# To Reserve or Not to Reserve: Optimal Online Multi-Instance Acquisition in IaaS Clouds

Wei Wang, Baochun Li, ***Ben Liang***

Department of Electrical and Computer Engineering

University of Toronto

# Growing Cloud-Computing Costs

- Drastic increase in enterprise spending on Infrastructure-as-a-Service (IaaS) clouds
  - 41.7% annual growth rate by 2016 [Cloud Times'12]
  - IaaS cloud is the *fastest-growing* segment

amazon web services™

ElasticHosts Flexible Servers in the Cloud

GOGRID Control in the Cloud™

# Tradeoffs in Cloud Pricing Options

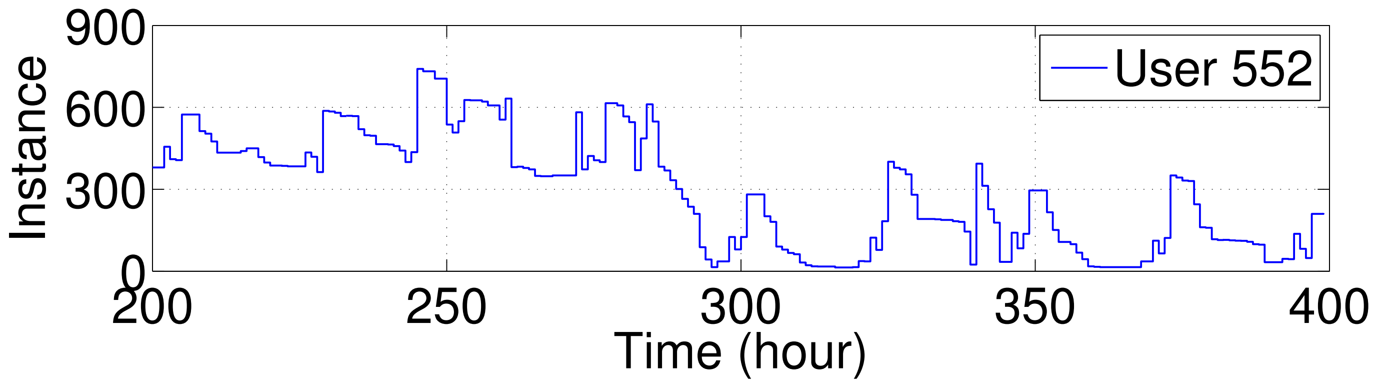
- On-demand Instances
  – No commitment
  – Pay-as-you-go
- Reserved Instances
  – Reservation fee + discounted price
  – Suitable for long-term usage commitment

amazon web services™

| Instance Type | Pricing Option | Upfront | Hourly |
|---|---|---|---|
| Standard Small | On-Demand | \$0 | \$0.08 |
| | 1-Year Reserved | \$69 | \$0.039 |
| Standard Medium | On-Demand | \$0 | \$0.16 |
| | 1-Year Reserved | \$138 | \$0.078 |

# Multi-Instance Acquisition Problem

- Workload (demand) is time-varying




- When should I reserve an instance?
- How many instances should I reserve?

# Predict the Future?

- Existing works rely on prediction of future demand
  - [Hong SIGMETRICS'11, Bodenstein ICIS'11, Vermeersch Thesis'11, Wang ICDCS'13]
- However…
  - Prediction is needed for long-term future
    - Instance reservation period is typically months to years
  - Precise prediction not possible
  - Demand history may be limited
    - E.g., startup companies, new services

How well can we make instance reservation decisions online, without any *a priori* information about the future demand?

# Our Main Contributions

- Propose two online reservation algorithms that offer the best provable cost guarantees
  - Deterministic: $(2\text{-}\alpha)$-competitive
  - Randomized: $e/(e\text{-}1+\alpha)$-competitive
    - $\alpha$ : normalized discounted price under reservation $(0 \le \alpha \le 1)$
- Study practical performance gains using Google cluster workload traces

# Problem Formulation

# Pricing of On-Demand and Reserved Instances

- On-demand Instances
  - Fixed hourly price: $p$
  - Cost of running for $h$ hours: $ph$
- Reserved Instances
  - Upfront reservation fee + discounted price
  - Normalized reservation fee: $1$
  - Reservation period: $\tau$ hours
  - Cost of running for $h$ hours: $1 + \alpha ph$
    - $\alpha$ : normalized discounted price under reservation $(0 \leq \alpha \leq 1)$

# User Demand and Reservation

At time $t$ (discrete time), the user

- Has demand for $d_t$ instances (time-varying)
- *Newly* reserves $r_t$ instances
  - Available reserved instances: $\sum_{i=t-\tau+1}^{t} r_i$
- Launches $o_t$ on-demand instances
  - Total available instances:

$$o_t + \sum_{i=t-\tau+1}^{t} r_i \geq d_t$$

# Optimal Offline Algorithm

- Make instance purchase decisions $o_t$ and $r_t$ with knowledge of all future demands $d_{t+1}$, $d_{t+2}$, $\cdots$

$$\min_{\{r_t, o_t\}} \quad C = \sum_{t=1}^{T} \Big( \overset{\text{On-demand cost}}{\boxed{o_t p}} + \overset{\text{Reservation cost}}{\boxed{r_t + \alpha p (d_t - o_t)}} \Big) \ ,$$

$$\text{s.t.} \quad o_t + \sum_{i=t-\tau+1}^{t} r_i \geq d_t \ ,$$

$$o_t, r_t \in \{0, 1, 2, \dots\}, t = 1, \dots, T \ .$$

– Can be solved by dynamic programming, but is *computationally prohibitive*

# Online Instance Reservation

- Make instance purchase decisions $o_t$ and $r_t$ without seeing future demands $d_{t+1}$, $d_{t+2}$, …

On-demand cost Reservation cost

$$\min_{\{r_t, o_t\}} \quad C = \sum_{t=1}^{T} \left( \boxed{o_t p} + \boxed{r_t + \alpha p (d_t - o_t)} \right) ,$$

$$\text{s.t.} \quad o_t + \sum_{i=t-\tau+1}^{t} r_i \geq d_t ,$$

$$o_t, r_t \in \{0, 1, 2, \ldots\}, t = 1, \ldots, T .$$

– What is the best that one can do?

# Measure of Optimality

- Compare an online reservation algorithm with the optimal offline reservation
- An online algorithm $A$ is *$\gamma$-competitive* if it incurs at most $\gamma$ times the optimal offline cost
  – For any demand sequence $\mathbf{d} = d_1, d_2, \ldots$

$$C_A(\mathbf{d}) \leq \gamma \, C_{\mathrm{OPT}}(\mathbf{d})$$

  – Aims to minimize the competitive ratio $\gamma$

# The Best Possible Outcome

**Lemma 1:** The best achievable competitive ratio is $2\text{-}\alpha$ for *deterministic* online algorithms, and is $e/(e\text{-}1+\alpha)$ for *randomized* online algorithms.

Bahncard problem [Fleischer TCS'01]:

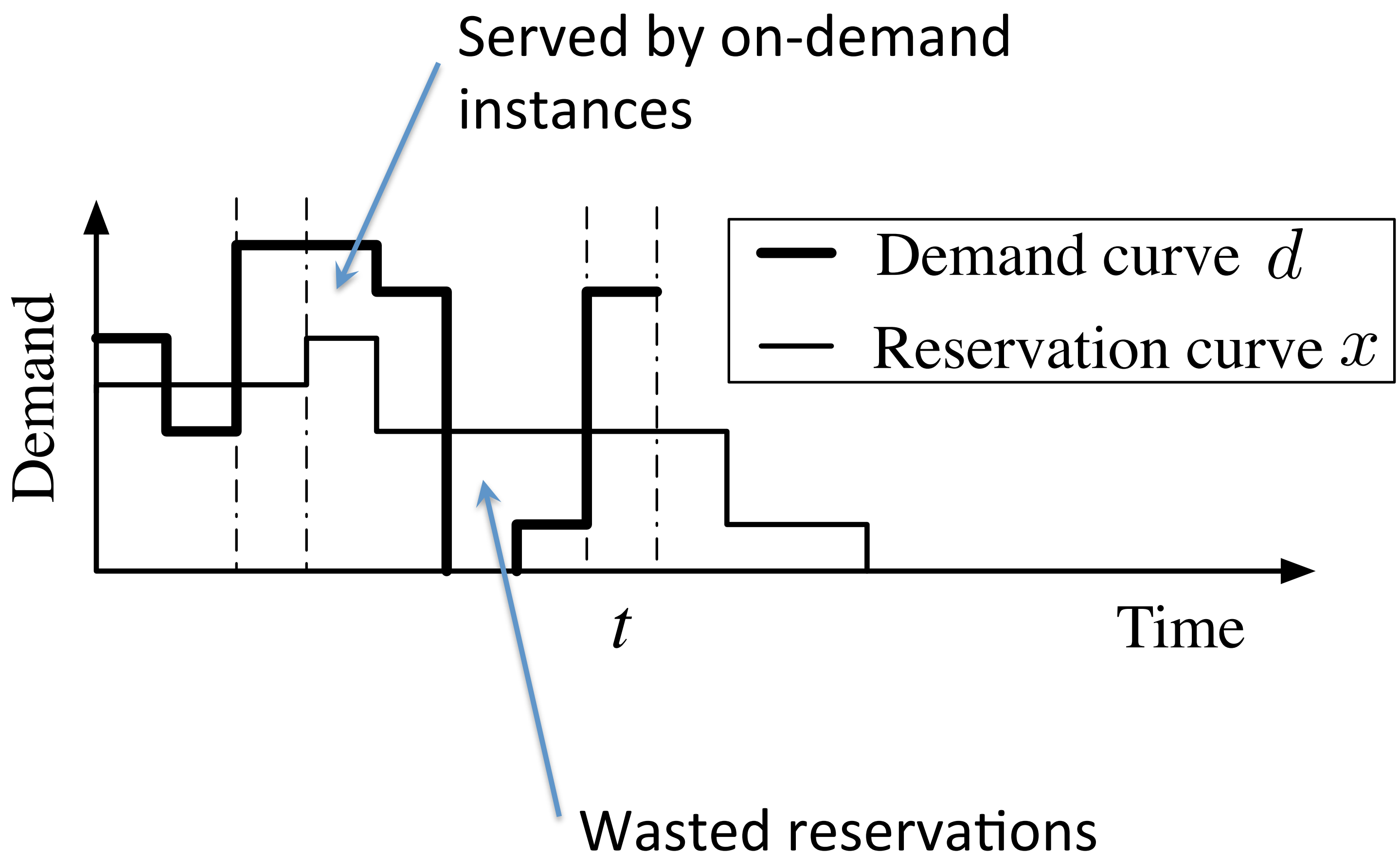
- Goal: reduce cost of using the Deutsche Bahn
- User may buy tickets on-demand or buy an annual Bahncard to enjoy discounted tickets
- No knowledge about user's travel plans or travel frequency

# Is the optimal competitive ratio achievable with multiple instances?

- “Multi-Bahncard” problem
- Naïve extension: separate Bahncards
  - Does not work

# Optimal Deterministic Online Algorithm

# Demand and Reservation Curves

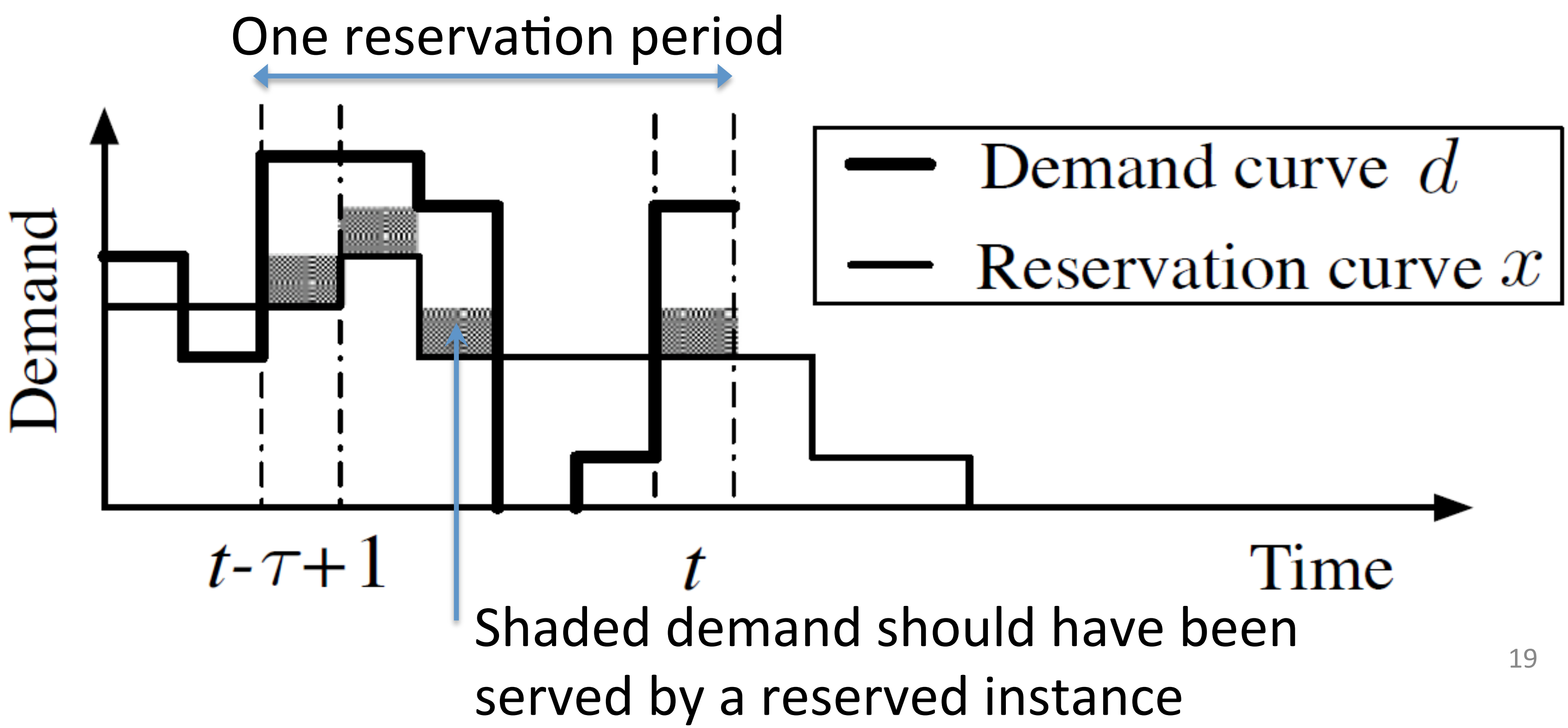


# Break-Even Point

- Let $c$ be the cost of one on-demand instance to serve workload that spans a reservation period.
- Using a reserved instance instead, the cost is $1+\alpha c$
- Break-even point: $c = 1+\alpha c$
  - Let $\beta=1/(1- \alpha)$
  - $c = \beta$ : Break even
  - $c < \beta$ : On-demand is better
  - $c > \beta$ : Reservation is better

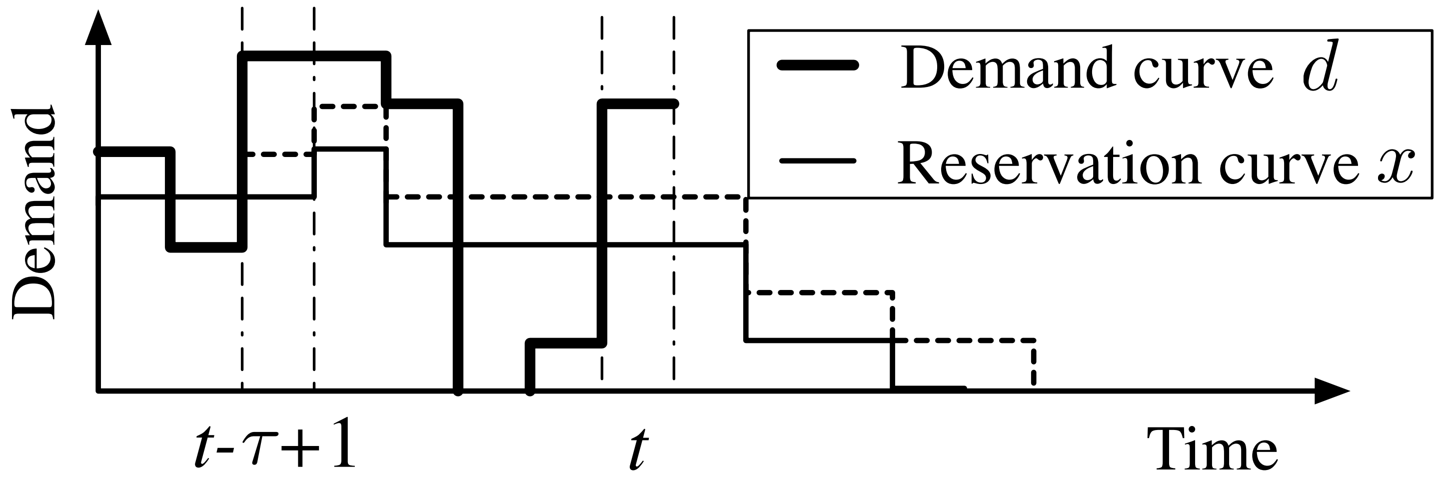
# Regret and Compensation

- At time $t$, look back for one reservation period.
- If the incurred on-demand cost $> \beta$, reserve a new instance: $r_t = r_t + 1$.



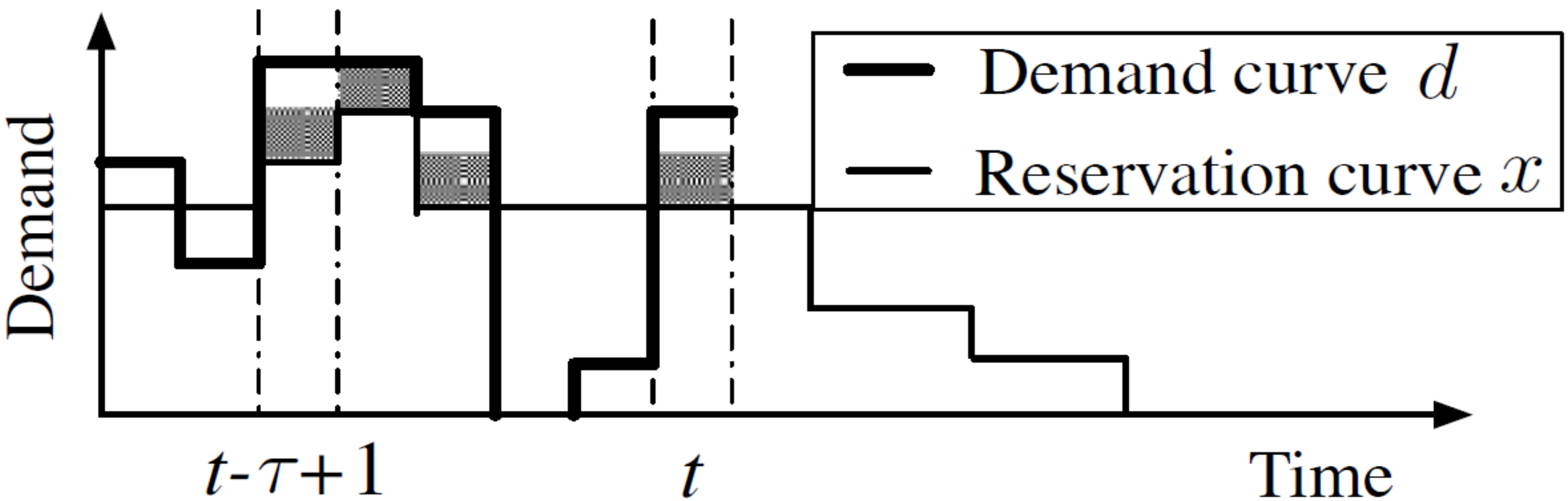
# Update Reservation Curve

- If a new instance is reserved, update the reservation curve, both forward and backward.



# Repeat until No Regret

- Repeat to reserve more new instances, until the (virtual) incurred on-demand cost $< \beta$.



**Proposition 1:** The deterministic online algorithm is ($2$-$\alpha$)-competitive, and hence is *optimal* among all deterministic online algorithms.

# Optimal Randomized Online Algorithm

# Basic Idea

- Can use different thresholds $z$ (other than the break-even point $\beta$) to decide whether to reserve an instance
  - A family of deterministic algorithms $\{A_z\}$
- The smaller $z$, the more aggressive the reservation strategy
  - $z = 0$: All-reserved
  - $z = +\infty$: All-on-demand

# Basic Idea (Cont'd)

- Randomly choose from the family of deterministic algorithms $\{A_z\}$
  - Strike balance between reserving too aggressively and too conservatively
  - Randomly pick threshold $z$ according to the following density function

$$f(z) = \begin{cases} (1-\alpha)e^{(1-\alpha)z}/(e-1+\alpha), & z \in [0,\beta), \\ \delta(z-\beta)\cdot\alpha/(e-1+\alpha), & \text{o.w.}, \end{cases}$$

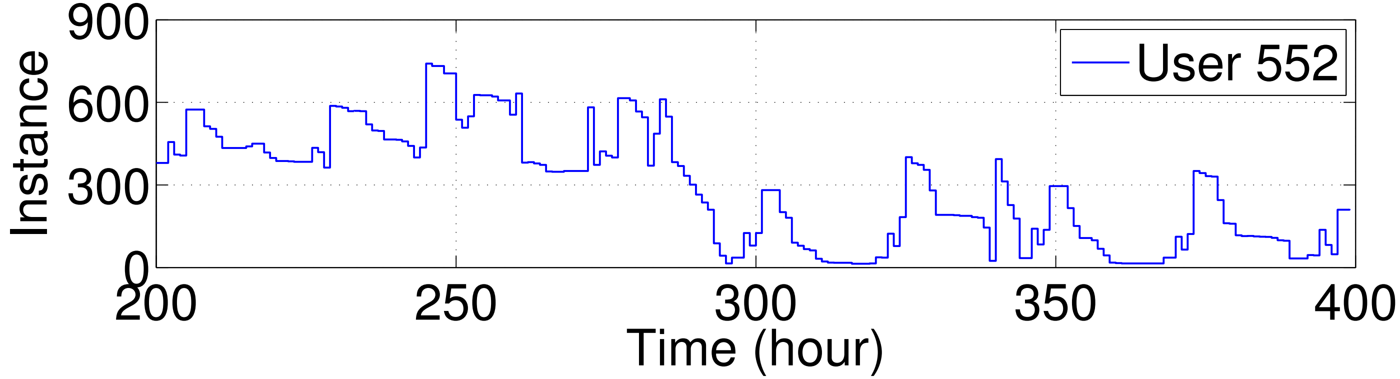
  - Make instance reservation decisions based on deterministic algorithm $A_z$

- **Proposition 2:** The randomized online algorithm is $e/(e\text{-}1+\alpha)$-competitive, and hence is *optimal* among all online algorithms.

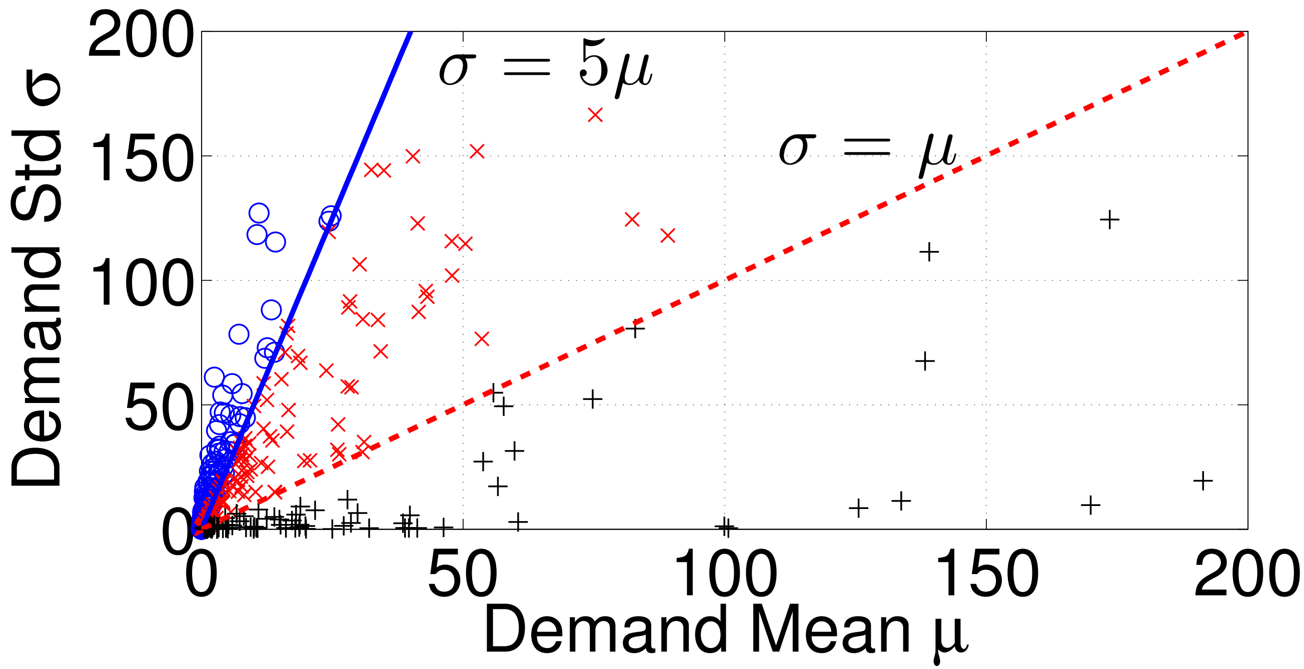
# Trace-Driven Simulations

# Dataset and Preprocessing

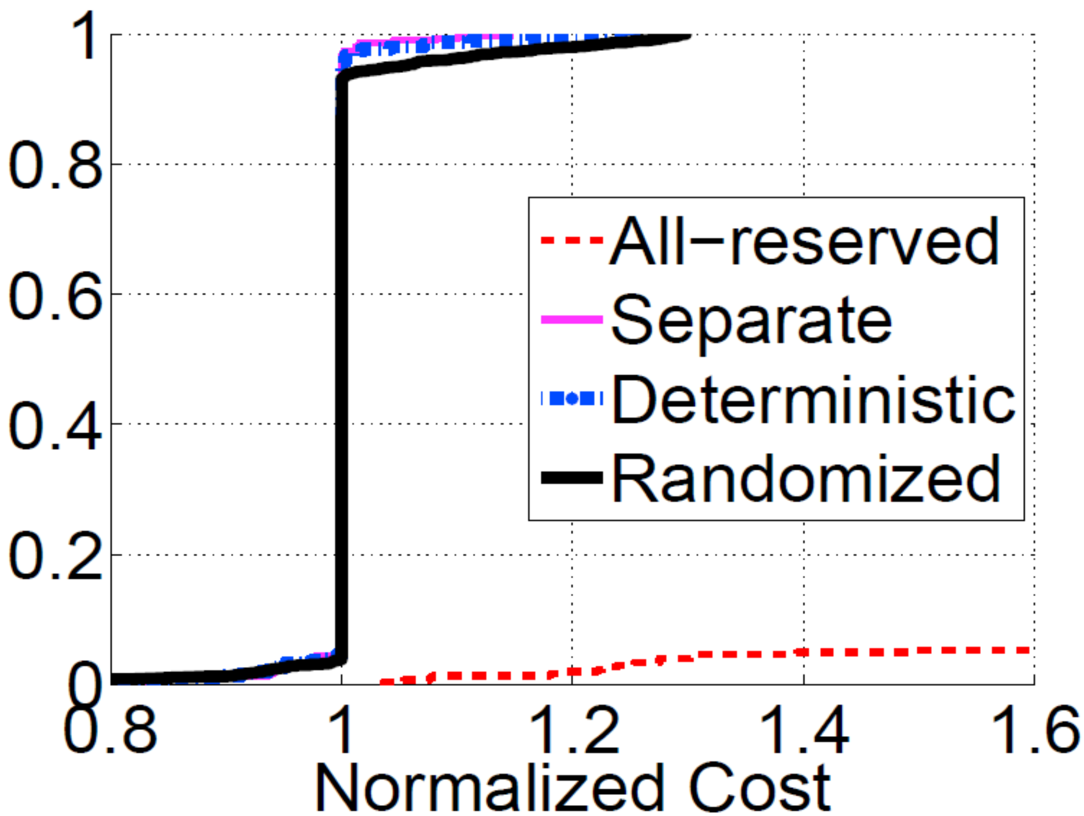
- Google cluster traces
  - 900+ users' usage traces in 1 month
  - We convert users' computing demand data to IaaS instance demands



- Users are classified into 3 groups based on demand fluctuation level
  - Standard deviation vs. mean in hourly demand



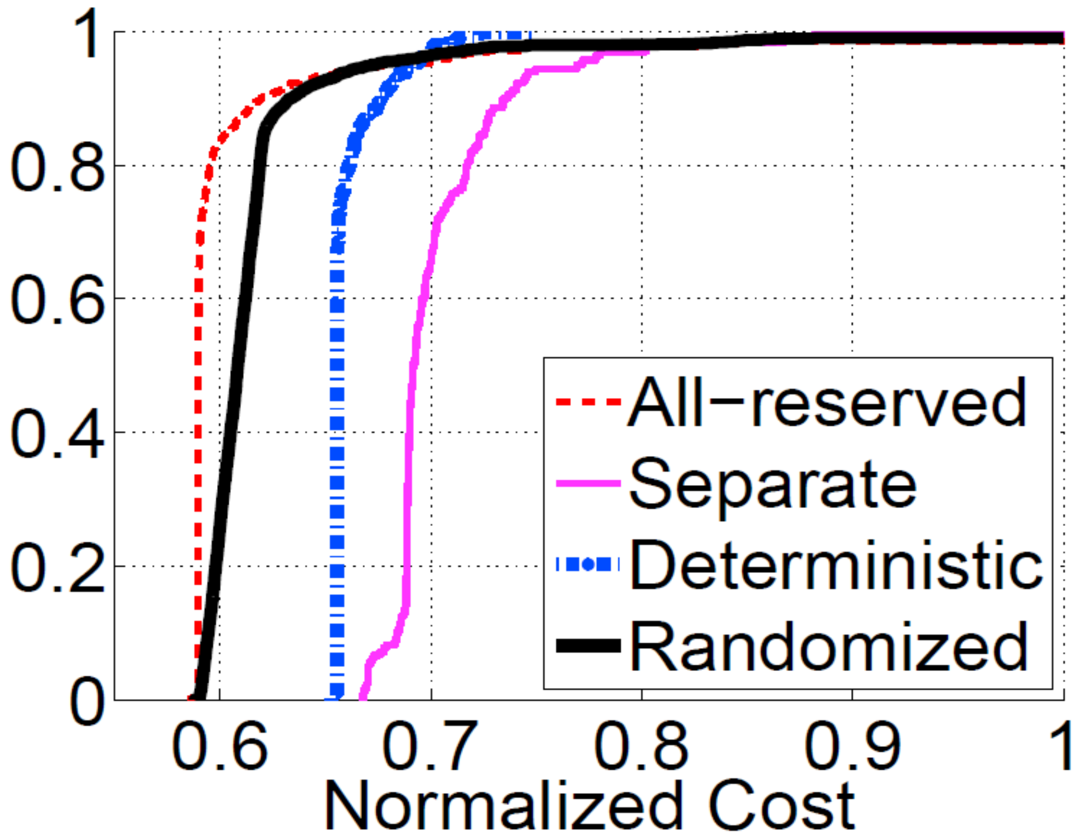
# CDF of Cost Normalized to All-On-Demand




High demand fluctuation

"*Separate*": stack demands and treat each layer as a virtual user, each individualy solving the Bahncard problem.

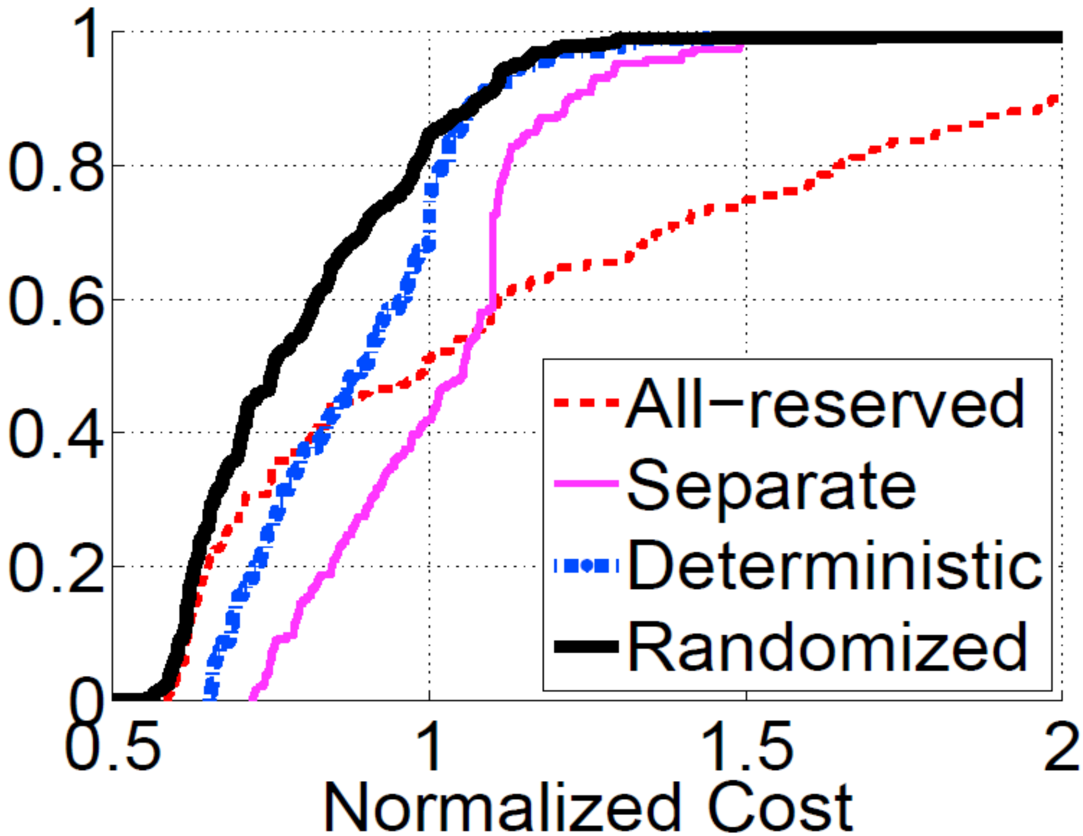
# CDF of Cost Normalized to All-On-Demand




Low demand fluctuation

"*Separate*": stack demands and treat each layer as a virtual user, each individually solving the Bahncard problem.

# CDF of Cost Normalized to All-On-Demand




Medium demand fluctuation

"*Separate*": stack demands and treat each layer as a virtual user, each individually solving the Bahncard problem.

# Conclusions

- Deterministic and randomized online multi-instance reservation algorithms without future demand information
  - Optimal competitive ratio vs. optimal offline algorithm
  - Substantial performance gain over a wide range of demand fluctuation levels
- Extension to cases where short-term predictions are reliable
- Open problem: multiple reservation options