Paper: On Designing and Deploying Internet-Scale Services

Tuesday, March 25, 2008 at 5:58AM

Todd Hoff in Paper, deployment, management, operations

Greg Linden links to a heavily lesson ladened LISA 2007 paper titled *On Designing and Deploying Internet-Scale Services* by James Hamilton of the Windows Live Services Platform group. I know people crave nitty-gritty details, but this isn't a how to configure a web server article. It hitches you to a rocket and zooms you up to 50,000 feet so you can take a look at best web operations practices from a broad, yet practical perspective. The author and his team of contributors obviously have a lot of in the trenches experience. Many non-obvious topics are covered. And there's a lot to learn from.

The paper has too many details to cover here, but the big sections are:

- Recommendations
- Automatic Management and Provisioning
- Dependency Management
- Release Cycle and Testing
- Operations and Capacity Planning
- Graceful Degradation and Admission Control
- Customer Self-Provisioning and Self-Help
- Customer and Press Communication Plan

In the recommendations we see some of our old favorites:

- Expect failure and design for failure.
- Implement redundancy and fault recovery.
- Depend upon a commodity hardware slice.

- Keep things simple and robust.
- Automate everything.

Personally, I'm still trying to figure out how to make something simple.

Next are some good thoughts on how to design operations friendly software:

- Quick service health check. This is the services version of a build verification test.
- Develop in the full environment.
- Zero trust of underlying components.
- Do not build the same functionality in multiple components.
- One pod or cluster should not affect another pod or cluster.
- Allow (rare) emergency human intervention.
- Enforce admission control at all levels.
- Partition services.
- Understand the network design.
- Analyze throughput and latency.
- Treat operations utilities as part of the service.
- Understand access patterns.
- Version everything.
- Keep the unit/functional tests from the last release.
- Avoid single points of failure.
- Support single-version software. Have all your customers run the same version.
- Implement multi-tenancy. Apparently a lot of software requires cloning hardware installations to support multiple customers. Don't do that. Have your software work for multiple customers all on the same hardware.

And the paper continues along the same lines in each section. Good detailed advice on lots of different topics.

You'll undoubtedly agree with some of the advice and disagree with some.

Greg wants faster release cycles, thinks having server affinity for some things is OK, and thinks the advice on allowing humans to throttle load won't work in a crisis. Perfectly valid points, but what's fun is to consider them. Some companies, for example, have a dead-man's switch that must be thrown before one master can failover to another in a multi-datacenter situation. Is that wrong or right? Only the shadow knows.

The advice to "document all conceivable component failures and modes and combinations" sounds good but is truly difficult to do in practice. I went through this process once on a telco project and it took months just to cover all the failure scenarios on a few cards. But the spirit is right I think.

My favorite part of the whole paper is:
We have long believed that 80% of operations issues originate in design and development, so this section
on overall service design is the largest and most important. When systems fail, there is a natural tendency
to look first to operations since that is where the problem actually took place. Most operations issues,
however, either have their genesis in design and development are best solved there.

Understand this and I think much of the rest follows naturally.
