*.

[11] Shaw, P. . (2004). A Constraint for Bin Packing. *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*. DBLP.

[12] Trick, M. A. . (2003). A dynamic programming approach for consistency and propagation for knapsack constraints. *Annals of Operations Research, 118*(1-4), 73-84.

[13] Prud'homme, C. (2019). Choco solver, a CP library. [online] Choco-solver.org. Available at: http://www.choco-solver.org/ [Accessed 18 Mar. 2019].

[14] Farnum, N. R. . (1992). Exponential smoothing: behavior of the exme series with outliers. *Aging SLA Journal of Forecasting*, 11.

[15] Gelper, S., & Croux, C. . (2007). Robust forecasting with holt-winters smoothing.

**Acknowledgements**

This work was supported by Shanghai Science and Technology Commission's Scientific Research Program (#17DZ1204903).