Availability Knob
Flexible User-Defined Availability in the Cloud

Mohammad Shahrad and David Wentzlaff

October 5, 2016
IaaS Providers and Availability Guarantees

One thing in common:
Fixed 99.95% availability!
What’s wrong with fixed availability?

Cloud customers:
- Various downtime demands
- Different WTP*

Cloud infrastructures:
- Heterogeneous HW & SW reliability

* WTP= Willingness to Pay
The Availability Knob (AK)

Let’s have clients ask for their desired availability and be charged correspondingly.
What should change in cloud to support AK?

Cloud management
• Gathering failure data and build failure stats
• Avail-aware scheduling
How do SLAs look with AK?

1. Desired Avail. / Period (e.g. 99.8% / 7 days)
2. Availability price scale e.g. (99.95%, 1.00), (99.9%, 0.95)
3. Variable service credit (penalty)
The AK Scheduler

1. Check for available resources
2. Find the cheapest resource considering possible penalties using:

- Expected PM time-to-next-failure
- VM size and expected DT** length in case of failure
- User’s experienced vs. requested DT

* PM= Physical Machine
** DT= Downtime
AK-Specific Scheduler Features

Extra Knowledge on user availability demand enables new scheduling features:

Benign VM\(^*\) Migration (BVM)

Deliberate Downtimes (DDT)

\(^*\) VM= Virtual Machine
Benign VM Migration (BVM)

- VMs can be over-served
  - Low failure rate
  - Assignment to HR resources (resource shortfall)

Periodic migration of over-served VMs to cheaper resources

* DTF = Downtime Fulfillment

** SLO = Service Level Objective
Deliberate Downtimes (DDT)

- Providers can deliberately fail VMs near the end of period.

Motivations:
- Building market incentives
- Lowering energy consumption
- Bidding redeemed resources
- etc.

![Diagram showing relationship between Requested Avail. and Delivered Avail. with a safety margin.](image)
Economics of AK

How to set prices to ensure mutual benefit?

How does AK make money?
Incentive Compatibility

Providers can:
- neglect meeting SLOs*
- run buggy VMs

Clients may:
- cause deliberate DTs**.

Pricing for incentive compatibility

Using game theory to ensure:
- Providers maximize profit margin by not violating SLOs
- Clients pay less by asking their true demands

* SLO= Service Level Objective  ** DT = Downtime
How does AK make money?

1. Adapting service to real demand:
   Higher market efficiency through supply chain flexibility

2. More efficient resource utilization:
   Lowering OpEx, Extra Bidding/Sprinting

3. Variable profit margins:
   Compensates risks & supply/demand disparity

~10% Cost Reduction
~20% Profit Increase
AK Deployment

• No hardware change required
  • Low technology adoption cost
• Existing fixed availability a subset of AK
  • Can be offered as an optional feature
  • Easy shift to the new model
How to evaluate AK?

Infrequency of Failures

Accelerated testing

Simulations

Data center scale

1. Stochastic simulations in MATLAB
2. Prototype implementation with OpenStack

AKSim: Stochastic Cloud Simulator

Scalability

Various Machine Types (cost/resilience trade-off)

Diverse Applications

Multiple VMs

Resolution/Accuracy trade-off
OpenStack AK Prototype
Availability-aware Scheduler

1000 machines, 12000 users, Normal demand dist., 6 month BVM every 1hr for top 10% of over-served clients
Benign VM Migration (BVM)

Benefits of BVM depend on **machine type blend** and **data-center utilization**.

- ~7% Cost Reduction
- Increased Miss Rate 0.19% → 0.34%

1000 machines, 12000 users, Uniform demand dist. [3 nines,5 nines], 30 days
BVM every 1hr for top 10% of over-served clients
Deliberate Downtimes (DDT)

Benefits of DDT depend on demand distribution.

1000 machines, 12000 users, Normal demand dist. [3 nines,5 nines], 6 month BVM every 1hr for top 10% of over-served clients
Improved Service Satisfaction

\[ S_{sat} = D_{sat} \times P_{sat} \]

\[ WTP = k\alpha^\beta(A-A_c) \]

* WTP= Willingness to Pay
Things to Remember

• Supply chain flexibility -> market efficiency
• Knowing user demand can enable new techniques
• Game theory to ensure mutual economic incentive
• Leveraging reliability/cost trade-offs
The Availability Knob

Mohammad Shahrad
mshahrad@princeton.edu
Back-up Slides
What if client’s demand changed?

Upper bound of SC given arbitrary P

<table>
<thead>
<tr>
<th></th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>No change; Fixed $A_1$</td>
<td>$P_{A_1}$</td>
</tr>
<tr>
<td>deliberate failures by user to earn cash back</td>
<td>$P_{A_1} - \text{SC}(\alpha A_1 + (1-\alpha)A_2)$</td>
</tr>
<tr>
<td>Change to $A_2$</td>
<td>$\alpha P_{A_1} + (1-\alpha)P_{A_2}$</td>
</tr>
</tbody>
</table>

Plan update condition:

$\forall \alpha \in [0, 1], \forall A_2 \text{ s.t. } A_2 < A_1$

$\text{SC}(\alpha A_1 + (1-\alpha)A_2) < (1-\alpha)(P_{A_1} - P_{A_2})$
Nash Equilibrium

<table>
<thead>
<tr>
<th></th>
<th>Defective Client</th>
<th>Healthy Client</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Request higher availability and cause DT</td>
<td>Request the desired availability</td>
</tr>
<tr>
<td>Responsive Provider</td>
<td>$(M_{A_2} - SC_{A_1}(A_2), P_{A_1} - SC_{A_1}(A_2))$</td>
<td>$(M_{A_2}, P_{A_2})$</td>
</tr>
<tr>
<td>Lazy Provider introducing excess DT</td>
<td>$(M_{A_3} - SC_{A_1}(A_3), P_{A_1} - SC_{A_1}(A_3))$</td>
<td>$(M_{A_3} - SC_{A_2}(A_3), P_{A_2} - SC_{A_2}(A_3))$</td>
</tr>
</tbody>
</table>

Nash equilibrium:

\[
\forall A_1, A_2, A_3 \text{ s.t. } A_3 \leq A_2 \leq A_1:
\]

1. \( M_{A_3} - M_{A_2} \leq SC_{A_1}(A_3) - SC_{A_1}(A_2) \)
2. \( SC_{A_1}(A_3) - SC_{A_2}(A_3) \leq P_{A_1} - P_{A_2} \)
Catastrophic Failure & AK

• When the whole cloud service is down.

\[
DT_{\text{crit}} = T \times (1 - A_c)
\]

\[
DT_{\text{crit}} = 30 \times 24 \times (1 - 0.995) = 3.6\text{hr}
\]
Why OpenStack

- VM migration (unlike Eucalyptus)
- Diverse hypervisor support (KVM)
- AWS Compatibility
- Big community (good support)
- Real world adoption in public/private/hybrid clouds
## Some More Results

<table>
<thead>
<tr>
<th>Range</th>
<th>Dist.</th>
<th><strong>No DDT</strong></th>
<th></th>
<th></th>
<th><strong>DDT</strong></th>
<th></th>
<th></th>
<th><strong>ΔC(%)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Miss(%)</td>
<td>DTF</td>
<td>Miss(%)</td>
<td>DTF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Two 9’s, Four 9’s]</td>
<td>N</td>
<td>0</td>
<td>0.999</td>
<td>0</td>
<td>0.300</td>
<td>0.116</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>U</td>
<td>0.004</td>
<td>0.997</td>
<td>0.001</td>
<td>0.299</td>
<td>0.263</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.004</td>
<td>0.997</td>
<td>0.007</td>
<td>0.300</td>
<td>0.125</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Three 9’s, Five 9’s]</td>
<td>N</td>
<td>0.015</td>
<td>0.987</td>
<td>0.013</td>
<td>0.301</td>
<td>-0.021</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>U</td>
<td>0.342</td>
<td>0.972</td>
<td>0.424</td>
<td>0.303</td>
<td>0.102</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.458</td>
<td>0.966</td>
<td>0.529</td>
<td>0.305</td>
<td>0.294</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Four 9’s, Six 9’s]</td>
<td>N</td>
<td>2.572</td>
<td>0.878</td>
<td>2.783</td>
<td>0.344</td>
<td>0.220</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>U</td>
<td>4.222</td>
<td>0.835</td>
<td>4.456</td>
<td>0.343</td>
<td>0.074</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>5.178</td>
<td>0.806</td>
<td>5.382</td>
<td>0.332</td>
<td>0.136</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Service Credit Reshaping

\[ DTF = \frac{DT_{SLO} - DT_{Delivered}}{DT_{SLO}} = \frac{(a - A)}{(1 - A)} \]

\[ DTF_{A1} = DTF_{A2} \Rightarrow \frac{(a_1 - A_1)}{1 - A_1} = \frac{(a_2 - A_2)}{1 - A_2} \]

\[ SC_{A2}(a) = SC_{A1}(ka + (1 - k)), \quad k = \frac{(1 - A_1)}{(1 - A_2)} \]
Availability Monitoring Tools

• There are some performance monitoring tools AK can use to gather avail data:
  • Nagios (used in AWS)
  • Zabbix
  • Ganglia