# Probabilistic Demand Allocation for Cloud Service Brokerage

Chenxi Qiu, Haiying Shen and Liuhua Chen

Holcombe Department of Electrical and Computer Engineering

CLEMSON
UNIVERSITY

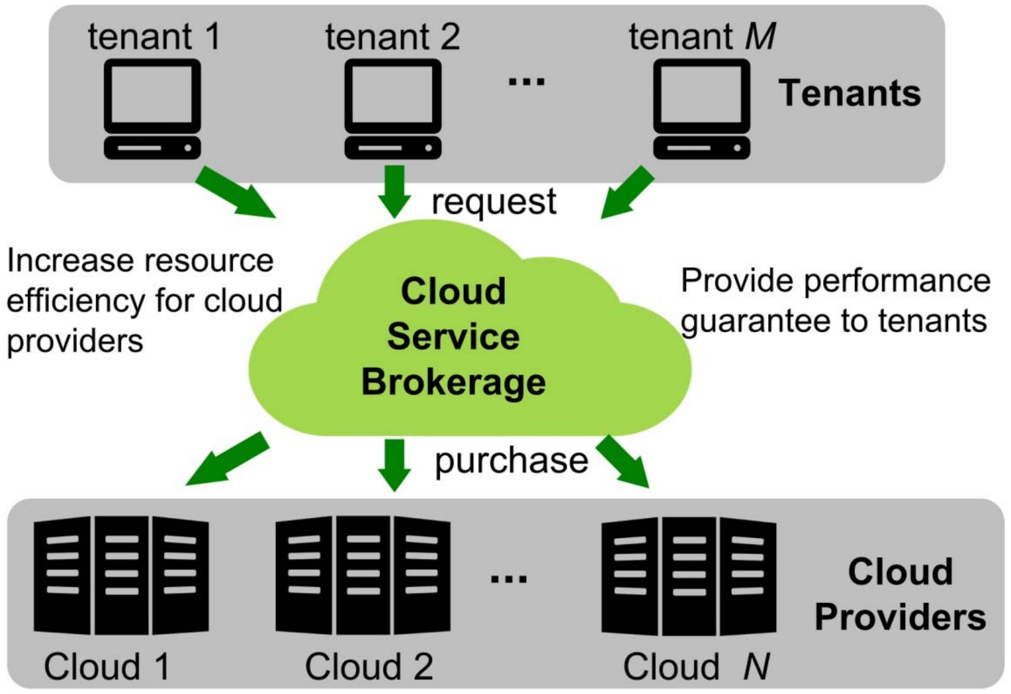
## Outline

1. Introduction
2. System Design
3. Performance Evaluation
4. Conclusion

## Introduction

### What is Cloud Service Brokerage

Definition: A third-party individual or business that acts as an intermediary between the tenants and the cloud providers.



CLEMSON UNIVERSITY

## Introduction

### Why Cloud Service Brokerage?

1. Helping users determine the best framework for each individual request.
2. A simplified interface, with interoperability benefits.
3. Cost-effective resources and infrastructure advantages.
4. Enhanced security.

## Introduction

### Our goal

Challenge: how to reserve servers and distribute tenant demands to the reserved servers such that the total reservation cost is minimized while the reserved servers can satisfy all the demands based on tenants' service level agreement (i.e., satisfy all the demands with a given probability)?

Implementing in two steps:

**Demand prediction**: we model the historical data of tenant demands by the seasonal auto-correlated moving average (SARMA) model for demand prediction.

**Demand allocation**: we formulate a stochastic programming problem for the challenge and find the problem solution through theoretical work.

## Introduction
### Related Work

Two groups:

1. Minimize the number of allocated servers while satisfying all demands.
    1. [Srikantaiah, HotPower 2008] : uses Euclidean distance between resource demands and residual capacity as a metric for consolidation.
    2. [Tang, WWW 2007]: combines the CPU and memory into a scalar by taking the ratio of the CPU demand to the memory demand.
    3. [Wood, NSDI 2007]: combines the CPU and memory into a scalar by taking the product of CPU, memory, and network loads.

2. Predict the demands before allocating the demands to servers
    1. [Shen, SOCC 2011]: predicts the seasonal period using FFT

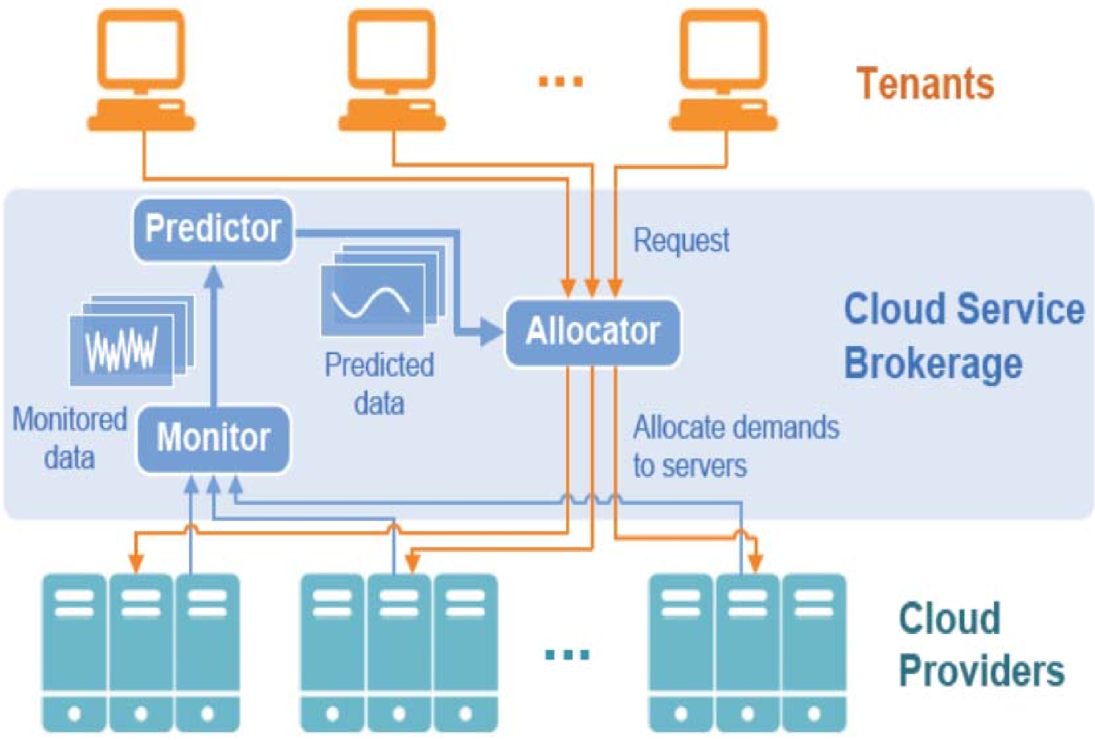
## System Design

### Architecture

1. First, CSB analyzes the history of each demand consumption from cloud monitoring, and then uses the historical data to predict the expected value of each demand by the predictor.

2. Then, CSB delivers the predicted value to allocator, which allocates the demands to different servers.

3. The CSB sells cloud service to tenants individually, it jointly reserves the different types of cloud resources from multiple cloud providers using the allocator.

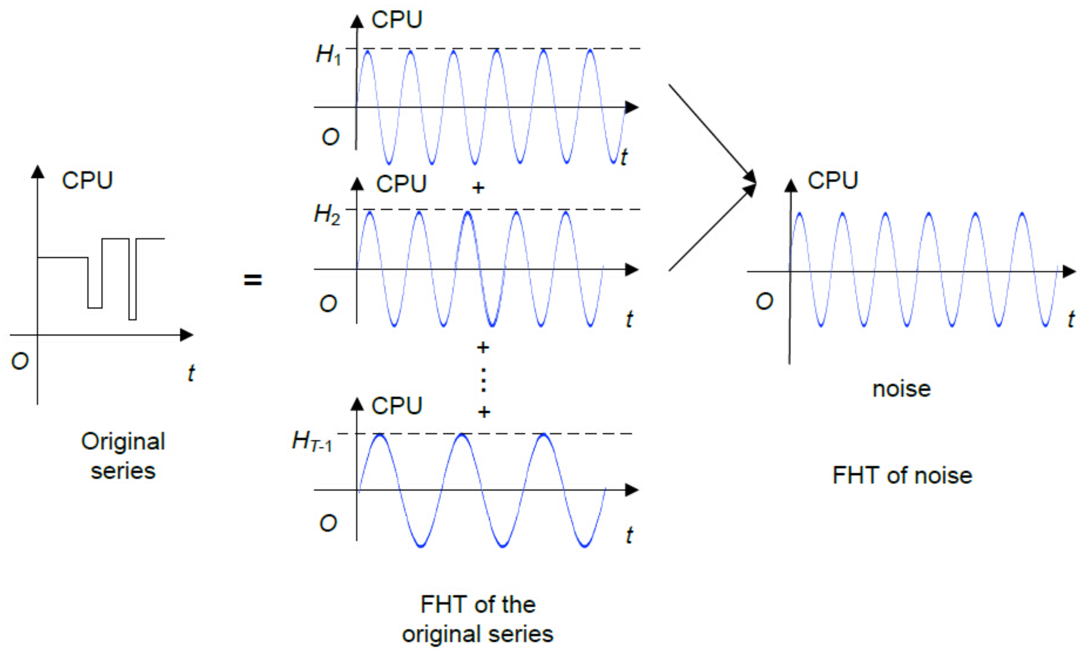


## System Design

### **Prediction:** the seasonal autoregressive moving-average (SARMA)

**Seasonal period estimation**: use Fast Hartley Transform (FHT) to estimate the seasonal period.

Employ FHT to calculate the dominant frequencies of resource demand, and then pick the lowest dominant frequency.



## System Design

**Prediction:** the seasonal autoregressive moving-average (SARMA)

**Prediction Error Estimation**: maximum likelihood estimation.

Find the value of $\sigma^{l,k}$ that maximizes the likelihood function

$$L^{l,k}\left(\mathbf{v}^{l,k}|\sigma^{l,k}\right) = \ln f\left(\sigma^{l,k}\right)$$

where $\sigma^{l,k}$ represents the prediction error variance.

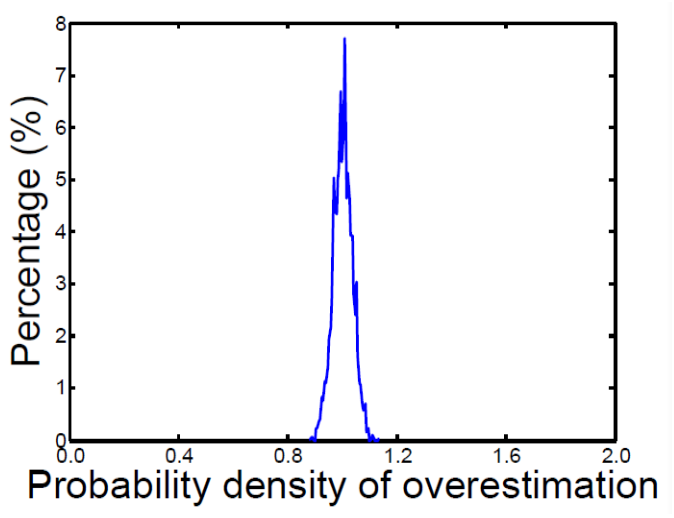
## System Design
### Demand allocation

Rationale of our demand allocation method:

Run our prediction method on the CPU utilization of 3000 demands in Google cluster and measure the ratio of overestimation over underestimation in the prediction of each demand.

Observation: overestimation and underestimation are equally likely to happen for each demand in most cases.

We allocate demands with similar underestimation and overestimation together in a server, so that the underestimation and overestimation can offset each other and hence the server resources are less likely to be either overutilized or underutilized.

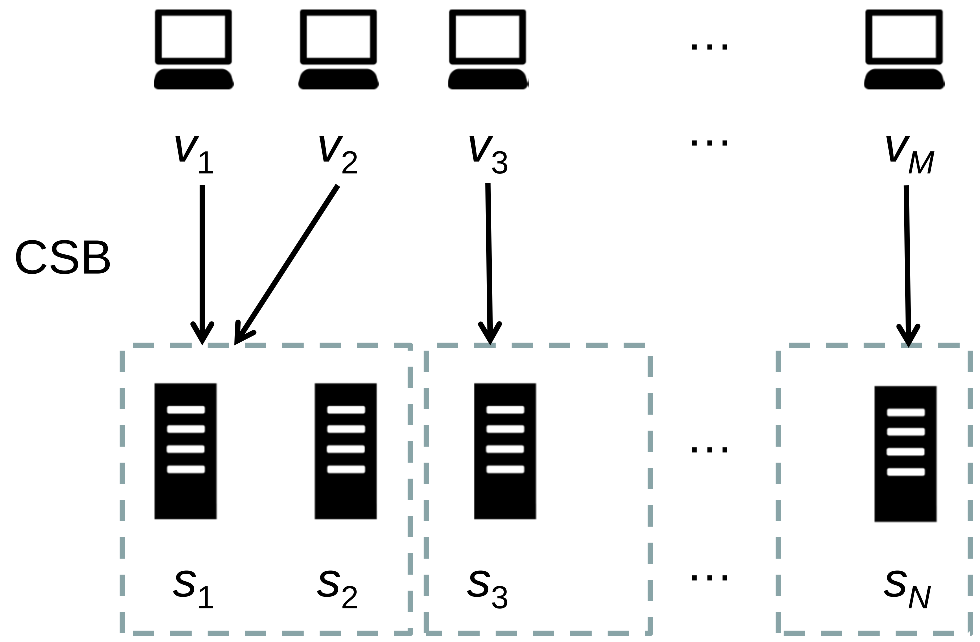


# System Design

## Allocation

Model:

*M* demands from tenants:



estimated demand vector

$$\hat{\mathbf{w}}_T^l = [w_T^{l,1}, \ldots, w_T^{l,K}]^\top$$

prediction error vector

$$\hat{\sigma}_T^l = [\hat{\sigma}_T^{l,1}, \ldots, \hat{\sigma}_T^{l,K}]^\top$$

*N* heterogeneous servers

## System Design

### Allocation

Formulate the Probabilistic Demand Allocation (PDA) problem:

$$\begin{array}{ll} \min & \mathbf{c}^{\top}\mathbf{x} \\ \text{s.t.} & \Pr\left(\omega_{k,T}^{\top}\mathbf{y}_i \leq b_{i,k}x_i\right) \geq 1-\varepsilon,\ \forall i,k \\ & \mathbf{y}^j\mathbf{1} = 1 \end{array}$$

variable $x_i$ to represent whether $s_i$ is purchased by the CSB: if yes, $x_i$ = 1; otherwise $x_i$ =0.

variable $y_{i,l}$ to denote whether demand $v_l$ is distributed to server $s_i$ : if yes, $y_{i,l}$ = 1; otherwise $y_{i,l}$ = 0.

## System Design

### Allocation

The PDA problem can be also written as a Second Order Conic Mixed Integer Programming (SOCMIP) problem:

$$
\begin{aligned}
\min \quad & f(\mathbf{z}) = \sum_{i=1}^{N} \mathbf{c}_i^\top \mathbf{z}_i \\
\text{s.t.} \quad & g_i(\mathbf{z}) = \phi(\varepsilon)\|\Sigma_k'^{1/2}\mathbf{z}_i\| - \mathbf{b}_{i,k}^\top \mathbf{z}_i \le 0, \\
& i = 1,\dots,N, \; k = 1,\dots,K \\
& h(\mathbf{z}) = E^\top \mathbf{z} - \mathbf{1} = 0,
\end{aligned}
$$

SOCMIP has been proved NP-hard. Hence, it is impossible to get the optimal solution of SOCMIP within polynomial time. As a solution, we first relax the feasible region of SOCMIP from integers to real numbers.

However, even for the relaxed SOCMIP, the subgradient algorithm is still not time-efficient.

## System Design

### Allocation

Distributed method: Decentralized Subgradient Algorithm

Step 1: Derive a dual problem of the relaxed SOCMIP, denoted by DSOCMIP.

$$\begin{aligned}\Lambda(\bar{\mathbf{z}},\lambda,\nu) &= \sum_{i=1}^{N}\mathbf{c}_i^\top\bar{\mathbf{z}}_i+\sum_{i=1}^{N}\lambda_i g_i(\bar{\mathbf{z}})+\nu h(\bar{\mathbf{z}})\\ &= \sum_{i=1}^{N}\left(\mathbf{c}_i^\top\bar{\mathbf{z}}_i+\lambda_i\left(\|\Sigma_i^{1/2}\bar{\mathbf{z}}_i\|\right)-\mathbf{b}_i^\top\bar{\mathbf{z}}_i\right) \quad [i]\\ &+ \nu\left(E^\top\bar{\mathbf{z}}-\mathbf{1}\right) \quad [ii]\end{aligned}$$

define the Lagrangian dual function by $\Theta(\lambda,\nu)=\inf\Lambda(\bar{\mathbf{z}},\lambda,\nu).$

Therefore, DSOCMIP is to find the solution

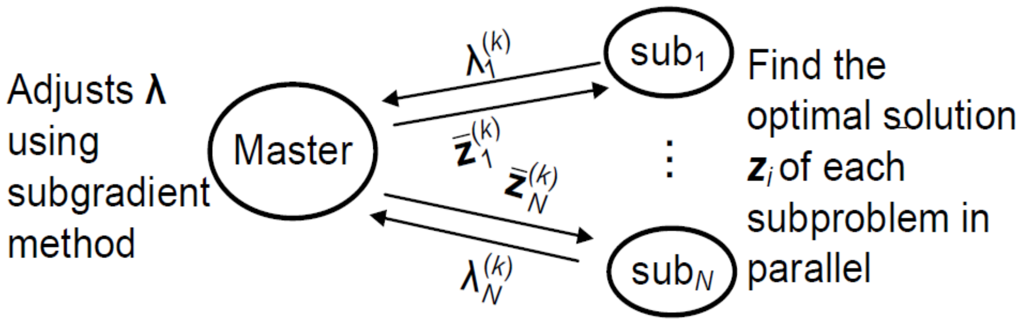
$$\begin{aligned}\max \quad & \Theta(\lambda,\nu)\\ \text{s.t.} \quad & \lambda\succeq\mathbf{0}\end{aligned}$$

## System Design

### Allocation

Distributed method: Decentralized Subgradient Algorithm

Step 2: Decompose the dual problem of the relaxed SOCMIP.
DSOCMIP can be decomposed to one master problem and a set of subproblems:


Adjusts $\lambda$ using subgradient method
Master
$\lambda_1^{(k)}$
$\bar{z}_1^{(k)}$
$\bar{z}_N^{(k)}$
$\lambda_N^{(k)}$
sub$_1$
sub$_N$
Find the optimal solution $z_i$ of each subproblem in parallel

the master problem and the subproblem collects and compares each other's solutions and sends feedback to each other to adjust the solutions if conflicts exist. The solution is obtained when the conflict is small enough.

Step 3: round fractional solution to get the final solution.

## Performance Evaluation

Conduct both simulation and real-world experiments (on Amazon EC2) driven by the Google Cluster trace

**Compared method**:

**Prediction**:

Cloud-Scale: estimates the magnitude of underestimation from prediction using FFT and adds a padding on predicted demands to avoid the impact of underestimation.

**Demand allocation**:
BFDSum and FFDSum: all servers' capacity vectors and demands' consumption vectors are mapped into singular scales by sum. Then use BFD or FFD to allocate the demand.

*BFD: Best Fit Decreasing*
*FFD: First Fit Decreasing*

## Performance Evaluation

**Metrics:**

**Prediction:**

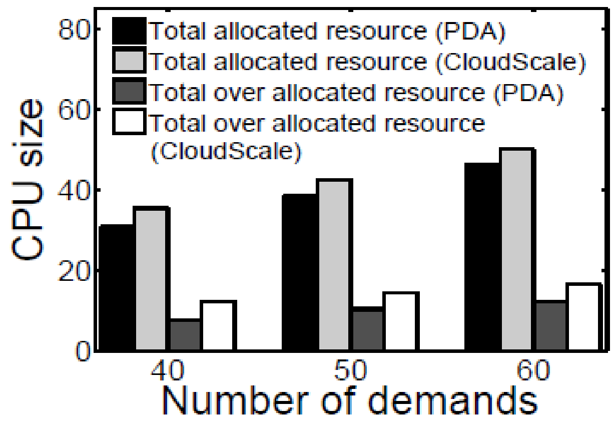
1) Total allocated resource:  the total amount of all the allocated resource.
2) Total over allocated resource: the total allocated resource minus the total amount of predicted demands.
3) Probability of overload: the percentage of periodical recoding times that the actual total demand exceeds the total allocated resource.

**Allocation**:

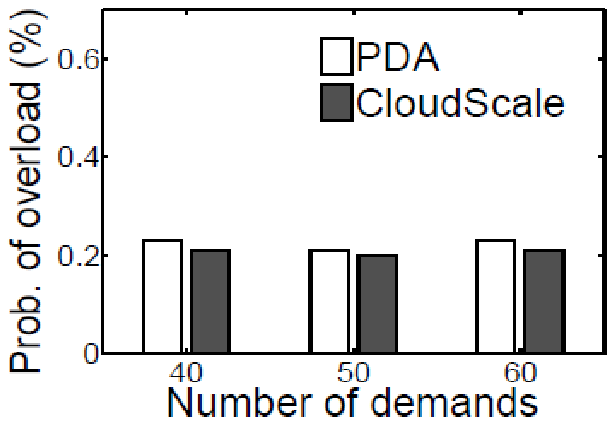
1) Total reservation cost: the total fee that the CSB needs to pay to all the cloud providers;
2) CPU overload rate: the percentage of time that a server's CPU is overloaded;
3) SLA violation rate: which is defined as the percentage of demands that do not satisfy SLA during the testing time;
4) CPU/memory utilization: which is defined as the percentage of a server's CPU/memory capacity that is actually consumed.

## Performance Evaluation

### Prediction



(a) Prediction for CPU



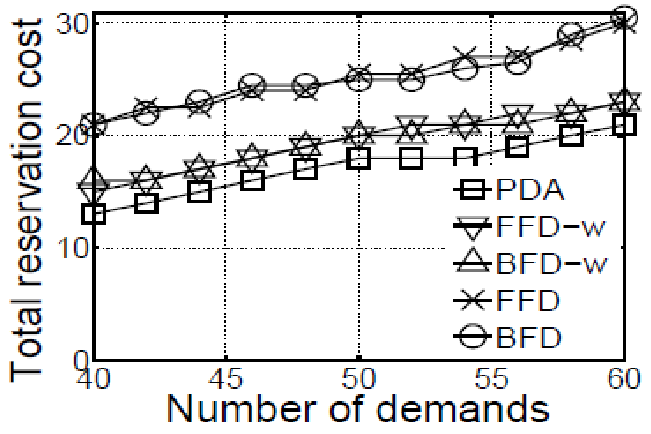
(b) Prob. of CPU overload

1) PDA needs less total allocated resource than CloudScale,
2) the total over allocated resource of PDA is smaller than that of CloudScale,
3) the probabilities of overload of PDA and CloudScale are similar.

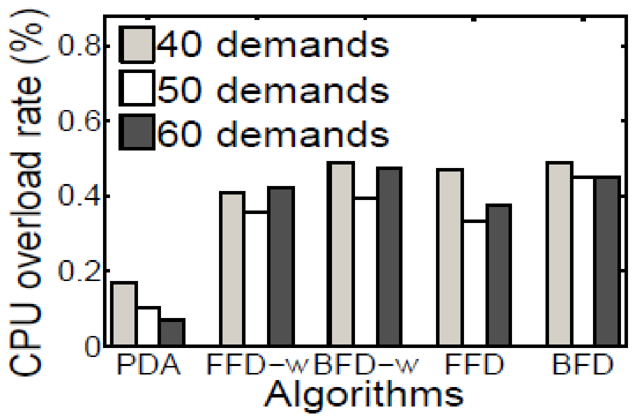
Reason: When allocating demands, PDA considers both overestimation and underestimation of the predictions to offset each other, which leads to less prediction errors.

## Performance Evaluation

### Demand allocation



(a) Total reservation cost



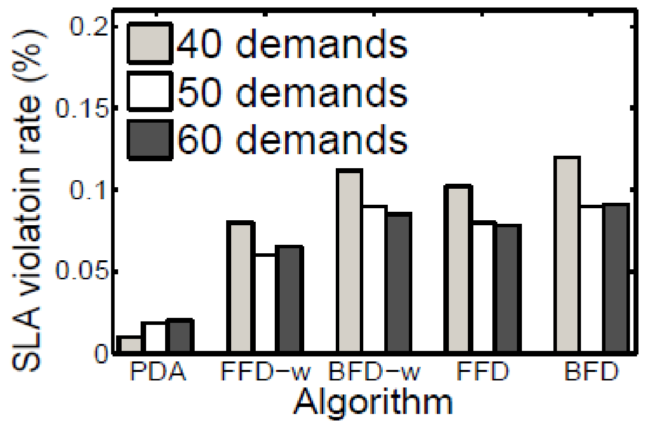
(b) CPU overload rate

1) The total reservation cost of servers follows: FFD, BFD>FFD-w, BFD-w>PDA,
2) CPU overload rate follows: FFD, BFD, FFD-w, BFD-w>PDA

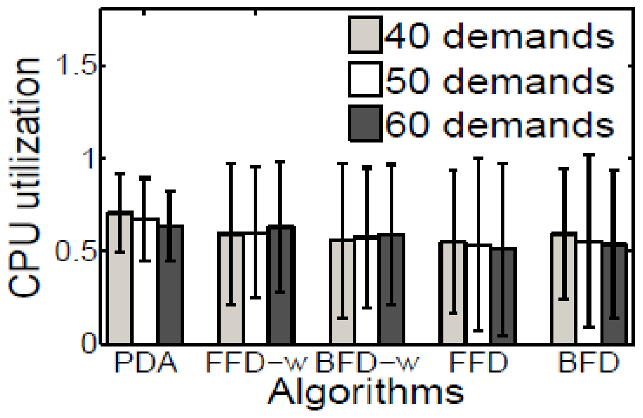
CLEMSON UNIVERSITY

## Performance Evaluation

### Demand allocation



(c) SLA violation rate



(d) CPU utilization

1) the SLA violation rate follows: FFD, BFD, FFD-w, BFD-w>PDA,
2) CPU utilization follows: FFD, BFD, FFD, BFD<PDA

Conclusion: PDA can more fully utilize resources in each server without overloading them and hence saves total reservation cost in Amazon EC2.

## Conclusion

1. A demand allocation system for CSB to determine how to reserve servers and allocate demands into the reserved servers, such that all the demands from tenants can be satisfied with a given probability and the total reservation cost is minimized.
2. A predictive model that can dynamically predict different types of resources of demands based on historical data.
3. In the demand allocation part, given the prediction results from prediction part, we formulated a probabilistic demand allocation problem.
4. Both simulation and real-world experimental results demonstrate the superior performance of our system in achieving the objective in comparison with some previous methods.

**Future work:**

1. When predicting the demands, we will further consider the correlation among the demands from different tenants and different types of resources.
2. We will consider the demand migration among different servers.

CLEMSON UNIVERSITY

*Thank you!*

*Questions & Comments?*

**Haiying Shen, Associated Professor**

**shenh@clemson.edu**

**Pervasive Communication Laboratory**

**Clemson University**