Robust Software in a Failing World
The Netflix solution

Kjell Jørgen Hole
Simula@UiB

HID, Bergen, Norway 11 Feb. 2014
Overview

- Why web-scale solutions should be cloud based
- How to avoid catastrophic cascading failures
Why cloud?
Everything is broken

<table>
<thead>
<tr>
<th>Hardware will fail</th>
<th>Everything is broken</th>
</tr>
</thead>
<tbody>
<tr>
<td>slow change</td>
<td>rapid change</td>
</tr>
<tr>
<td>large scale</td>
<td>large scale</td>
</tr>
<tr>
<td>Telcos</td>
<td>Web-scale</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Everything works</th>
<th>Software will fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>slow change</td>
<td>rapid change</td>
</tr>
<tr>
<td>small scale</td>
<td>small scale</td>
</tr>
<tr>
<td>Enterprise IT</td>
<td>Startups</td>
</tr>
</tbody>
</table>

Figs from Netflix blog
Monolithic solutions

- Insufficient *availability* due to **cascading failures**
- Many dependencies make it hard to upgrade software
Availability tradeoff

Availibility

99.999%
99.99%
99.9%
99%

Rate of change
Challenge: shift the cure
Problem: subsystem dependencies

Direction of dependency

Local uptime

sub-system 99.99%

sub-system 99.99%

sub-system 99.99%
Break dependencies

**Isolation**—failure in one subsystem should *never* result in cascading system failure
Why cloud?

- **Availability**: The cloud provides a cost-effective way to leverage the *redundancy* and *diversity* needed to break dependencies.
- **Scalability**: Server virtualization supports the needed scalability.
- **Performance**: The use of multiple cloud regions facilitate low-latency service all over the world.
How to isolate failures
(according to Netflix)

1. Introduce redundancy and diversity to **isolate**
   impact of local failures, and

2. Induce failures to **learn** how to make a system
   increasingly robust to cascading failures
Replacement

Redundant services with timeout and failover
The tool **Chaos Monkey** disables random production instances to make sure the Netflix solution survives this common type of failure without any customer impact.
Latency monkey

**Latency Monkey** tests what happens when the delay becomes too long.
Default fallback response

Timeout with fallback default response used when *all* instances are affected
Canary push

• Since a web-scale solution supports users all over the world, there is no good time to take down the system and upgrade its software

• An alternative is to introduce new code by keeping both old and new code running and switch user requests to new code
Simple canary push

dependent system

Timeout

dependence

Canary instance new code
Red/black deployment

dependent system

Fallback to old code

dependence v2.3

dependence v2.2
Standby blue system

- Software error in both red and black deployment
- Blue system is an indecently authored system delivering a minimal solution
- Used when all resent versions of the code fail
Zone isolation

Chaos Gorilla generates zone failures

Local balancer

dependent system

dependence

Zone A

Zone B

dependent system

dependence
Region isolation

Chaos Kong is used to test region failures
Information from techblog.netflix.com