

Learning Cloud Dynamics to Optimize Spot Instance Bidding Strategies

Misha Khodak

Joint with Liang Zheng, Andrew Lan, Carlee Joe-Wong, and Mung Chiang






Overview

- The popularity of cloud computing services has led to the rise of dual-market pricing schemes:
 - Providers sell some instances at a fixed “on-demand” price.
 - Excess capacity is sold at a variable “spot-price” determined via an auction.
- We propose a nonlinear dynamical system model to understand spot-price behavior in this environment.
- We verify our model using five months of Amazon EC2 data and demonstrate its potential to inform strategic bidding between heterogeneous cloud resources.

The Amazon EC2 Spot Market

1. Select the type of instance

 Amazon Linux Free tier eligible	<input type="checkbox"/>	Memory optimized	r3.large	2	15	1 x 32 (SSD)	-	Moderate
	<input type="checkbox"/>	Memory optimized	r3.xlarge	4	30.5	1 x 80 (SSD)	Yes	Moderate
 Red Hat Free tier eligible	<input type="checkbox"/>	Memory optimized	r3.2xlarge	8	61	1 x 160 (SSD)	Yes	High
	<input type="checkbox"/>	Memory optimized	r3.4xlarge	16	122	1 x 320 (SSD)	Yes	High
 SUSE Linux Free tier eligible	<input type="checkbox"/>	Memory optimized	r3.8xlarge	32	244	2 x 320 (SSD)	-	10 Gigabit

2. Configure and set a bid price

Number of instances

Purchasing option Request Spot Instances

Current price

us-east-1a	0.1582
us-east-1b	0.1587
us-east-1d	0.1821
us-east-1e	0.1856

Maximum price \$

3. Amazon sets a spot price

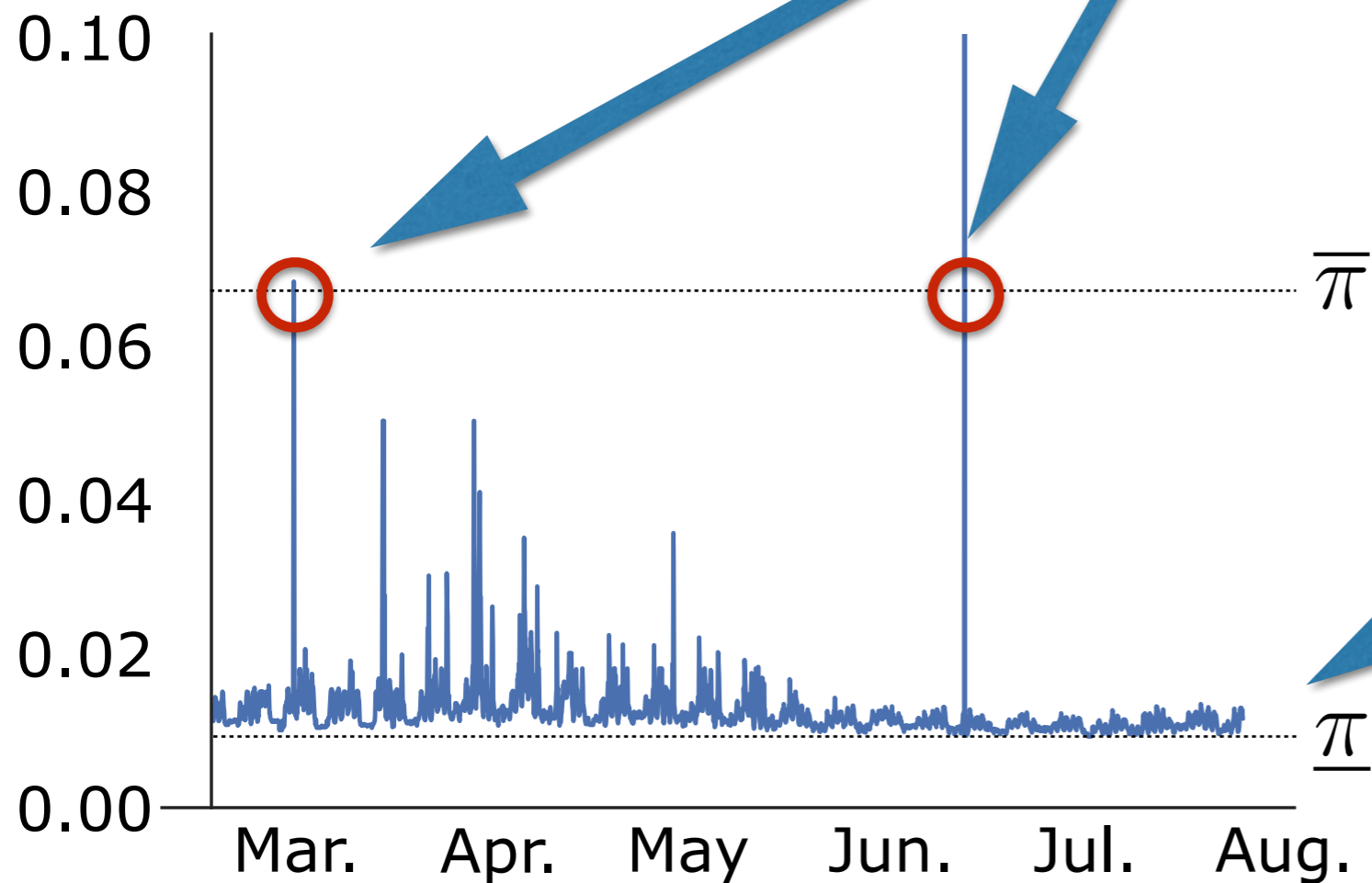
4. User receives instance if bid was above spot price

Motivation

- Spot price dynamics are poorly understood, with past economic modeling focusing mainly on global behavior:
 - Zheng et al. (SIGCOMM 2015) study the spot price distribution at equilibrium.
 - Hoy et al. (WINE 2016) explain the optimality of a two-market design as stemming from variable user risk aversion.
- Understanding temporal dynamics can better inform strategic bidding.

Spot Price Observations



Sometimes goes above the on-demand price $\bar{\pi}$. Occurs when on-demand users take up too much capacity.



The spot price π_t tends to hover above a constant lower bound price $\underline{\pi}$.

m3.medium spot price
over 5 months in 2017

Provider Profit Maximization

on-demand instances		spot-market instances
		
$(\overline{\pi} - \underline{\pi}) N_t^{(d)}$	+	$(\pi_t - \underline{\pi}) N_t^{(s)}$
<hr/>		<hr/>
on-demand profit		spot-market profit

Need a constraint on the number of instances (N):

- Profit-Maximizing: $N_t^{(d)} + N_t^{(s)} \leq N$
- Usage-Maximizing: $N_t^{(d)} + N_t^{(s)} = N$

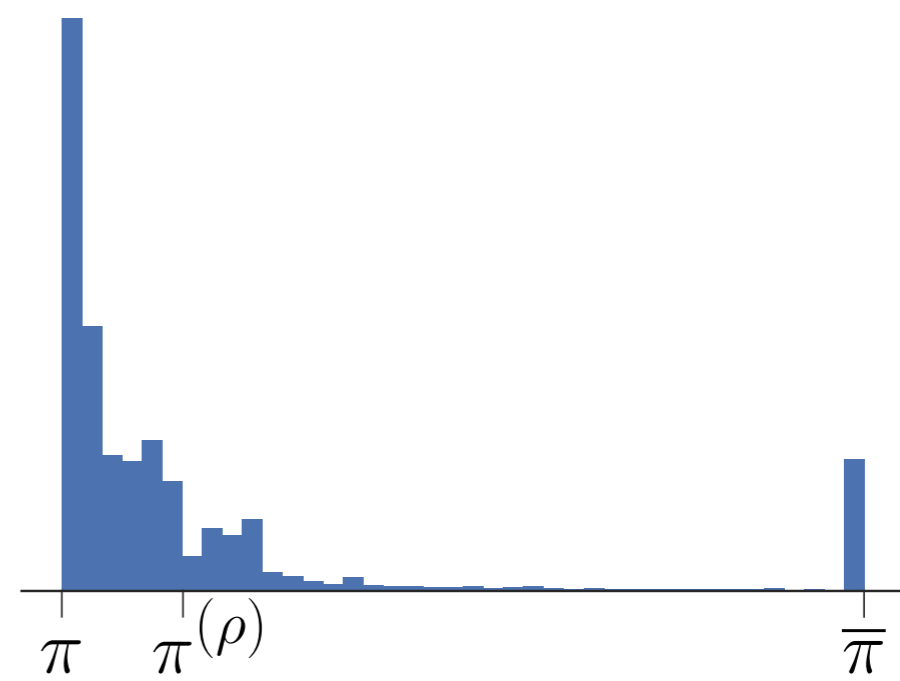
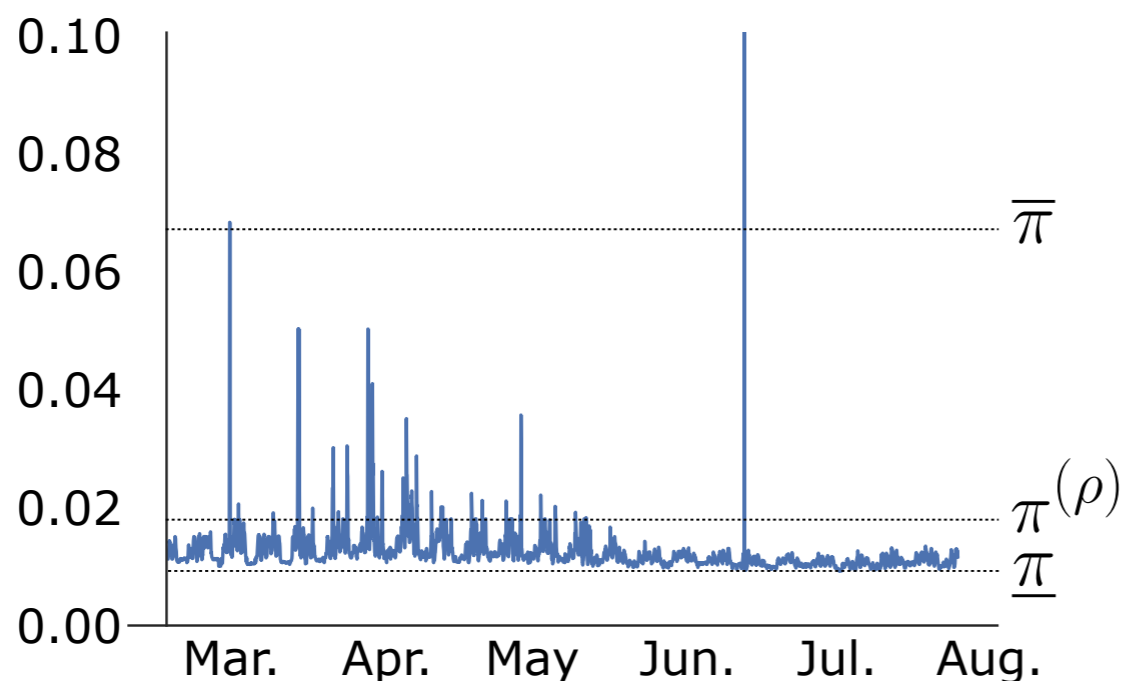
Maximize Profit or Usage?

Proposition [KZLJC'18]: If B_t bids are drawn independently from a distribution that weakly stochastically dominates the uniform distribution over $[\underline{\pi}, \bar{\pi}]$, then if the provider uses the profit-maximizing constraint we have

$$\mathbb{P} \left(\pi_t \leq \pi^{(\rho)} \right) \leq \exp \left(-2 \left(\frac{1}{2} - 2\rho \right)^2 B_t \right)$$

for $\rho \in [0, 1/4]$ and $\pi^{(\rho)} = \rho\bar{\pi} + (1 - \rho)\underline{\pi}$.

So a profit-maximizing provider will not set π_t close to $\underline{\pi}$ very often, which contradicts the data and motivates the choice of a usage-maximizing constraint:



Observed Spot Price Model

At time t cloud provider sees B_t bids, which we model as being i.i.d. draws from $\mathcal{U}[\underline{\pi}, \bar{\pi}]$ (Zheng et al., 2015). Then in the limit $B_t \rightarrow \infty$ this the spot price is distributed as

$$\pi_t = \begin{cases} \underline{\pi} & n_t + b_t \leq 1 & \text{(not enough users)} \\ \bar{\pi} - (\bar{\pi} - \underline{\pi}) \frac{1-n_t}{b_t} + \varepsilon_t & 0 < 1 - n_t < b_t \\ \bar{\pi} & n_t \geq 1 & \text{(too many on-demand users)} \end{cases}$$

where we define:

$$\begin{aligned} n_t &= N_t^{(d)} / N && \text{on-demand usage} \\ b_t &= B_t / N && \text{spot usage} \\ \varepsilon_t &\sim \mathcal{N}\left(0, \frac{\sigma^2}{b_t}\right) && \text{observation noise} \end{aligned}$$

Job Arrival and Departure

We model two hidden variables:

1. n_t , the number of running on-demand jobs at time t , normalized by N
2. b_t , the number of active spot bids at time t , normalized by N

At each time step, $\Lambda_t^{(d)}$ on-demand jobs arrive, $\Lambda_t^{(s)}$ spot jobs arrive, $\tilde{\Lambda}_t^{(d)}$ on-demand jobs complete, and $\tilde{\Lambda}_t^{(s)}$ spot jobs complete. This yields the dynamical system

$$\begin{aligned}n_{t+1} &= n_t + \lambda_t^{(d)} - \tilde{\lambda}_t^{(d)} \\ b_{t+1} &= b_t + \lambda_t^{(s)} - \tilde{\lambda}_t^{(s)}\end{aligned}$$

for all $\lambda_t = \Lambda_t/N$ modeled as i.i.d. draws from exponential distributions.

Combined Model

Our spot price model is a hidden Markov model (HMM) with hidden state transition governed by the job arrival/departure model:

$$\begin{aligned}n_{t+1} &= n_t + \lambda_t^{(d)} - \tilde{\lambda}_t^{(d)} \\b_{t+1} &= b_t + \lambda_t^{(s)} - \tilde{\lambda}_t^{(s)}\end{aligned}\quad \text{hidden state}$$

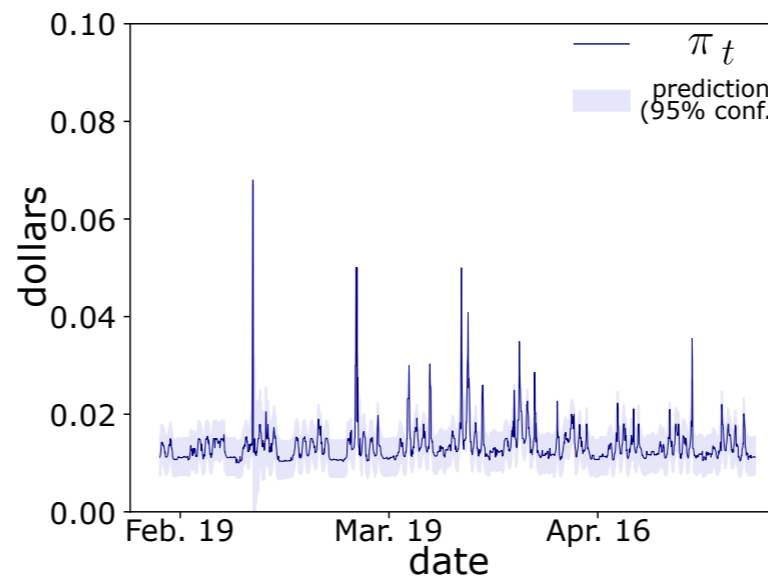
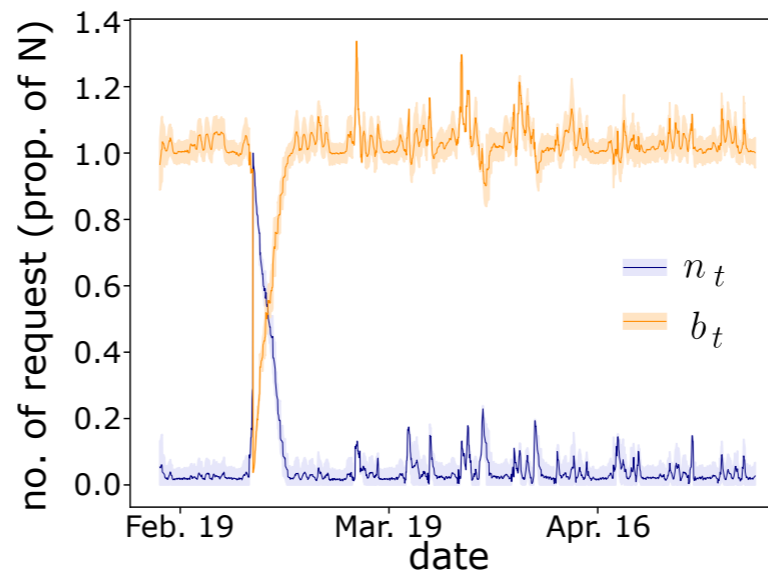
and the spot price distribution:

$$\pi_t = \begin{cases} \underline{\pi} & n_t + b_t \leq 1 \\ \overline{\pi} - (\overline{\pi} - \underline{\pi}) \frac{1-n_t}{b_t} + \varepsilon_t & 0 < 1 - n_t < b_t \\ \overline{\pi} & n_t \geq 1 \end{cases} \quad \text{observation}$$

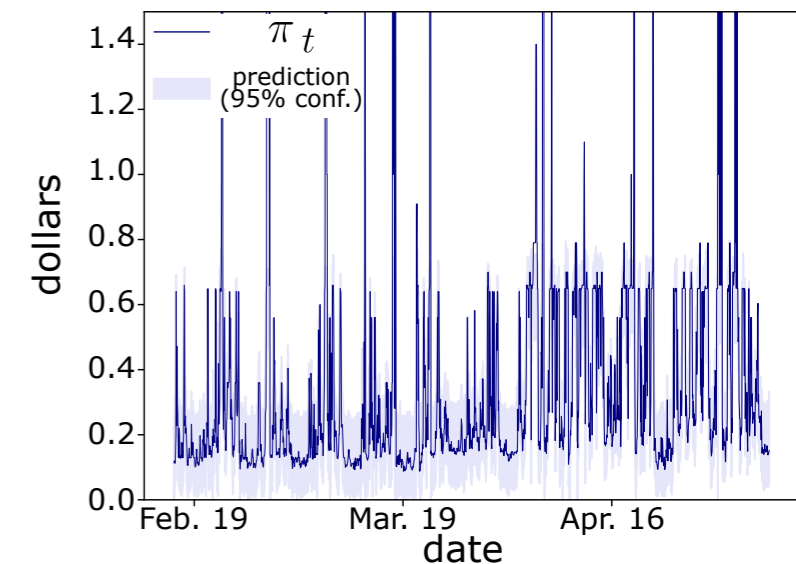
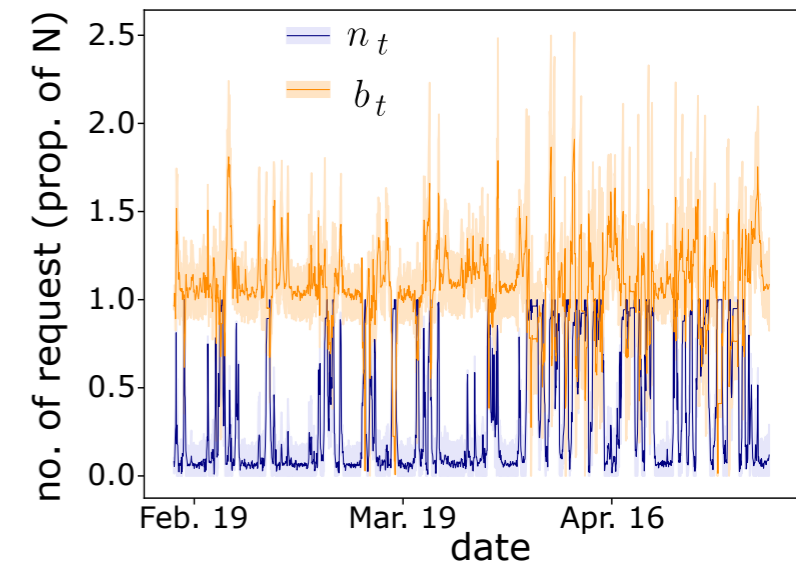
Five model parameters: a scale parameter for each exponentially-distributed λ_t for job arrival/departure and variance σ^2 of the Gaussian observation noise ε_t .

Parameter Estimation

- Model parameters and hidden states are jointly estimated using Expectation-Maximization (EM).
- The E-step is conducted using a sequential Monte Carlo (“particle filter”) approach:
 - Better suited than Kalman-type filters for non-smooth, singular models.
 - Can handle hidden state constraints.



m3.medium,
spring 2017



g2.2xlarge,
spring 2017

Strategic Bidding

- We consider the setting where we want to start a job immediately.
- In the single-instance setting, the optimal strategy is to bid the on-demand price, if one can afford it.
- Instead we can bid within a class of instances by assuming jobs can be easily parallelized.

Region:

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
c4.large	2	8	3.75	EBS Only	\$0.1 per Hour
c4.xlarge	4	16	7.5	EBS Only	\$0.199 per Hour
c4.2xlarge	8	31	15	EBS Only	\$0.398 per Hour
c4.4xlarge	16	62	30	EBS Only	\$0.796 per Hour
c4.8xlarge	36	132	60	EBS Only	\$1.591 per Hour

family of compute-optimized instances

price scales linearly with resources

Choosing Between Instance Families

Given a price $\pi_\tau^{(i)}$ at time τ for each instance type i in a family \mathcal{I} of instances, we wish to minimize the instance cost:

$$P_\tau^{(i)} = \sum_{t=\tau+1}^{t+T_i} \pi_t^{(i)}$$

for T_i the amount of time it takes to finish a job on type i .

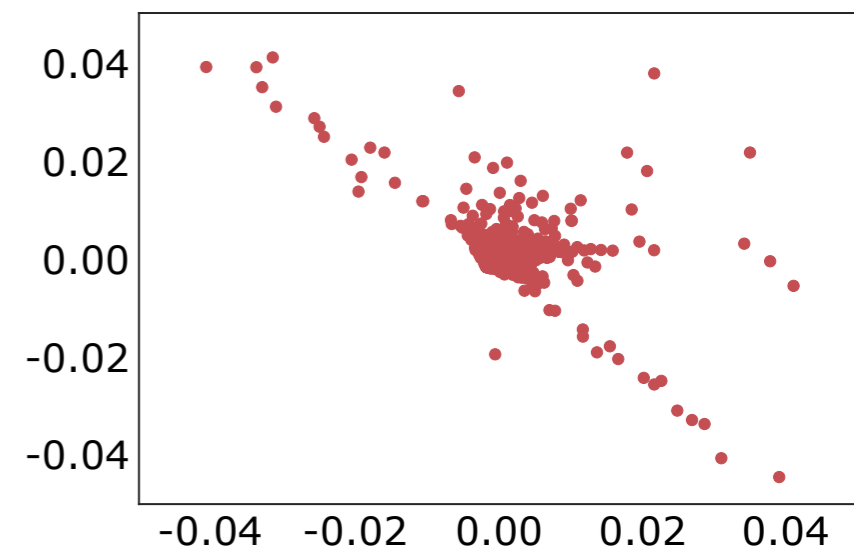
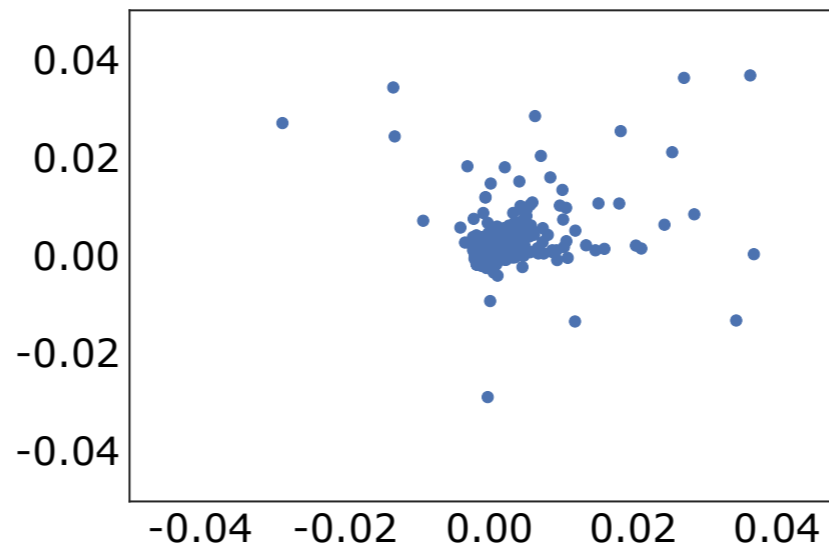
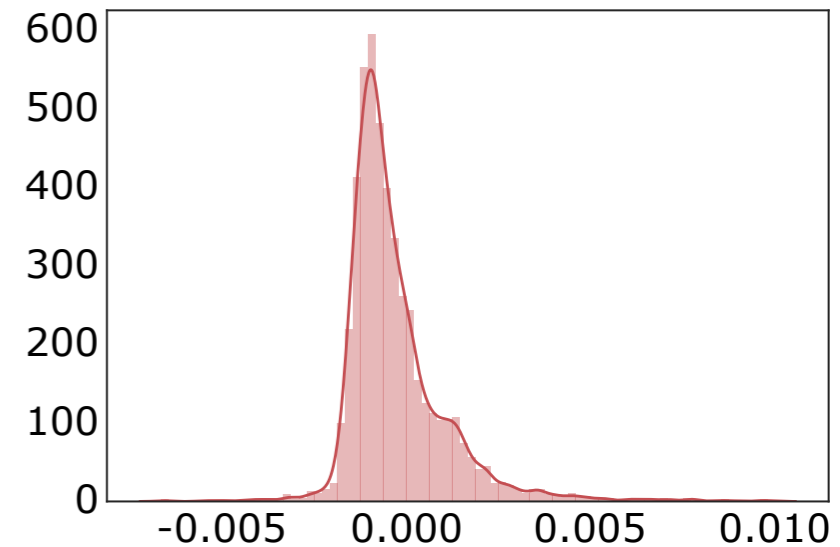
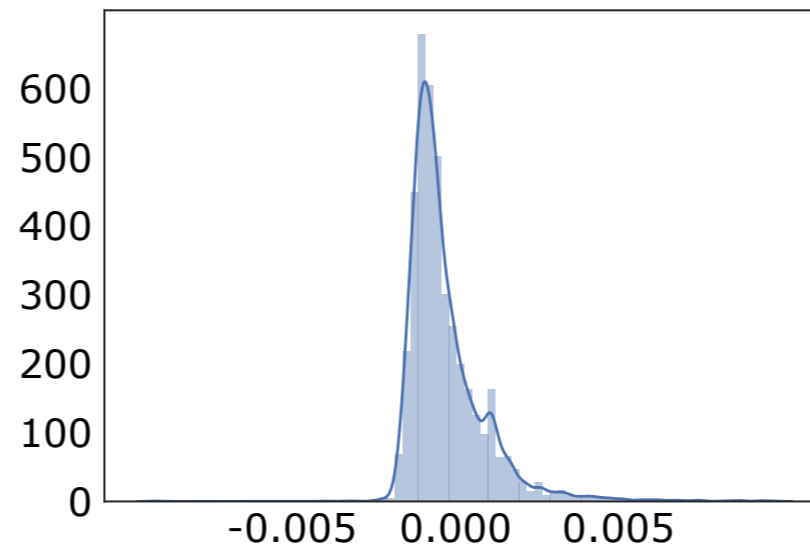
- Solved by using a spot-price model to find $i_\tau = \arg \min \mathbb{E}_{\pi_\tau^{(i)}} P_\tau^{(i)}$.
- In experiments we assume jobs that can be run completely in parallel.
- Can be extended to bidding on non perfectly-parallel jobs and strategic bidding across geographic regions.

Expected Instance Cost: Leveraging Our Model

- **Simulation:** use learned parameters to compute the expected instance cost by simulating multiple trajectories.
 - Requires a lot of computation for high accuracy.
 - Empirically useful on shorter job lengths.
- **Approximation:** approximate the expected instance cost using a second-order Taylor expansion.
 - Cheap to compute.
 - Empirically useful on longer timescales.
 - Assumes the job arrival/departure rates are about the same in both the on-demand and spot market (empirically true).

Expected Instance Cost: Linear Auto-Regression

- Baseline AR(p) model - the price at each time step is some noisy linear combination of the price at p previous time steps.
- Data does not satisfy standard Gaussian error assumptions and uncorrelated residuals.



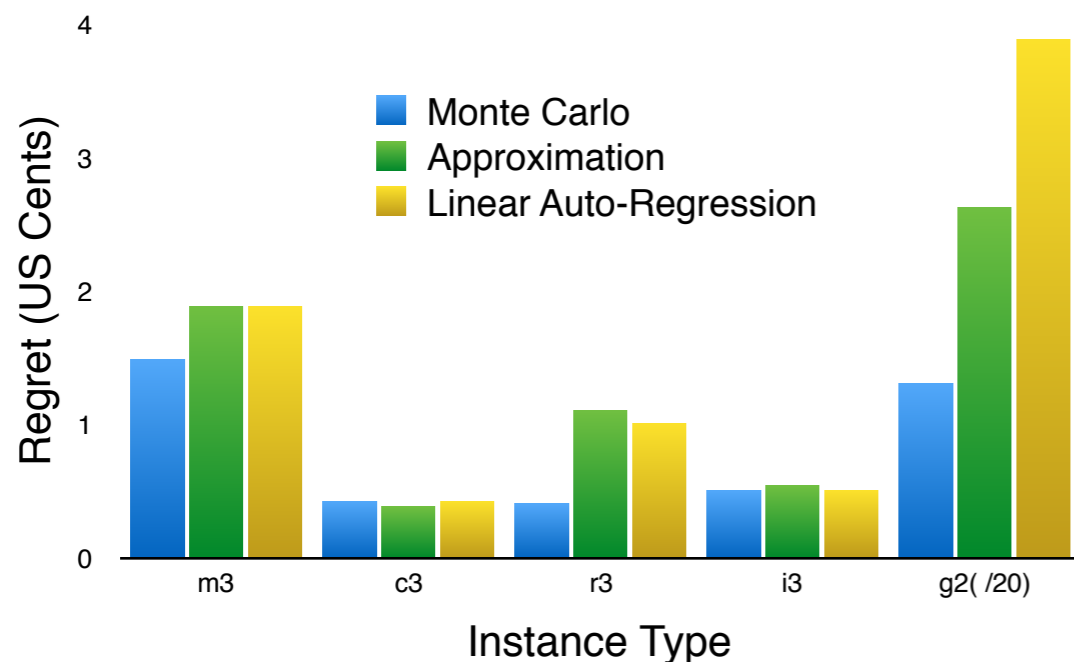
AR(1)

AR(17)

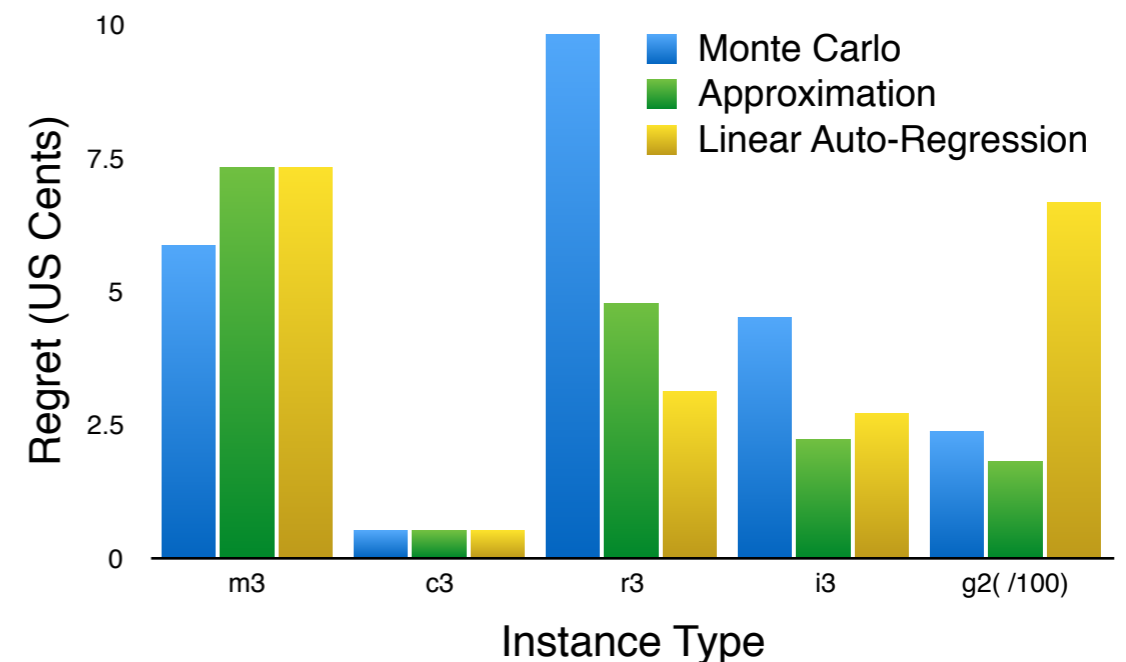
Evaluating Bidding Strategies

- The performance of each bidding strategy is evaluated using regret: the difference between the cost of the chosen action and that of the best action in hindsight.
- Model-based methods succeed especially well on shorter-term (e.g. 16-hour) jobs.

16 Hours



64 Hours

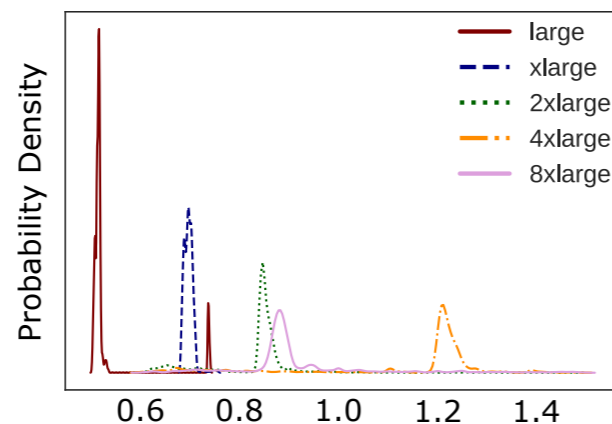
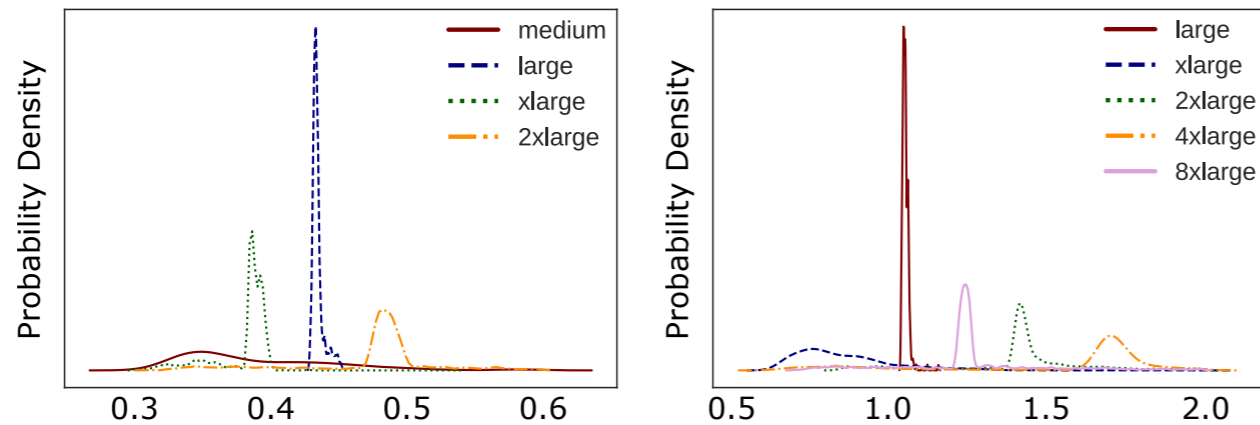


Performance and Volatility

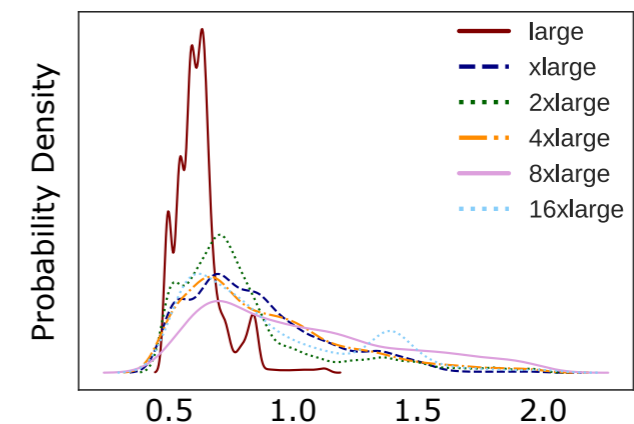
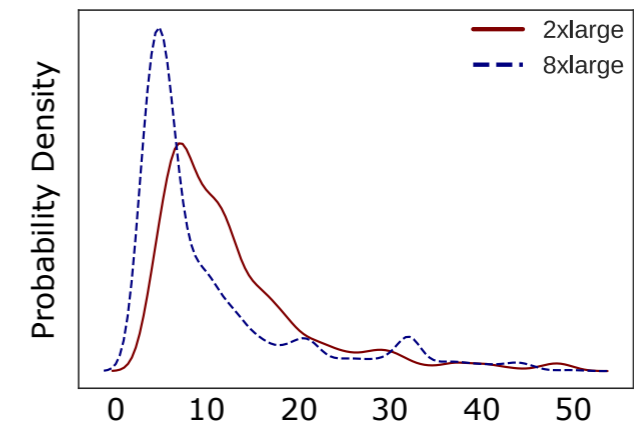
- Model is consistently better than AR on more volatile instances.
 - More monetary gain to be had from strategic bidding.
 - More overlap between the realized cost distributions of different instances.

Payment Distribution for Different Instances

Non-Volatile Instances (m3, c3, r3)



Volatile Instances (g2, i3)



Summary

- We model spot-pricing in cloud computing as a nonlinear dynamical system.
- Amazon EC2 data was used to analyze the problem and learn model parameters.
- We describe strategies for strategic bidding between instances that can make use of the model.

Open Questions

- How do we examine the problem in the setting where dynamics can be influenced across instance families or different regions?
- Provide a model for job departure that explicitly depends on the recent job arrival random variables.
- Devise more sophisticated bidding strategies requiring lighter assumptions concerning job parallelism.

Thank you!

Questions?

Contact e-mail: mkhodak@princeton.edu