# Dynamic Cloud Resource Reservation via Cloud Brokerage

**Wei Wang***, Di Niu+, Baochun Li*, Ben Liang*
* Department of Electrical and Computer Engineering, University of Toronto
+ Department of Electrical and Computer Engineering, University of Alberta

July 10, 2013

# Growing Cloud Computing Costs

## Drastic increase in enterprise spending on Infrastructure-as-a-Service (IaaS) clouds

- 41.7% annual growth rate by 2016 [CloudTimes'12]
- IaaS cloud will be the *fastest-growing* segment among all cloud services

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

| | Linux/UNIX Usage | Windows Usage |
|---|---|---|
| **Standard On-Demand Instances** | | |
| Small (Default) | $0.080 per Hour | $0.115 per Hour |
| Medium | $0.160 per Hour | $0.230 per Hour |
| Large | $0.320 per Hour | $0.460 per Hour |
| Extra Large | $0.640 per Hour | $0.920 per Hour |

## Reserved instances

Reservation fee + discounted price

Suitable for long-term usage commitment

# On-demand v.s. Reservation

| | Pros | Cons |
|---|---|---|
| **On-demand** | 1. Flexible<br>2. Fits sporadic workload | Expensive for long-term usage |
| **Reservation** | Cost efficient for long-term usage | 1. Long-term usage commitment<br>2. Expensive for sporadic workload |

# User's Problem

## Hard to choose among different pricing options

- Lacks sufficient expertise

## Cost savings due to the reservation option are not always possible

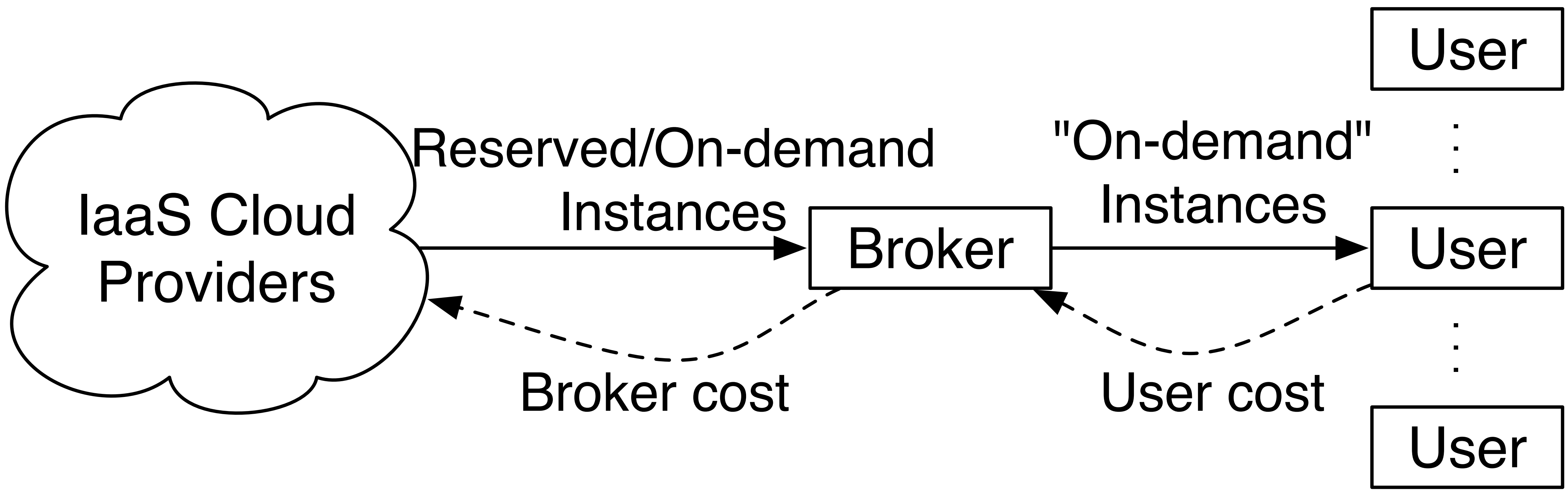
- Depends on the user's own demand pattern
- Must be long-term and heavy usage

# Can we go beyond the limitation of demand pattern of a single user and lower the cost?

# A Cloud Brokerage Service

## A cloud broker reserves a large pool of instances

## Users purchase instances from the broker in an "on-demand" fashion



# Why cloud broker?

# Better Exploiting Reservation Options

## Statistical multiplexing increases the utilization of reserved instances

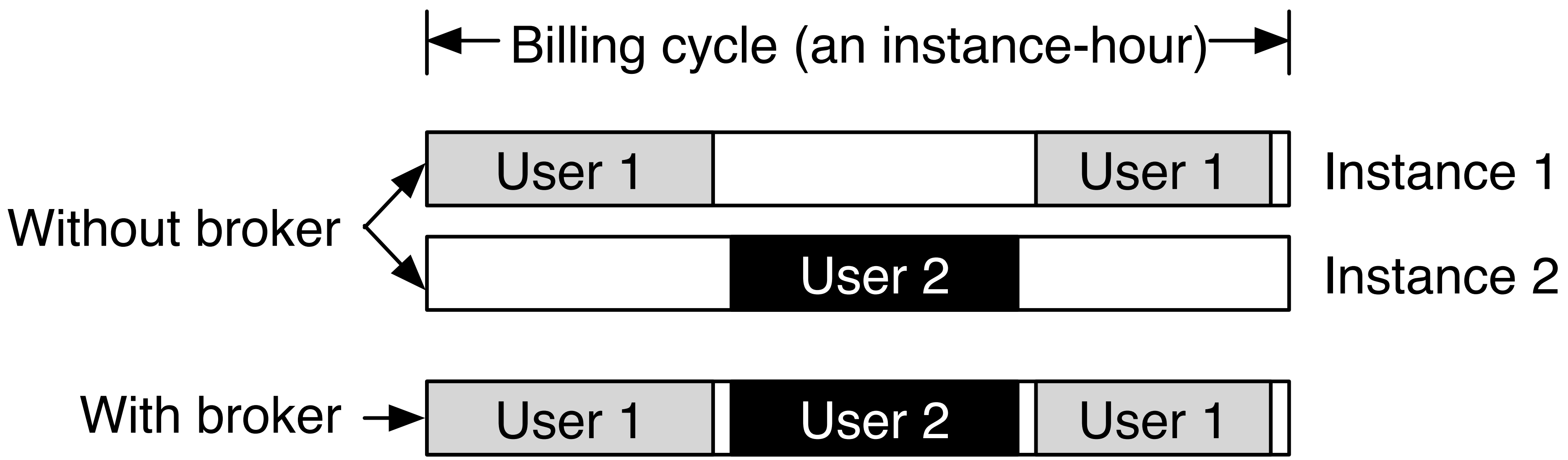
- Aggregating all users' demands smoothes out the "bursts"
- A flat demand curve is more friendly to reserved instances
- The "true cost" of reserved instance is reduced due to the increased instance utilization

# Reducing Wasted Cost Due to Partial Usage

## Alleviate the pricing inefficiency of on-demand instances

Partial usage is counted as a full billing cycle

The broker can *time-multiplex* partial usage



# Enjoying Volume Discounts

## Most IaaS clouds offer significant volume discounts

- Amazon provides 20% or even higher volume discounts in EC2
- The sheer volume of the aggregated demand makes cloud broker easily qualify for such discounts


# A Win-Win Solution

## Users receive a lower price when trading with the broker

- No upfront payment for reservation
- No money wasted on idled reservation instances

## Broker makes profit by leveraging the wholesale (reservation) model

- A significant price gap between on-demand and reserved instances
- Aggregate demand is more amenable to the reservation option

# How many instances should a broker reserve?

# On-demand and Reserved Pricing

## On-demand instances

Fixed hourly rate *p*

## Reserved instances

Upfront reservation fee: $\gamma$

Reservation period: $\tau$

## Instances reserved at time t: $r_t$

## # of reserved instances that are effective at time t

$$n_t = \sum_{i=t-\tau+1}^{t} r_i$$

# Dynamic Resource Reservation

**Cloud users submit demand predictions to the broker**

**Broker reserves instances based on the aggregate demand**

$d_1, \ldots, d_T$

**Total cost = Reservation cost + On-demand cost**

$$\sum_{t=1}^{T} r_t \gamma + \sum_{t=1}^{T} (d_t - n_t)^+ p$$

where,

$$n_t = \sum_{i=t-\tau+1}^{t} r_i$$

# of reserved instances that are effective at *t*

# The Cost Minimization Problem

**Make dynamic reservation decisions $r_1, \ldots, r_T$ to accommodate demands $d_1, \ldots, d_T$**

$$\min_{\{r_1,\ldots,r_T\}} \quad \text{cost} = \sum_{t=1}^{T} r_t \gamma + \sum_{t=1}^{T} (d_t - n_t)^+ p$$

**This is an integer program!**

# Optimal Solution: Dynamic Programming

# The Curse of Dimensionality

## High dimensional dynamic programming

High dimensional state: $\mathbf{s}_t := (t, x_1, \ldots, x_{\tau-1})$

$x_i$ : # of instances that are reserved no later than *t* and remain effective at *t+i*

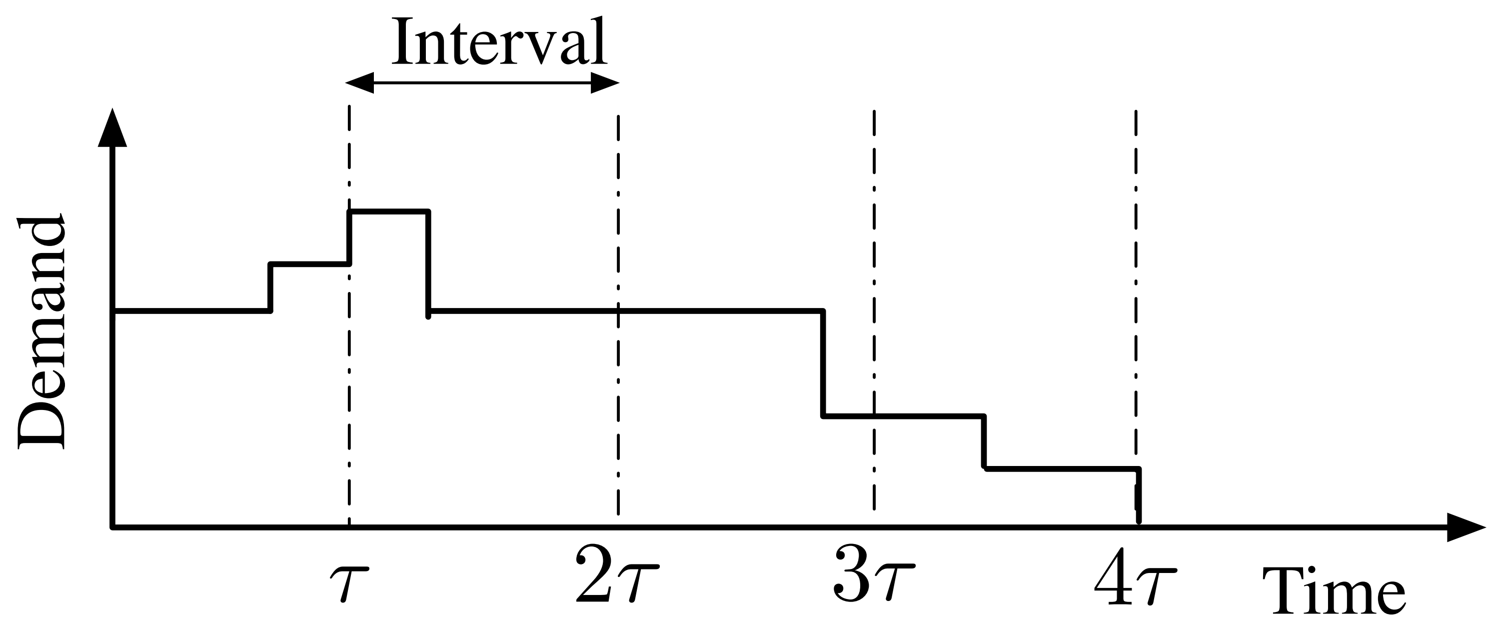
## Exponential time and space complexity

The curse of dimensionality

# Approximate Solution

# A 2-Competitive Heuristic

## Segment the demand into intervals each spanning one reservation period




## Make optimal instance reservation decisions per interval

# Optimal Instance Reservation within an Interval

**Stratify demand into levels**

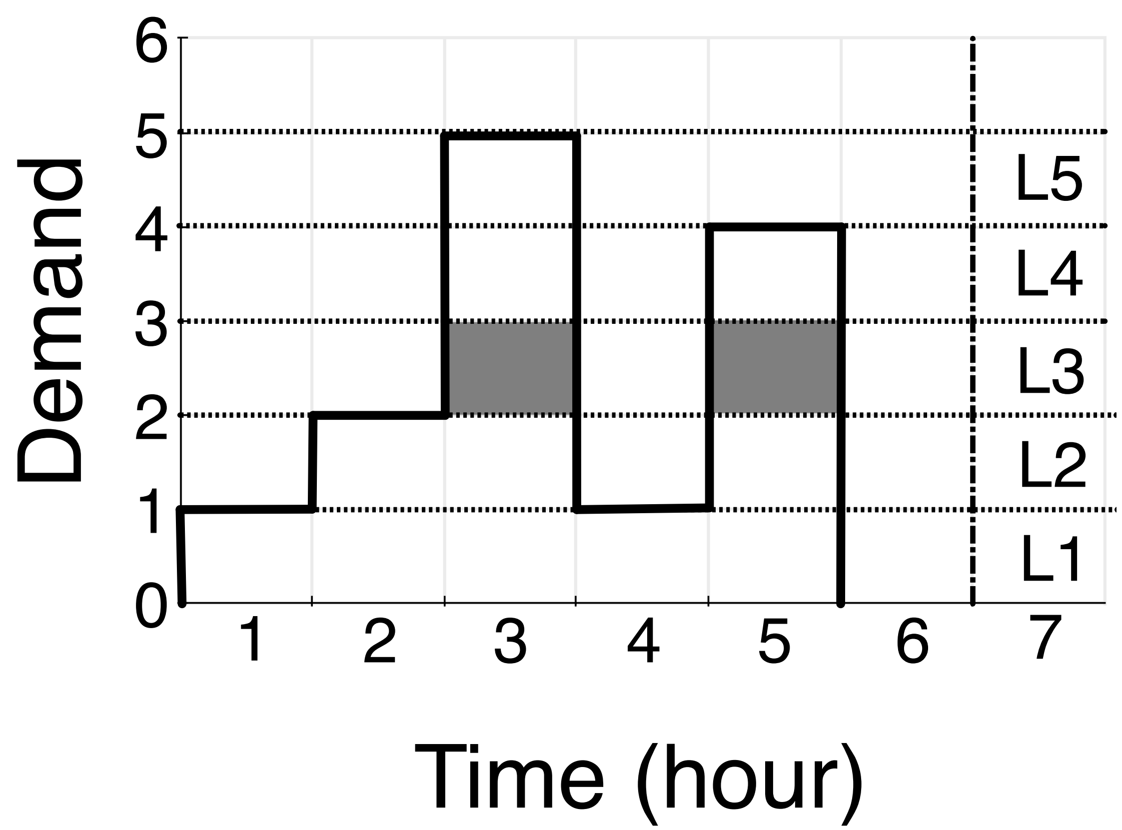
**For each level, decide if a reserved instance should be used**

**Example**

On-demand rate: $1 per hour

Reservation: $2.5 for 6 hours

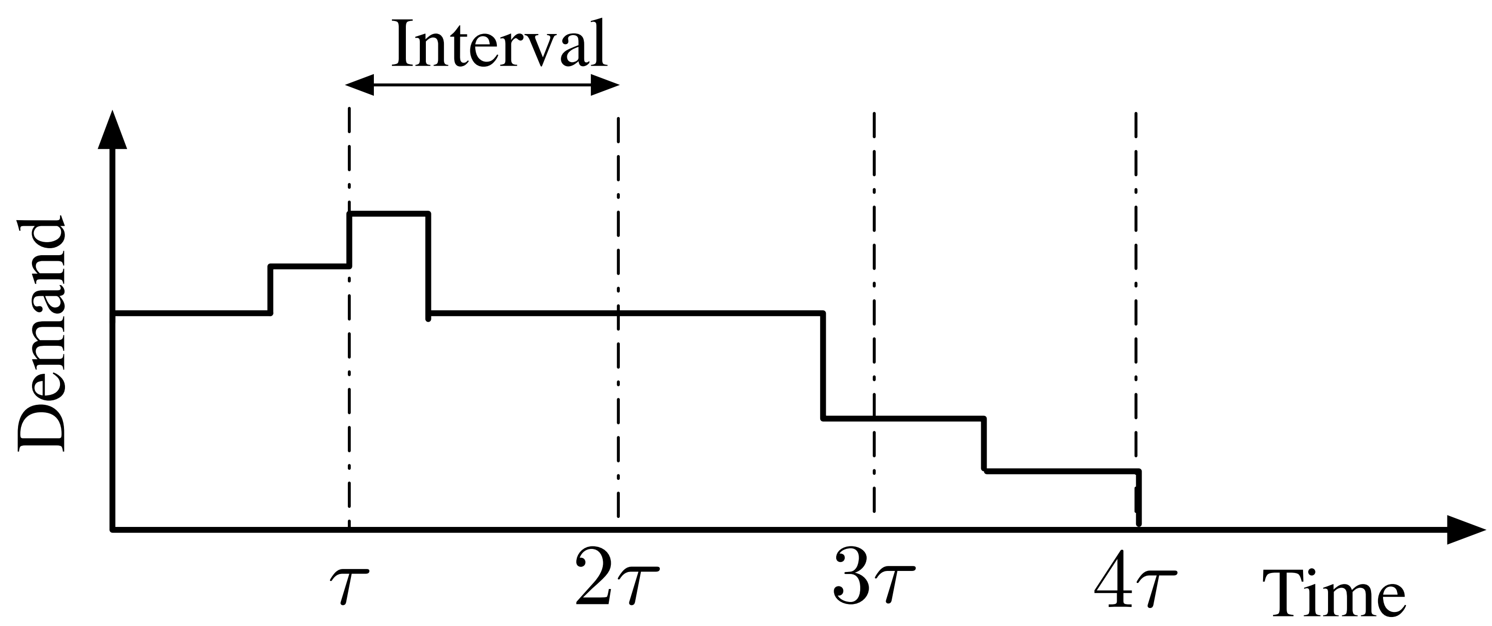
Should reserve when instance usage >= 3 hours



# Cost Performance

## Per-interval reservation is 2-competitive

Incurs *at most* twice the optimal cost in the worst case

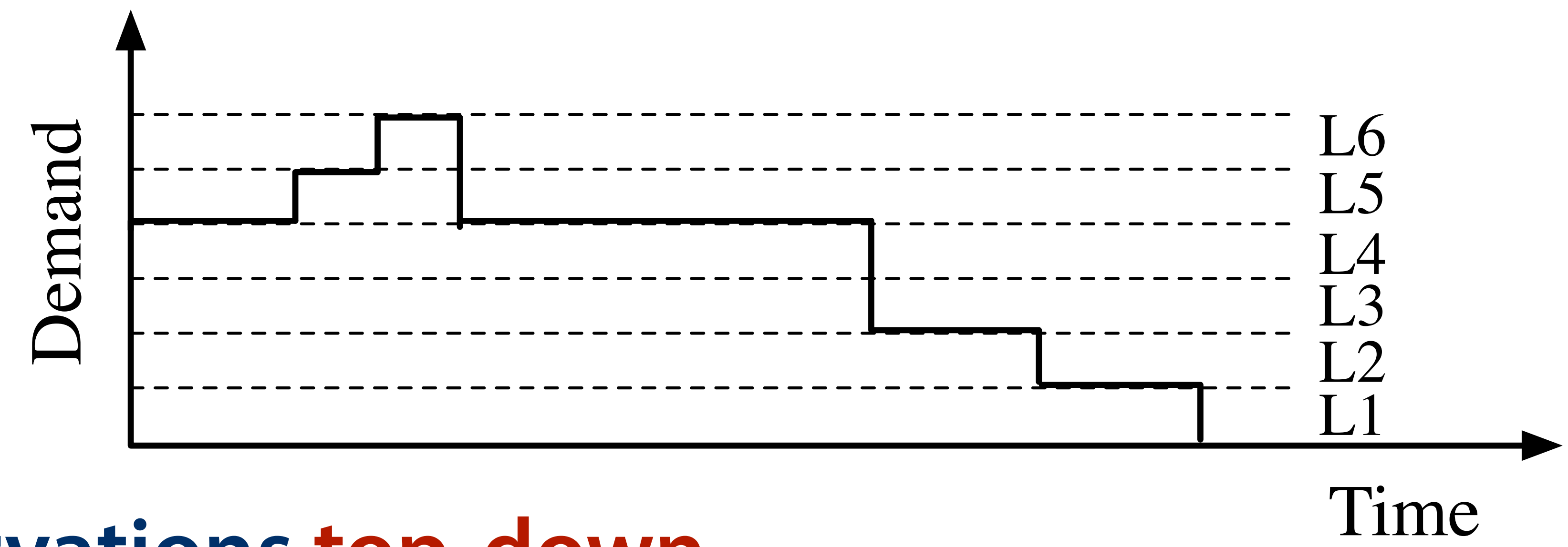



All reservations are made at the beginning of the interval

# An Improved Greedy Algorithm

## Do not segment demand into intervals

## Stratify demands into levels




## Make reservations top-down

At each level, apply dynamic programming

$$V_l(t) = \min\{V_l(t-\tau)+\gamma,\ V_l(t-1)+c_l(t)\}$$

## Strictly better than Per-Interval Reservation, and is also 2-competitive

# When demand predictions are unavailable

# Online Algorithm

## Make instance reservation decisions without future information

**Algorithm 3** Online Reservation Made at Time $t$

1. Let $g_i = (d_i - n_i)^+$ for all $i = t - \tau + 1, \ldots, t$.
2. Run Algorithm 1 with $g_{t-\tau+1}, \ldots, g_t$ as the input demands. Let $x$ be its output.
3. Reserve $r_t = x$ instances at time $t$.
4. Update $n_i = n_i + r_t$ for all $i = t - \tau + 1, \ldots, t + \tau - 1$.

## The best that we can do [Wang et al. ICAC'13]

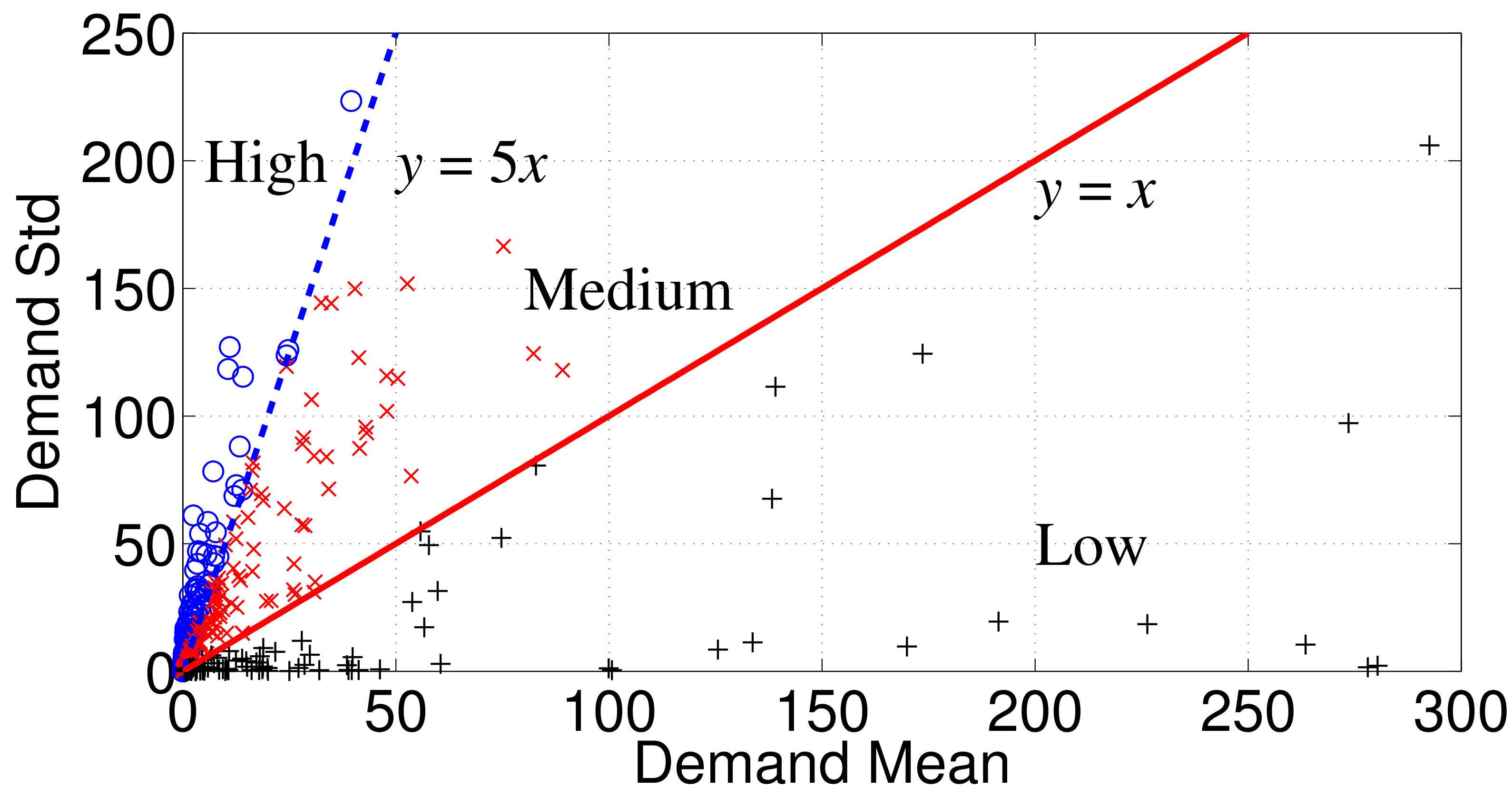
2-competitiveness for the *deterministic* online algorithm

# Trace-Driven Simulations

# Dataset and Preprocessing

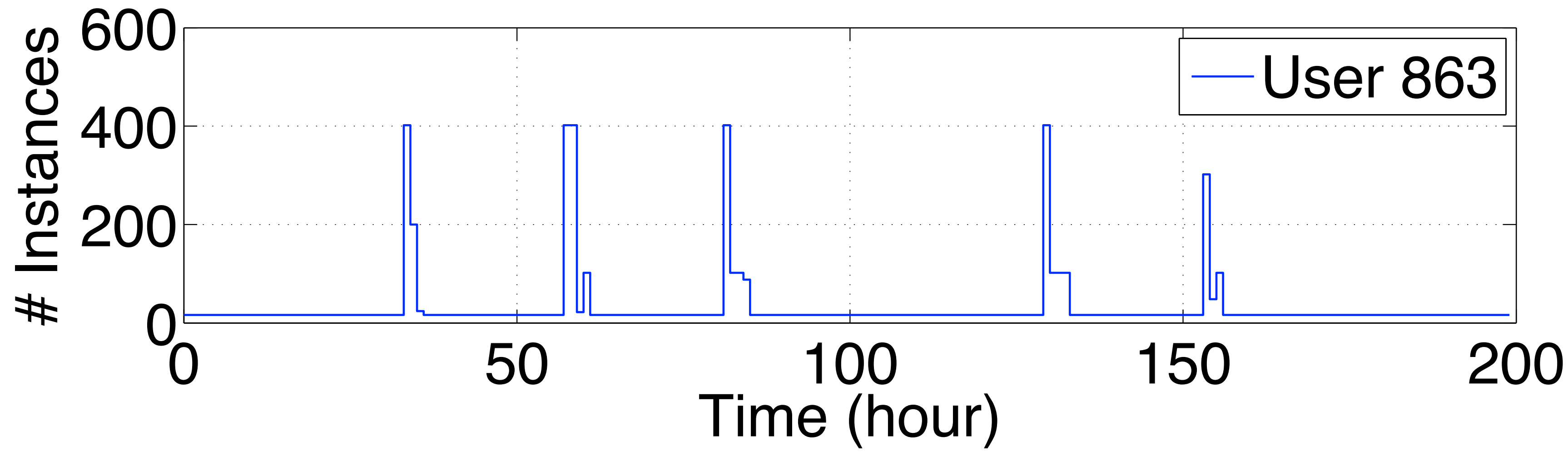
## Google cluster-usage traces

- 900+ users' usage traces on a 12K-node Google datacenter
- We convert users' computing demand data to IaaS instance demand
- Users are classified into 3 groups based on demand fluctuation level
  - Standard deviation vs. mean in hourly demand

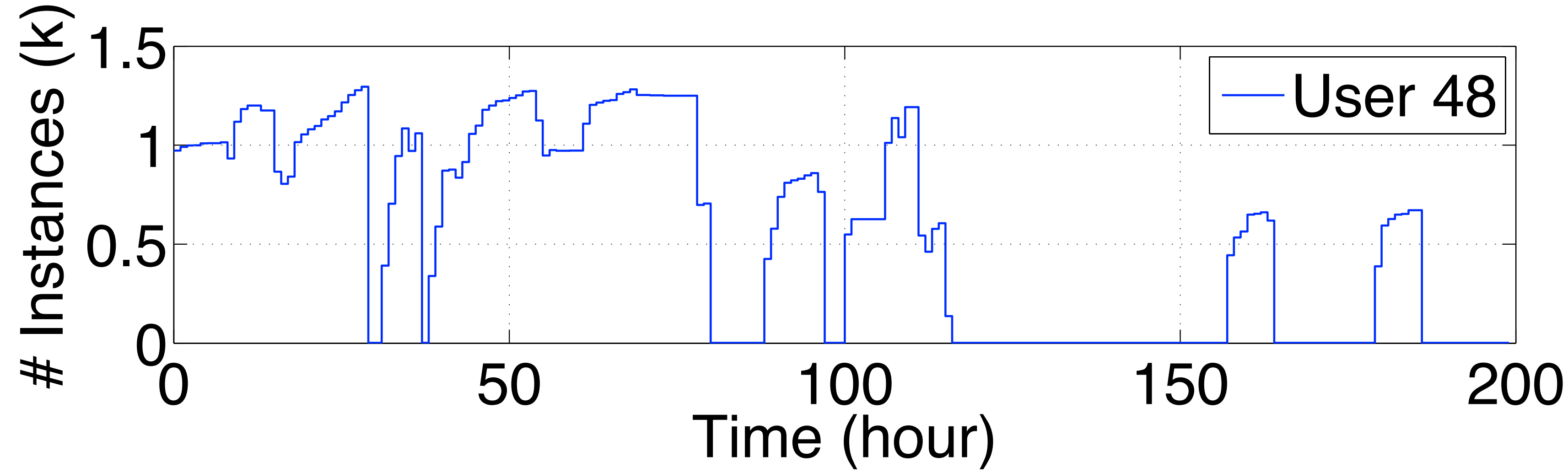


# Demand Curve

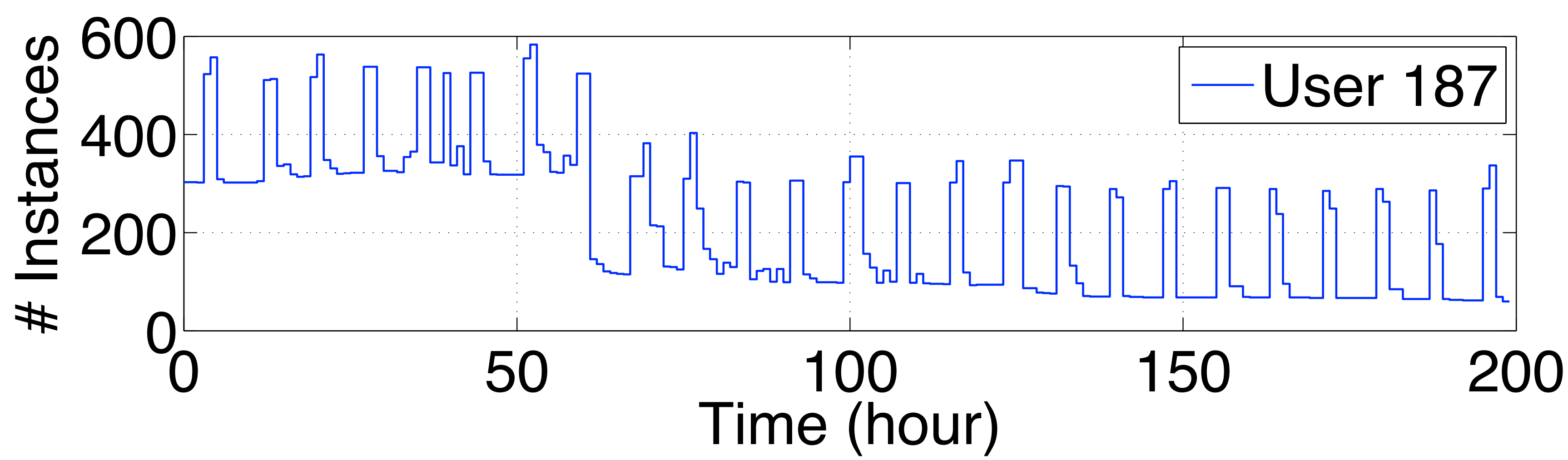
Highly fluctuated



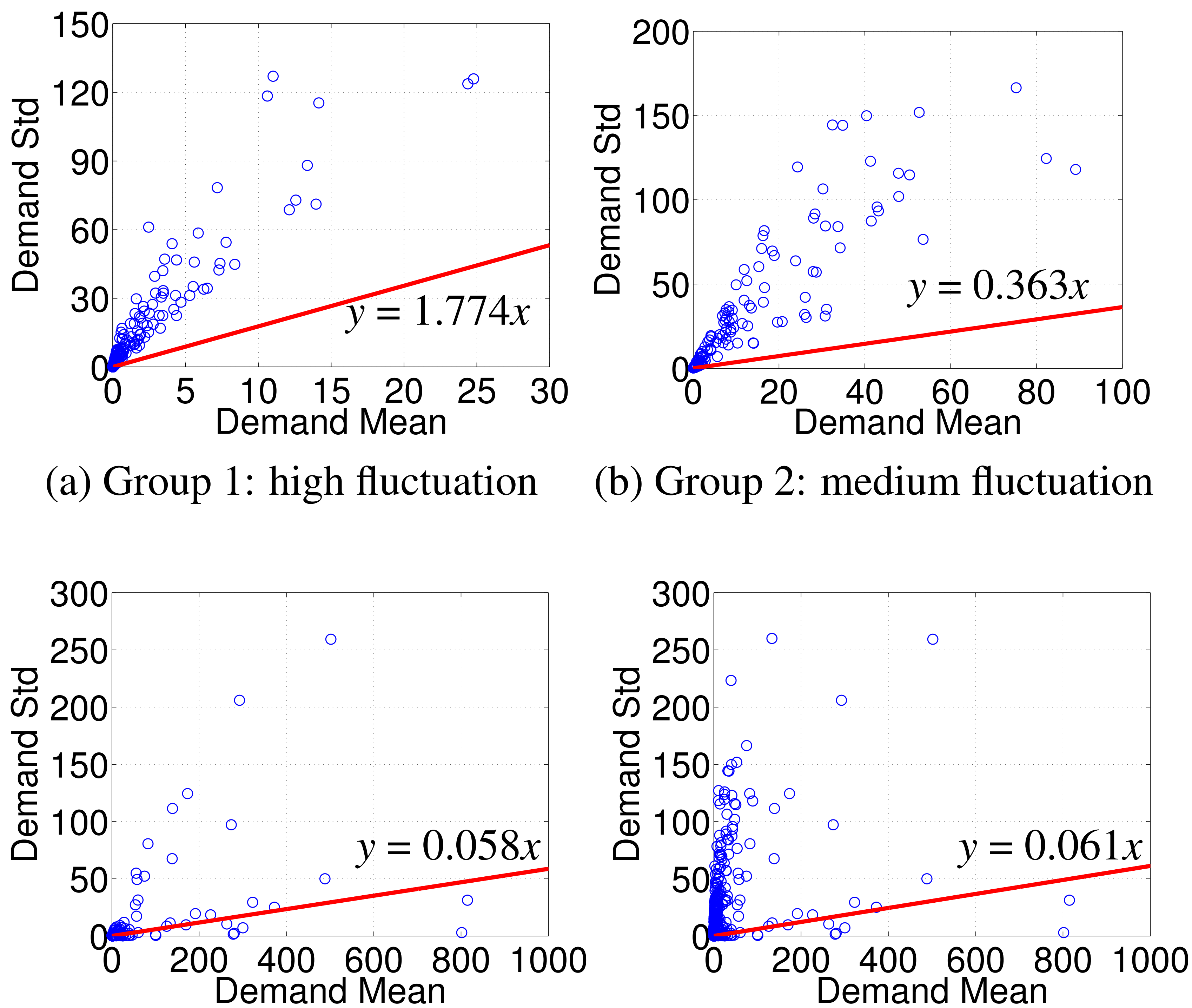

Medium fluctuation




Relatively stable



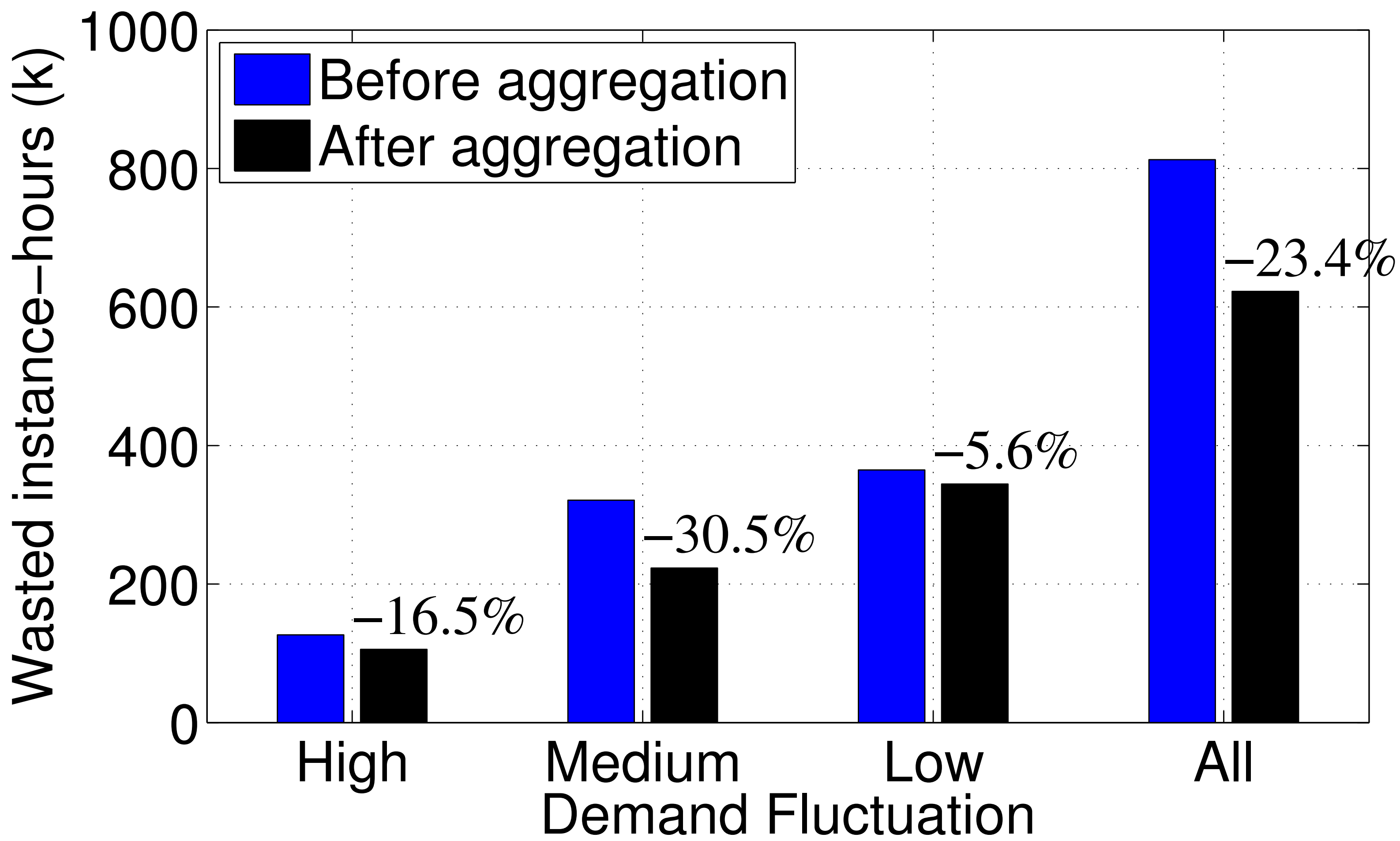
# Aggregation Smoothes Out Demand Bursts



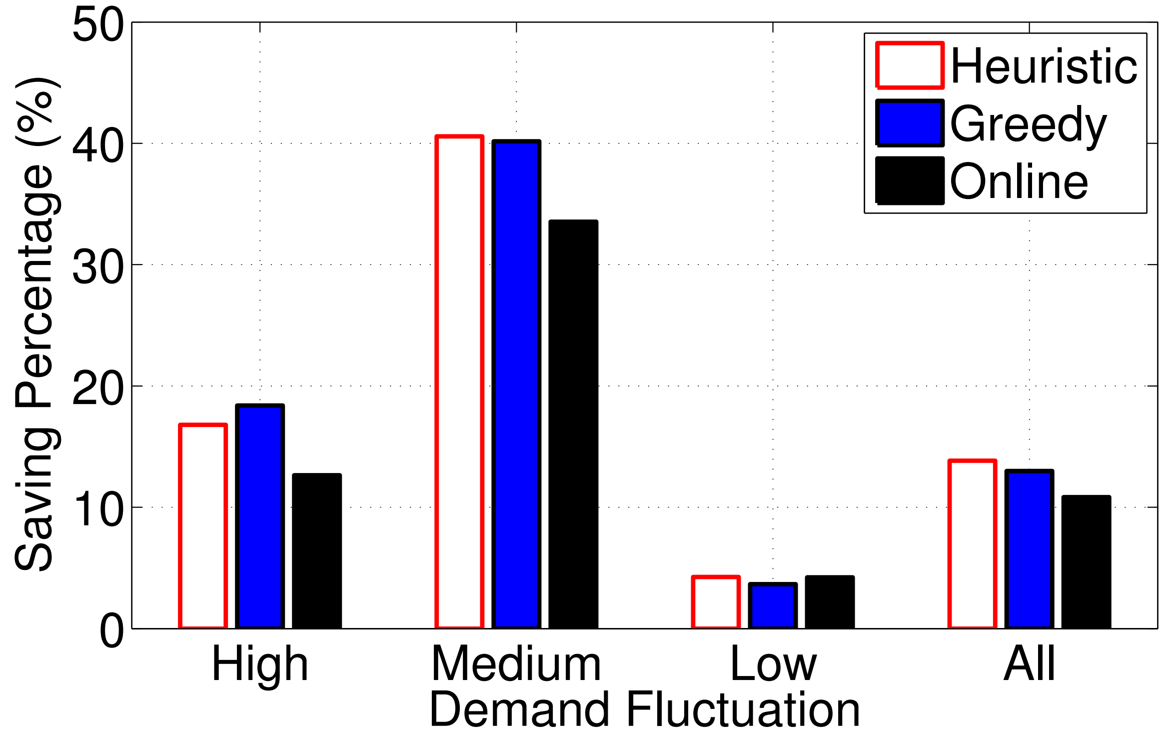

(a) Group 1: high fluctuation (b) Group 2: medium fluctuation

(c) Group 3: low fluctuation (d) All the users.

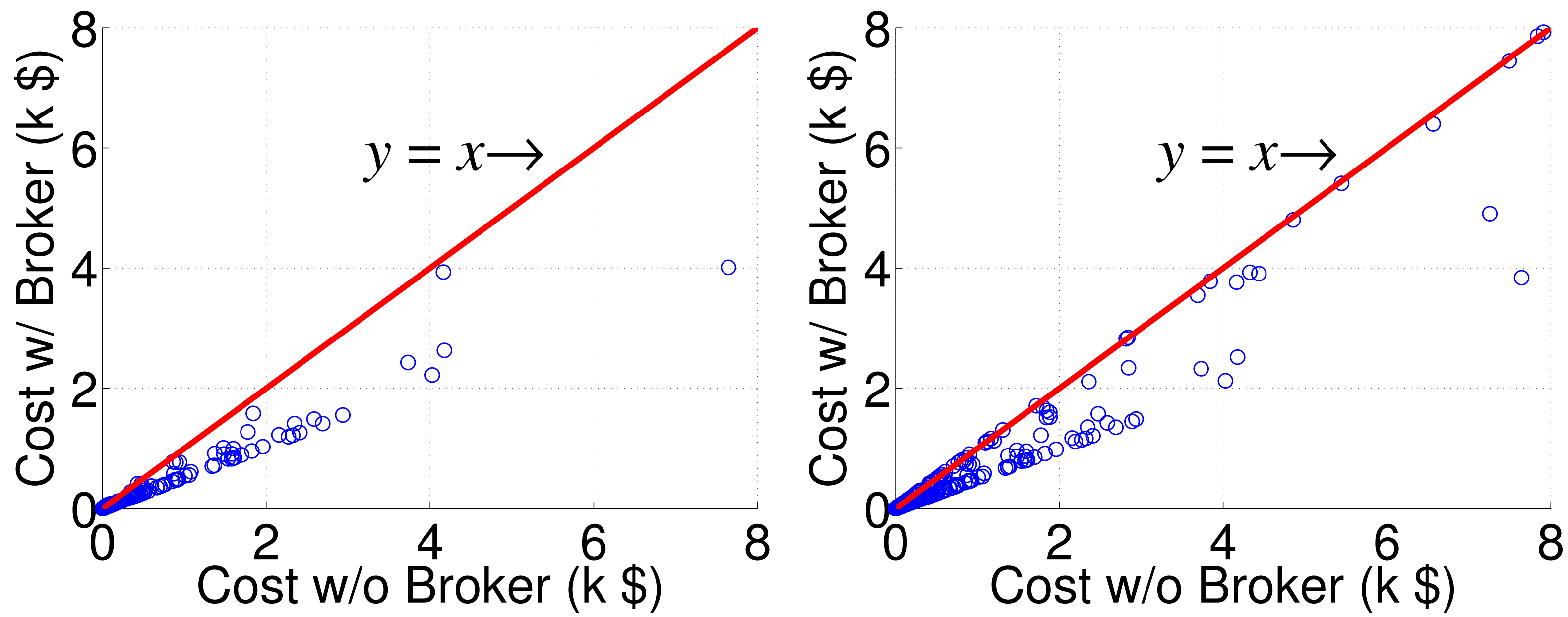
# The Reduction of Partial Usage



# Cost Savings Due to the Broker




No volume discount




(a) Group 2: medium fluctuation

(b) All the users

# Conclusions

## We propose a smart cloud brokerage service

- Reserves a pool of instances to serve the aggregated demand
- Leverages the price gap between the wholesale and retail model to reap the profit while offering lower price to cloud users
- Cloud users purchase instances from the broker as if instances were offered on demand

## Design and analyze three instance reservation algorithms for the broker and evaluate them via trace-driven simulations

- More detailed analysis of online algorithms are given in our follow-up work [Wang et al. ICAC'13]

# Thanks!

http://iqua.ece.toronto.edu/~weiwang/