Optimal Online Multi-Instance Acquisition in IaaS Clouds

Wei Wang, Baochun Li, Ben Liang

USENIX ICAC'13

TPDS'14

Tradeoffs in Cloud Pricing Options

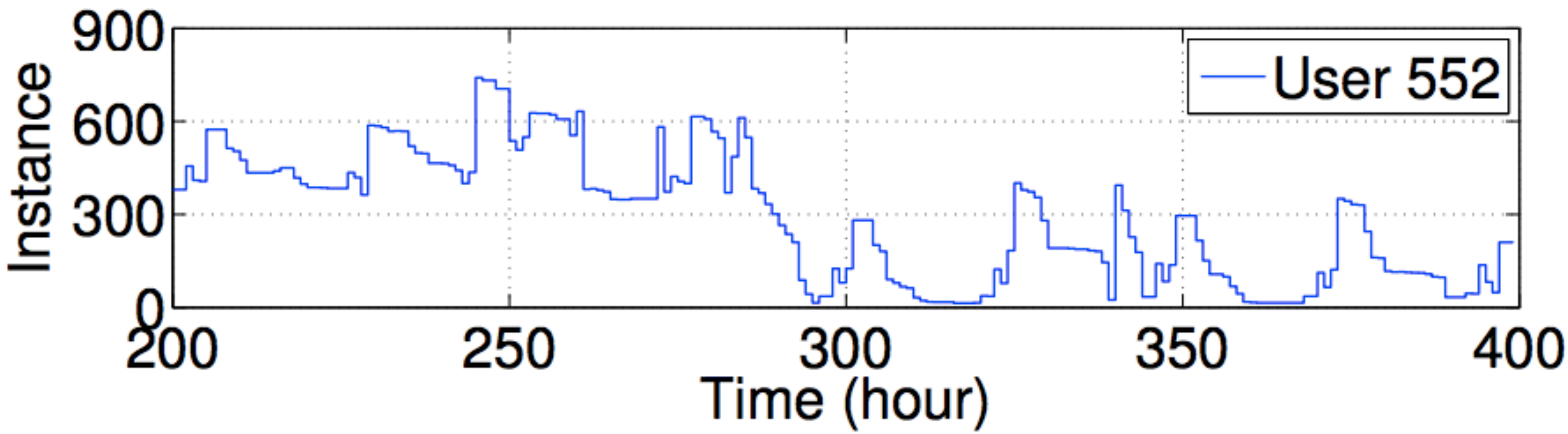
- On-demand Instances
  - ➢ No commitment
  - ➢ Pay-as-you-go
- Reserved Instances
  - ➢ Reservation fee + discounted price
  - ➢ Suitable for long-term usage commitment

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 Future?

- Existing work relies on prediction of future demand
- However…

➢ Prediction is needed for long-term future

- Instance reservation period is typically months to years

➢ Precise prediction is impossible

➢ Demand prediction is limited

- E.g., start-up companies, new services

How well can we make reservation decisions online, without any a *prior* information of the future demand?

Main Contributions

- Propose two online algorithms that offer the best provable cost guarantees
  - Deterministic: (2-α)-competitive
  - Randomized: (e/(e-1+α))-comepetitive
    - α is the hourly discount due to reservation ($0 \leq \alpha \leq 1$)
    - They only consider one instance type

Main Contributions

- Propose two online algorithms that offer the best provable cost guarantees
  - Deterministic: (2-α)-competitive
  - Randomized: (e/(e-1+α))-comepetitive
    - α is the hourly discount due to reservation (0 ≤ α ≤ 1)
    - They only consider one instance type

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

Main Contributions

- Propose two online algorithms that offer the best provable cost guarantees
  - Deterministic: (2-α)-competitive
  - Randomized: (e/(e-1+α))-comepetitive
    - α is the hourly discount due to reservation (0 ≤ α ≤ 1)
    - They only consider one instance type
- Study practical performance gains using Google workload traces

Problem Model

Pricing of On-demand Instances and Reserved Instances

- On-demand Instances
  - Fixed hourly price: p
  - Cost of running for h hours: ph
- Reserved Instances
  - Upfront reservation fee + discounted hourly price
  - Reservation fee is normalized to 1
  - Reservation period: τ
  - Cost of running for h hours: $1+ \alpha ph$
  - α is the hourly discount due to reservation ($0 \leq \alpha \leq 1$)

User demand and reservation

At t time slot, the user

- Has demand for d_t instances (time-varying)
- Newly reserves r_t instances
  - Available reserved instances: $\sum_{i=t-\tau+1}^{t} r_i$
- Purchases o_t on-demand instances
  - Total # of instances the user could launch at t:

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

Offline Problem Formulation

Cost of on-demand instances

Cost of reserved instances

$$
\begin{aligned}
\min_{\{r_t, o_t\}} \quad & C = \sum_{t=1}^{T} \left(\boxed{o_t p} + \boxed{r_t + \alpha p(d_t - o_t)} \right), \\
\text{s.t.} \quad & o_t + \sum_{i=t-\tau+1}^{t} r_i \geq d_t \ , \\
& o_t, r_t \in \{0, 1, 2, \ldots\}, t = 1, \ldots, T \ .
\end{aligned}
$$

Offline Problem Formulation

Cost of on-demand instances

Cost of reserved instances

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

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

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

Hourly fee of reserved instances is based on usage t = 1: $d_t = 7$, $r_t = 5$, $o_t = 2$, $d_t - o_t = 5$, $\tau=2$

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

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

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

Hourly fee of reserved instances are based on usage

$t = 1$: $d_t = 7, r_t = 5, o_t = 2, d_t - o_t = 5, \tau=2$

$t = 2$: $d_t = 4, r_1 = 5, r_2 = 0, o_t = 0, d_t - o_t = 4$

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

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

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

Hourly fee of reserved instances are based on usage

t = 1: $d_t = 7$, $r_t = 5$, $o_t = 2$, $d_t - o_t = 5$, $\tau=2$

t = 2: $d_t = 4$, $r_1 = 5$, $r_2 = 0$, $o_t = 0$, $d_t - o_t = 4$

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

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

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

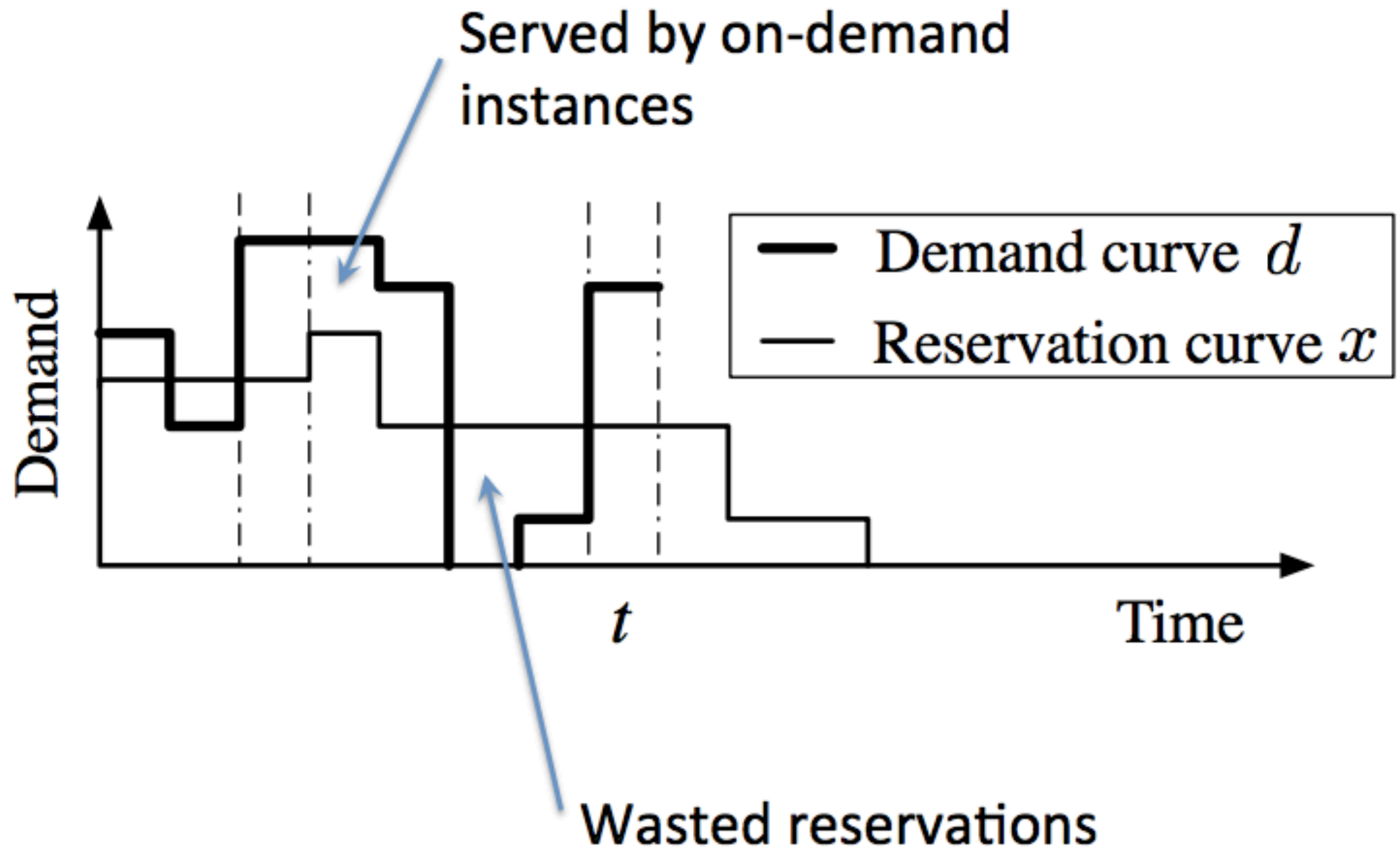
Hourly fee of reserved instances are based on usage

Lower-bound of competitive ratio

- Lemma 1: The best competitive ratio of this problem is at least $2-\alpha$ for deterministic online algorithms, and is at least $e/(e-1+\alpha)$ for randomized algorithms.
- R. Rleischer, ''On the Bahncard Problem'', 2001

Optimal deterministic online algorithm

Demand and Reservation Curves



Break-Even Point

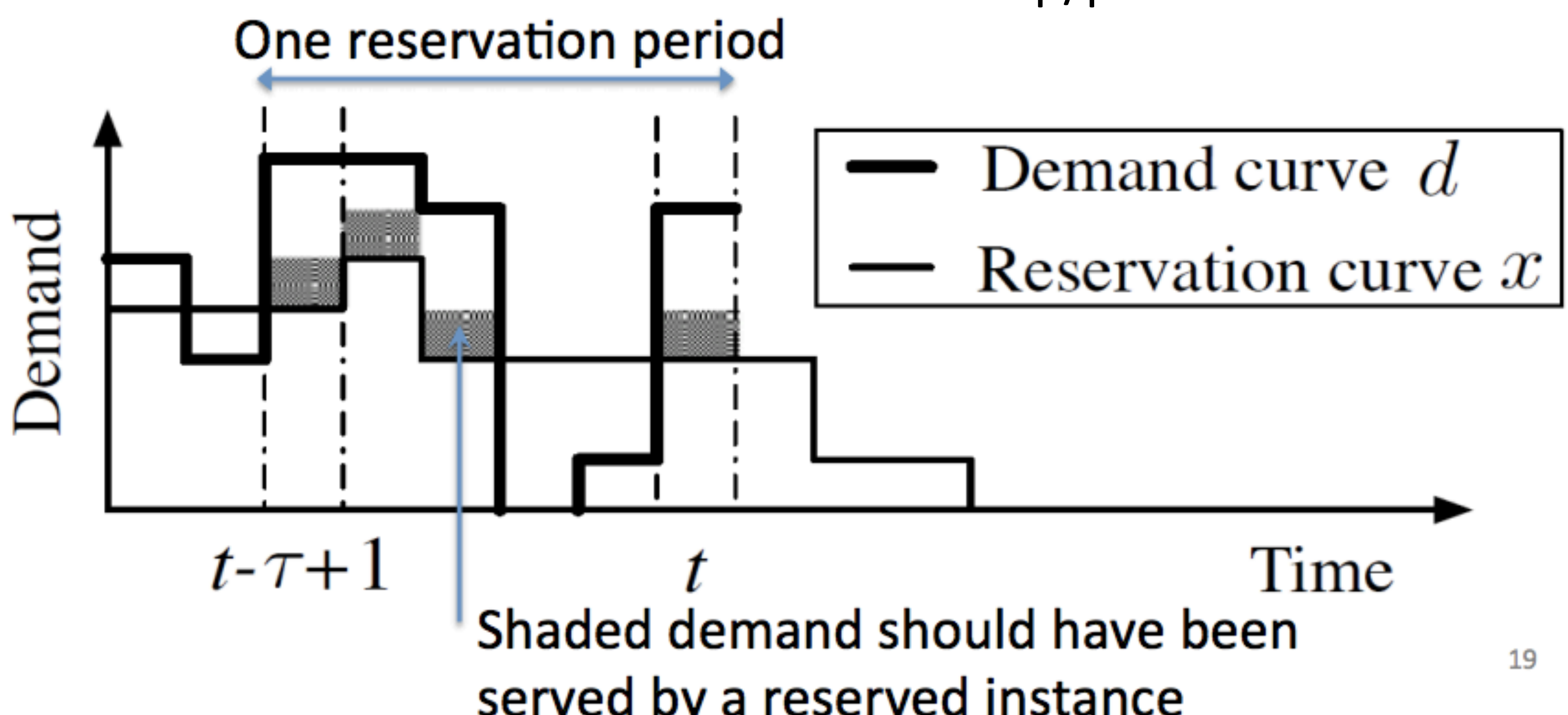
- Let c be the cost of running an on-demand instance that within a reservation period
- Using a reserved instance instead, the cost is

  $1+ \alpha c$
- Break-even point: $\beta = 1+\alpha\beta \Rightarrow \beta = 1/(1-\alpha)$
  - If $c = \beta$, $c=1+\alpha c$, break even
  - If $c < \beta$, $c<1+\alpha c$, use an on-demand instance
  - If $c> \beta$, $c>1+\alpha c$, use a reserved instance

Regret and Compensation

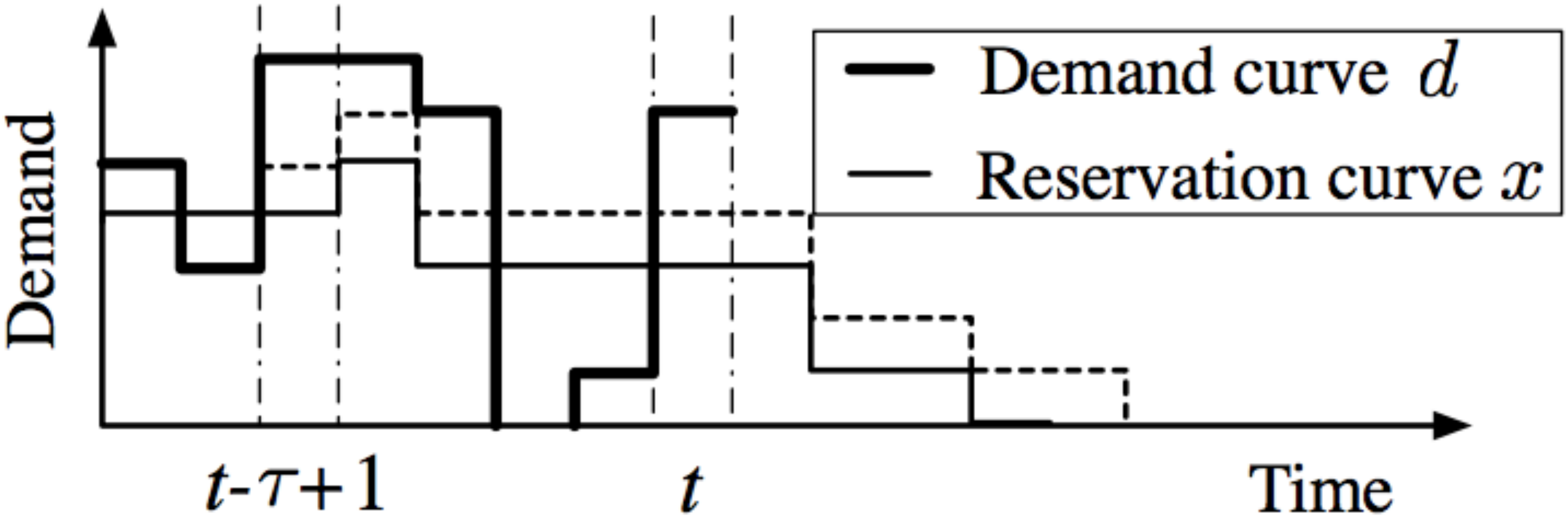
- At time t, look back for a reservation period
- If the incurred on-demand cost > β, reserve a new instance $r_t := r_t + 1$

$\beta/p = 4$



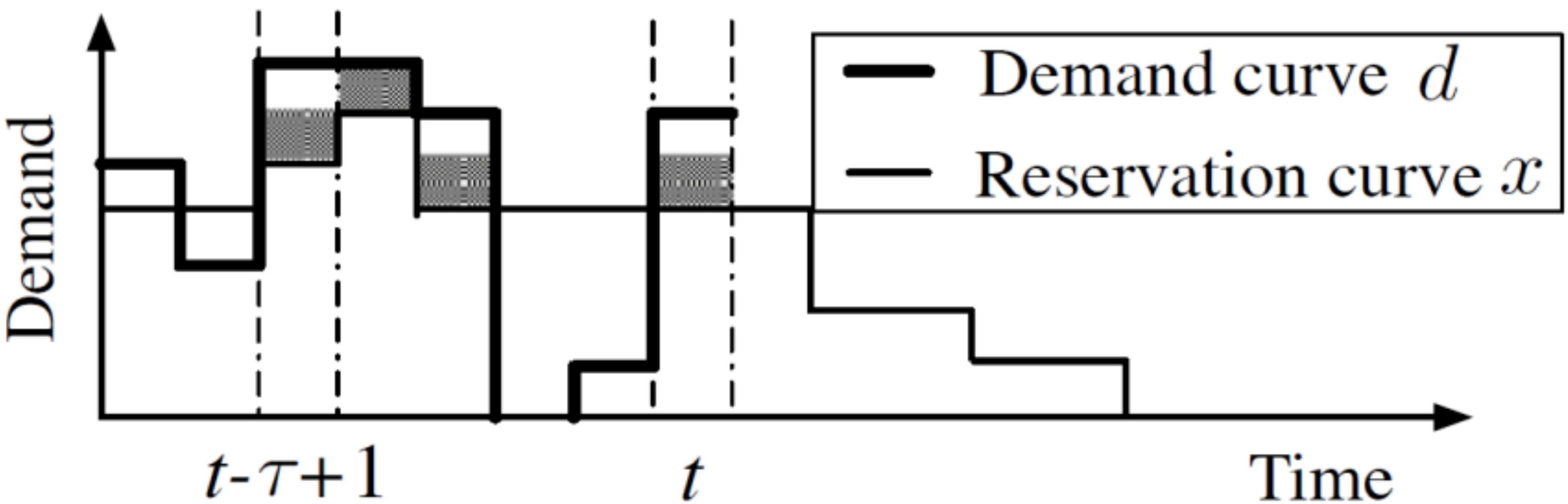
Update reservation curve

If a new instance is reserved, lift the reservation curve (mark the compensated demand) by 1, both backward and forward for τ time slots



Repeat until no regret

- Repeat to reserve more instances, until the incurred on-demand instances cost < β.



- Proposition 1: This deterministic algorithm is (2-α)-competitive, hence is the optimal among all the deterministic algorithms of this problem

Optimal Randomized Online Algorithm

Basic Idea

- Strike balance between reserving too aggressively or too conservatively
- Randomly pick a threshold z instead of the break-even point β according to the density function:

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

Basic Idea

- Strike balance between reserving too aggressively or too conservatively
- Randomly pick a threshold z instead of the break-even point β according to the density function:

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

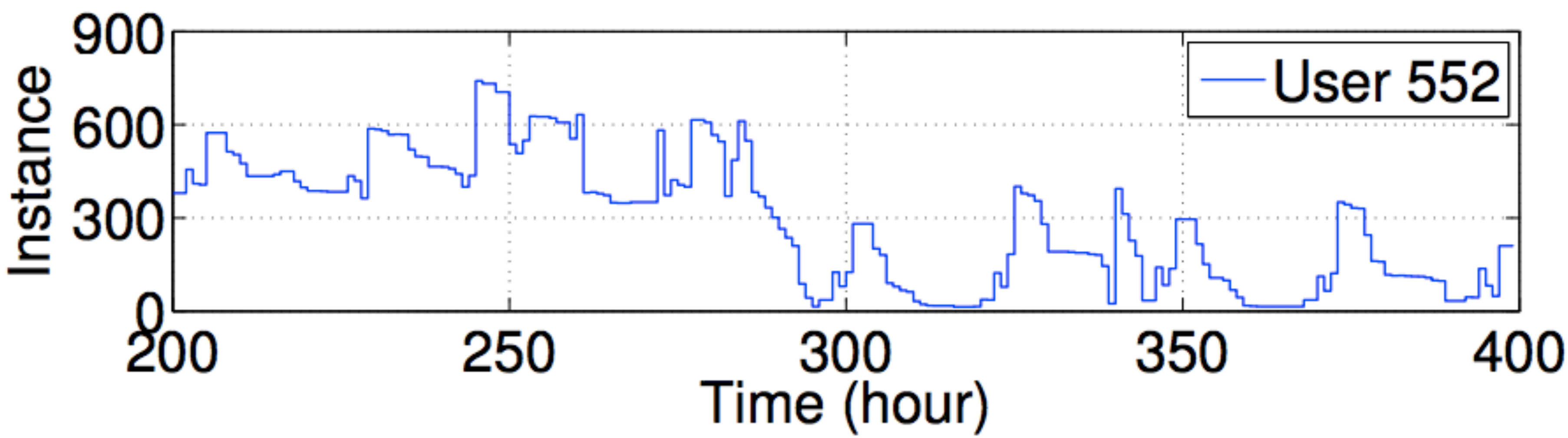
- Pick z = β with probability of 1

- Proposition 2: This randomized algorithm is (e/(e-1+))-competitive, and hence is an optimal among all the randomized algorithms of this problem

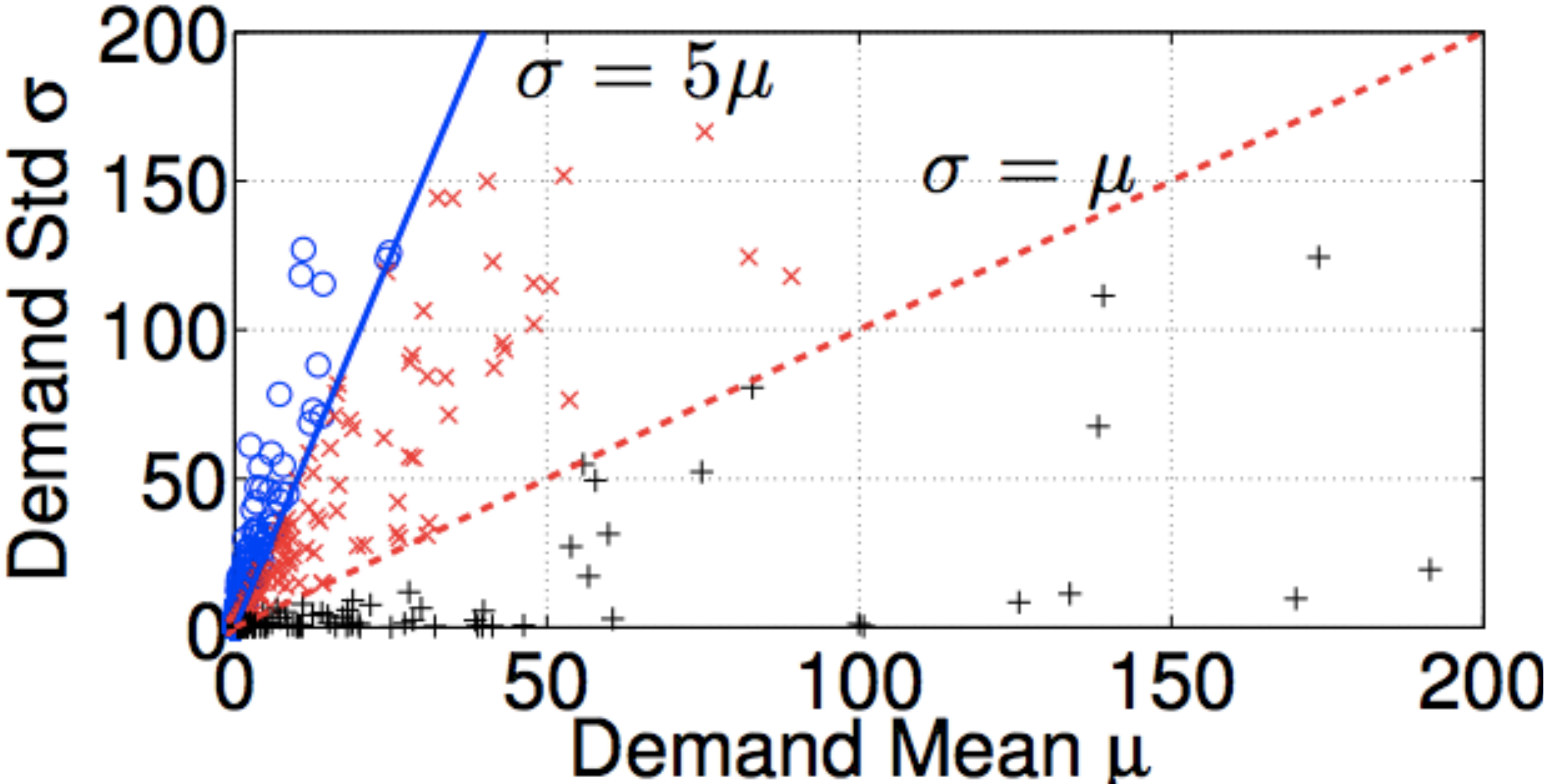
Trace Driven Simulation

Dataset and Preprocessing

- Google Cluster Traces
  - 900+ user's usage traces in a month
  - Users' computing demand data converted to IaaS instance demands



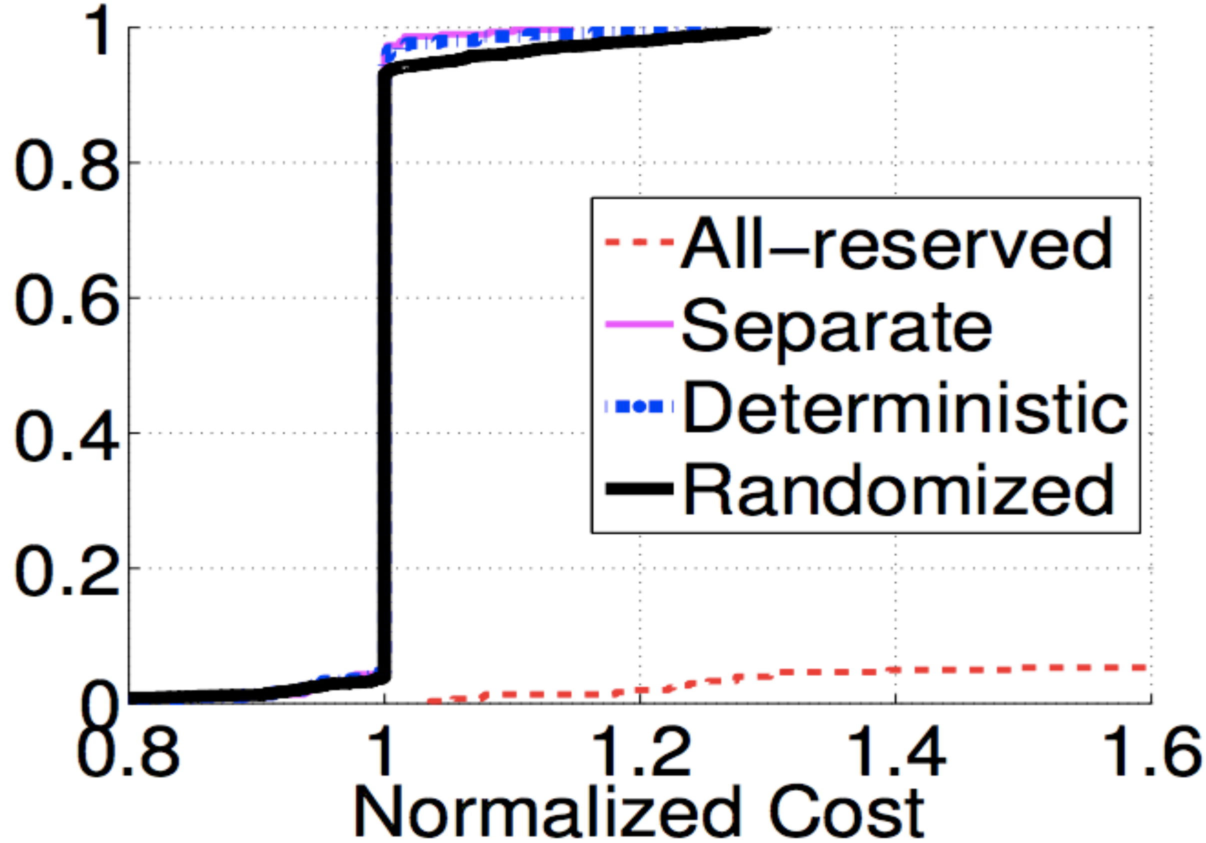
Users are classified into 3 groups based on demand fluctuation level



Five Algorithms

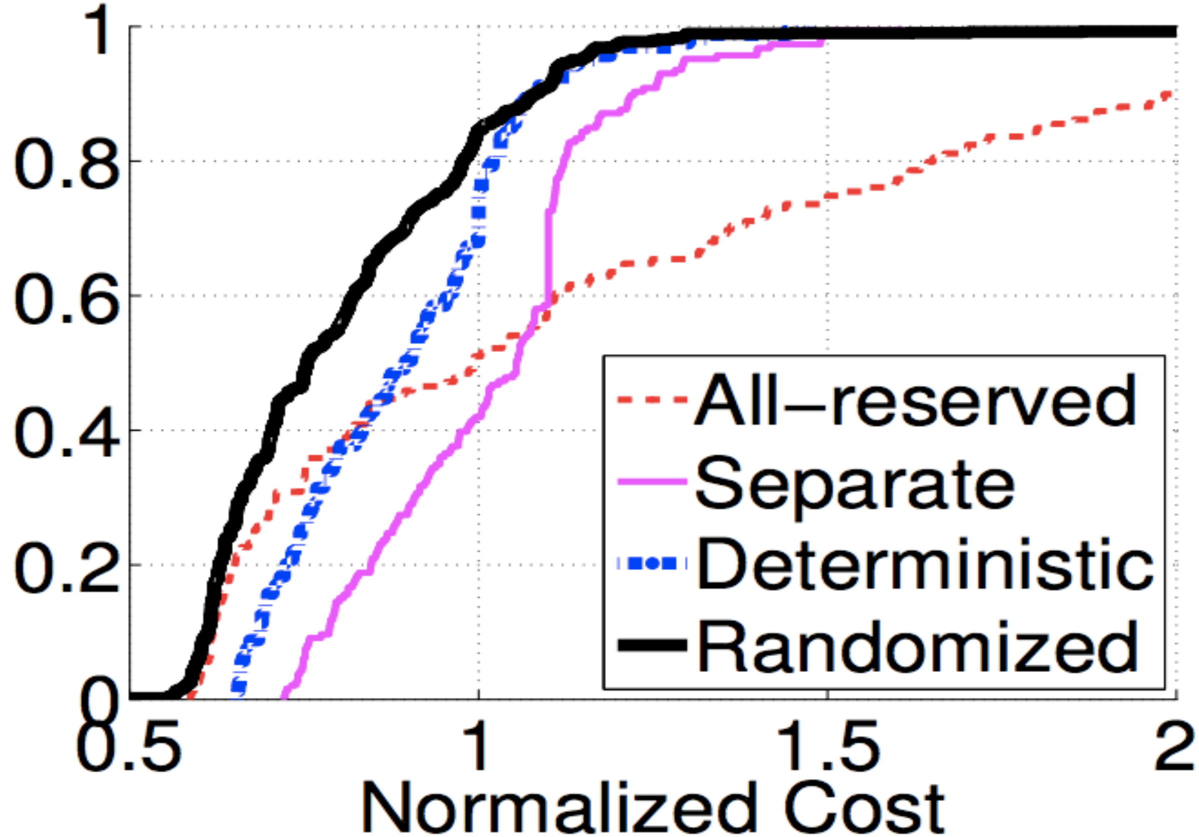
- All-On-Demand
- All-Reserved
- Deterministic
- Randomized
- Separate (At each demand level, run a Bahncard Algorithm)

CDF of Cost Normalized to the All-On-Demand



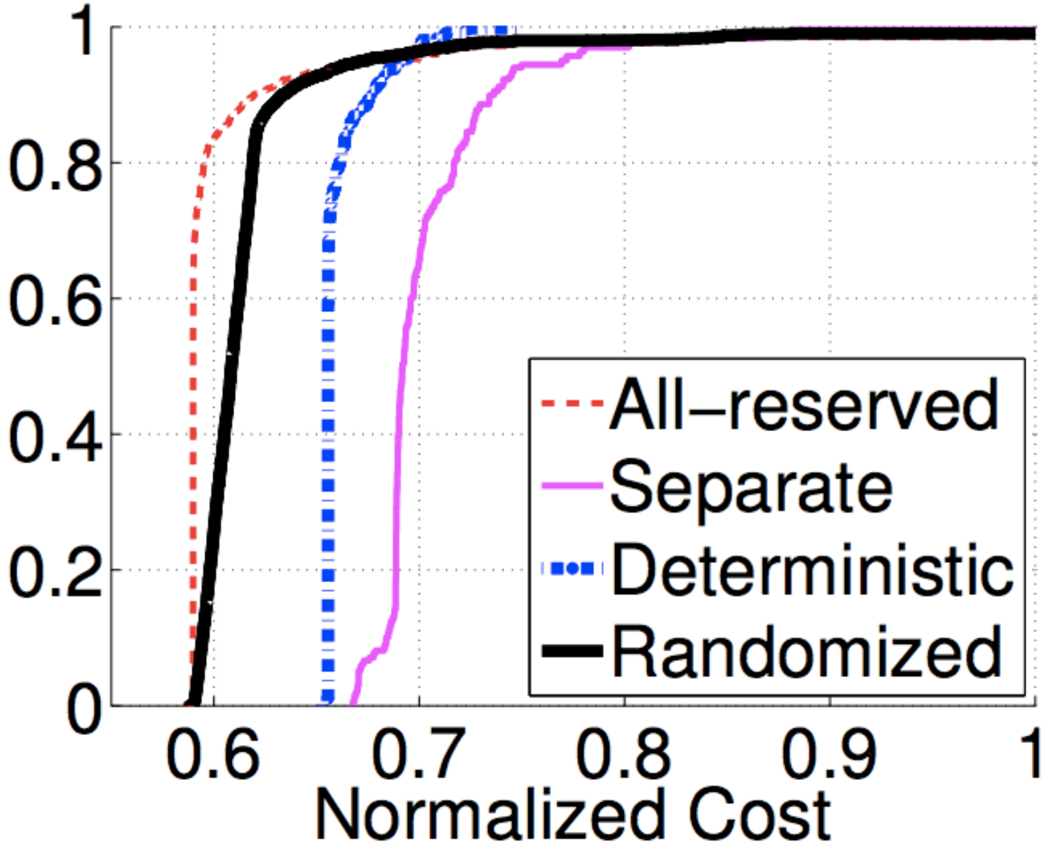
(b) Cost CDF (high fluctuation)

CDF of Cost Normalized to the All-On-Demand



(c) Cost CDF (medium fluctuation)

CDF of Cost Normalized to the All-On-Demand



(d) Cost CDF (stable demands)

Thank you
Q & A