Bidding for Highly Available Services with Low Price in Spot Instance Market

Weichao Guo, Kang Chen, Yongwei Wu, and Weimin Zheng

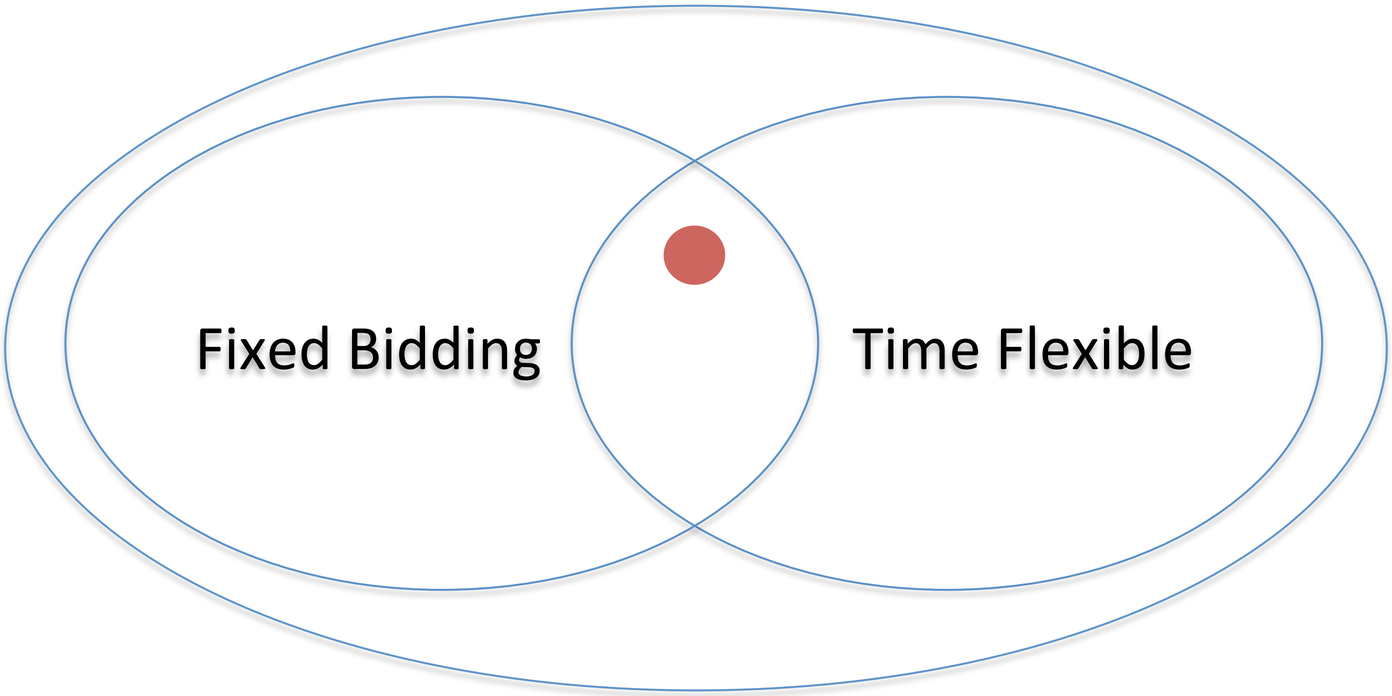
Tsinghua University

On-Demand Instance vs. Spot Instance

- Price Model
  - On-Demand: pre-define
  - Spot: fluctuate based on supply & demand
- Failure Model
  - On-Demand: SLAs
  - Spot: terminated when spot price exceeds bid

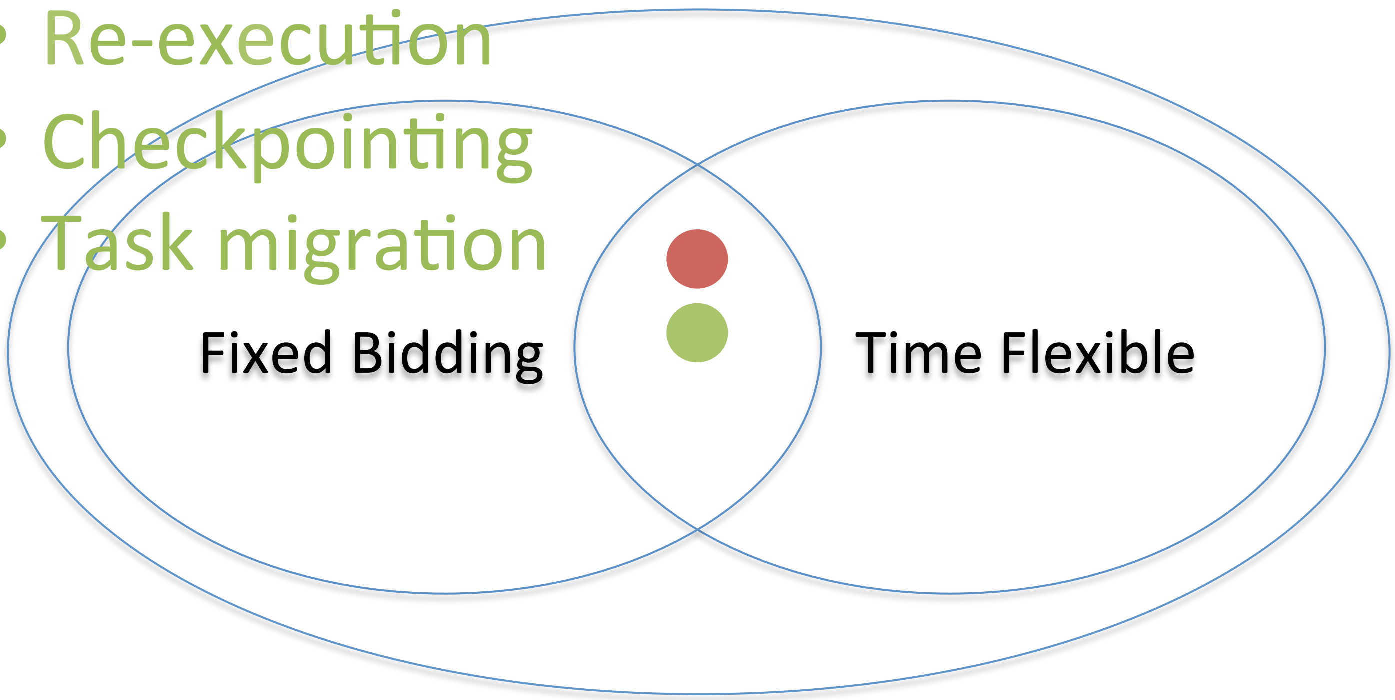
Cost-Efficient Computing with Spot Instances

- Optional accelerators for MapReduce jobs [HotCloud '10]



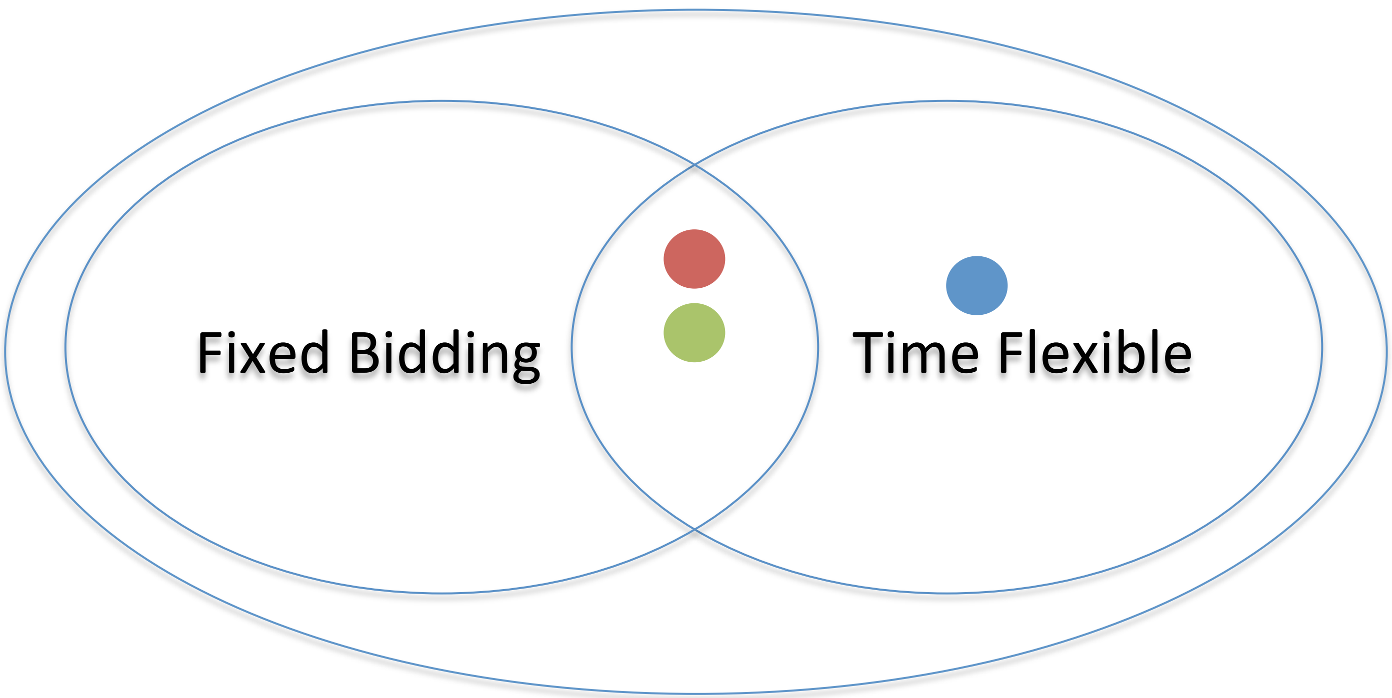
Cost-Efficient Computing with Spot Instances

- Adapting FT techniques for divisible parallel jobs [CLOUD ‘10, HotCloud ’11, etc.]
  - Re-execution
  - Checkpointing
  - Task migration

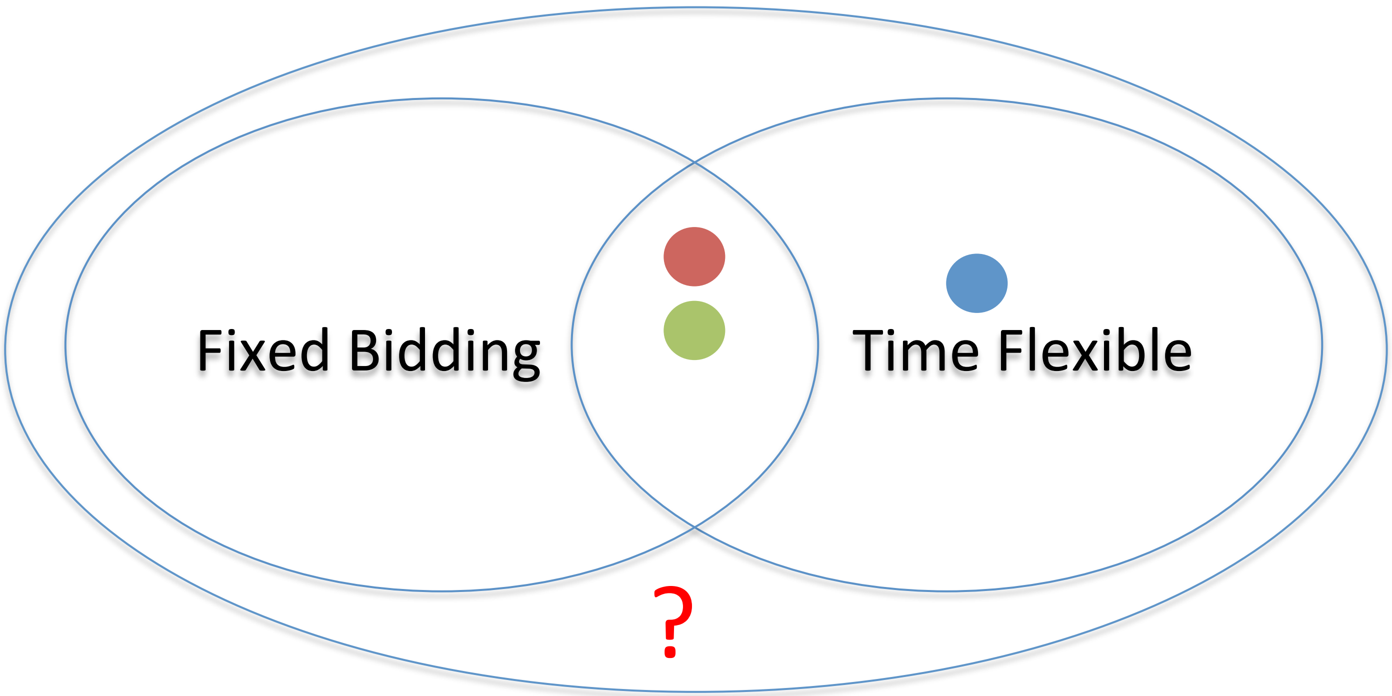


Cost-Efficient Computing with Spot Instances

- Profit aware dynamic bidding from a cloud service broker's perspective [INFOCOM '12]



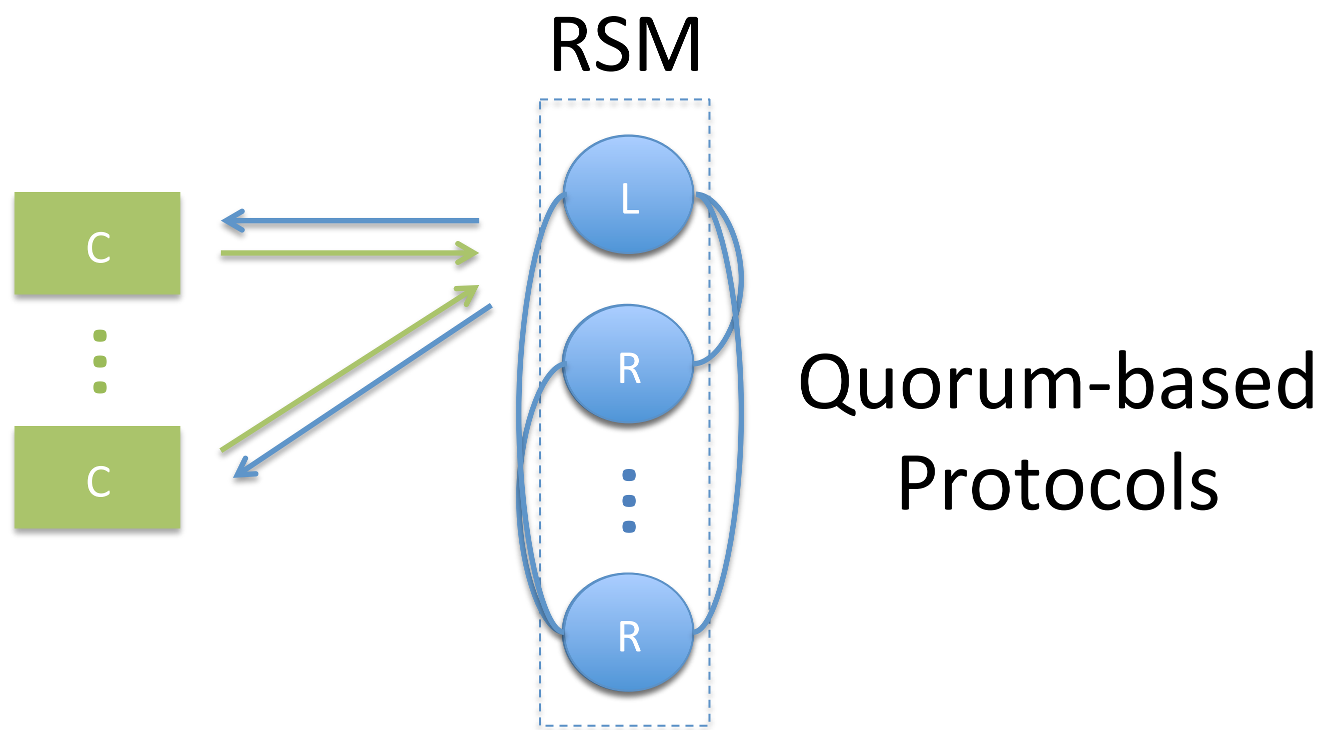
Cost-Efficient Computing with Spot Instances



High available services with low prices?

Distributed Service Basics

- State Machine Replication



Distributed Service Basics

- Acceptance Set $\mathcal{A}$
  - Intersection $\forall S, T \in \mathcal{A}, S \cap T \neq \emptyset$
  - Monotonicity $\forall T \supseteq S, S \in \mathcal{A} \longrightarrow T \in \mathcal{A}$

- Availability

$$A_{\mathcal{A}} = \sum_{S \in \mathcal{A}} \left(\prod_{i \in S} (1 - p_i) \prod_{j \in \overline{S}} p_j \right)$$

Distributed Service with Spot Instances

RSM	O	0.044 $/H	RSM	S	0.008 $/H
	O	0.044 $/H		S	0.008 $/H
	O	0.047 $/H		S	0.009 $/H
	O	0.044 $/H		S	0.009 $/H
	O	0.044 $/H		S	0.009 $/H
		~25.5 s			>1500 s

Distributed Service with Spot Instances

RSM			RSM			
	O	0.044 $/H		S	0.008 $/H	
	O	0.044 $/H		S	0.008 $/H ↑	
	O	0.047 $/H		S	0.009 $/H ↑	
	O	0.044 $/H		S	0.009 $/H	S
	O	0.044 $/H		S	0.009 $/H	S

~25.5 s

>1500 s

To Improve Availability:

- Higher Bids
- More nodes

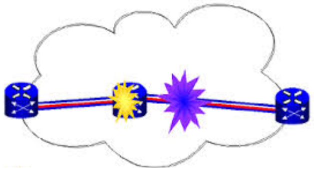
Contributions

- Spot Instance Failure Model
  - Availability analysis
  - Failure probability estimation
- Bidding Framework
  - Cost minimization problem
  - Online bidding strategy

Outline

- Problem Formalization
  - Spot Instance Failure Model
  - Cost Minimization Problem
- Bidding Framework
  - Failure Probability Estimation
  - Online Bidding
- Experiment

Spot Instance Failure Model

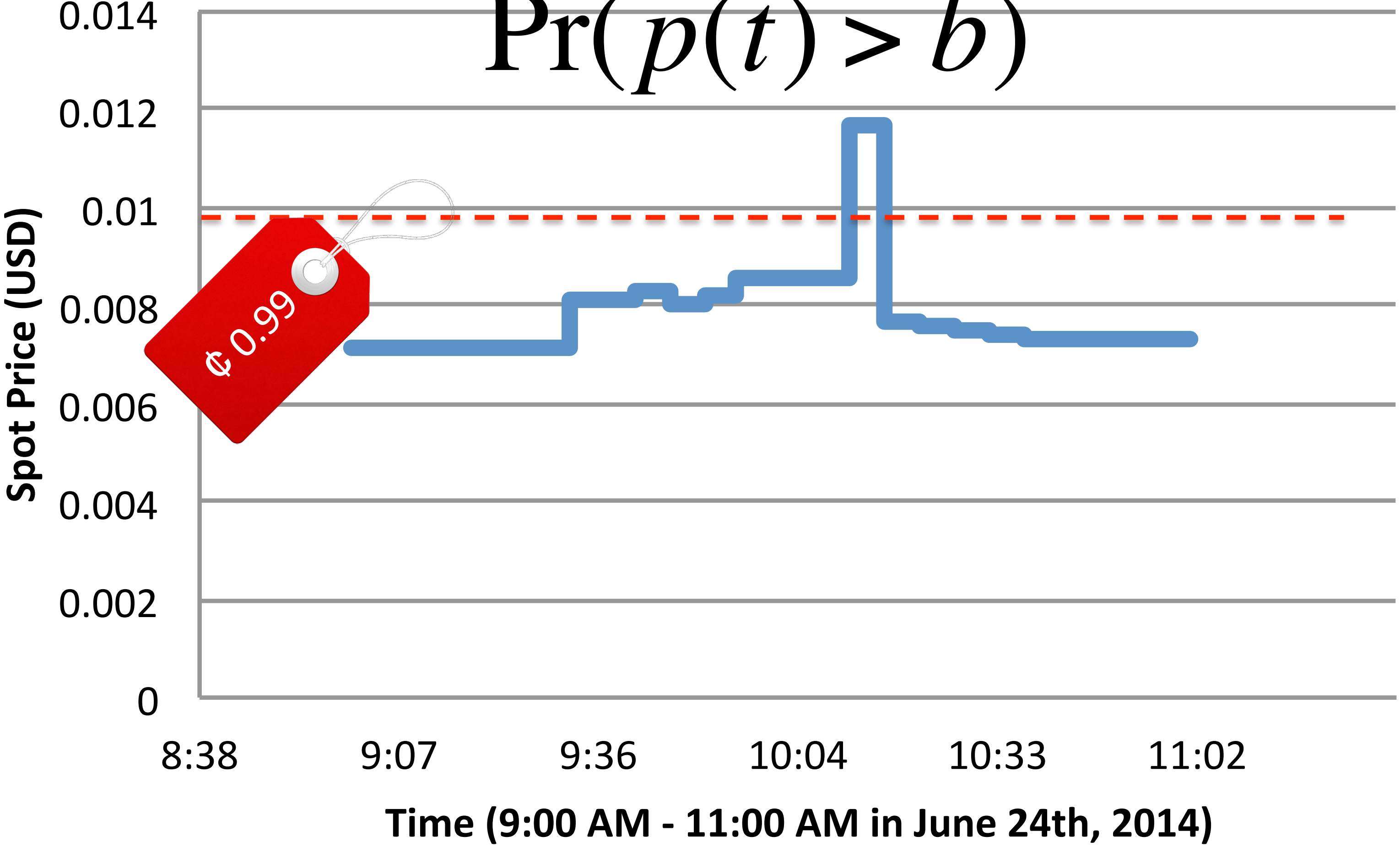


On-Demand Instance Failures

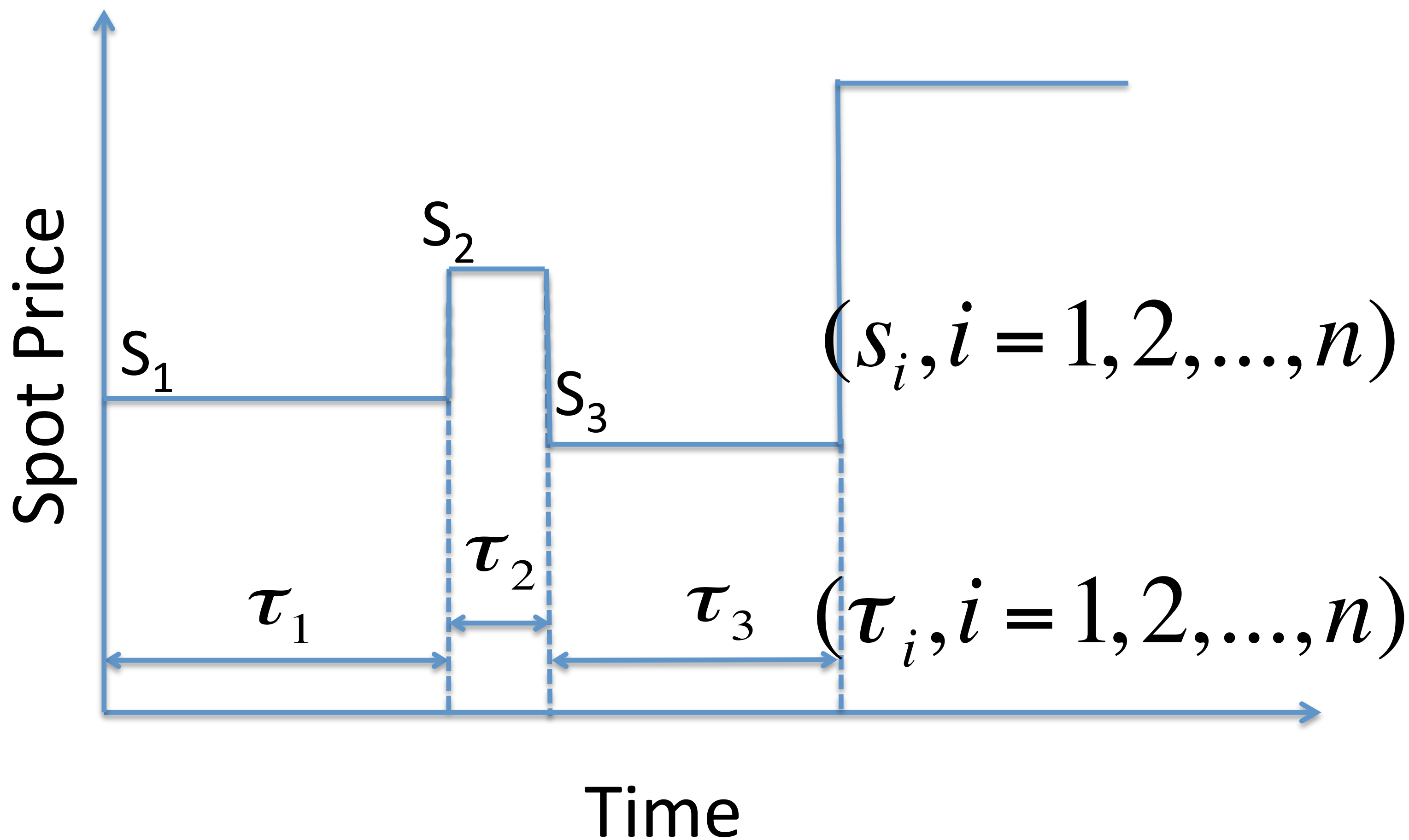
Out-of-bid Failure

Spot Instance Failure Model

$$\Pr(p(t) > b)$$



Spot Instance Failure Model



Spot Instance Failure Model

- Semi-Markovian Chain's Stochastic Kernel

$$Q(i,j,k) = (q_{i,j,k}; s_i, s_j \in \mathcal{S}, k \in \mathcal{T})$$

where

$$q_{i,j,k} = Pr(S_{n+1} = s_j, S_n = s_i, \tau_n = k)$$

Spot Instance Failure Model

- Failure probability at time *t*

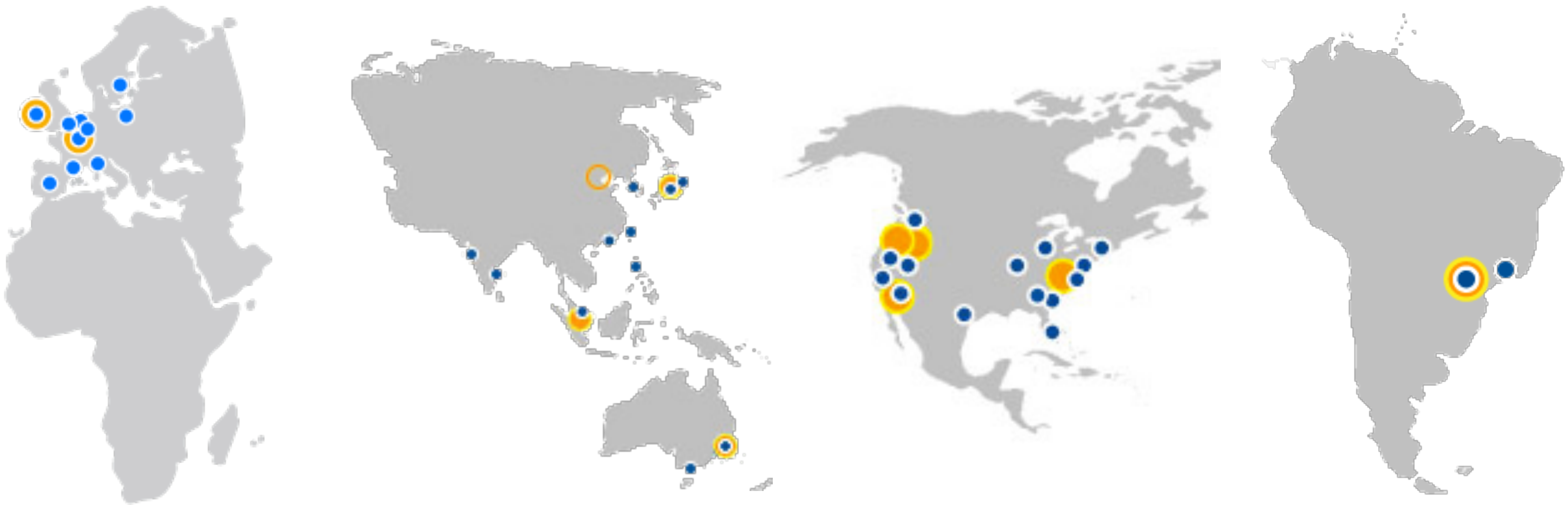
$$FP(t) = 1 - (1 - FP') \cdot (1 - Pr(p(t) > b))$$

- Failure probability in time duration *d*

$$\int_0^d FP(t)dt$$

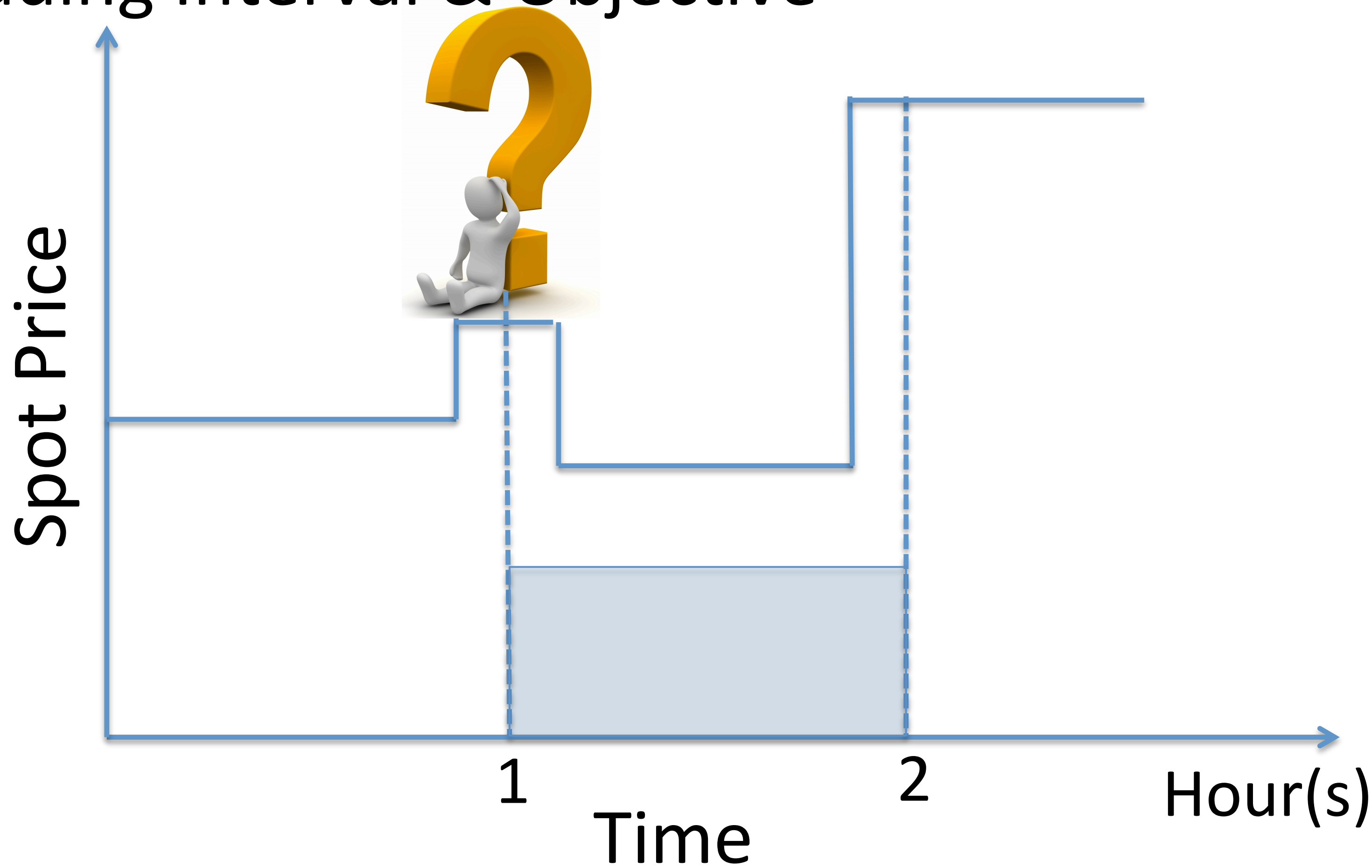
Cost Minimization Problem

Availability Zones & i.i.d.



Cost Minimization Problem

- Bidding Interval & Objective



Cost Minimization Problem

$$\min \sum_{i=1}^{n} b_i$$

s.t.

$$\sum_{i=1}^{n} \epsilon(b_i - p_i) \geq m$$

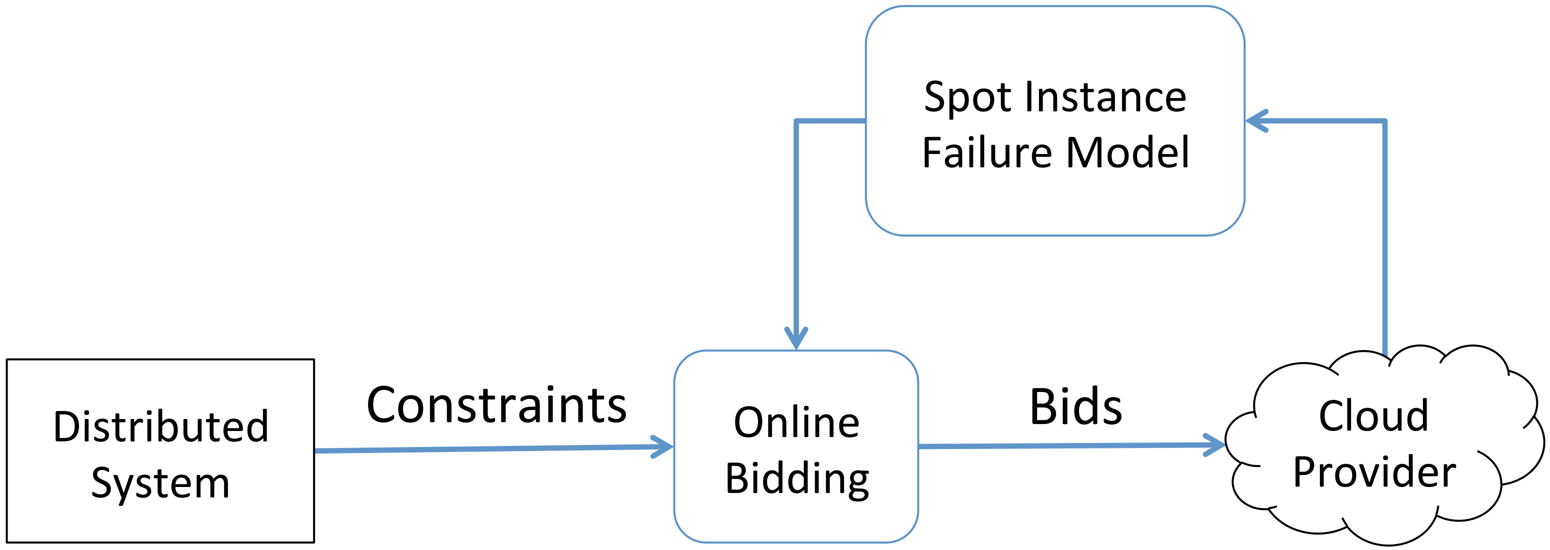
and

$$A_{\mathcal{A}_o(S_o,\mathbf{FP}')} - A_{\mathcal{A}_o(S_s,FP(\mathbf{b}))} < \varepsilon$$

Outline

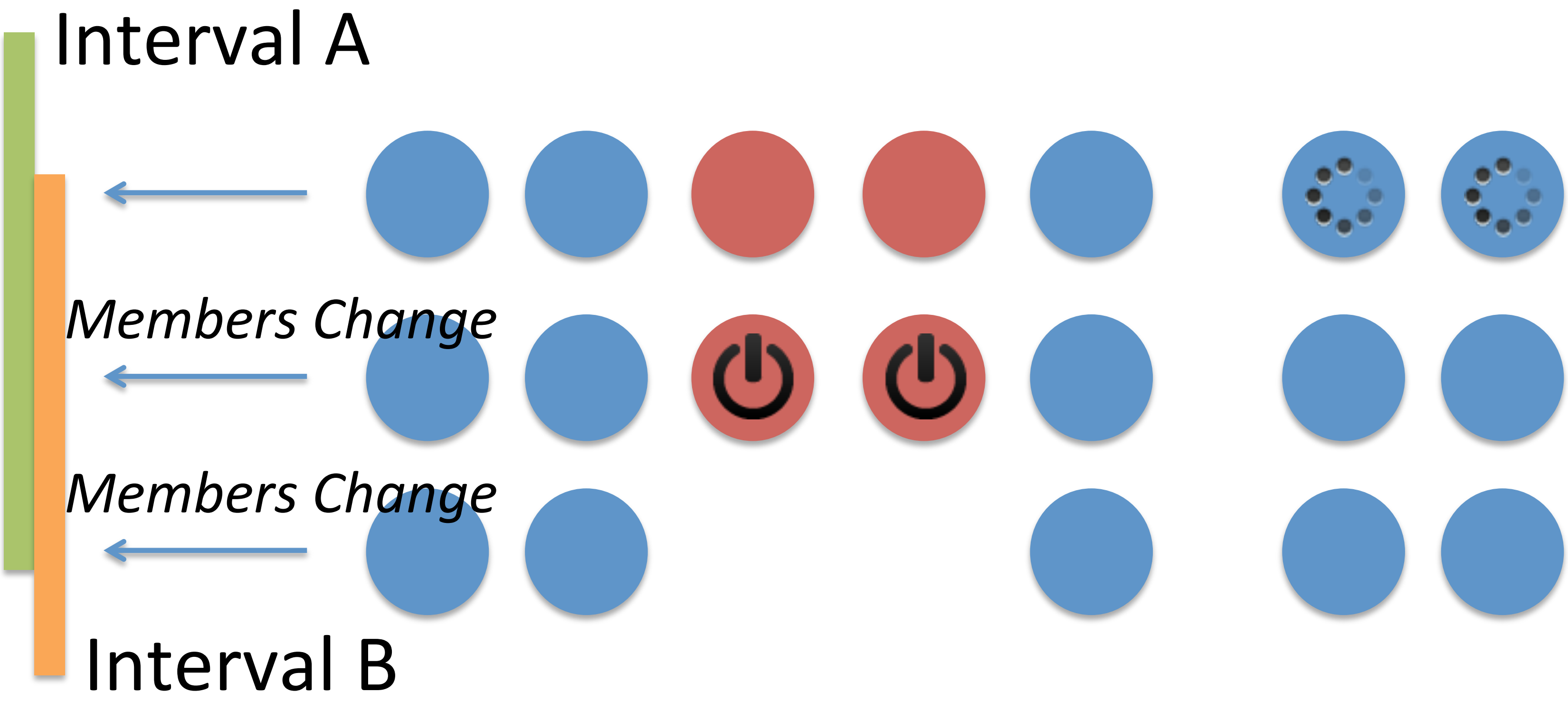
- Problem Formalization
  - Spot Instance Failure Model
  - Cost Minimization Problem
- Bidding Framework
  - Failure Probability Estimation
  - Online Bidding
- Experiment

Bidding Framework



Bidding Framework

- Keeping Safety at Reconfiguration



Failure Probability Estimation

- Maximum Likelihood Estimator (MLE)

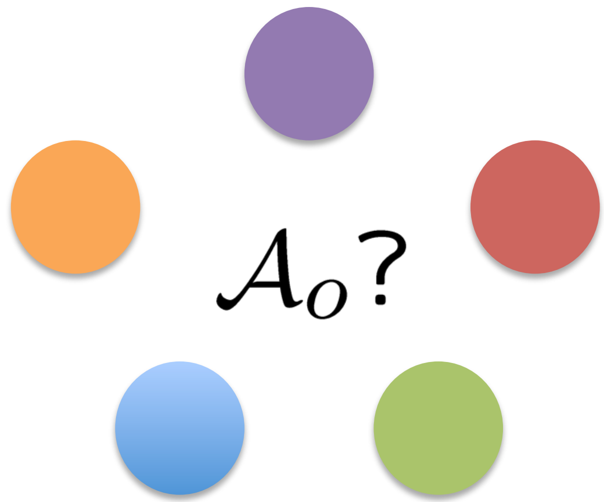
$$\widehat{q_{i,j,k}} = \frac{N_{i,j}^{k}}{N_i}, if\, N_i \neq 0 \text{, otherwise } \widehat{q_{i,j,k}} = 0$$

- Failure Probability (in a time unit)(FP)

$$1 - (1 - FP') \cdot \sum_{j=p}^{b} \widehat{q_{p,j,k}}, p < b < o$$

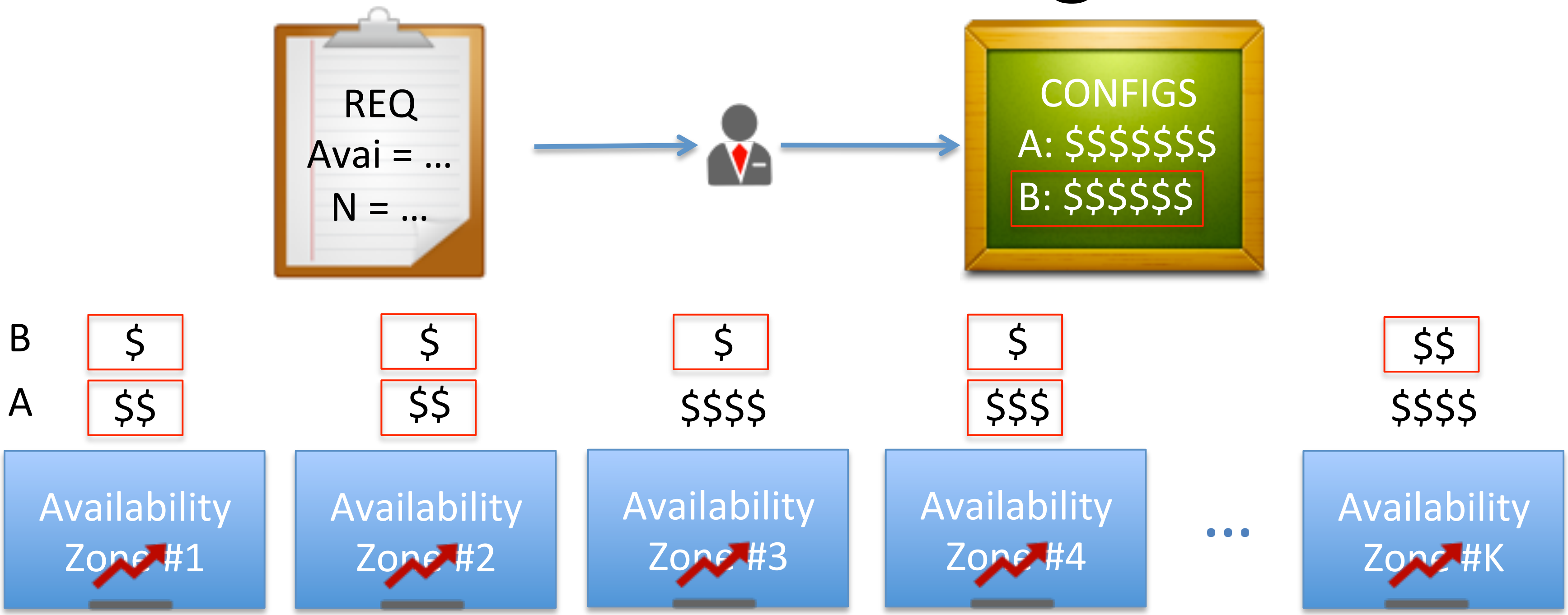
Online Bidding

- Constraint Without Analytic Expression



- Exhaustive search ?
  - Traverse space m^n
- Keeping FPs same & bidding greedily

Online Bidding



*For each possible **n**, get **FP** with given **Availability***

- *For each AZ, get min **bid** with given **FP***
- *Select **bids** of AZs in a greedy way*

Choose the lowest bidding CONFIG

Outline

- Problem Formalization
  - Spot Instance Failure Model
  - Cost Minimization Problem
- Bidding Framework
  - Failure Probability Estimation
  - Online Bidding
- Experiment

Experiment

- Whether cost has been reduced?

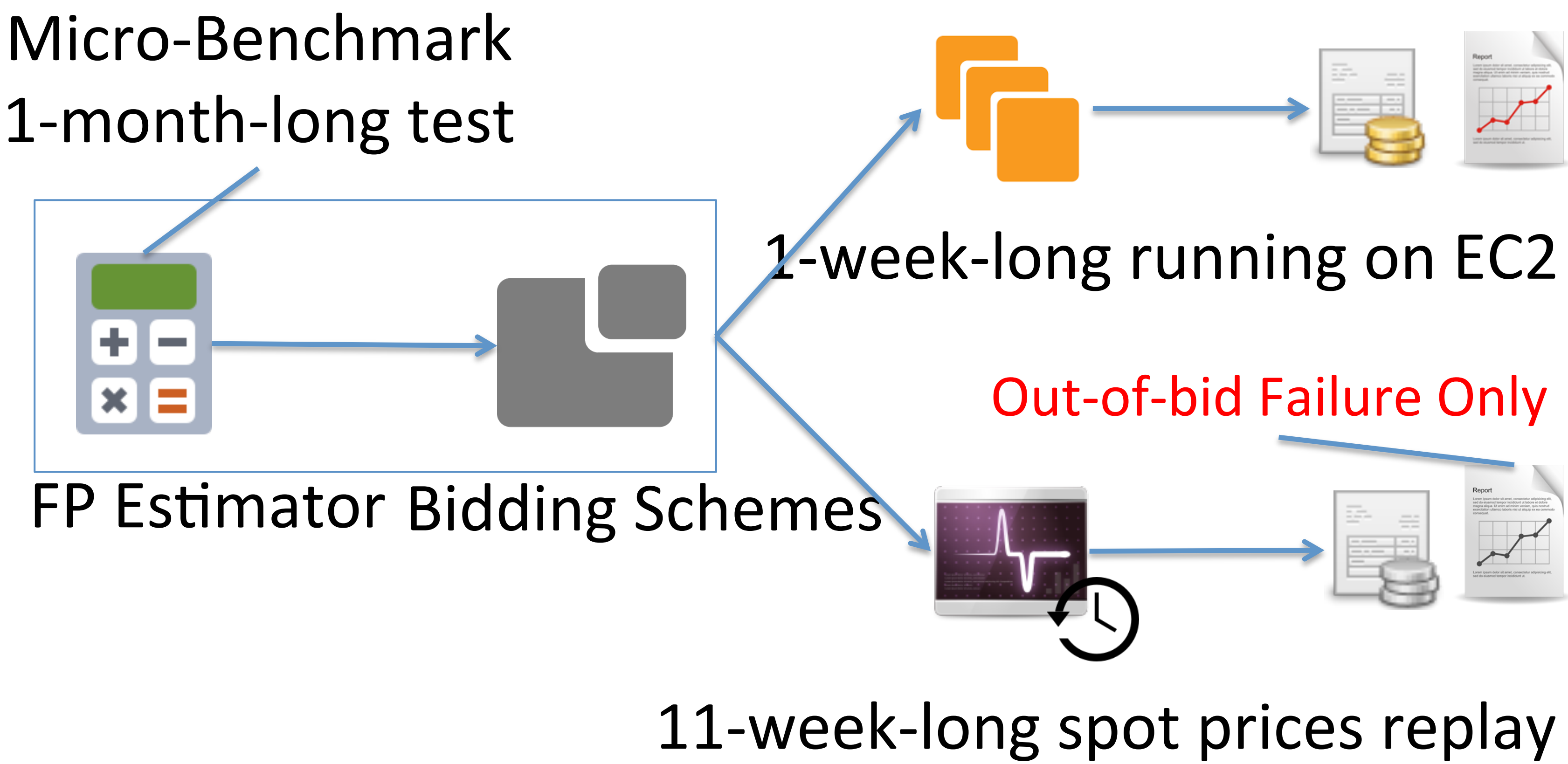
- What about the availability achieved?

Experiment Setup

- Experimental Systems
  - Distributed Lock Service (‘linux.m1.small’)
  - Erasure Code Based Distributed Storage Service (linux.m3.large)
- Estimator Training
  - ~ 3 months spot price data
- Baseline
  - 5 On-Demand Instances
- Straw-man Scheme
  - *Extra(m, p):* Adding *m* extra nodes & setting bids as spot price + extra portion *p*

Experiment Setup

- Test Cases

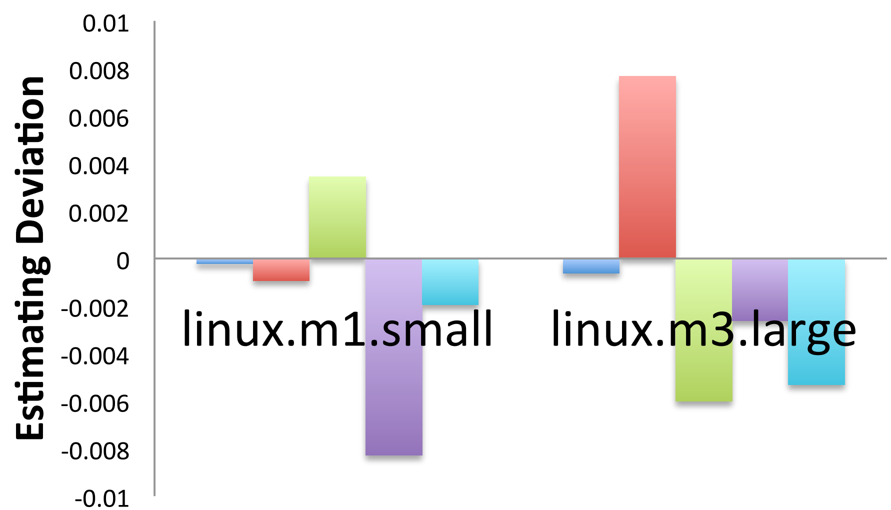


Experiment Results

- Feasibility

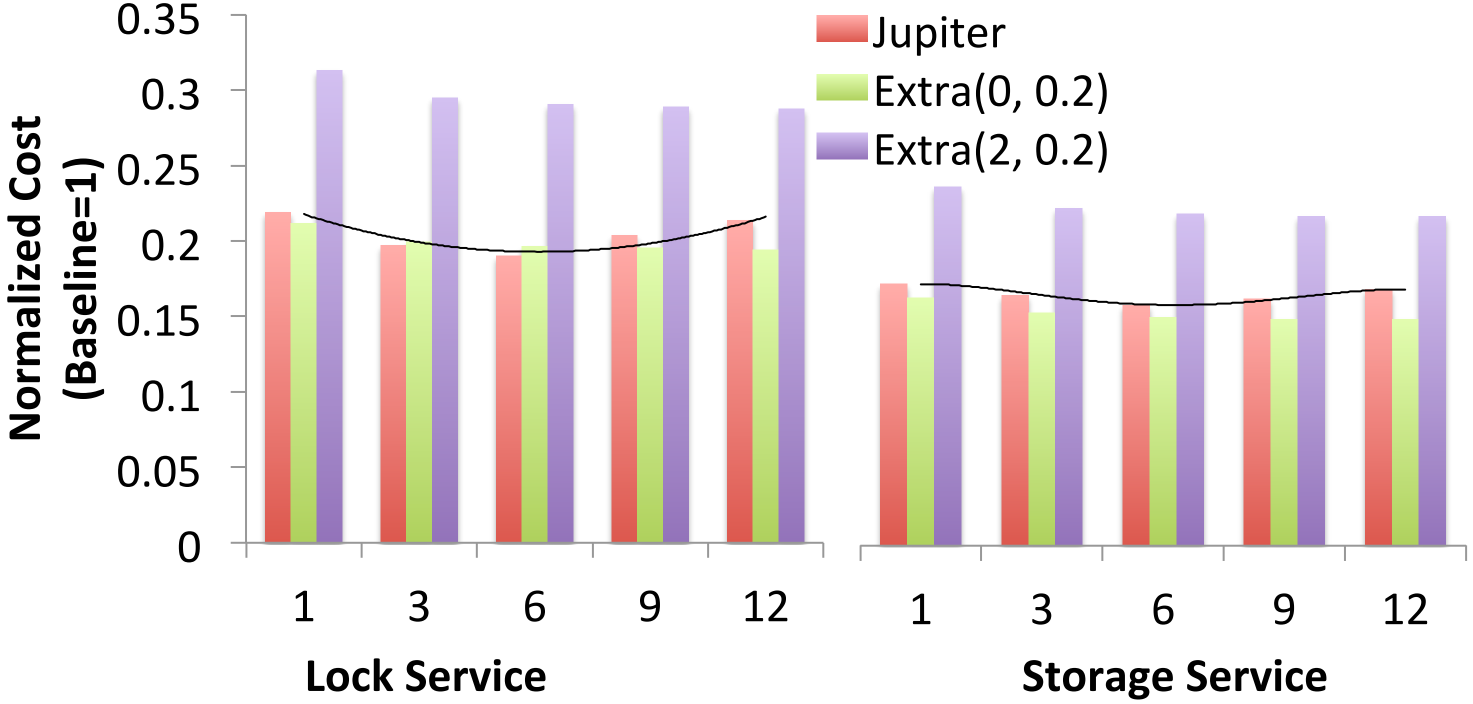
	COST	AVAILABILITY
Jupiter	☺	☺
Extra	☺	☹

- Micro-Benchmark



Experiment Results

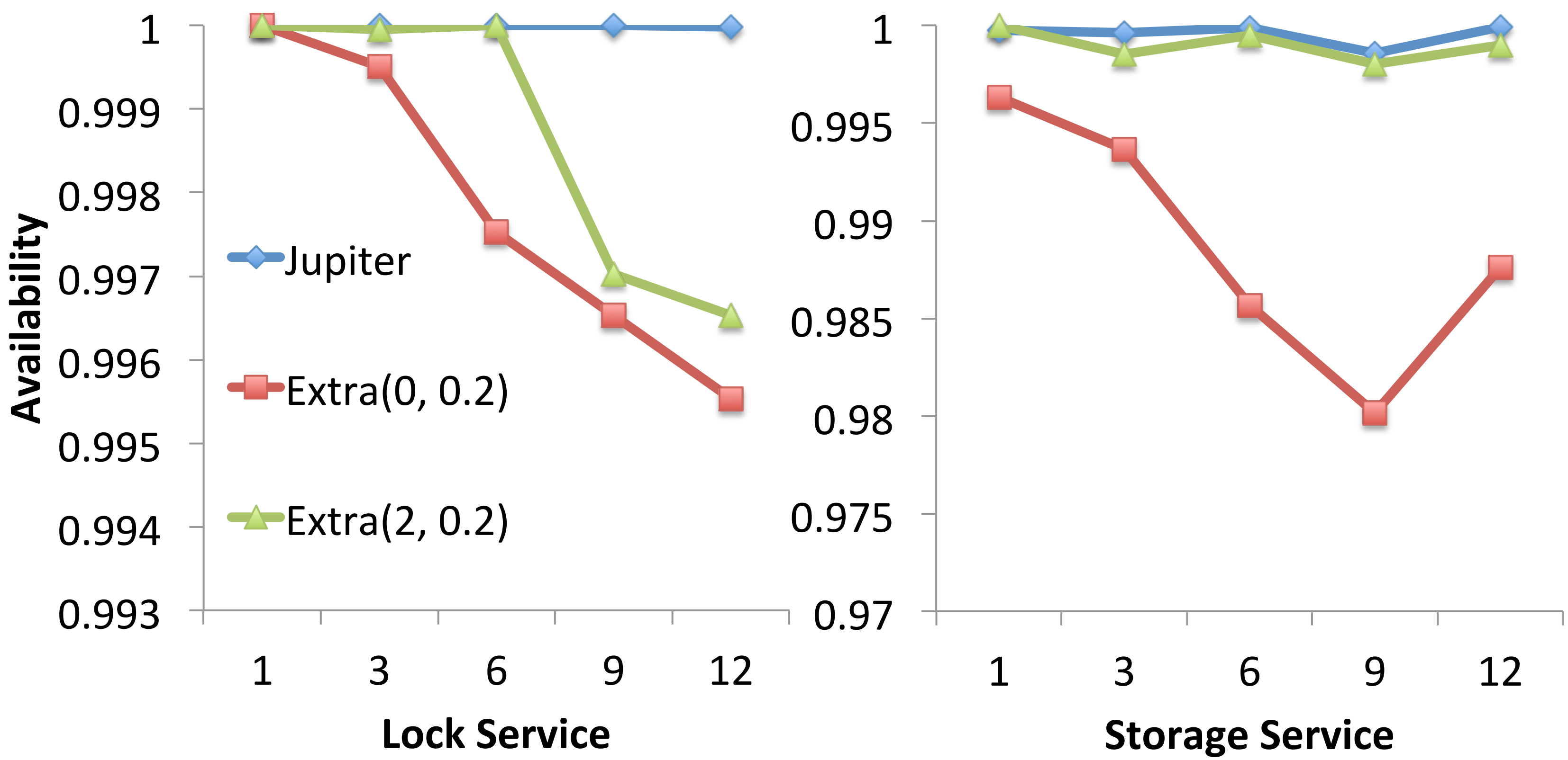
- Cost under different bidding intervals



Jupiter costs only ~ 1/5 and ~1/6 of the baseline

Experiment Results

- Availability under different bidding intervals



Jupiter kept the service availability level close to the baseline

Summary

- Market pricing has bring a new vision of Cloud Computing
- Spot instance failure model challenges the reliability of quorum-based system
- The problem is formalized by Spot Instance Failure Model and Non-linear Programming
- Our bidding framework can obtain cost efficiency while still keeping high availability

THANKS FOR YOUR ATTENTION!

Questions ?