A Day Late and a Dollar Short

The Case for Research on Cloud Billing Systems

Robert Jellinek       Yan Zhai       Thomas Ristenpart       Michael Swift
{jellinek | yanzhai | rist | swift}@cs.wisc.edu
Outline

• Motivation: Why study cloud billing?
• Our contributions
• Our study
• Results
• Conclusions and future work
Motivation: Cloud Billing

• Many performance, reliability, and cost efficiency studies of the cloud.

• Little attention has been paid to their billing systems.

• Pay-as-you-go pricing model relies upon complex, large-scale billing systems.
Motivation: Cloud Billing

• Resource accounting is an interesting challenge.

• How to track all compute resource usage in
  • real time,
  • at fine granularity
  • maintaining accuracy,
  • and not hurting performance?
Study of cloud billing mechanisms.

- We were able to:
  - **Disambiguate billing** by reverse engineering
  - **Uncover bugs**:
    - Race conditions in EC2
    - Inconsistencies across billing interfaces in EC2
    - Rackspace bug causing overcharges
  - **Detect systematic undercharging** from caching/aggregation
  - **Characterize performance** of billing latency.
Study Overview

• Guiding question:
  How accurate, timely, and predictable are customer-facing billing interfaces?

• Measured billing for:
  • **compute time**
  • **storage** (IOPS and capacity)
  • **network** usage

• Experimented on AWS, GCE, and Rackspace
• Calculated **billing latency** of billing interfaces.
Methodology

• **Instrument providers’ API calls** to record timestamps of all operations.

• **Launch an instance and execute one of several workloads**

• **Fetch instance’s OS-based resource-usage data**

• **Terminate instance after workload completion**

• **Poll for billing updates** over all measured resources.
Billing Interfaces

- **EC2:**
  - Web-based GUI *management console*
  - Programatically accessible CSVs:
    - *Hourly*
    - *Monthly* (to date)
    - *Cost-allocation* (allows user to tag resources and filter costs by tag)

- **GCE:**
  - Web-based GUI interface

- **Rackspace:**
  - Web-based GUI interface
# Amazon Web Billing Interface

## Details

### AWS Service Charges

<table>
<thead>
<tr>
<th>Service</th>
<th>Charge</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Amazon Elastic Compute Cloud</strong></td>
<td>$0.95</td>
</tr>
<tr>
<td><strong>US East (Northern Virginia) Region</strong></td>
<td>$0.95</td>
</tr>
<tr>
<td>Amazon EC2 running Linux/UNIX</td>
<td></td>
</tr>
<tr>
<td>$0.020 per Micro Instance (1h micro) instance-hour (or partial hour)</td>
<td>33 Hrs</td>
</tr>
<tr>
<td>Amazon EC2 EBS</td>
<td></td>
</tr>
<tr>
<td>$0.100 per GB-month of provisioned storage (blended price)*</td>
<td>2.367 GB-Mo</td>
</tr>
<tr>
<td>$0.10 per 1 million I/O requests</td>
<td>76,834 IOs</td>
</tr>
<tr>
<td>$0.125 per GB-Month of snapshot data stored (blended price)*</td>
<td>0.344 GB-Mo</td>
</tr>
<tr>
<td><strong>Amazon Simple Notification Service</strong></td>
<td>$0.00</td>
</tr>
<tr>
<td><strong>US East (Northern Virginia) Region</strong></td>
<td>$0.00</td>
</tr>
<tr>
<td>First 100,000 Amazon SNS API Requests per month are free</td>
<td>54 Requests</td>
</tr>
<tr>
<td><strong>AWS Data Transfer (excluding Amazon CloudFront)</strong></td>
<td>$0.00</td>
</tr>
<tr>
<td>$0.00 per GB - data transfer in per month</td>
<td>0.034 GB</td>
</tr>
<tr>
<td>$0.00 per GB - first 1 GB of data transferred out per month</td>
<td>0.001 GB</td>
</tr>
<tr>
<td>$0.00 per GB of regional data transfer in/out (blended price)*</td>
<td>0.000020 GB</td>
</tr>
</tbody>
</table>
Billing Latency

- We define as: the time between when a resource is consumed, and when the corresponding charge for that usage become visible to the customer.

- Important that billing updates are:
  - Timely
  - Predictable
  - Atomic

- But this was not the case
EC2 Web/CSV Billing Latency

Web Console
Avg latency: 6:41 hours
Std dev: 4:10 hours

CSV
Avg latency: 8:15 hours
Std dev: 3 hours
GCE/Rackspace Billing Latency

Lower bounds on GCE billing latency for 13 instances, in DAYS. Error bars indicate upper bounds.

Rackspace billing latency for 21 instances in HOURS, +/- 10 minutes. All billing updates occurred between 9-10am UTC.
EC2: Why such latency?

We deliberately staggered the start times of instances. The billing update schedule suggests periodic batch processing.
What is “Compute Time”?

Major events in an instance lifetime
EC2 Compute Time Results

Fract. of instances billed 2 hours

$T_{kill} - T_{launch}$
Compute Time Anomalies

Other anomalies in EC2 compute billing

Measured uptime $T_{down} - T_{up}$ for 272 EC2 instances run with $3590 \leq T_{kill} - T_{launch} \leq 3600$ versus the number of hours billed.
Compute Time

• We created a special “fast boot” kernel that booted and immediately sent heartbeat messages to our control server.
• Terminated instances $\Delta$ seconds after launch.

<table>
<thead>
<tr>
<th>$\Delta$</th>
<th># launched</th>
<th># ran</th>
<th>Avg. uptime per instance (s)</th>
<th>Hours billed</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\leq 16$</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>20</td>
<td>1</td>
<td>115</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>20</td>
<td>1</td>
<td>116</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>4</td>
<td>117</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>6</td>
<td>118</td>
<td>3</td>
</tr>
</tbody>
</table>

• Race condition causes some instances to not get billed, but yield roughly 2 minutes of free uptime.
Storage Billing

• Provider charges based on its view of storage ops

• Bugs:
  • In Rackspace deleting a volume before detaching from an instance caused it to hang and accrue charges.

• Anomalies:
  • I/O charges in EC2 are lower than /proc/diskstats would suggest; caching or aggregation?
  • Example:
    • Write (4kb); write(4kb); -> 1 storage op
    • Write (4kb); seek(1 million); write(4kb) -> 2 storage ops
IOPS Aggregation/Caching?

Ratio of the number of storage ops measured by `/proc/diskstats` to the number of ops billed by EC2
Network Billing

• EC2 underbilled by an average of 5.6% for Internet-outbound traffic.

• Measured traffic using Netfilter/iptables

• Some Rackspace instances were greatly underbilled: 125 MB / 1 GB not billed. Another bug?
Summary

• Current billing systems are:
  • Not timely
  • Often unpredictable
  • Not transparent
  • Largely inaccessible via APIs

• Customer must rely on provider for ground truth
• Cannot use billing information for real-time deployment decisions.
Conclusions

• Future research should investigate the tradeoffs between performance, and accurate, timely, transparent resource accounting and billing.

• This will likely necessitate collaboration with industry.

• Providers should expose a billing API to enable programmatic queries of billing information.
Thank you!