Cloud Computing @ WashU with a closer look at (spot) pricing

Roch Guérin

UC Riverside – December 1st, 2017

Cloud Computing @ WashU

- Multiple faculty and students working on a number of cloud computing problems*
  - Middleware (real-time messaging – RTM)
  - Virtualization
    - Scheduling and networking stack (RT-Xen & VATC)
  - Distributed computing support
  - And pricing

- Today, I'll give a very brief overview of some of those works and then talk mostly about pricing (joint work w/ J. Song)

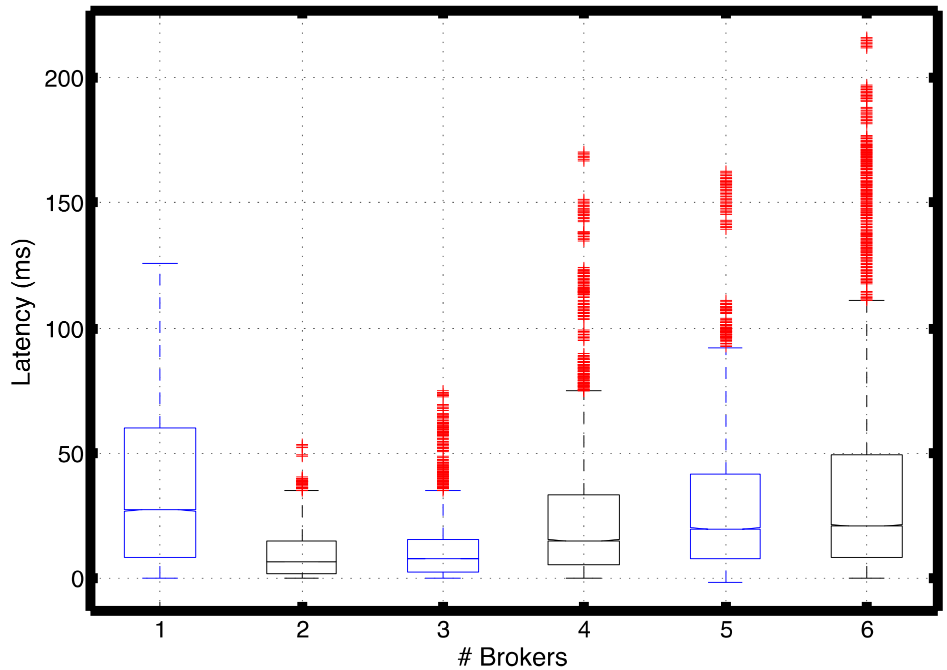
* **C. Gill, R. Jain, B. Kocoloski** (joining on Jan. 1st), C. Li, J. Liu, **C. Lu**, J. Song, etc.

Real-Time Messaging Middleware

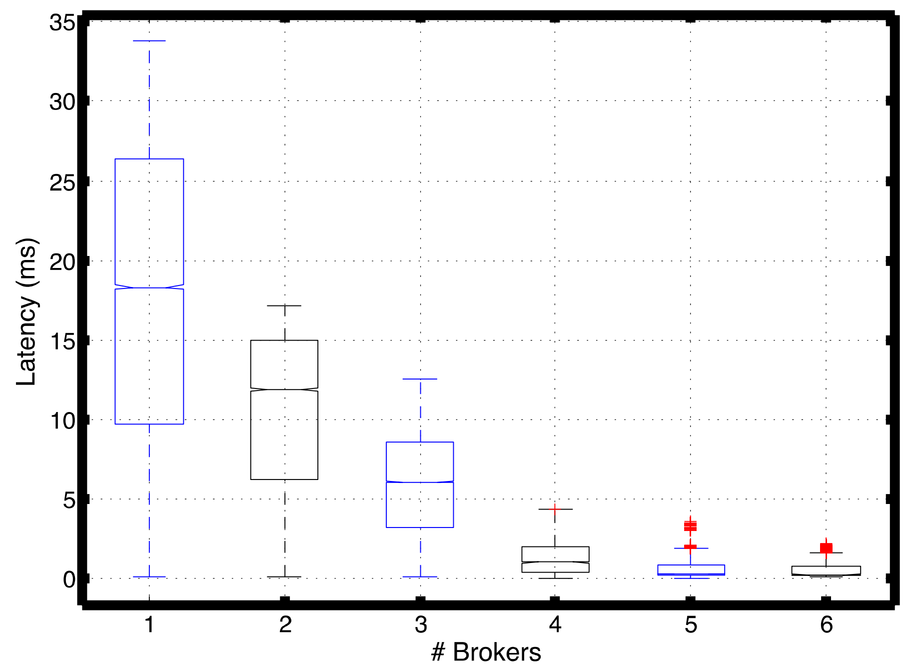
- Why? – Increasingly important in platform as a service offerings
  - Numerous IoT and 5G (edge computing) applications rely on it
- Basic requirements include
  - Scalability (large number of devices and connections)
  - Service differentiation (both latency sensitive and throughput sensitive applications)
- Current RTM system under development
  - Based on NSQ code base
  - Extensions include prioritization (per topic) and rate-limiting (for high-priority topics)
  - Scalability through distribution of publishers across multiple brokers
  - **Main challenges**: Address tension between load-balancing and rate limiting as a function of application profile

Load-Balancing vs. Rate Limiting

- Asynchronous publishers



- Synchronous publishers



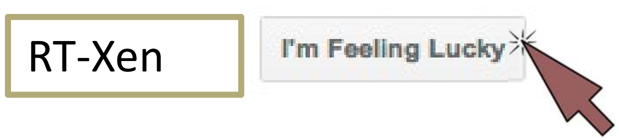
- Trade-off between processing and rate-limiting delays for asynchronous publishers, while synchronous publishers "always" benefit from distribution across more resources
  - Basic issue: How is application burstiness affected by load distribution?

From Clouds to Real-Time Clouds

- Why real-time clouds? They are critical to many large-scale Internet-of-Things applications
  - Smart transportation, smart manufacturing, smart grid
  - Industry trend: AWS IoT, IBM IoT Foundation
- Cloud systems for real-time applications.
  - Latency guarantees for tasks running in virtual machines (VMs).
  - Resource sharing between real-time and non-real-time VMs.
  - Coordinated scheduling, traffic shaping, rate limiting
  - **End-to-end resource guarantees, from CPU to network**
- Service differentiation in Xen
  - **RT-Xen** → real-time VM scheduling in a virtualized host.
  - **VATC** → real-time network I/O in a virtualized host

Real-Time Virtualization with RT-Xen

- Real-time schedulers in the Xen hypervisor.
- Provide real-time guarantees to tasks in VMs.
- Incorporated in **Xen 4.5**.



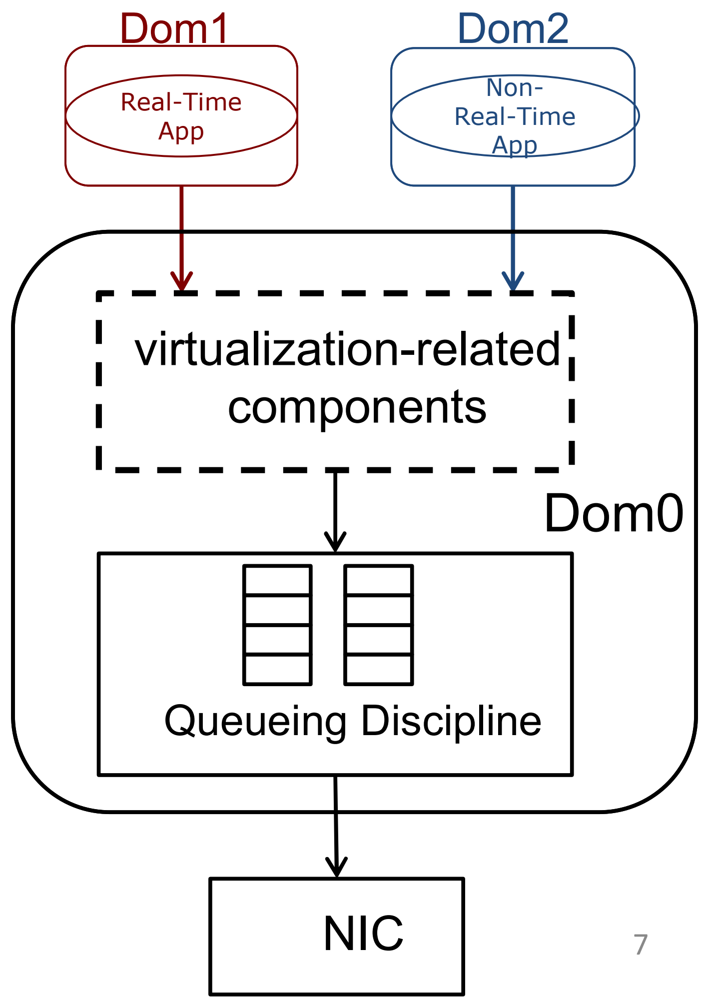
RT-Xen
Real-Time Virtualization

https://sites.google.com/site/realtimexen/

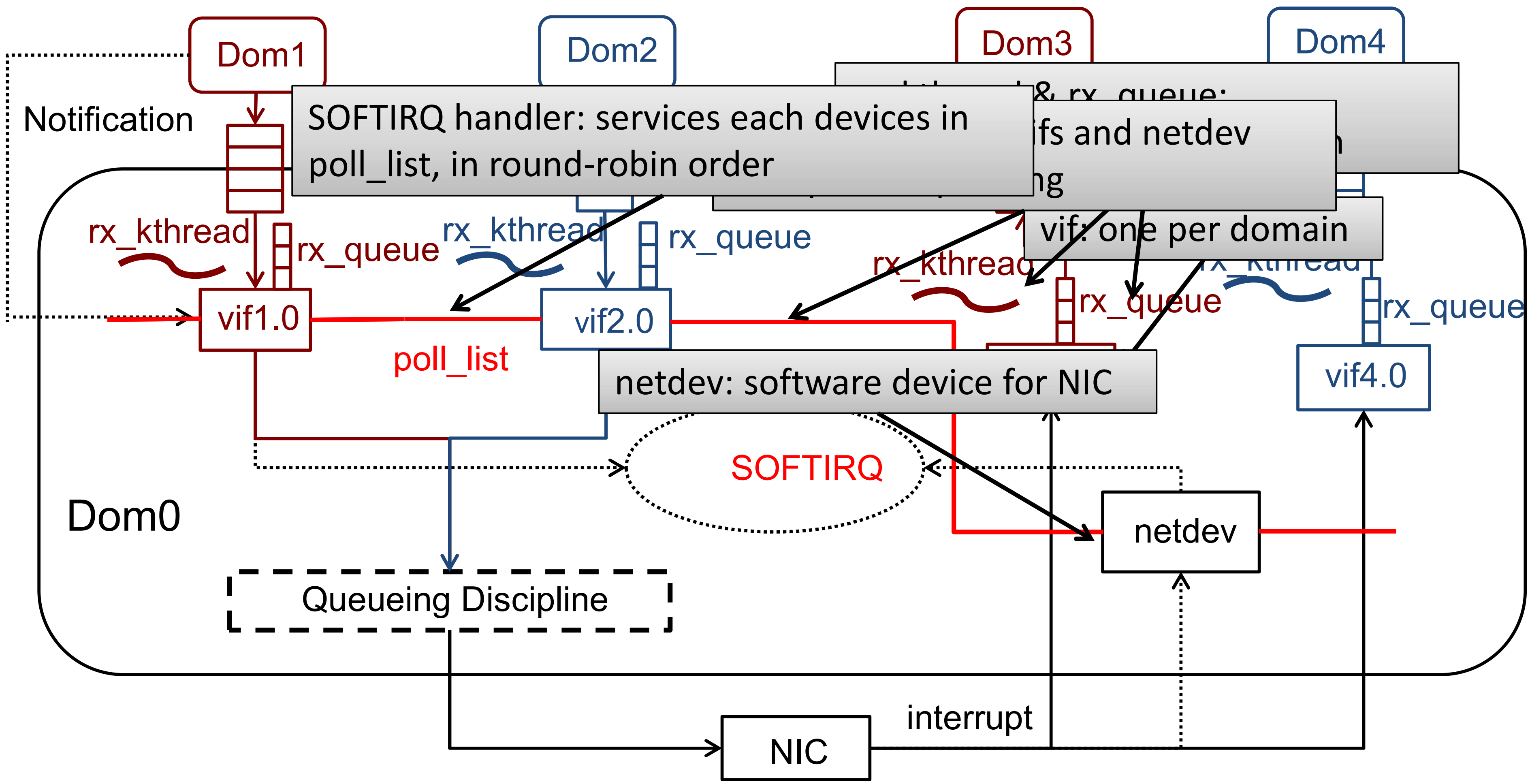
S. Xi, M. Xu, C. Lu, L. Phan, C. Gill, O. Sokolsky and I. Lee, Real-Time Multi-Core Virtual Machine Scheduling in Xen, ACM International Conference on Embedded Software (EMSOFT'14), October 2014.

VATC: Service Differentiation in Xen's Networking Stack

- Networking in Xen (and many other virtualization systems) is realized through a "special" Linux VM (dom0)
  - Virtual interfaces (vif's) provide VMs with network access through dom0
  - dom0 leverages Linux components, *e.g.*, Qdisc layer



Network Access Through Dom0



Problem Statement & Solution

- **Current Limitations**
  - Transmit side
    - Poll list is served in round-robin order ⇒ Possibility of priority inversions
  - Receive side
    - Rx_ktrhead can only be scheduled after SOFTIRQ handler completes ⇒ Possibility of priority inversions
- **Approach**
  - Rely on kernel threads instead of softirq
    - Each priority assigned dedicated & prioritized tx and rcv kernel threads
    - Dedicated rcv kernel thread to handle interrup from NIC
    - Threads scheduled by SCHED_FIFO preemptive scheduler
  - Dedicated per priority tx and rx queues
  - First version implemented and described in

    C. Li, S. Xi, C. Lu, C. Gill, R. Guerin , *"Prioritizing Soft Real-Time Network Traffic in Virtualized Hosts Based on Xen."* RTAS'15

Distributed Computing in the Cloud

Reserved instances busy/idle

On-demand instances

Unallocated → opportunistic instances

- Can we leverage both unallocated and idle reserved instances to do useful work, *i.e.*, a la HTCondor?
- Requirements
  - "Transparent" to current services
  - Little to no changes to user code running on opportunistic instances

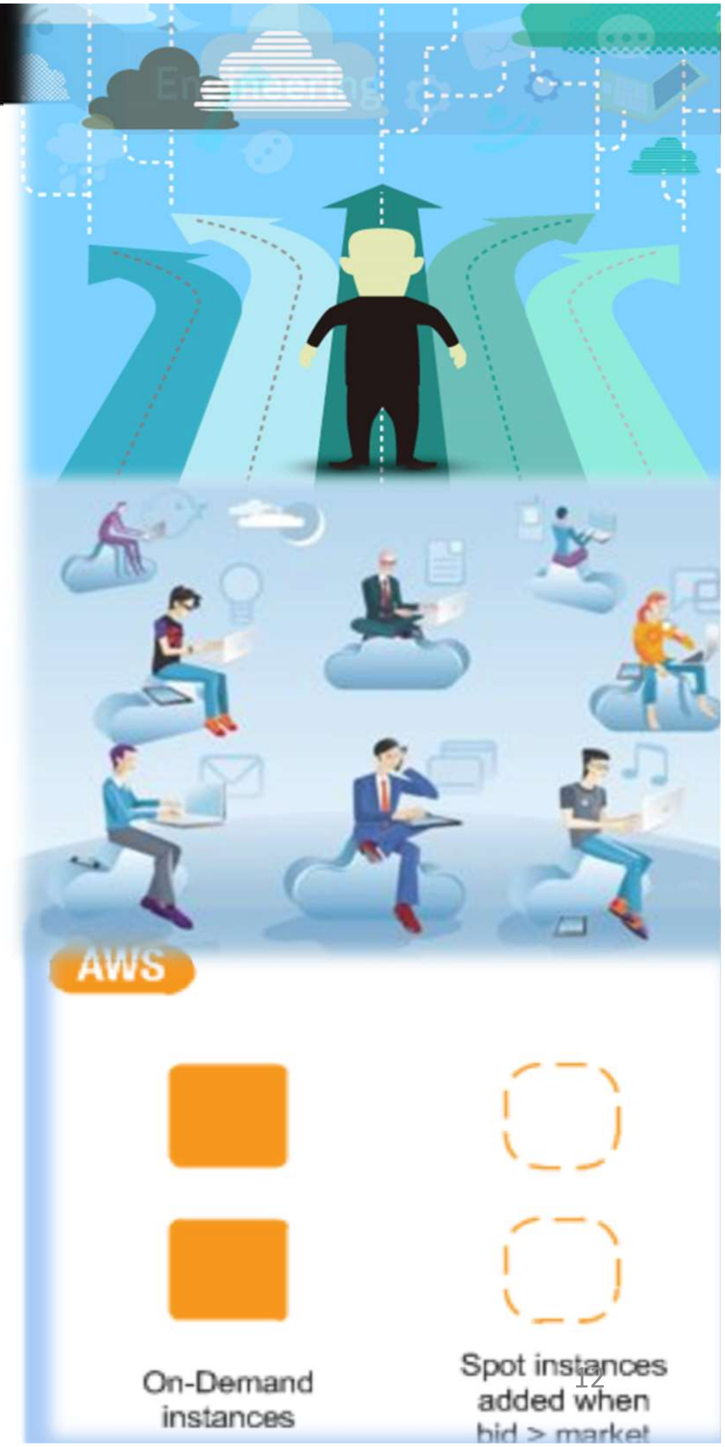
A Two-Prong Effort

1. Scheduler that determines the usage classification of different resources
2. An efficient check-pointing mechanism for opportunistic jobs that need to be terminated

- Some challenges
  - Partitioning of unallocated resources, *e.g.,* available to opportunistic instances or not
  - Pre-configuration of opportunistic instances, *e.g.,* based on user job profiles
  - Allocation of opportunistic jobs to available instances, *e.g.,* favoring free opportunistic instances over idle reserved instances
  - Job specific check-pointing decisions, *e.g.,* omit check-pointing of "short" jobs, dynamically vary check-pointing interval based on system load, etc.
  - Termination decisions, *i.e.,* which opportunistic instance to terminate when required?

Pricing in Clouds

- Motivated by
  - As mentioned earlier, ubiquity of clouds as compute platform
  - Diversity of cloud customers in terms of job valuation, size, timeliness of execution, etc.
  - And the fact that pricing is a powerful knob to match users to services and improve "efficiency"
- And as alluded to, there are indeed a variety of cloud service offerings
  - *e.g.*, reserved, on-demand, and spot instances



Our Focus

- A semi-monopolistic cloud provider like AWS
- A range of services, but in particular services that trade-off price for timeliness of execution
  - On-demand vs. spot instances (more on this in a moment)
- Questions we seek to answer
  - When does having both services help the provider improve revenue?
    - How should prices be set?
  - What are effective bidding strategies for users?
    - Generate highest "utility"?

Our Setup – On the Provider Side

- Monopoly, *i.e.,* we ignore the possible impact of competition
- Focus on spot service (when is it useful?)
  - Spot price is periodically updated
    - Customers register bids ahead of each period
    - Jobs run (stop) whenever their bid exceeds (falls below) the spot price
    - Jobs are charged spot price (not bid) whenever they run
- Unconstrained cloud resources (capacity is not a constraint)
  - A reasonable assumption for most large cloud providers and supported by recent empirical work*

    * O. A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 spot instance pricing," ACM Trans. Econ. Comput., vol. 1, no. 3, September 2013.
- Spot prices are selected randomly from a given set of prices
  - Known price distribution
  - Also supported by empirical findings*, and Amazon makes historical spot prices available
- Goal: pick prices and price distribution to maximize expected revenue
  - Note: If answer is to use a ***single*** price, then a spot = on-demand

Our Setup – On the Customer Side

- Heterogeneous job requirements:
  – Job valuation (v)
  – Job timelinsess / sensitivity to delay (κ)
  – Job execution time (t)

Job profiles (v,κ,t) are private information, but their distribution is known to the service provider

- Customer Decisions:
  – Whether or not to purchase the (spot) service
  – How to bid for the service
- Goal: For each job, pick a bidding strategy Γ that maximizes a job's expected *utility* (over possible spot price realizations)

$$U(t, v, \kappa, \Gamma) = V(t, v, \kappa, \Gamma) - P(t, v, \kappa, \Gamma) - D(t, v, \kappa, \Gamma)$$

where

– $V(t, v, \kappa, \Gamma)$: job valuation (realized only at job completion)
– $P(t, v, \kappa, \Gamma)$: expected payment (for the spot service)
– $D(t, v, \kappa, \Gamma)$: expected delay penalty (given bidding strategy Γ)

Customers bid if and only if $U(v, t, \kappa, \Gamma) > 0$

Two Primary Questions

- How should provider select prices to maximize expected revenue given known distribution of customer/job profiles?
  - Assuming rational users
- How should customers decide whether or not to bid, and if they bid, how to bid to maximize a job's expected utility?
  - Assuming known price distribution and knowledge of job profile

Model Parameters

- Service provider
  - A discrete set of prices $p_1 < p_2 < \dots < p_n$ from which to choose spot prices
  - Distribution $\pi_1, \pi_2, \dots, \pi_n$ for prices
- Customers: Job profiles (v,κ,t) and bidding strategy (Γ)
  - v and κ have joint density function $q(v, \kappa)$ and are independent of t
    - Job sensitivity to execution delay depends on its valuation, but not its execution time (big jobs are more valuable, but not necessarily more or less sensitive to delay)
    - **Variable correlation coefficient**, $\rho \in [-1,1]$
  - t distributed according to $f(t)$
  - Γ is a function of (v,κ,t) and pricing

Optimization Framework

Service provider

- Maximize expected revenue
- Find $p_1 < p_2 < \ldots < p_n$ and $\pi_1 < \pi_2 < \ldots < \pi_n$

↓

Customer

- Maximize expected utility given $(p_1, p_2, \ldots, p_n)$, $(\pi_1, \pi_2, \ldots, \pi_n)$, and (v, κ, t)
- Find a bidding strategy

Optimization Framework

Service provider

- Maximize expected revenue
- Find $p_1 < p_2 < \ldots < p_n$ and $\pi_1 < \pi_2 < \ldots < \pi_n$

Customer

- Maximize expected utility given $(p_1, p_2, \ldots, p_n)$, $(\pi_1, \pi_2, \ldots, \pi_n)$, and (v, κ, t)
- Find a bidding strategy

As usual, work backwards

Customer's Optimization

$$\Gamma^* = \arg\max_{\Gamma} U\left(v, t, \kappa, \Gamma\right)$$

- Some simplifying assumptions
  - Linear delay penalty

$$U(t, v, \kappa, \Gamma) = vt - P(t, v, \kappa, \Gamma) - \kappa(T(t, v, \kappa, \Gamma) - t),$$

  where $T(t, v, \kappa, \Gamma)$ is the expected completion time
  - Jobs are not terminated once bidding starts (positive utility in expectation over jobs with the same profile)
- Numerical exploration when relaxing those assumptions

Optimal Bidding Strategy

$$\Gamma^* = \arg\max_{\Gamma} U(v, t, \kappa, \Gamma)$$

- Fixed bidding strategy is optimal for jobs of size 1
  - In other words, a job profile (1, v, κ) maps to a static bidding value b^*
- Can be extended to a job of arbitrary size by induction
- b^* can be obtained through a simple linear search
  - It belongs to the set of spot prices $[p_1, p_2, \ldots, p_n]$
- Result is, however, fragile to relaxations of our simplifying assumptions, *i.e.*, job termination and non-linear delay penalties

Properties of Optimal Bidding Strategy

$$b^* = \min \arg\min_{p_i \le p_n} \left\{ \frac{\sum_{p_j \le p_i} \pi_j p_j}{\alpha(p_i)} + \kappa \left(\frac{1}{\alpha(p_i)} - 1 \right) \right\}$$

where $\alpha(p_i) = \sum_{p_j \le p_i} \pi_j$ (fraction of time the job executes if bidding a p_i)

- **If** a customer decides to bid for job (t, v, κ)
  - b^* is determined solely by κ (independent of v and t)
  - b^* increases with κ
- The decision to bid, however, depends on v (job's valuation affects its ability to generate positive utility)

Service Provider's Optimization

$$\left(p^*, \pi^*\right) = \arg\max_{p,\pi} R(p, \pi)$$

Where $R(p,\pi)$ is expected revenue given pricing (p,π)

Recall:

– t is independent of v and κ

– v and κ are correlated.

$$R_{p,\pi} = \iiint_{v,\kappa,t} f(t) q(v,\kappa) P(t, \Gamma^*_{p,\pi}(t,v,\kappa)) \, dv \, d\kappa \, dt$$

where

$f(t)$: density function of job length

$q(v, \kappa)$: joint density function of v and κ

Gaining Insight with a Discrete Model

- Users belong to four different "categories"

κ_1, v_1: low

κ_2, v_2: high

	κ_1	κ_2	
v_1	q_{11}	q_{12}	a
v_2	q_{21}	q_{22}	1-a
	b	1-b	

$$\rho = \frac{E[v\kappa] - E[v]E[\kappa]}{\sqrt{\mathrm{var}(v)\mathrm{var}(\kappa)}}$$

high/low valuation + high/low sensitivity to delay

- Fix marginal
- Vary correlation

⇨ Effect of correlation

Optimal Pricing Strategy

- For a given density function $q(v, \kappa)$ with fixed marginal and correlation coefficient ρ, there exists $\rho^* \in (0,1]$ such that
  - When $\rho \leq \rho^*$, a single price strategy is optimal, *i.e.,* a spot service generates no more expected revenue than an on-demand service
  - When $\rho > \rho^*$, a two-price strategy is optimal, *i.e.,* a spot service can offer greater revenue than an on-demand service alone

Note 1: $\rho^* \in (0,1]$ implies that if jobs' valuation and sensitivity to delay are independent, then a spot service is not useful

Note 2: The result is not robust to changes in our simplifying assumptions, *i.e.*, in general ρ^* can span the full range [-1,1]

Some Intuition

Perfectly negatively correlated

	κ_1 can bid low	κ_2 has to bid high
v_1 can't afford high bid	0	1/2
v_2 can afford high bid	1/2	0

Perfectly positively correlated

	κ_1 can bid low	κ_2 has to bid high
v_1 can't afford high bid	1/2	0
v_2 can afford high bid	0	1/2

A two-price spot service has a positive impact if jobs with large delay sensitivity pay more. This in turn has the potential to 1) exclude jobs with large delay sensitivity and small valuation, and 2) extract a smaller price from jobs with small delay sensitivity and large valuation. 1) and 2) have to remain small

Testing for Robustness

- Two main results
  1. Optimality of fixed bidding strategy
  2. Presence of a correlation threshold below which spot service is of no benefit
- Two primary assumptions and one secondary
  1a. Jobs are never terminated once they start bidding
  1b. Delay penalty increases linearly
  2. Binary job profile
- Which results still hold when relaxing assumptions?
  – Allow termination, non-linear delay penalties, continuous job profiles
  – Because optimality of fixed bidding is easily found to be fragile, focus is on existence of correlation threshold (currently investigating how bad fixed bidding can be)
- Approach is numerical in nature
  – Test for threshold where single price solution stops being optimal. In other words, we don't identify the optimal policy, only that single price stops being optimal

Allowing Job Termination

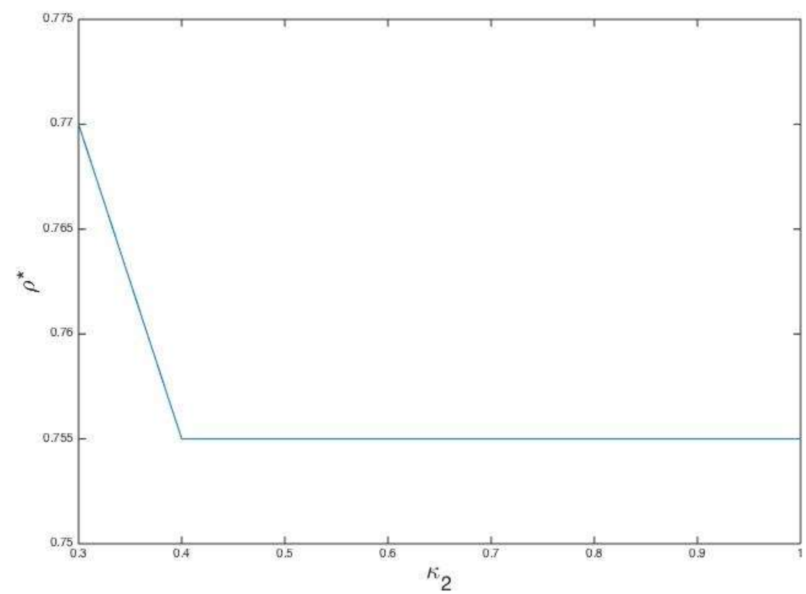
- Jobs are terminated when their expected residual utility becomes negative

$$\text{terminate if } vt - p(p_i)(t - t_0) - \kappa\left(\tau - t + \frac{t - t_0}{\alpha(p_i)}\right) \leq 0$$

  for linear penalty and fixed bidding (t_0 is execution time so far, and τ is elapsed time)
- Both fixed and dynamic bidding are considered
  - Dynamic bidding policy calls for solving a dynamic program
- Tested for different binary job profiles and combinations of job sizes (termination depends on job size)

Allowing Job Termination

- Preliminary findings
  - Threshold remains present under termination for both fixed and dynamic bidding
  - Termination appears to lower provider revenue though less so when dynamic bidding is used (revenue drop from early termination exceeds higher participation that termination allows)
  - Dynamic bidding seems to increase ρ^*



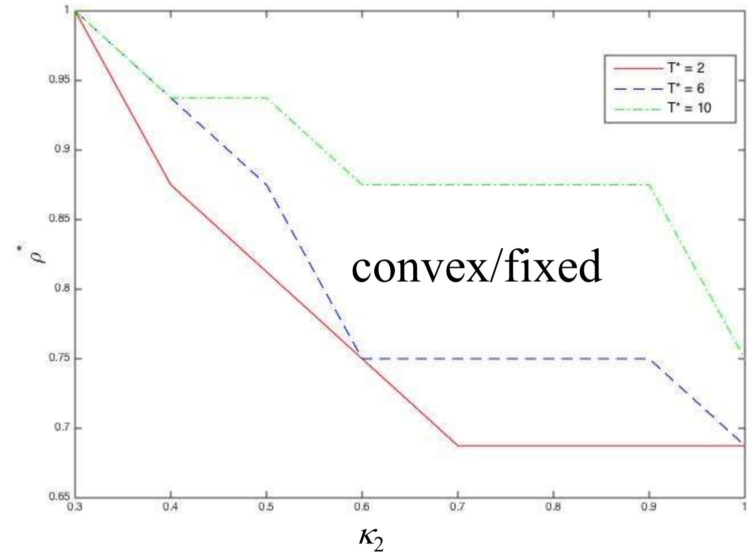
Nonlinear Delay Penalty Functions

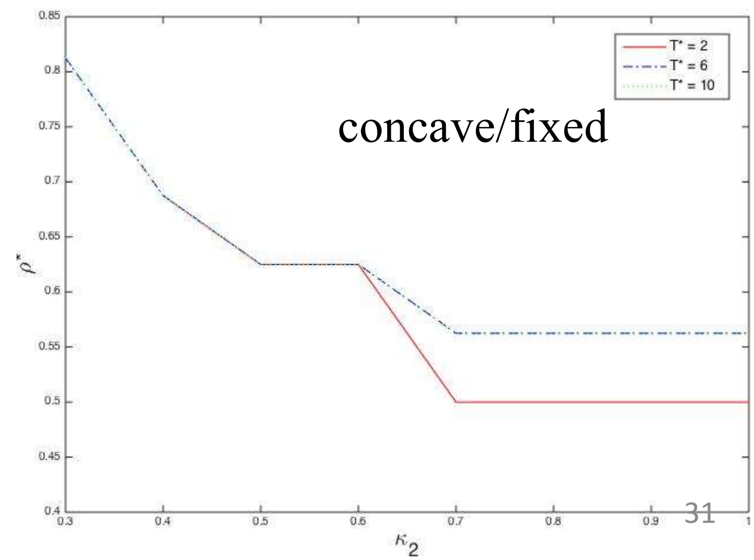
- Piecewise-linear delay penalty function
  - "Convex" delay function
    - $D_1(\kappa, t) = \kappa \max\{0, T(t) - t - T^*\}$
  - "Concave" delay function
    - $D_2(\kappa, t) = \kappa \min\{T(t) - t, T^*\}$

  T^* is a threshold, t is the job's execution time, and $T(t) > t$ is the job's total completion time
- Again, we investigate fixed and dynamic bidding configurations
  - For simplicity, we initially preclude termination

Nonlinear Delay Penalty Functions

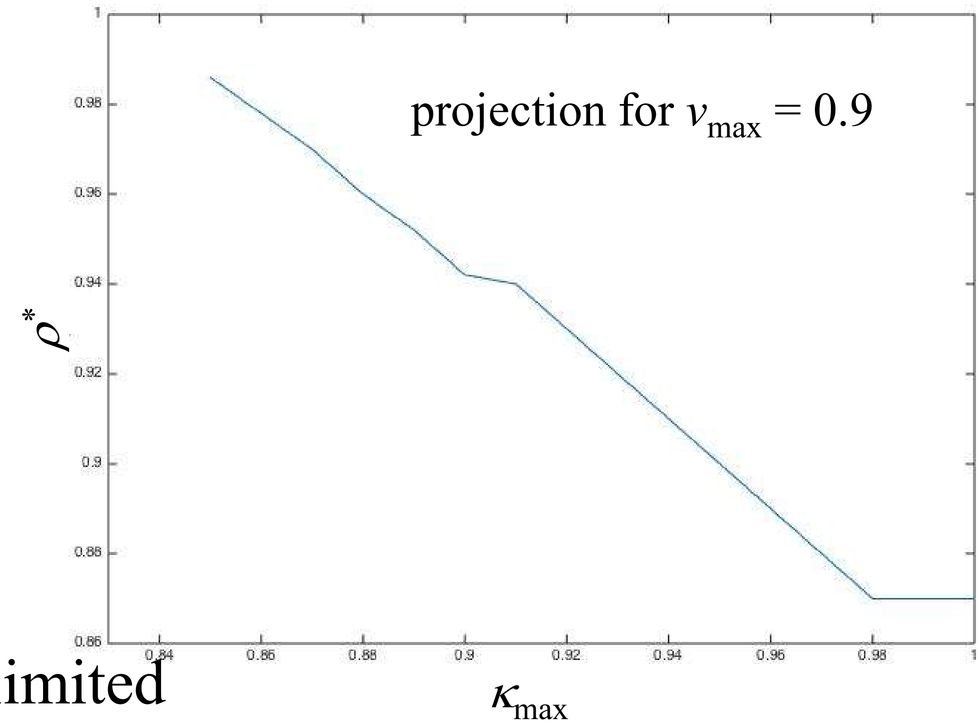
- Preliminary findings
  - Threshold remains present under termination for both fixed and dynamic bidding
  - Dynamic bidding seems to again yield larger ρ^*





More General User Profiles

- $v_{\min} = 0, \kappa_{\min} = 0.$
- $v_{\max} \in [0.1, 1.5],$
- $\kappa_{\max} \in [0.1, 1.5].$



- Gaussian copula
  - marginals: uniform distributions
- Optimal pricing search limited to one and two prices

For all ($v_{\max}$, $\kappa_{\max}$) pairs, the result still holds, *i.e.*, optimality of one vs. more than one price depends on $\rho > \rho^*$

Some Closely Related Works

Fixed and Market Pricing for Cloud Services

-V. Abhishek, I.A. Kash, and P. Key, NetEcon 12
(updated and expanded 2017 version)

Revenue Maximization for Cloud Computing Services

- C. Kilcioglu and C. Maglaras, SIGMETRICS 15

- Cloud computing services under (mostly) infinite capacity.
- Jobs are heterogeneous in valuation, delay sensitivity.
- Result: spot service is useful under some conditions.

- Neither work explicitly studies the role of correlation.

Summary and Extensions

- Summary
  - Results highlight the role of correlation in determining the value of offering a spot service
- Extensions: Develop more systematic guidelines to assess
  - Penalty of using fixed bidding
  - Benefits and disadvantages (to users and providers) of early job termination
  - Impact of non-linearities in delay sensitivity
- Explore other pricing mechanisms, *e.g.*, auctions when supporting opportunistic jobs