Cost-Aware VM Purchasing for Application Service Providers with Arbitrary Demands

Presenter: Shengkai Shi

May 30, 2014

IaaS cloud

Application service providers

Challenges

- Cost management: problem of fundamental importance.
- Service guarantee.

Pricing options

- On-demand instances.
  - No commitment.
  - Pay as you go.
- Reserved instances.
  - Reservation fee + discounted price.
  - Suitable for long-term usage commitment.
- Spot instances.
  - Substantially lower hourly rate.
  - Risk job interruptions.

Amazon spot instances

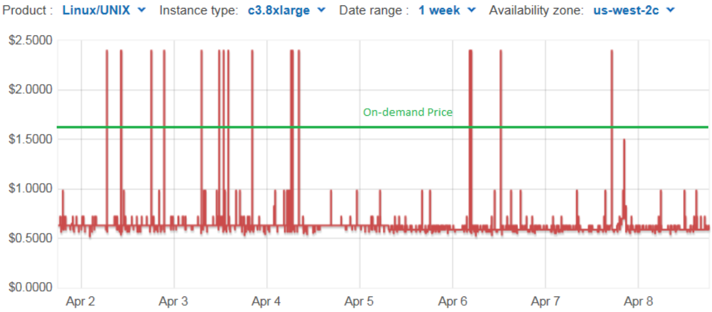

Figure: The variation in Amazon EC2 spot prices for Linux/UNIX c3.8xlarge instances in the US-West-2c region from April 2 to April 8, 2014.

A pricing example

TABLE I

PRICING OF RESERVED INSTANCE, ON-DEMAND INSTANCE AND SPOT INSTANCE (LINUX, US WEST) IN AMAZON EC2, AS OF MARCH 18, 2014.

Instance Type	Pricing Option	Up-front	Hourly
m3.medium	1-year reserved	$317	$0.041
	on-demand	$0	$0.124
	spot	$0	$0.0333
m3.large	1-year reserved	$633	$0.081
	on-demand	$0	$0.248
	spot	$0	$0.0662

Related work

- Dynamic Server Provisioning to Minimize Cost in an IaaS Cloud.
    - Hong *et al.*, SIGMMETRICS 2011.
- Optimal Resource Rental Planning for Elastic Applications in Cloud Market.
    - Zhao *et al.*, IPDPS 2012.
- Dynamic Cloud Resource Reservation via Cloud Brokerage.
    - Wang *et al.*, ICDCS 2013.
- Optimal Online Multi-Instance Acquisition in IaaS Clouds.
    - Wang *et al.*, ICAC 2013.
- Dynamic Resource Allocation for Executing Batch Jobs in the Cloud
    - Jain *et al.*, ICAC 2014.

Motivation

- Integrate all available pricing options.
- Design an efficient online algorithm to guide VM purchasing decisions.

System model overview

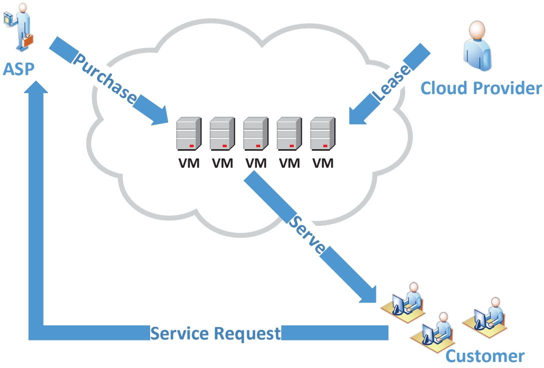

Figure: System overview.

Job model

- Job, or service request:
  - The type of required VM: s_g.
  - The Service Level Agreement: l_g.
  - The number of required time slots: m_g.

Scheduling model

- Number of type-g job arrivals at t: $r_g(t)$.
- Number of newly scheduled type-g jobs at t: $u_g(t)$.
- Number of leftover type-g jobs at t: $u_g(t^-)$.
- Number of dropped type-g jobs at t: $D_g(t)$.
- Workload queue dynamics:

$$\begin{aligned} Q_g(t+1) =& max\{Q_g(t) - u_g(t) - u_g(t^-) - m_g D_g(t), 0\} \\ &+ m_g r_g(t). \end{aligned} \tag{1}$$

Virtual queue

- Virtual queue $Z_g(t)$, associated with $Q_g(t)$, starts from $Z_g(0) = 0$ and evolves as:

$$\begin{aligned} Z_g(t+1) = \max\{ & Z_g(t) + \mathbf{1}_{\{Q_g(t)>0\}} \cdot [\epsilon_g - u_g(t) - u_g(t^-)] \\ & - m_g D_g(t) - \mathbf{1}_{\{Q_g(t)=0\}} u_g^{max}, 0\}, \forall g \in [1, G]. \end{aligned} \tag{2}$$

VM provisioning model

- Mix of three pricing options.
  - Type-s reserved instances at t: $a_s(t)$.
  - Type-s on-demand instances at t: $b_s(t)$.
  - Type-s spot instances at t: $f_s(t)$.
  - Reservation period: N.
  - Number of reserved instances that are effective at t: $\sum_{\tau=t-N+1}^{t} a_s(\tau)$.
- The total supply should always accommodate the total demand:

$$\sum_{\tau=t-N+1}^{t} a_s(\tau) + b_s(t) + f_s(t) \geq \sum_{g:s_g=s} [u_g(t) + u_g(t^-)], \forall t, \forall s \in [1, S]. \tag{3}$$

Cost model

- Reserved instances
  - Upfront one-time payment: h_s.
- On-demand instances
  - Charge per billing cycle at t: $\beta_s(t)$
- Spot instances
  - Spot price at t: $\gamma_s(t)$

How to manage interruption

- Spot price updates periodically: every 5 minutes on Amazon EC2.
- Replace the terminated spot instance by a new on-demand instance.
- The probability of an interruption event within $[t, t+1]$: $P_s(t)$.
- Expected cost incurred by one spot instance:
  $[1 - P_s(t)]\gamma_s(t) + P_s(t)\beta_s(t)$.

Problem formulation

- Total VM cost in time slot t = reservation cost+on-demand cost+ spot cost+penalty.

$$Cost(t) = \sum_{s\in[1,S]} \{h_s a_s(t) + \beta_s(t)b_s(t) + P_s(t)\beta_s(t)f_s(t) + [1 - P_s(t)]\gamma_s(t)f_s(t)\} + \sum_{g\in[1,G]} D_g(t)\sigma_g. \tag{4}$$

- VM cost minimization pursued by an ASP:

$$\min \quad \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} E[Cost(t)]. \tag{5}$$

Lyapunov optimization: drift-plus-penalty framework

- One-shot optimization problem.

$$\min \quad \varphi_1(t) + \varphi_2(t) \tag{6}$$

where

$$\varphi_1(t) = \sum_{g \in [1,G]} D_g(t)[V\sigma_g - m_g Q_g(t) - m_g Z_g(t)]$$

$$\begin{aligned}\varphi_2(t) = V \sum_{s \in [1,S]} \{ & h_s a_s(t) + \beta_s(t) b_s(t) + [1 - P_s(t)]\gamma_s(t) f_s(t) \\ & + P_s(t)\beta_s(t) f_s(t)\} - \sum_{g \in [1,G]} u_g(t)[Q_g(t) + Z_g(t)],\end{aligned}$$

and $V > 0$ is a user-defined parameter for gauging the optimality of the time-averaged cost.

Job dropping

- The number of dropped jobs $D_g(t)$, $\forall g \in [1, G]$, is obtained by solving the following optimization problem:

$$\min \quad D_g(t)[V\sigma_g - m_g Q_g(t) - m_g Z_g(t)] \tag{7}$$

Job scheduling and instance purchasing

- The decisions on the number of scheduled jobs, the number of newly reserved instances, the number of launched on-demand instances, and the number of acquired spot instances, can be got by solving the following optimization problem:

$$\begin{aligned} \min \quad & V \sum_{s \in [1,S]} \{ h_s a_s(t) + \beta_s(t) b_s(t) + [1 - P_s(t)] \gamma_s(t) f_s(t) \\ & + P_s(t) \beta_s(t) f_s(t) \} - \sum_{g \in [1,G]} u_g(t) [Q_g(t) + Z_g(t)] \end{aligned} \tag{8}$$

Online algorithm

- Make VM purchasing decisions without future information.

Algorithm 1 Dynamic VM Purchasing Cost Minimization at Time t

Input: $r_g(t)$, $Q_g(t)$, $Z_g(t)$, $u_g(t^-)$, σ_g, h_s, $\beta_s(t)$, $\gamma_s(t)$, $\forall s \in [1, S]$, $\forall g \in [1, G]$.
Output: $D_g(t)$, $u_g(t)$, $f_s(t)$, $a_s(t)$, $b_s(t)$, $\forall s \in [1, S]$, $\forall g \in [1, G]$.

1: **Job dropping**: Decide $D_g(t)$ solving optimization problem (7);
2: **Job scheduling and instance purchasing**: Decide $u_g(t)$, $f_s(t)$, $a_s(t)$, and $b_s(t)$ solving optimization problem (8);
3: Update $Q_g(t)$ and $Z_g(t)$ with Eqn. (1) and (2).

Queueing delay bound

Theorem 1. (Queueing Delay Bound) If $m_g D_g^{max} > \max\{m_g r_g^{max}, \epsilon_g\}$, each workload queue $Q_g(t)$ and each virtual queue $Z_g(t)$ are upper bounded by $Q_g^{max} = V\sigma_g/m_g + m_g r_g^{max}$ and $Z_g^{max} = V\sigma_g/m_g + \epsilon_g$, respectively, $\forall t, \forall g \in [1, G]$. The SLA of each job can be guaranteed by $\frac{Q_g^{max}+Z_g^{max}}{\epsilon_g}$, $\forall g \in [1, G]$, if we set $\epsilon_g = \frac{Q_g^{max}+Z_g^{max}}{l_g}$.

Performance optimality

Theorem 2. (Performance Optimality) Suppose $N > m^{max}$, under our online algorithm we have :

$$\lim_{\kappa \to \infty} \frac{1}{\kappa N} \sum_{x=0}^{\kappa-1} \sum_{t=xN}^{(x+1)N-1} E[Cost(t)]$$
$$\leq Cost^* + \frac{B}{V} + \frac{(N - m^{max})(N - m^{max} - 1)}{2VN} B_1$$
$$+ \frac{N-1}{2V} \sum_{g \in [1,G]} [(\epsilon_g)^2 + (m_g)^2 (r_g^{max})^2]$$
$$+ \frac{m^{max}}{N} \sum_{s \in [1,S]} (h_s a_s^{max} + \beta_s^{max} b_s^{max} + \beta_s^{max} f_s^{max})$$
$$+ \frac{(N - m^{max})(N - m^{max} - 1)}{2N} \sum_{s \in [1,S]} (f_s^{max})(\beta_s^{max} - \gamma_s^{min}), \qquad (9)$$

Conclusion

- We propose an efficient VM purchasing strategy.
  - Addresses possible terminations of spot VMs.
  - Leverage three pricing options to fully exploit the economic advantages of IaaS cloud.
  - Achieves a time-averaged VM cost arbitrarily close to the offline optimum.
- Future work
  - Trace-driven simulations.

Thanks!