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

Authors: Liuhua Chen and Haiying Shen

Presenter: Haiying Shen

IEEE BigData

Washington D.C., USA

December 2016

Object

To ensure resource provisioning for guaranteeing service level objectives (SLOs).

Demand-prediction based resource provisioning schemes

- Avoid over-provisioning

- Avoid under-provisioning

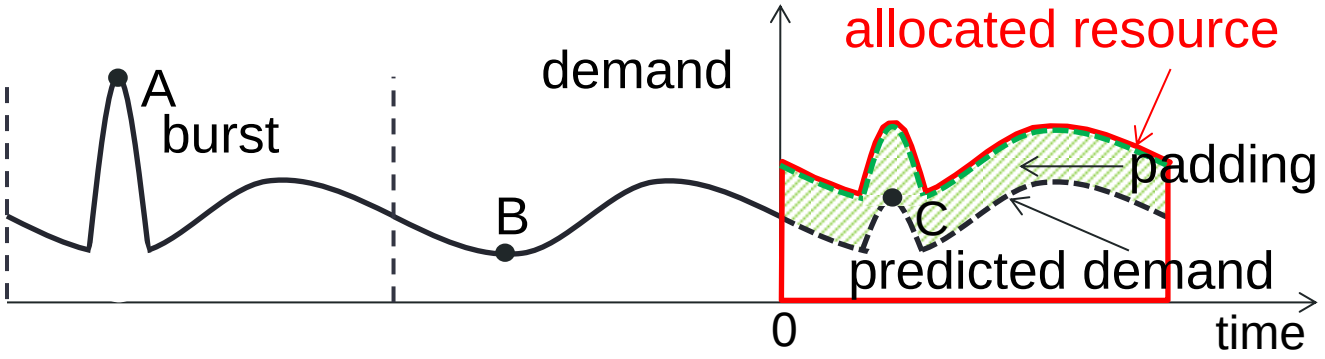
Overview

- A brief review of current approaches
- Our approach: Resource-efficient Predictive Resource Provisioning system in clouds (RPRP)
    1) Burst-exclusive prediction
    2) Load-dependent padding
    3) Responsive padding
- Experimental results
- Conclusion with future directions

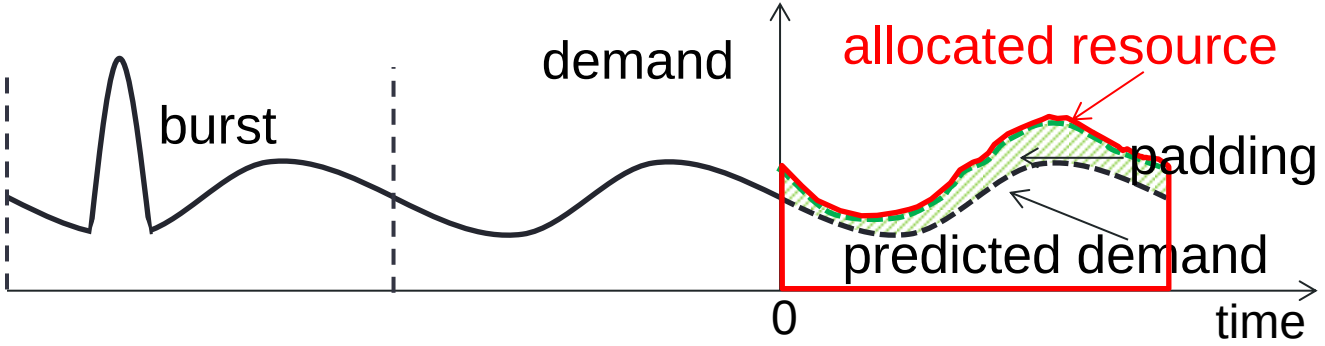
Current approach

CloudScale

not exclude bursts

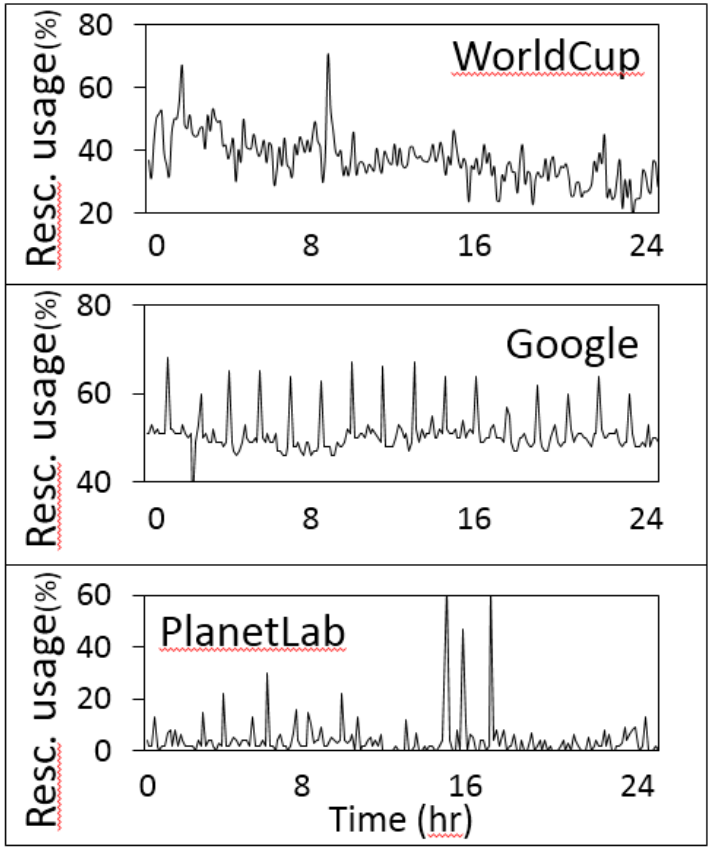


RPRP

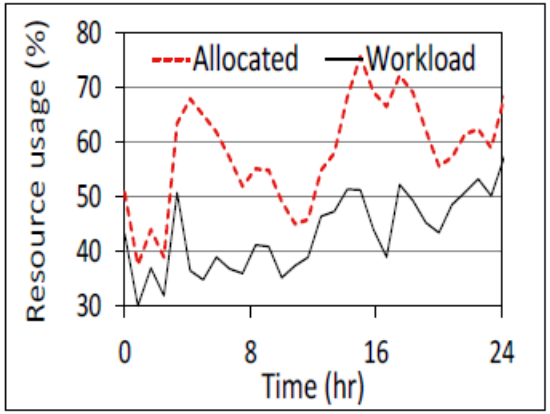


Burst-Exclusive Prediction

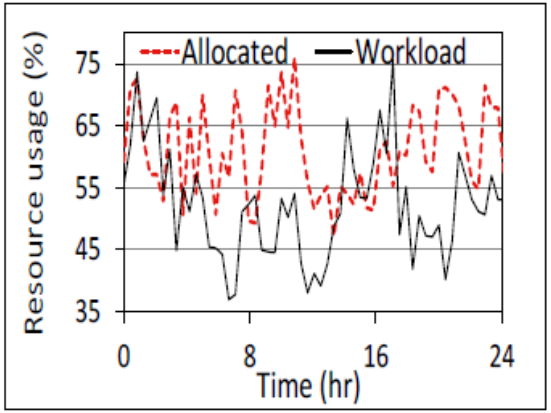
- Trace analysis
  - To confirm the prevalence of bursts.
  - To find whether CloudScale has relatively low resource utilization.



Resource usage bursts.



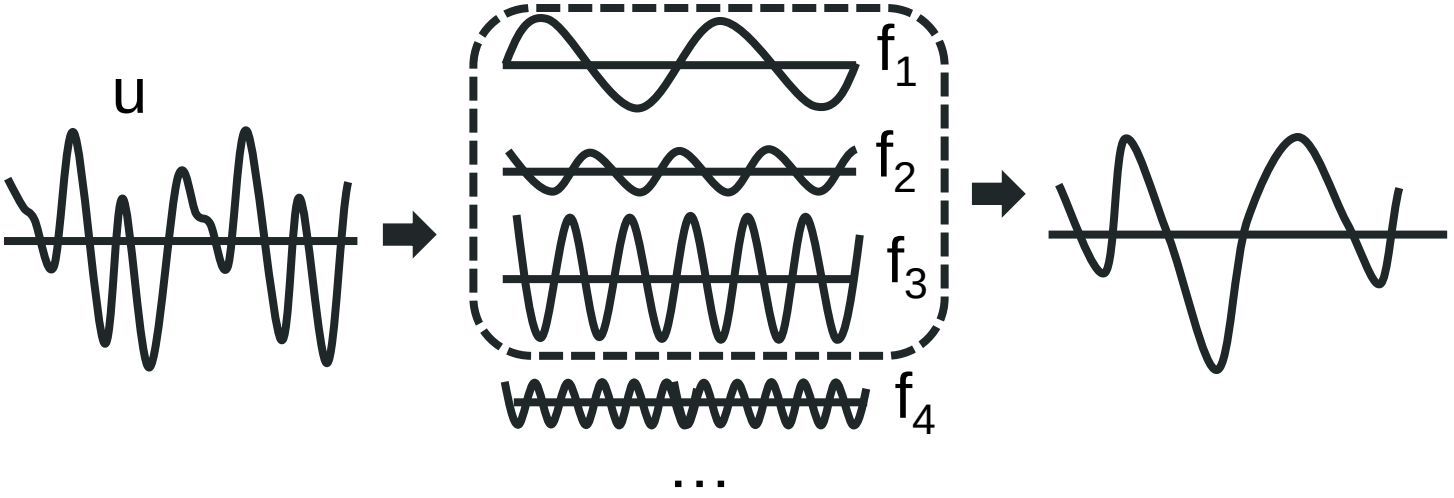
(a) Low burst density.



(b) High burst density.

Burst-Exclusive Prediction (cont.)

- Predicts demands based on history records.
  -- Fast Fourier Transform (FFT)



Load-dependent Padding

- Given

  (1) the probability distribution of the predicted demand levels $\hat{p}_j$,

  (2) the probability distribution of the actual demands for each level (i.e., $\hat{p}_j$) and

  (3) allowed violation rate ε from SLO,

  how can we determine the padding value $\delta(\hat{p}_j)$ for each level $\hat{p}_j$ to

  (a) achieve $\overline{P_r} \geq 1 - \varepsilon$

  (probability that the allocated resource is sufficient) and meanwhile

  (b) minimize the expected total allocated resource.

Load-dependent Padding (cont.)

Historical predicted and actual demand series during a time period $[t_1, t_N]$.

M different predicted demand levels:

$$\mathscr{D}_{\hat{p}_i} = \{d_1,\ d_2,\ \ldots,\ d_{n_{\hat{p}_i}}\}$$

N is the total number of workload demands in the demand series.
$n_{\hat{p}_i}$ is the number of workload demands in demand level $\hat{p}_i$

$$N = \sum_{j=1}^{M} n_{\hat{p}_j}$$

Load-dependent Padding (cont.)

Padding value for $\hat{p}_i$: $\delta(\hat{p}_i)$

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

$$Pr(\hat{p}_i) = \frac{|\{d_j \le \hat{p}_i + \delta(\hat{p}_i) | d_j \in \mathscr{D}_{\hat{p}_i}\}|}{n_{\hat{p}_i}}.$$

The expected probability that the allocated resource is sufficient ($\overline{P_r}$) is

$$\overline{Pr} = \sum_{i=1}^{M} Pr(\hat{p}_i)\frac{n_{\hat{p}_i}}{N} = \sum_{i=1}^{M} \frac{|\{d_j \le \hat{p}_i + \delta(\hat{p}_i) | d_j \in \mathscr{D}_{\hat{p}_i}\}|}{n_{\hat{p}_i}} \frac{n_{\hat{p}_i}}{N}$$

The expected allocated resource amount can be calculated by

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

Load-dependent Padding (cont.)

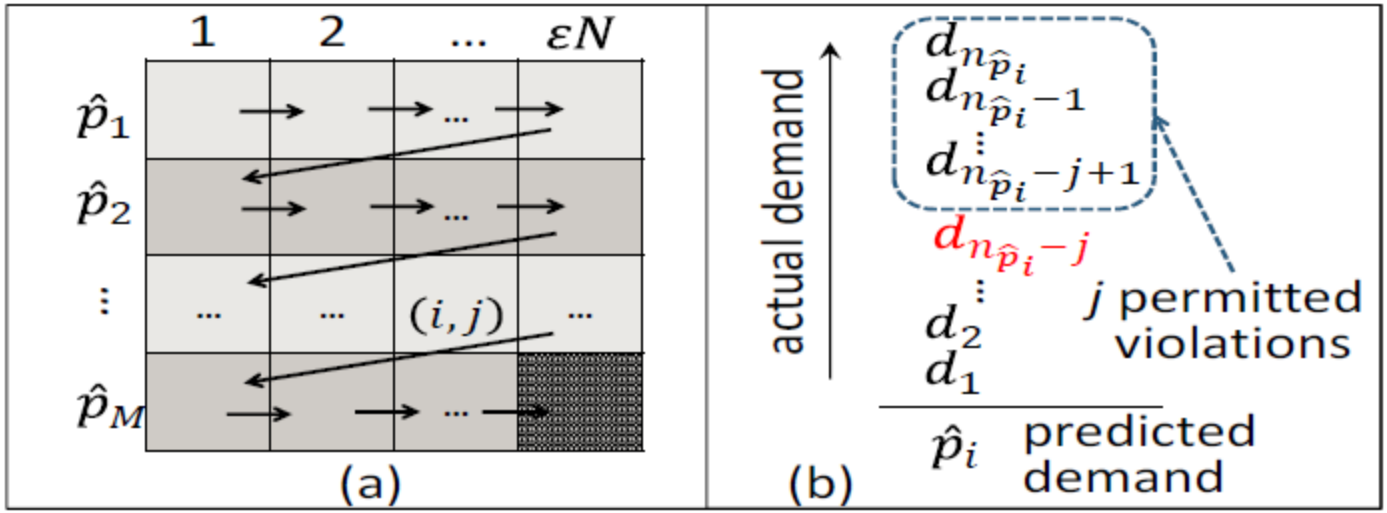


Figure 6: Dynamic programming algorithm. (a) an $M \times \varepsilon N$ dynamic programming matrix $\mathbb{M}$. (b) procedure to determine the allocated resource if we place j permitted violations on predicted demand level $\hat{p}_i$.

Responsive Padding

- Keeps the resource utilization efficiency while satisfying SLO dynamically.

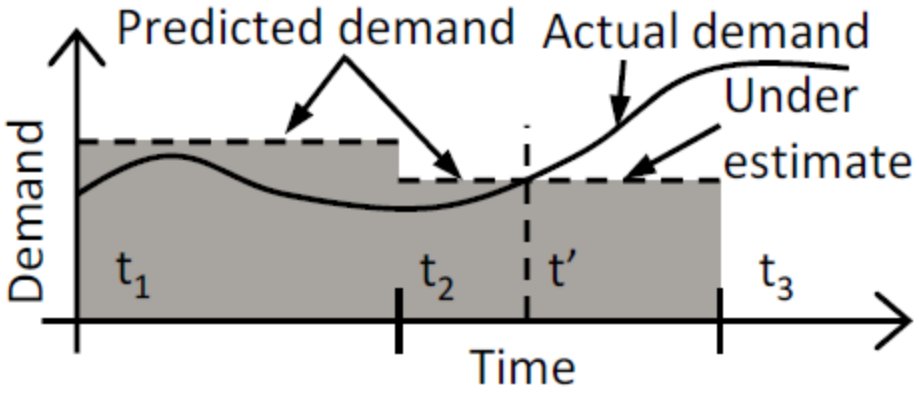


Figure 7: Underestimate correction. Demand prediction and resource allocation are performed at time t_1, t_2 and t_3. Responsive padding is performed at time t' where the allocated resource becomes insufficient for the demand before next prediction and allocation.

Responsive Padding (cont.)

- If underestimation is flagged (resource utilization efficiency > T_u) at time t', then

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

- If the resource utilization efficiency is lower than T_l after raising (t''), then

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

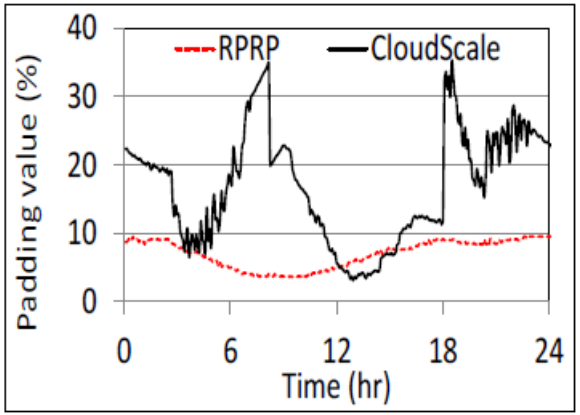
Experimental results

- Analytical Performance Evaluation
  - Evaluate algorithm based on traces
- Trace-driven Simulation
  - CloudSim simulation toolkit
  - 1000 PMs
  - 2000 VMs
- Real testbed
  - Based on XenAPI
  - 5 PMs (2.00GHz Intel(R) Core(TM)2 CPU, 2GB memory, 60GB HDD)
  - 11 VMs (1VCPU, 256MB memory, 8.0GB virtual disk, running Debian Squeeze 6.0)

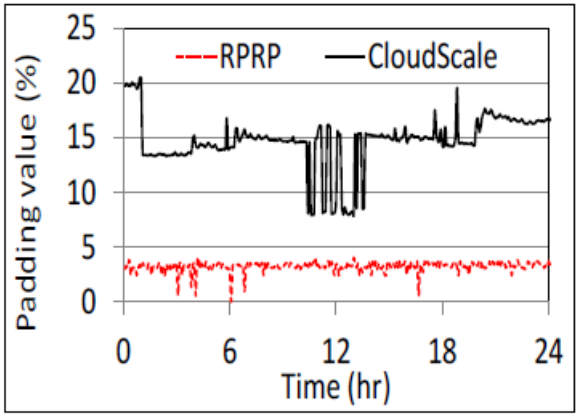
Our algorithm: RPRP

Comparison algorithms: Sandpiper, CloudScale

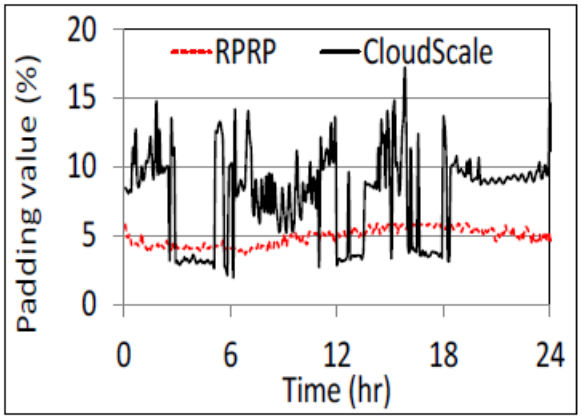
Analytical Performance Evaluation



(a) Google Cluster trace



(b) PlanetLab trace

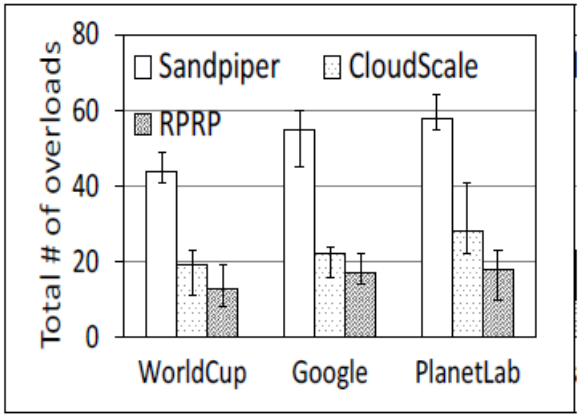


(c) WorldCup trace

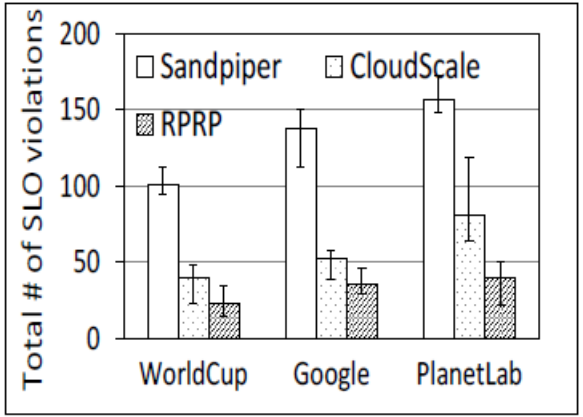
Performance of the load-dependent padding algorithm

Result: Lower padding while satisfying SLO.

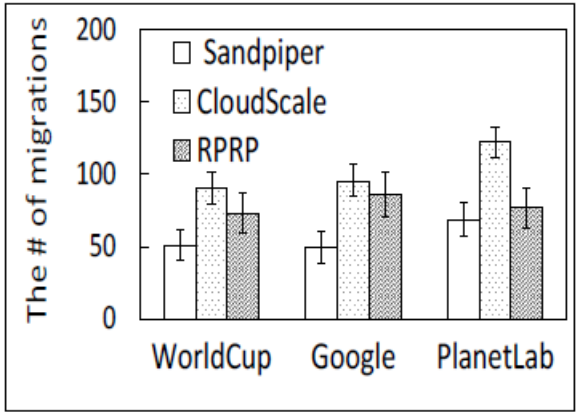
Trace-driven Simulation



(a) Total number of overload PMs



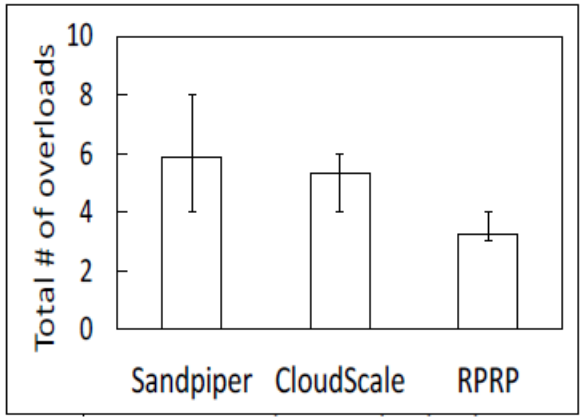
(b) Total number of SLO violations



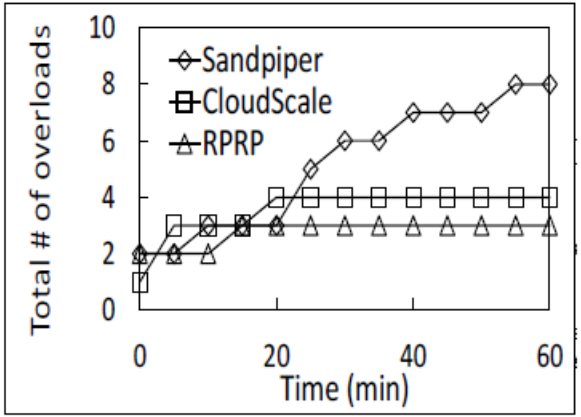
(c) The number of migrations

Result: Lower number of overloads, lower number of SLO violations, fewer number of migrations.

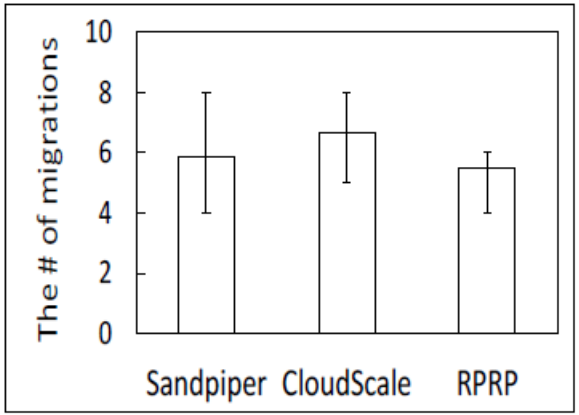
Real-World Testbed Experiments



(a) The number of overloads



(b) Cumulated number of overloads



(c) The number of migrations

Result: Confirm trace-driven simulation results.

Conclusion and future work

- In this paper, we proposed RPRP, which consists of three algorithms:
  - 1) a burst-exclusive prediction algorithm,
  - 2) a load-dependent padding algorithm,
  - 3) a responsive padding algorithm.

- We conducted extensive experiments to show the effectiveness of the proposed algorithms and that RPRP achieves higher resource utilization, more accurate demand prediction, and fewer SLO violations than previous schemes.

Future work: extend RPRP to deal with VMs with various priorities besides CPU and memory.

Thank you!
Questions & Comments?

Liuhua Chen, Ph.D.

liuhuac@clemson.edu

Pervasive Communication Laboratory

Clemson University