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

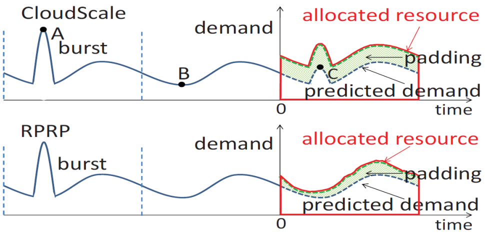
Outline

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[1], 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.

[1] 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[1]

- 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.

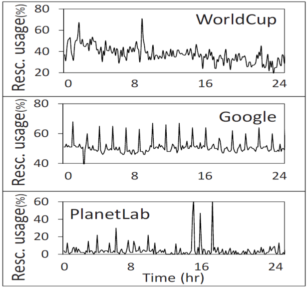
[1] RPRP: Resource-efficient Predictive Resource Provisioning system

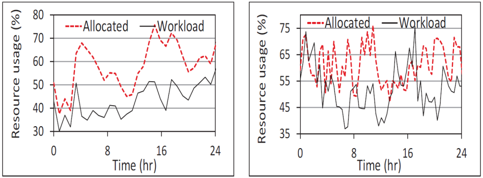
Objective

- Denote a VM's records:
  - *workload demand*: $D = \{d_{t_1}, ..., d_{t_i}, ..., d_{t_N}\}$
  - *allocated resource*: $A = \{a_{t_1}, ..., a_{t_i}, ..., a_{t_N}\}$
  - *utilized resource*: $U = \{u_{t_1}, ..., u_{t_i}, ..., u_{t_N}\}$
  - *resource capacity*: C
- And from the historical records, we have:
  - *predict demand*: $P = \{p_{t_{N+1}}, p_{t_{N+2}}, ..., p_{t_{N+T}}\}$
  - *allocated resource*: $A = \{a_{t_{N+1}}, a_{t_{N+2}}, ..., a_{t_{N+T}}\}$
- 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.

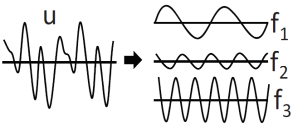




(a) Low burst density. (b) High burst density.

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.



The first three components

Algo.2: Load-dependent Padding

- The variation of prediction errors is dependent on load levels in cloud.
- Formulate the problem of padding determination to achieve both *resource efficiency* and *SLO guarantee*.
- Use $\mathscr{P} = \{\hat{p_1}, \hat{p_2}, ..., \hat{p_M}\}$ $(\hat{p_1} < \hat{p_2} < ... < \hat{p_M})$ to represent the M different predicted demand levels.
- Use $\mathscr{D}_{\hat{P_i}} = \{d_1, d_2, ..., d_{n_{\hat{p_i}}}\}$ $(d_1 < d_2 < ... < d_{n_{\hat{p_i}}}, n_{\hat{p_i}} =\mid \mathscr{D}_{\hat{p_i}} \mid)$ 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

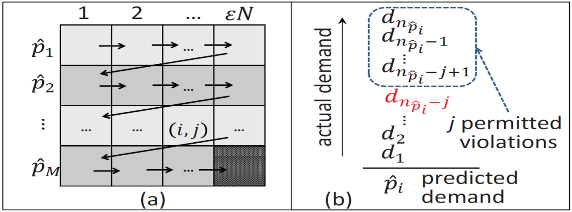
$$Pr(\hat{p}_i) = \frac{|\{d_j \leq \hat{p}_i + \delta(\hat{p}_i) \mid d_j \in \mathscr{D}_{\hat{p}_i}\}|}{n_{\hat{p}_i}} \tag{1}$$

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

- The expected allocated resource amount can be calculated by

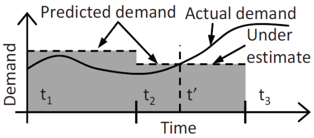
$$\sum_{\hat{p}_i \in \mathscr{P}} [\hat{p}_i + \delta(\hat{p}_i)] \frac{n_{\hat{p}_i}}{N} \tag{3}$$

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.
- $\mathbb{M}(i, j) = min_{0 \leq x \leq j}\{\mathbb{M}(i-1, j-x) + d_{n_{\hat{p}_i}-x} \times n_{\hat{p}_i}\}$

Algo.3: Responsive Padding



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

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

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

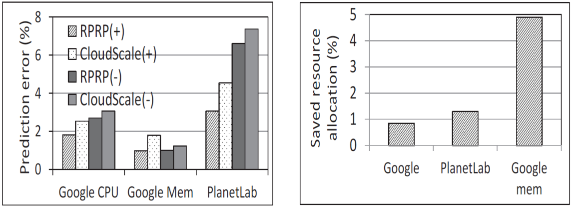
$$a_{t''+\Delta} = a_{t''} - \frac{1}{2}(a_{t''} - u_{t''}) \tag{5}$$

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

Analytical Performance Evaluation

- Use 48-hours history utilization data from each trace to predict the resource demand at every 5 minutes in the next 24 hours.
- Performance evaluation
  - Burst-exclusive Prediction
  - Load-dependent Padding
  - Resource Provisioning
  - Responsive Padding

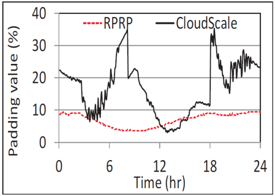
Performance of Burst-exclusive Prediction



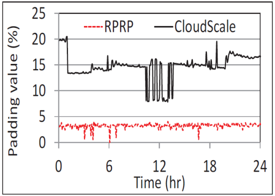
(a) Prediction error. (b) Saved resource allocation.

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

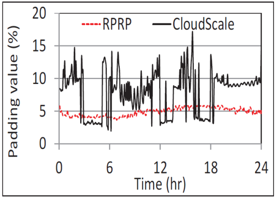
Performance of Load-dependent Padding



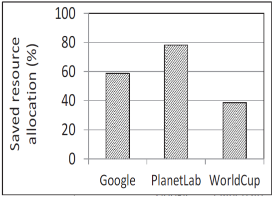
(a) Google Cluster trace



(b) PlanetLab trace

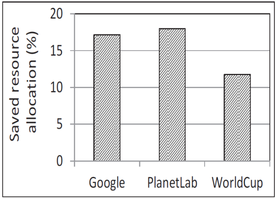


(c) WorldCup trace

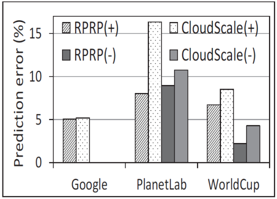


(d) Saved resource allocation

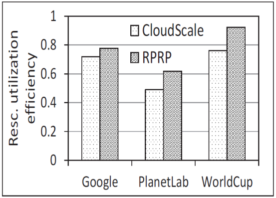
Performance of Resource Provisioning



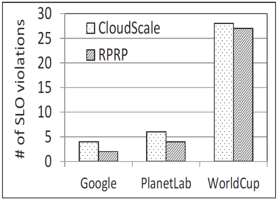
(a) Saved resource allocation



(b) Prediction error



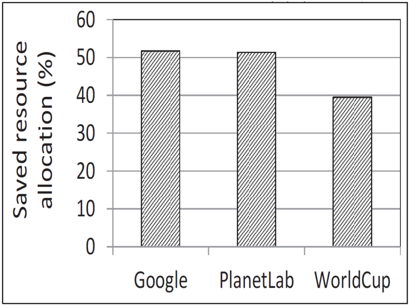
(c) Resource utilization efficiency



(d) The number of SLO violations

Performance of Responsive Padding

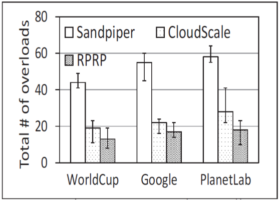
- RPRP's responsive padding algorithm can scale the resource cap up and down.
- CloudScale scales the resource cap to and stays at a high level.



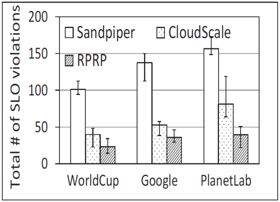
Trace-driven Simulation

- Evaluate the performance of *RPRP* in comparison with *SandPiper* and *CloudScale* on the *CloudSim* simulator.
- *SandPiper* conducts VM migration from overloaded PMs based on the current VM loads.
- Both *RPRP* and *CloudScale* employ resource demand prediction and conduct VM migration from PMs that are predicted to be overloaded.
- In the simulation, allocate *2000* VMs to *1000* PMs.

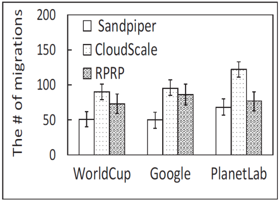
Trace-driven Simulation



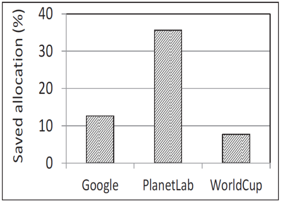
(a) Total number of overload PMs



(b) Total number of SLO violations



(c) The number of migrations

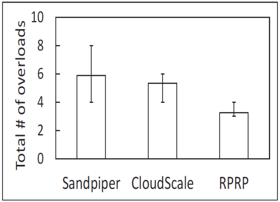


(d) Saved resource allocation

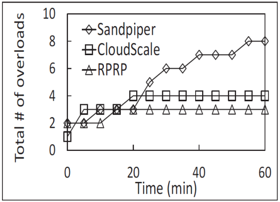
Real-World Testbed Experiments

- Use the experiment in real-world testbed to verify the simulation results.
- In the experiment, allocate *11* VMs to *5* PMs.

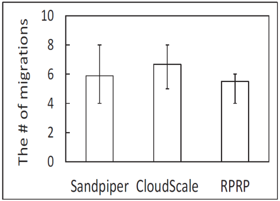
Real-World Testbed Experiments



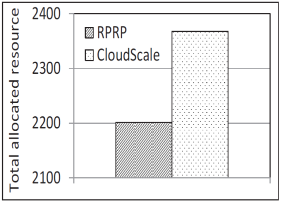
(a) The number of overloads



(b) Cumulated number of overloads



(c) The number of migrations



(d) Total allocated resource

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.
- RPRP achives 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.