

# Toward Resource-Efficient Cloud Systems: Avoiding Over-Provisioning in Demand-Prediction Based Resource Provisioning

Liuhua Chen

*ECE, Clemson University*

Haiying Shen

*CS, University of Virginia*

2016 IEEE International Conference on Big Data

Presenter: Yi-Ning Chang

October 19, 2017

# Outline

- 1 Introduction
- 2 System Design
- 3 Evaluation
- 4 Conclusion

## 1 Introduction

## 2 System Design

## 3 Evaluation

## 4 Conclusion

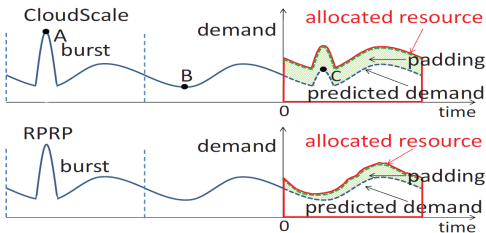
# Introduction

- In cloud systems, cloud providers abstract resources in physical machines into virtual machines and sell them to the tenants.
- To ensure resource provisioning for guaranteeing SLOs<sup>1</sup>, clouds can use *demand-prediction based resource provisioning schemes*.
- Achieving the tradeoff between the penalties associated with *SLO violations* and *high resource utilization* requires an accurate demand prediction methodology.

---

<sup>1</sup>SLO: Service Level Objectives

# Previous Work - CloudScale



- CloudScale predicts the demand at a time period based on a historical record.
- Padding: using the high-frequency spectrum or the average of the latest prediction error.
- Online Adaptive: to handle underestimation, raising the resource allocation by  $\alpha > 1$  until an error is corrected.

# RPRP<sup>1</sup>

- RPRP excludes bursts in demand prediction and specifically handles bursts to avoid resource over-provisioning.
- Algorithm
  - *burst-exclusive prediction algorithm*
  - *load-dependent padding algorithm*
  - *responsive padding algorithm*
- Algorithm 1 and 2 aim to exclude bursts, and algorithm 3 aims to handle bursts.

---

<sup>1</sup>RPRP: Resource-efficient Predictive Resource Provisioning system

- 1 Introduction
- 2 System Design
- 3 Evaluation
- 4 Conclusion

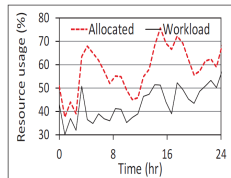
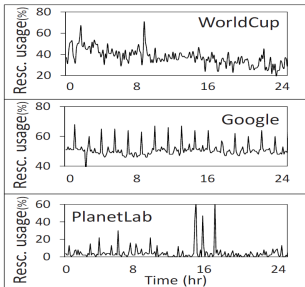
# Objective

- Denote a VM's records:
  - *workload demand*:  $D = \{d_{t_1}, \dots, d_{t_i}, \dots, d_{t_N}\}$
  - *allocated resource*:  $A = \{a_{t_1}, \dots, a_{t_i}, \dots, a_{t_N}\}$
  - *utilized resource*:  $U = \{u_{t_1}, \dots, u_{t_i}, \dots, u_{t_N}\}$
  - *resource capacity*:  $C$
- And from the historical records, we have:
  - *predict demand*:  $P = \{p_{t_{N+1}}, p_{t_{N+2}}, \dots, p_{t_{N+T}}\}$
  - *allocated resource*:  $A = \{a_{t_{N+1}}, a_{t_{N+2}}, \dots, a_{t_{N+T}}\}$
- Goal: determine allocated resource  $A$  such that
  - $d_{t_i} \leq a_{t_i} \leq C$
  - and meanwhile to minimize  $a_{t_i} - d_{t_i}, \forall t_i > t_N$

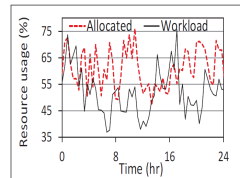


# Algo.1: Burst-exclusive Prediction

- Trace analysis and CloudScale prediction + padding.



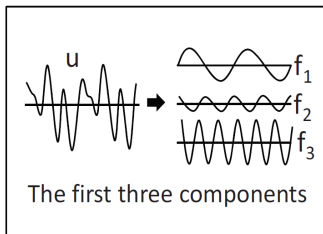
(a) Low burst density.



(b) High burst density.

# Algo.1: Burst-exclusive Prediction

- RPRP relies on FFT to exclude the burst.
- FFT is applicable for predicting workload demand in repeated periodic patterns  $P$  based on the historical utilization series  $U$ .



## Algo.2: Load-dependent Padding

- Formulate the problem of padding determination to achieve both *resource efficient* and *SLO guarantee*.
- The variation of prediction errors is dependent on load levels in cloud.
- Use  $\mathcal{P} = \{\hat{p}_1, \hat{p}_2, \dots, \hat{p}_M\}$  ( $\hat{p}_1 < \hat{p}_2 < \dots < \hat{p}_M$ ) to represent the M different predicted demand levels.
- Use  $\mathcal{D}_{\hat{p}_i} = \{d_1, d_2, \dots, d_{n_{\hat{p}_i}}\}$  ( $d_1 < d_2 < \dots < d_{n_{\hat{p}_i}}$ ,  $n_{\hat{p}_i} = |\mathcal{D}_{\hat{p}_i}|$ ) to indicate the demands that were predicted to be  $\hat{p}_i$ .
- And  $N = \sum_{j=1}^M n_{\hat{p}_j}$  is the total number of workload demands in the demand series.

## Algo.2: Load-dependent Padding

- The probability that the allocated resource ( $a_{t_i} = \hat{p}_i + \delta(\hat{p}_i)$ ) is sufficient to meet the demand is

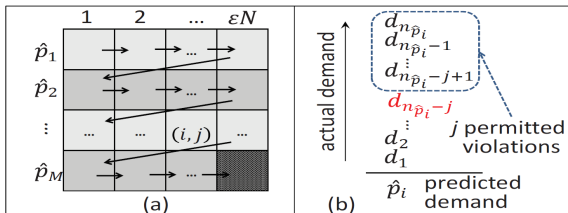
$$Pr(\hat{p}_i) = \frac{|\{d_j \leq \hat{p}_i + \delta(\hat{p}_i) \mid d_j \in \mathcal{D}_{\hat{p}_i}\}|}{n_{\hat{p}_i}} \quad (1)$$

$$\bar{P}r = \sum_{i=1}^M Pr(\hat{p}_i) \frac{n_{\hat{p}_i}}{N} = \sum_{i=1}^M \frac{|\{d_j \leq \hat{p}_i + \delta(\hat{p}_i) \mid d_j \in \mathcal{D}_{\hat{p}_i}\}|}{n_{\hat{p}_i}} \frac{n_{\hat{p}_i}}{N} \geq 1 - \epsilon \quad (2)$$

- The expected allocated resource amount can be calculated by

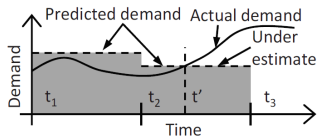
$$\sum_{\hat{p}_i \in \mathcal{P}} [\hat{p}_i + \delta(\hat{p}_i)] \frac{n_{\hat{p}_i}}{N} \quad (3)$$

# Algo.2: Load-dependent Padding



- Solved by an  $M \times \epsilon N$  dynamics programming.
- In the matrix,  $\mathbb{M}(i, j)$  represents the minimum total allocated resource when distributing  $j$  violations to the first  $i$  predicted demand levels.
- $$\mathbb{M}(i, j) = \min_{0 < x < j} \{ \mathbb{M}(i-1, j-1) + d_{n_{\hat{p}_i-x}} \times n_{\hat{p}_i} \}$$

## Algo.3: Responsive Padding



- If the resource utilization is upper than  $T_u$  at time  $t'$ , then

$$a_{t'+\Delta} = a_{t'} + \frac{1}{2}(u_{max} - a_{t'}) \quad (4)$$

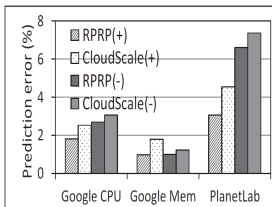
- If the resource utilization is lower than  $T_l$  after raising, then

$$a_{t''+\Delta} = a_{t''} - \frac{1}{2}(a_{t''} - u_{t''}) \quad (5)$$

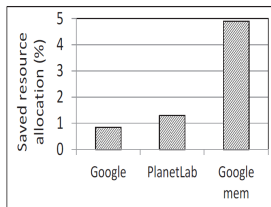
- $T_l$ ,  $T_u$  are lower and upper bound threshold.
- $u_{max}$  is the maximum recorded utilization.
- $\Delta$  is a monitoring interval.

- 1 Introduction
- 2 System Design
- 3 Evaluation**
- 4 Conclusion

# Performance of Burst-exclusive Prediction



(a) Prediction error.

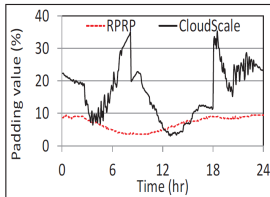


(b) Saved resource allocation.

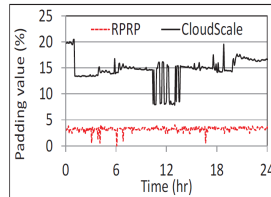
- Average prediction error is calculated by  $\frac{1}{n} \sum_{i=1}^n |\hat{p}_i - d_i|$
- Saved resource allocation is calculated by  $\frac{CloudScale - RPRP}{CloudScale}$



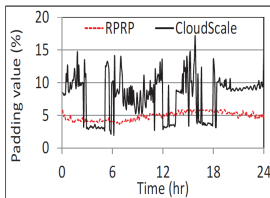
# Performance of Load-dependent Padding



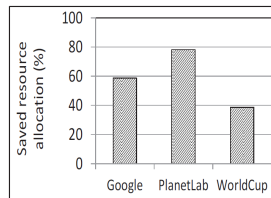
(a) Google Cluster trace



(b) PlanetLab trace

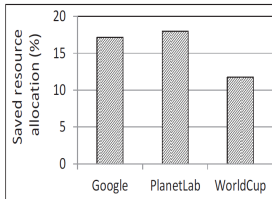


(c) WorldCup trace

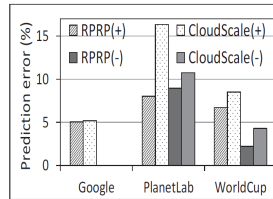


(d) Saved resource allocation

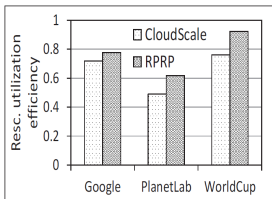
# Performance of Resource Provisioning



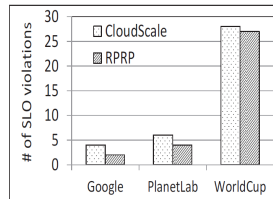
(a) Saved resource allocation



(b) Prediction error

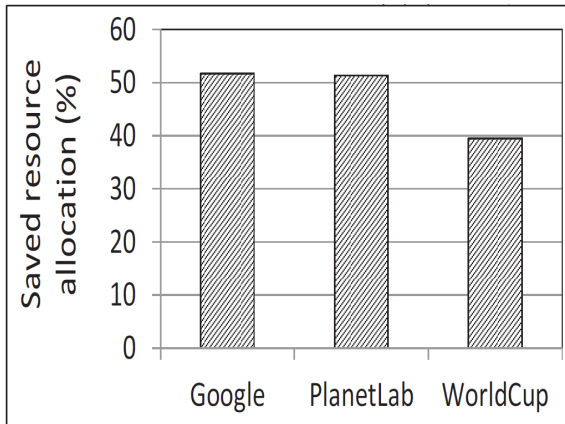


(c) Resource utilization efficiency

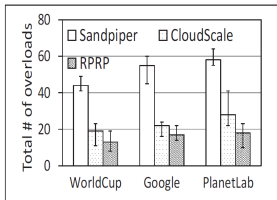


(d) The number of SLO violations

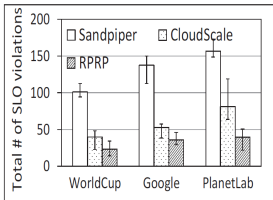
# Performance of Responsive Padding



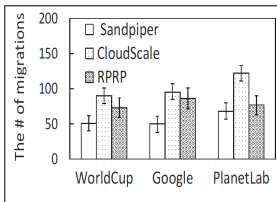
# Trace-driven Simulation



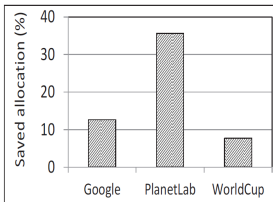
(a) Total number of overload PMs



(b) Total number of SLO violations

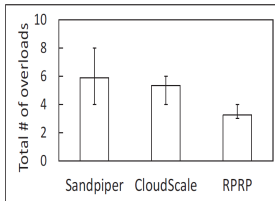


(c) The number of migrations

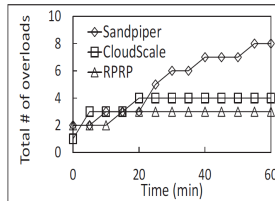


(d) Saved resource allocation

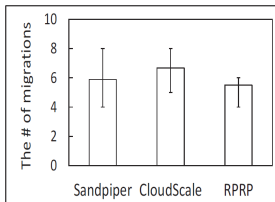
# Real-World Testbed Experiments



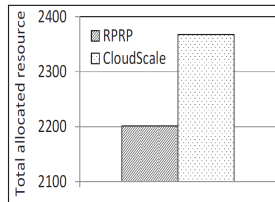
(a) The number of overloads



(b) Cumulated number of overloads



(c) The number of migrations



(d) Total allocated resource

- 1 Introduction
- 2 System Design
- 3 Evaluation
- 4 Conclusion

# Conclusion

- Experimental results show that by using algo. 1 and 2, RPRP reduces 18% of the allocated resource while reducing 30% of the total number of SLO violations compared to CloudScale.
- These reduced padding values are substantial in improving the resource utilization of a PM that hosts multiple VMs.
- RPRP achieves higher resource utilization, more accurate demand prediction, and fewer SLO violations than previous schemes.

# Future Work

- Use the technic mentioned in the paper and the position prediction to make a more accurate prediction in social VR.
- Read the referenced paper to research more about the resource scaling method in cloud system.