Practical steps to optimizing your use of public cloud capacity

Andrew Hillier, CTO
What is Densify?

- Densify is a predictive analytics service that optimizes public cloud, bare metal cloud and on-premise virtual environments in real-time, enabling customers to operate with less cloud cost, less infrastructure and better performing applications.

- It is a unique combination of SaaS-based analytics and expert guidance.
But isn’t public cloud cheap?

Many assume that public cloud is cheap, but that isn’t always the case.

- Batch job that runs hot and then turns off
- Continuous business service that runs 24x7
- Scale-out app that dynamically starts and stops instances
To make things even more complicated…

This problem is compounded by several effects:

- Lack of visibility
- Complexity of cloud offerings
- Lack of processes and controls

The result: the monthly bill is typically far higher than expected—It is also very unpredictable. This is a monthly cost, not a sunk cost.

“The when deploying an Amazon Web Services Elastic Cloud Compute instance, there are more than 1.7 million potential considerations, and EC2 is only one of 90 services offered.”

Source: Gartner
The knee-jerk reaction to “sticker shock” is to buy a product that can read the bill and make sense of all the granular billing data.

But that is just the start – there are multiple strategies for cloud cost optimization, and the further you go the higher the savings:

1. **Read the bill** and assign costs to users/LOBs
2. **Identify risks** and eliminate resource pressure
3. **Right-size** instances, identify deadwood and turn off
4. **Modernize** and align with latest services and catalogs
5. **Assess alternatives** including other hosting providers
6. **Leverage bare metal** with hypervisor, dedicated hosts, etc.
7. **Leverage containers** and other advanced hosting strategies

---

**Increasing Savings**

- Reserved Instances
  - 20%
  - 35%
  - 50%
  - 80%

Public cloud optimization
1. Reading the bill, budgeting, and allocating costs

- Monthly view of historical and projected cloud cost against budget
- Cost allocation breakdown by business group or service type, with drill-down to billing details
### Practical considerations

- Tagging becomes critical to ensuring that there is accountability/chargeback
  - CMDBs and ITIL-style discipline are often thrown out the window

There are usually many cloud accounts
  - Some orgs have dozens of accounts that were created by different groups
  - These need to be cleaned up or aggregated under a master account

<table>
<thead>
<tr>
<th>Location</th>
<th>Department</th>
<th>Backup Window</th>
<th>DR Category</th>
<th>Application Tier</th>
<th>Environment</th>
<th>Maintenance Window</th>
<th>Owner</th>
<th>Primary Application</th>
<th>Primary Business Service</th>
<th>UPS Category</th>
<th>Acquisition Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>London</td>
<td>Shared Services</td>
<td>12-6am</td>
<td>Resilient</td>
<td>Tier 3 - Database Server</td>
<td>Prod</td>
<td>Weekend Fixed Time</td>
<td>ITSS</td>
<td>Oracle</td>
<td>Online Banking</td>
<td>Non-Essential</td>
<td>1 hour</td>
</tr>
<tr>
<td>London</td>
<td>Engineering</td>
<td>12-6am</td>
<td>Live</td>
<td>Tier 3 - Database Server</td>
<td>Dev</td>
<td>Weekday Fixed Time</td>
<td>Dev</td>
<td>Oracle</td>
<td>Settlements</td>
<td>Low Availability</td>
<td>1 hour</td>
</tr>
<tr>
<td>London</td>
<td>Engineering</td>
<td>12-6am</td>
<td>Live</td>
<td>Tier 2 - Database Server</td>
<td>Dev</td>
<td>Weekday Fixed Time</td>
<td>Dev</td>
<td>Oracle</td>
<td>Settlements</td>
<td>Low Availability</td>
<td>1 hour</td>
</tr>
<tr>
<td>Boston</td>
<td>Engineering</td>
<td>12-6am</td>
<td>Restore based</td>
<td>Tier 3 - Database Server</td>
<td>Staging</td>
<td>Weekend Fixed Time</td>
<td>ITSS</td>
<td>Oracle</td>
<td>Online Banking</td>
<td>Low Availability</td>
<td>1 hour</td>
</tr>
<tr>
<td>Boston</td>
<td>Operations</td>
<td>12-6am</td>
<td>Restore based</td>
<td>Tier 3 - Database Server</td>
<td>Staging</td>
<td>Weekend Fixed Time</td>
<td>ITSS</td>
<td>Oracle</td>
<td>Online Banking</td>
<td>Low Availability</td>
<td>1 hour</td>
</tr>
<tr>
<td>Boston</td>
<td>Operations</td>
<td>12-6am</td>
<td>Live</td>
<td>Tier 3 - Database Server</td>
<td>Staging</td>
<td>Weekday Fixed Time</td>
<td>Dev</td>
<td>Oracle</td>
<td>Settlements</td>
<td>Near Continuous</td>
<td>2 hours</td>
</tr>
</tbody>
</table>
Modeling cloud utilization

APIs provide access to utilization data for public cloud workloads, and it is important to track the details and to model cloud workloads and on-prem systems in a consistent way. This allows workload patterns to be tracked and normalized using benchmarks, enabling accurate analysis of operational patterns and business cycles, precise catalog optimization, and what-if analysis between providers.
Even basic visibility is very valuable

- Be prepared to lose a decade or two of maturity

- The provider APIs have many quirks and nuances
  - E.g. Amazon CloudWatch – no memory
  - Azure – still advancing their APIs
2. Identifying risks

- Relatively simple: high utilization = risk, so increase the instance size
Not so fast…

- Fairly advanced policies are needed to properly identify risks
  - You could easily get into a “bump-up loop”

- Scale groups and other constructs also impact the actions taken

Batch jobs

Memory usage…

..vs “active” memory usage
3. Identifying waste and right-sizing

- Relatively simple: low utilization = waste, so decrease the instance size
Not so fast…

Last Month of Activity:

Business cycle has peaks of high utilization throughout the month

Busiest Day:

Densify analysis sees 85% and recommends a bump-up

Using 90\textsuperscript{th} percentile gives 3.25% and recommends a \textit{bump-down}, which would be catastrophic to the app.
Practical considerations

- Need to properly analyze historical patterns

- Need to analyze against all instance types
  - Standard
  - CPU optimized
  - Memory optimized
  - Micro/burstable

- Need to normalize using benchmarks
  - Necessary to go between “instance classes”

- Other considerations
  - Need to properly identify Deadwood/Zombie instances
  - Scheduling on and off may be a strategy
4. Modernization

- There may be newer offerings that you can leverage to save money

- Like a cell phone plan, the provider won’t necessarily notify you…

- …and even if they did you could never keep up

- But with the right analytics this becomes an *unprecedented opportunity*
Right-sizing and modernizing instances

Recent customer example: downsizing saved 20%, modernization saved 36%
Reserved Instances

- If you have them it will constrain what you can do
  - Need a roll-over plan to get to optimal state over time

- If you don’t have them then there is a big opportunity
  - But you don’t want to lock into the wrong sizes

- Very complex rules, changing all the time
  - Availability zones, instance classes
  - Splitting and merging
  - Selling and disposal

- Optimizing RIs requires an understanding of what you will need *in the future*
  - Not the mentality of most cloud users – many think they never need to plan again
5. Assess alternatives

Example 1: the impact of cloud catalogs on hosting decisions

In order to get enough Memory in AWS you are forced to over-configure CPU resources.

In order to get enough CPU capacity in Azure you are forced to over-configure memory resources.

<table>
<thead>
<tr>
<th></th>
<th>Current</th>
<th>AWS</th>
<th>Azure</th>
<th>SoftLayer</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Memory</td>
<td>16384</td>
<td>16384</td>
<td>28672</td>
<td>16384</td>
</tr>
<tr>
<td>Catalog Type</td>
<td>--</td>
<td>m4.xlarge-4X16-Windows</td>
<td>Standard_GS1-2X28.672-Windows</td>
<td>soft.2x16-2X16-Windows</td>
</tr>
</tbody>
</table>

SoftLayer allows independent sizing of CPU and Memory, giving a better fit.
Assess alternatives

Example 2: the impact of **cloud infrastructure performance** on hosting decisions

Because AWS has faster cores than Azure or SoftLayer, this workload will fit in a 4-way instance, making it cheaper than in other providers for this specific workload.

Benchmarks must be used to automatically account for this.

<table>
<thead>
<tr>
<th></th>
<th>Current</th>
<th>AWS</th>
<th>Azure</th>
<th>SoftLayer</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Memory</td>
<td>16384</td>
<td>31232</td>
<td>28672</td>
<td>32768</td>
</tr>
<tr>
<td>Cost</td>
<td>--</td>
<td>$3,749</td>
<td>$12,158</td>
<td>$5,895</td>
</tr>
<tr>
<td>Cint 2006</td>
<td>315.38</td>
<td>199</td>
<td>256</td>
<td>329.2</td>
</tr>
<tr>
<td>Catalog Type</td>
<td>--</td>
<td>r3.xlarge-4X30.5-Windows</td>
<td>Standard_D4-8X28.672-Windows</td>
<td>soft.8x32-8X32-Windows</td>
</tr>
</tbody>
</table>
Automatically assessing hosting alternatives
6. Leveraging bare metal

- T-Shirt instance sizing model
- Cost based on catalog size
- Typically sized to peak utilization
- User pays for capacity whether it is used or not (no overcommit)

- Bare-metal server model
- User rents a server, not a VM
- Hypervisor allows workload stacking
- User has opportunity to dovetail workloads and leverage overcommit
Analysis example

VMware on SoftLayer Bare Metal
Analysis Results for Warehouse apps to external cloud

Consolidation Summary

<table>
<thead>
<tr>
<th>Public cloud using “T-Shirt” sizing</th>
<th>Same provider using bare metal (with hypervisor)</th>
</tr>
</thead>
<tbody>
<tr>
<td>60 SoftLayer Virtual Instances Various Sizes</td>
<td>4 SoftLayer Bare Metal Nodes Xeon E5-2690 128GB</td>
</tr>
<tr>
<td>$13,531/month</td>
<td>$6,681/month</td>
</tr>
</tbody>
</table>

Net Savings: 51%

Overcommit enables over 4X higher utilization
What about on-prem?

“I&O leaders are being led to think that all IaaS workloads belong within the public cloud, causing some to attempt migrations that are not cost-effective or operationally effective.”

Source: Gartner
Analyzing cloud workloads back on-prem

Every 3 months in AWS would pay for the on-prem compute gear

AWS instances: $108K / month (on-demand pricing)

Hewlett Packard Enterprise
List price: $300K (compute only, internet pricing, Vmware license, enclosure, etc. extra)
7. Leveraging containers

983 Workloads: AWS 1-year Hosting Cost with Densify catalog optimization

$1,892,733

Extra large Amazon Instances with optimized container stacking using Densify

$325,285

Net Savings: 82%

S  M  L
EC2 Container Service (ECS) – Planned Integration

EC2 Instances → ECS API → AWS ECS

Real-Time Routing & Placement

Blox

EC2 Instances → ECS API → CloudWatch API

Blox Extensions

CloudWatch API
There is a huge opportunity to drive optimization in public cloud if you have the ability to properly crunch the numbers.

There are multiple strategies that can be employed, and the further you go the higher the savings you will realize:

1. **Read the bill** and assign costs to users/LOBs
2. **Identify risks** and eliminate resource pressure
3. **Right-size** instances, identify deadwood and turn off
4. **Modernize** and align with latest services and catalogs
5. **Assess alternatives** including other hosting providers
6. **Leverage bare metal** with hypervisor, dedicated hosts, etc.
7. **Leverage containers** and other advanced hosting strategies

---

**Conclusion**

Increasing Savings

- Reserved Instances
  - 20%
  - 35%
  - 50%
  - 80%

---

Copyright © 2017, Cirba Inc. DBA Densify. All Rights Reserved.
Questions?
Learn more about Densify

How does Densify optimize public cloud, bare metal cloud and on-premise infrastructure?

First 14 days are free
Get a free t-shirt when you sign up and connect

www.densify.com/cmg
SAVE THE DATE!

CMG imPACt 2017

November 6 – 9, 2017
Loews New Orleans Hotel