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
Tradeoffs in Cloud Pricing Options

On-demand instances

No commitment
Pay-as-you-go

<table>
<thead>
<tr>
<th></th>
<th>Linux/UNIX Usage</th>
<th>Windows Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard On-Demand Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small (Default)</td>
<td>$0.080 per Hour</td>
<td>$0.115 per Hour</td>
</tr>
<tr>
<td>Medium</td>
<td>$0.160 per Hour</td>
<td>$0.230 per Hour</td>
</tr>
<tr>
<td>Large</td>
<td>$0.320 per Hour</td>
<td>$0.460 per Hour</td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.640 per Hour</td>
<td>$0.920 per Hour</td>
</tr>
</tbody>
</table>

Reserved instances

Reservation fee + discounted price
Suitable for long-term usage commitment
### On-demand v.s. Reservation

<table>
<thead>
<tr>
<th></th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
</table>
| **On-demand**  | 1. Flexible  
2. Fits sporadic workload          | Expensive for long-term usage                             |
| **Reservation**| Cost efficient for long-term usage       | 1. Long-term usage commitment  
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 broker reserves a large pool of instances

Users purchase instances from the broker in an “on-demand” fashion

A Cloud Brokerage Service

Wei Wang, Department of Electrical and Computer Engineering, University of Toronto
Why cloud broker?
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
Alleviate the pricing inefficiency of on-demand instances

Partial usage is counted as a full billing cycle

The broker can *time-multiplex* partial usage

**Without broker**

- Billing cycle (an instance-hour)
- User 1
- User 1
- Instance 1
- Instance 2
- User 2

**With broker**

- User 1
- User 2
- User 1
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 \( d_1, \ldots, d_T \)

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

This is an integer program!
Optimal Solution: Dynamic Programming
The Curse of Dimensionality

High dimensional dynamic programming

High dimensional state: \( s_t := (t, x_1, \ldots, x_{T-1}) \)

\( x_i \) : # 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 $\geq 3$ 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

\[ V_i(t) = \min\{V_i(t - \tau) + \gamma, \ V_i(t - 1) + c_i(t)\} \]

**Strictly better than Per-Interval Reservation, and is also 2-competitive**
When demand predictions are unavailable
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
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
update the reservation history

Suppose the found value is $i$ that we would have to launch instances we should have reserved at time $t$. Clearly, all these requirements are covered in the reservation decision we make at time $t$ if and only if there is demand at $t$, i.e., $\delta_i(t) = 0$. We call the algorithm that makes reservation decisions based only on history. Revenue. After reservations have been made in level $t$, incurring a cost of $V_{\bar{t}}(\bar{m}) := (\sum_{i=t+1}^{\bar{t}} - \sum_{i=t+1}^{\bar{t}} \delta_i(t) + \sum_{i=t+1}^{\bar{t}} \delta_i(t))$, the number of reserved instances to be passed to level $x$ in each level requires only $O(1)$ to compute. At time $t$, if an instance is left over from upper levels to use at time $t$, we have to launch an on-demand instance to serve the demand at time $t$, yet no reserved instance is passed over. If an instance is left over from upper levels to use at time $t$ as if we had reserved $i$ instances at the current time $t$, the online algorithm makes a reservation gap of $\sum_{i=t}^{\bar{t}} \delta_i(t)$, yet no reserved instance is passed over. We now make a "regret" to calculate how many instances we should have reserved at time $t$ given by Algorithm 2 incurs a cost no more than $\sum_{i=t}^{\bar{t}} \delta_i(t)$, incurring a cost of $\sum_{i=t}^{\bar{t}} \delta_i(t)$. The algorithm incurs a cost no more than $\sum_{i=t}^{\bar{t}} \delta_i(t)$, incurring a cost of $\sum_{i=t}^{\bar{t}} \delta_i(t)$. The algorithm incurs a cost no more than $\sum_{i=t}^{\bar{t}} \delta_i(t)$, incurring a cost of $\sum_{i=t}^{\bar{t}} \delta_i(t)$.
Aggregation Smoothes Out Demand Bursts

We take such a dataset as input, and ask the question: How many computing instances would a user require in each hour?

We hence classify all 933 users into the following three groups, each of which has a different demand pattern:

1. **Group 1 (High Fluctuation):** Users in this group have a demand fluctuation level greater than 1, represented by “+” in Fig. 8. There are 286 users in this group, represented by “x” in Fig. 8. These users demand a high amount of instances, with a mean demand of less than 30 instances.

2. **Group 2 (Medium Fluctuation):** Users in this group have a demand fluctuation level less than 1, represented by “+” in Fig. 8. There are 348 users in this group, represented by “x” in Fig. 8. These users demand a medium amount of instances, with a mean demand of less than 100.

3. **Group 3 (Low Fluctuation):** Users in this group have a demand fluctuation level less than 100. They have small demands, with a mean less than 30 instances.

Our evaluations are carried out for each group. We start with the Group 1 (High Fluctuation) as an example.

- **Group 1 (High Fluctuation):** The demand pattern of this group is shown in Fig. 7. There are 286 users in this group, represented by “x” in Fig. 8. These users demand a high amount of instances, with a mean demand of less than 30 instances.

- **Group 2 (Medium Fluctuation):** The demand pattern of this group is shown in Fig. 7. There are 348 users in this group, represented by “x” in Fig. 8. These users demand a medium amount of instances, with a mean demand of less than 100.

- **Group 3 (Low Fluctuation):** The demand pattern of this group is shown in Fig. 7. There are 300 users in this group, represented by “x” in Fig. 8. These users demand a low amount of instances, with a mean demand of less than 30 instances.

Fig. 8. A typical user's demand is shown in the bottom graph of the diagram.

We see from Fig. 9a and 9b that aggregating bursty users (i.e., users in Group 1 and 2) results in a steadier curve, with a demand fluctuation level much smaller than that of any individual user.

Addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.

The demand fluctuation level less than 1, represented by “+” in Fig. 8, is observed in all cases. In addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.

Fig. 9. Aggregation suppresses the demand fluctuation of individual users. Each circle represents a user. The line indicates the demand fluctuation level (the ratio between the demand standard deviation and mean) in the aggregate demand curve.

We see from Fig. 9a and 9b that aggregating bursty users (i.e., users in Group 1 and 2) results in a steadier curve, with a demand fluctuation level much smaller than that of any individual user.

Addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.

The demand fluctuation level less than 1, represented by “+” in Fig. 8, is observed in all cases. In addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.

Fig. 9. Aggregation reduces the wasted instance-hours (k).

The demand fluctuation level less than 1, represented by “+” in Fig. 8, is observed in all cases. In addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.

The demand fluctuation level less than 1, represented by “+” in Fig. 8, is observed in all cases. In addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.

The demand fluctuation level less than 1, represented by “+” in Fig. 8, is observed in all cases. In addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.

The demand fluctuation level less than 1, represented by “+” in Fig. 8, is observed in all cases. In addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.

The demand fluctuation level less than 1, represented by “+” in Fig. 8, is observed in all cases. In addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.

The demand fluctuation level less than 1, represented by “+” in Fig. 8, is observed in all cases. In addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.

The demand fluctuation level less than 1, represented by “+” in Fig. 8, is observed in all cases. In addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.

The demand fluctuation level less than 1, represented by “+” in Fig. 8, is observed in all cases. In addition, Fig. 9d presents the result of aggregating all the users. In all cases, the aggregated demand is stabler and represents the fluctuation level of the aggregated demand.
The Reduction of Partial Usage

The figure shows the reduction in wasted instance-hours achieved through aggregation for different demand fluctuation levels: High, Medium, Low, and All users. The chart includes bars for both the 'Before aggregation' and 'After aggregation' scenarios. For each demand fluctuation category:

- **High Fluctuation**: 16.5% reduction in wasted instance-hours.
- **Medium Fluctuation**: 30.5% reduction.
- **Low Fluctuation**: 5.6% reduction.
- **All Users**: 23.4% reduction.

These reductions demonstrate the effectiveness of aggregation in mitigating partial usage, especially for users with high fluctuation levels.
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/