SUPERIOR MISSION SYSTEMS
Faster, Resilient, Secure & More Affordable
Dave Manley, Chief Mission Systems Architect
February 27, 2018
Emerging Requirements Are More Demanding

- Faster change velocities (sustainable)
- Better resiliency with higher availability
- Cybersecurity must outpace threats
- **All** within limited or reduced budgets

The challenge is monotonically increasing
The Critical Decision

The architecture decision determines a mission system’s obtainable performance.

The architecture decision determines a mission system’s obtainable performance.
Achieving Higher Availability
Downtime Allowed

<table>
<thead>
<tr>
<th>Availability %</th>
<th>Downtime per day</th>
<th>Downtime per month</th>
<th>Downtime per year</th>
<th>Downtime per 3 years</th>
<th>Downtime per 5 years</th>
<th>Downtime per 10 years</th>
<th>Downtime per 25 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>90% (&quot;one nine&quot;)</td>
<td>2.4 hours</td>
<td>72 hours</td>
<td>36.5 days</td>
<td>109.5 days</td>
<td>182.5 days</td>
<td>1 year</td>
<td>2.5 years</td>
</tr>
<tr>
<td>99% (&quot;two nines&quot;)</td>
<td>14.4 minutes</td>
<td>7.20 hours</td>
<td>3.65 days</td>
<td>10.95 days</td>
<td>18.25 days</td>
<td>36.5 days</td>
<td>91.25 days</td>
</tr>
<tr>
<td>99.9% (&quot;three nines&quot;)</td>
<td>1.44 minutes</td>
<td>43.8 minutes</td>
<td>8.76 hours</td>
<td>1.095 days</td>
<td>1.825 days</td>
<td>3.65 days</td>
<td>9.125 days</td>
</tr>
<tr>
<td>99.99% (&quot;four nines&quot;)</td>
<td>8.64 seconds</td>
<td>4.38 minutes</td>
<td>52.56 minutes</td>
<td>2.628 hours</td>
<td>4.38 hours</td>
<td>8.76 hours</td>
<td>21.9 hours</td>
</tr>
<tr>
<td>99.999% (&quot;five nines&quot;)</td>
<td>864.3 milliseconds</td>
<td>25.9 seconds</td>
<td>5.26 minutes</td>
<td>15.78 minutes</td>
<td>26.3 minutes</td>
<td>52.6 minutes</td>
<td>2.19 hours</td>
</tr>
<tr>
<td>99.9999% (&quot;six nines&quot;)</td>
<td>86.4 milliseconds</td>
<td>2.59 seconds</td>
<td>31.5 seconds</td>
<td>1.575 minutes</td>
<td>2.625 minutes</td>
<td>5.25 minutes</td>
<td>13.125 minutes</td>
</tr>
<tr>
<td>99.99999% (&quot;seven nines&quot;)</td>
<td>8.64 milliseconds</td>
<td>262.97 milliseconds</td>
<td>3.15 seconds</td>
<td>9.45 seconds</td>
<td>15.75 seconds</td>
<td>31.5 seconds</td>
<td>1.3125 minutes</td>
</tr>
<tr>
<td>99.999999% (&quot;eight nines&quot;)</td>
<td>0.864 milliseconds</td>
<td>26.297 milliseconds</td>
<td>315.7 milliseconds</td>
<td>947.0 milliseconds</td>
<td>1.5785 seconds</td>
<td>3.157 seconds</td>
<td>7.8925 seconds</td>
</tr>
<tr>
<td>99.9999999% (&quot;nine nines&quot;)</td>
<td>0.0864 milliseconds</td>
<td>2.6297 milliseconds</td>
<td>31.5569 milliseconds</td>
<td>94.6707 milliseconds</td>
<td>157.7845 milliseconds</td>
<td>315.569 milliseconds</td>
<td>788.9225 milliseconds</td>
</tr>
</tbody>
</table>

Always constrained by available funding, technology, and technique
The Resiliency Triad

**Fragile** things don’t like volatility
- They easily **break** under stress
- Over time, they fail, erode, deteriorate
- e.g., wine glasses, pottery, eggs

**Robust** things appear immune to volatility
- They **resist** stress
- But when they fail, they do so spectacularly
- e.g., castles, Lehman Brothers, Maginot Line

**Anti-fragile** things enjoy volatility
- They **benefit** from stress
- Over time, they evolve, improve, get better
- e.g., vaccines, athletes, silicon valley
**The Microservices Architecture “Secret Sauce”**

**Modularity**

Decompose application into modular set of services with **Bounded Contexts**

- Eliminate dependencies
  - Enforce implementation guidelines (e.g., event sourcing, aggregates, responsibility segregation, etc.)
  - Foster **independent** development, deployment, scaling, and technology stack choices
  - Pursue **simplicity** to ease learning, debugging, and enhancement

**Modularity facilitates disaggregation, redundancy, and geographic separation**
Additional Keys to Achieving Tougher Requirements
Disaggregation, Redundancy, Separation, Isolation, Awareness, Automation, and Testing

- Establish resiliency via disaggregation, redundancy, separation and isolation of services
- Realize continual awareness of entire stack
  - Characterize historic normal conditions, analyze current conditions, and merge results constantly
- Create robust continuous integration / continuous delivery (CI-CD) automation pipelines
- Establish and continuously improve self-healing and robust Defensive Cyberspace Operations (DCO) near real-time remediation capabilities
- Implement chaos testing and continuous security assurance

We can not keep putting all eggs in the same basket

Continuously prove your implementation is anti-fragile and secure
Cyber Chaos Testing

- Assume everything will fail
- Force failure to validate resiliency
- Don’t wait for random failure, remove its uncertainty by forcing it periodically
  - Seek confidence that single failure points don’t exist
  - Seek confidence that cascading failure vulnerabilities do not exist
  - Seek confidence that self-healing automation works
  - Seek confidence that DCO capabilities are robust
- Getting stronger through failure is the basis of anti-fragility

“I’ve failed over and over and over again in my life. And that is why I succeed.”

Michael Jordan

Avoiding failure at all costs makes you brittle, vulnerable, slow, and expensive
Rapid Response Essential

Netflix Chief Architect:
“The Chaos Monkey’s job is to **randomly kill instances and services** within our architecture. If we aren’t constantly testing our ability to succeed despite failure, then it isn’t likely to work when it matters most – in the event of an unexpected outage.”

Unlike ships, with software you can automatically replace a “flooded bulkhead” in **milliseconds** and the rest of the system is none-the-wiser.

If one component fails, but does not cascade, the problem can be isolated and healed/remediated while the rest of the system keeps working.
Economic Benefit Sources

- **Operational efficiency** due to reductions in resources allocated to development operations
- Increased **developer productivity** resulting from automation and elimination of unexpected dependencies
- Decreased downtime due to **higher quality software that is secure by default**, which allows for better mission support and higher mission satisfaction
- Ability to support **mission growth** through shortened release cycles and faster response times to market dynamics

We are what we repeatedly do. Excellence then is not an act but a habit.”

**Aristotle**
Summary

- Enable faster change velocity
- Improve resiliency with higher availability
- Enable cybersecurity to outpace threats
- All within limited or reduced budgets

“It’s not whether you get knocked down. It’s whether you get up.”

Vince Lombardi