TRIERS: traffic burst oriented adaptive resource provisioning in cloud

Junjie Chen¹, Xiaomin Zhu¹,²,*, Weidong Bao¹, Guanlin Wu¹, Hui Yan¹ and Xiongtao Zhang¹

¹ College of Systems Engineering, National University of Defense Technology, Changsha, Hunan, 410073, China
² State Key Laboratory of High Performance Computing, Beijing, China
* Corresponding author's e-mail: xmzhu@nudt.edu.cn

Abstract. A holistic solution for Traffic Burst Oriented Adaptive Resource Provisioning in Cloud (TRIERS for short) has been proposed to tackle the problem of maintaining performance scalability and service agility in the presence of sudden bursts of workload traffic. The TRIERS framework consists of three core components: workload classification by characterization, traffic prediction in class, and dynamic resource configuration and scheduling. Extensive experiments have been conducted to validate its practicality in terms of guarantee ratio (25.7% improved), resource utilization (5.6% improved) and total energy consumption (17.3% saved), indicating the effectiveness of TRIERS for adaptive resource provisioning in workload traffic burst.

1. Introduction
These Cloud computing has become a dominating service hosting platform for cost-effective and large-scale distributed computing capability in the fields of science, engineering, entertainment and other applications [1]. The cloud-based solutions can offer business agility and economics of scale by amortizing their front-end capital investment and energy cost over a large number of heterogeneous hardware and software resources. However, one of the biggest challenges confronted nowadays is the deficiency in coping with unexpected flood of workloads. This requires significantly larger amount of computing resources to maintain the performance stability, meeting the requirements of business continuity and Service Level Agreements (SLAs) [2]. Some recent research efforts have been dedicated to dynamic resource provisioning [3]. However, the prediction methods deployed in these existing efforts are general and straightforward application of baseline machine learning methods, which may no longer be effective for different domain specific tasks and application scenarios. Moreover, many resource consolidation strategies are lacking in adequate and adaptive reservation beforehand, which may result in high scheduling delay, and possibly serious service interruption [7].

To address this issue, we present a holistic solution for Traffic Burst Oriented Adaptive Resource Provisioning in Cloud (TRIERS). By design, TRIERS presents an integrated burst-handling approach, composed of three core components. First, a clustering-based learning module is employed to characterize the workloads, classifying them into clusters with similar resource demands. The second adopts a compound traffic-oriented prediction method for respective clusters, distinguishing the bursty scenarios from normal ones. In the adaptive resource reservation module, resources are provisioned in accordance with the trade-off between overall resource utilization and runtime performance.
We further conducted extensive experiments using complex traces combined by two popular real world workloads: Google cloud traces [9] and traces of online game World of Warcraft [10], and compared the results of TRIERS with those of another existing representative approach [11] and other variations of the proposed method. Our experimental results show that TRIERS handles workload traffic bursts and it outperforms other existing approaches by 25.7% improved in guarantee ratio, 5.6% improved in resource utilization, and 17.3% saved in total energy consumption.

In summary, the paper makes three original contributions: (1) substituting Mahalanobis distance for Euclidean distance, which eliminates the inter-relationships of varied attributes, thus sharpening its perception of a burst and then improving the accuracy of successive prediction; (2) establishing a mechanism, with a control knob that monitors arriving tasks, to distinguish workload burst and employing different prediction methods in different scenarios accordingly. (3) developing a novel heuristic-based algorithm for resource reservation and provisioning, which supports the aforehand prediction results, thus less resources being wasted in norm and more reserved in burst.

2. System Design

2.1 Architecture Overview

The design overview of TRIERS is presented in Fig. 1, and the three core functional modules are as follows: 1) workload classification module by characterization, 2) trends prediction module in individual classes, and 3) prediction-based adaptive resource reservation and configuration module.

2.2 Before-prediction Characterization

The most influential factors in workload classification is the given computation length and required memory size as well as expected running time of the arriving tasks since they determine the final quality of overall performance to a great extent. For each arriving task \( t_i \) in the task set \( T \), it can be modeled by \( t_i = (a_i, l_i, d_i, m_i) \), where \( a_i \), \( l_i \), \( d_i \) and \( m_i \) represent the arrival time, computation length, expected deadline and memory size of task \( t_i \), respectively.

Then we employ a low-complexity clustering method, \( k \)-means, to group the sampled training tasks at a coarser scale. In this paper, we set four clusters and we assign each task to a cluster with the highest similarity score, which is calculated by Mahalanobis distance. The scale-invariant Mahalanobis distance eliminates the relevance among the three chosen attributes, therefore presenting distinct features of each cluster of tasks. This distance calculation method can be formulated as:

\[
D_M(t_i, t_j) = \sqrt{(t_i - t_j)^T \Sigma^{-1} (t_i - t_j)}, \quad (1)
\]

\[
\Sigma_{ij} = \text{cov}(t_i, t_j) = E[(t_i - \mu)(t_j - \mu)], \quad (2)
\]

where \( \mu \) is the mean value of all the tasks, \( \Sigma_{ij} \) is the covariance value between two tasks.

2.3 Scenario-based Prediction

Aiming at designing the traffic burst prediction model as accurate as possible, we adopt the trend extrapolation prediction method predicting the trend of the incoming workloads when a burst comes and the moving average method forecasting the arrival rate of the incoming tasks based on the statistical properties of historical records at normal stage.

To make this integrated prediction method work more effectively, we further set an acceleration threshold as a control knob to switch on or off under the detection of the state transition between normal scenario and bursty scenario. When the workload acceleration exceeds the preset threshold several times in a row, the prediction method will regard it as an indicator of an arrival burst in the near future. However, if the acceleration is below the threshold value couples of time consecutively, it will signal a normal fluctuation and the prediction mode will be accordingly switched back to moving average method. To sum up, the integrated prediction method can be formulated as in (3).
\[ e_{t+s} = \begin{cases} b_1 \times \exp \left( -\left( \frac{(t + s) - b_2}{b_3} \right)^2 \right) & \text{marked as "BURST"}, \\ \sum_{i=1}^{\text{window}} w_i \times c_{t-i} & \sum_{i=1}^{\text{window}} w_i, \text{marked as "NORM"} \end{cases} \]

Two methods are applied in two different scenarios based on the real-time task count change. Therein, \( t \) represents the current time point, \( s \) represents the prediction time span, \( c_t \) is the current task count, \( e_{t+s} \) is the estimated task count at the \((t + s)\)-th second, \( b_1 \), \( b_2 \) and \( b_3 \) are all characteristic parameters of the fitting curve in the trend extrapolation method, \text{window} is the number of tasks from history for calculation, and \( w_i \) represents the weight of the \( i \)-th task in moving average method.

### 2.4 Prediction-based Reservation Strategy

In this strategy, we first target a virtualized cloud consisting of \( m \) physical hosts or PMs, and each PM is characterized by \( h_j = (r_j, o_j, f_j) \), where \( r_j \) is the memory size of the \( j \)-th PM \( h_j \), \( f_j \) is the CPU value of the \( j \)-th PM \( h_j \) and \( o_j \) is the maximum number of virtual machines in the \( j \)-th host. The \( k \)-th VM that is set on the host \( h_j \) can be modeled as \( v_{m_{j,k}} = (f_{j,k}, o_{j,k}) \), where \( f_{j,k} \) and \( r_{j,k} \) are respectively the CPU performance and memory required for \( v_{m_{j,k}} \). Then, we set the capacity constraints as follows:

\[ f_j^{\max} - \sum_{k=1}^{o_j} f_{j,k} \geq 0, \forall h_j \in H, \]

\[ m_j - \sum_{k=1}^{o_j} m_{j,k} \geq 0, \forall h_j \in H. \]

With these constraints satisfied, we hope to strike a balance between energy consumption saving and task accomplishment, and we propose a heuristic algorithm for this reservation strategy. Of that, the feedback of performance of current state will be returned to the system to make corresponding adjustments. In addition, a redeployment functional algorithm reserves and redeploy available resources to cope with coming bursts, and function allocation allocates coming tasks to compatible VMs in terms of their characteristic attributes and heterogeneity.

### 3. Experimental Evaluation

In this section, we first present our experimental results of the TRIERS algorithms under different settings and then compare the method with an existing representative approach and several other versions of itself to further demonstrate the effectiveness in terms of performance improvement.

#### 3.1 Result Before-prediction Characterization

For the test of this module, we sampled a segment of the trace as training data, which contains several workload spikes. Next, we calculated a representative covariation matrix and group the training tasks to four clusters, obtaining their features. Figures below demonstrate the results of our proposed algorithm.
Figure 2: Comparisons of Traits in Clusters Grouped by Two Different Distances.

Fig. 2 compares the tasks attributes grouped in the four clusters using different distance score computing methods, the attributes including computation lengths, deadlines required and memory sizes demanded. Figures on the left side show the results from Euclidean-distance computation, while the rest on the right-side show those from Mahalanobis-distance computation. We take the pair of Fig. 2(a) and Fig. 2(b) for an example. A significant portion of the tasks are marked with red color, which indicates bursty scenario, while other colors are mostly marked when the total of tasks are not spiraling. This indicates that tasks in the cluster marked as red are likely to occur when traffic burst in workload occurs and tasks marked as other colors have greater possibilities arriving at normal scenario when the task count does not fluctuate. The two highlighted blocks in the two figures illustrate that the Mahalanobis computation marks more tasks as “bursty” than Euclidean does, and thus the former one in a way prolongs the bursty scenario. Similar implication can be inferred in the other pairs.

3.2 Scenario-based Prediction.

Fig. 3 demonstrates the comparison between the prediction results and the real data of the four clusters. The blue curve represents the real value of task count for each cluster and the red curve represents the predicted value. As regards to the other three clusters, they barely have no prominent spiky bursts detected, thus as shown in Fig. 3(b), Fig. 2(c) and Fig. 2(d), their prediction method is not shifted to trend extrapolation method throughout the test. This finding indicates that the pre-grouping step manages in separating the characteristic “bursty tasks” from those in norm. Moreover, the “bursty tasks” also have a characteristic arriving curve, and if the controlling knob is set accordingly and appropriately, the prediction method will predict a more formfitting prediction curve.

3.3 Prediction-based Reservation Strategy.

We first choose another approach proposed in [11] for comparison, referred as RF (reservation-free). Then we excerpt and generate two “partial” versions of TRIERS, which represent a reservation solution without task characterization module, referred as GF (grouping-free) and a reservation strategy without workload burst consideration, PF (prediction-free). The purpose of comparing the full-fledged solution of TRIERS with these methods is to show that each module of our solution approach matters. We consider three metrics to evaluate our system’s performance, which are the guarantee ratio, resource utilization and total energy consumption.

The guarantee ratio in (6) represents the tasks accomplishment rate, which is also the primary objective of our approach.
\[ GR = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{o_i} \frac{x_{i,j,k}}{n} \]  \hspace{1cm} (6)

Of them, \( x_{i,j,k} \) represents whether the assigned task \( t_i \) have been accomplished in the \( k \)-th VM of the \( j \)-th PM. By design, a “accomplished” task indicates being done before its deadline. Experimental results of this metric are presented in Fig. 4.

\[ RU = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{o_j} y_{i,j,k} \cdot l_i / (\sum_{j=1}^{m} f_j \cdot w_{t_j})}{f_{t_j}} \]  \hspace{1cm} (7)

Of them, \( y_{i,j,k} \) represents whether the assigned task \( t_i \) have been finished in the \( k \)-th VM of the \( j \)-th PM. By designation, a “finished” task indicates literally being finished regardless of its running time. In addition, \( w_{t_j} \) represents the active time of PM \( h_j \). Experimental results of this metric are presented in Fig. 5.

\[ p_j^{active} \propto f_j^3 \]  \hspace{1cm} (8)
\[ t_{ecj} = \int_{st}^{et} \left( s_j \cdot p_j^{\text{max}} \cdot c_j^T + \frac{(1 - s_j) \cdot p_j^{\text{max}}}{(f_j^{\text{max}})^3} \cdot (f_j)^3 \right) dt \]  

(9)

4. Conclusions
In summary, TRIERS is a light-weight and non-intrusive approach, which can predict incoming workload burst by combining the moving average prediction method for normal workload scenario and the trend extrapolation prediction method for workload burst scenario, and thus trigger the refinement of resource reservation strategies. Extensive experiments were conducted to validate the effectiveness of TRIERS for workload traffic bursts adaptive resource provisioning in cloud.

Acknowledgments
This research was partially supported by the National Natural Science Foundation of China under Grants No. 61872378, No. 61572511 and No. 91648204, also partially by the China Postdoctoral Science Foundation under Grant 2016M602960 and Grant 2017T100796, and partially by Science Fund for Distinguished Young Scholars in Hunan Province under Grant 2018JJ1032.

References
[1] Mell, P., Grance, T. (2011) The nist definition of cloud computing. Commun. ACM, 53(6): 50–50.
[2] (2013) Amazon elastic computing cloud. http://aws.amazon.com/ec2.
[3] Piraghaj, S. F., Dastjerdi, A. V., Calheiros, R. N., Buyya, R. (2015) Efficient virtual machine sizing for hosting containers as a service. In Proceeding of IEEE Services. pp. 31–38.
[4] Zhang, Q., Boutaba, R. (2014) Dynamic workload management in heterogeneous cloud computing environments. In: IEEE Network Operations and Management Symposium (NOMS). pp. 1–7.
[5] Powers, R., Goldszmidt, M., Cohen, I. (2005) Short term performance forecasting in enterprise systems. In: ACM SIGKDD. pp. 801–807.
[6] Shen, D., Hellerstein, J. L. (2000) Predictive models for proactive network management: Application to a production web server. In: IEEE/IFIP Network Operations and Management Symposium. pp. 833–846.
[7] Sharma, B., Chudnovsky, V., Hellerstein, J. L., Rifaat, R., Das, C. R. (2011) Modeling and synthesizing task placement constraints in google compute clusters. In: ACM Symposium on Cloud Computing (SoCC). pp. 3-3.
[8] Zhang, Q., Hellerstein, J. L., Boutaba, R., et al. (2011) Characterizing task usage shapes in goolges compute clusters. In: Large-Scale Distributed Systems and Middleware (LADIS). pp. 1–6.
[9] Google cluster data v2. https://code.google.com/p/googleclusterdata/wiki/.
[10] Lee, Y. T., Chen, K. T., Cheng, Y. M., Lei, C. L. (2011) World of warcraft avatar history dataset. In: ACM Multimedia Systems Conference (MMSys). pp. 123–128.
[11] Lu, H., Chen, H., Ma, S., Dai, W., Xing., P. (2014) Dynamic virtual resource management in clouds coping with traffic burst. In: IEEE Services Computing (SCC). pp. 590–596.