How Liferay Is Improving Quality Using Hundreds of Jenkins Servers

James Min
Sr. Consultant, Liferay, Inc.
liferay.com

Liferay Background

- Open Source Portal & Collaboration
- Java Platform
- 250 employees worldwide
- In its 11th year of development, Liferay Portal is today's most popular and widely downloaded open source portal platform
  - Over 4 Million downloads and 50,000 downloads per month
  - 46,034 registered users on Liferay.com
  - 15,758 forum participants
  - 161,641 forum posts
  - Over 50 active volunteer contributors

Challenges at Liferay

Deployment Compatibility

Operating Systems

- Linux (CentOS, RHES, SUSE, Ubuntu, and others)
- Unix (AIX, HP-UX, Mac OS X, Solaris, and others)
- Windows

Servlet Containers

- Jetty
- Resin
- Tomcat

Application Servers

- Geronimo
- GlassFish
- JBoss
- JOnAS
- OracleAS
- SUN JSAS
- WebLogic
- WebSphere

Java Runtimes

- Java Standard & Enterprise Edition (SE/EE) 5
- Java Standard & Enterprise Edition (SE/EE) 6

Databases

- IBM DB2
- MySQL
- Oracle
- PostgresSQL
- SQL Server
- Sybase

Public & Private Clouds

Liferay Portal is deployable to the cloud and virtualized environments, including EC2, Elastic Beanstalk, and VMWare.

Liferay Portal Needs CI For:

- Liferay Portal Core
- Liferay Portal Plugins
- Deployment platforms and environments (provisioning)

Before Jenkins

- Manual Setup of Environments
- Manual Testing
- As Liferay supports more platforms, the effort to keep up increases exponentially!

Liferay's Early Jenkins Setup

- 100 unclustered servers
- Each physical machine set-up to run up to 4 VMs with 13 VMs (vmdk files) loaded onto an extra hard disk
- Hardware Specs:
  - INTEL CPU C2Q Q9550s 2.83ghz
  - INTEL MB G41MJ MITX LGA775
  - Intel® X25-M Mainstream SATA Solid State Drive 80GB 2.5" 9.5mm SSD

Jenkins Manages Tasks/Tests

- Deploy every permutation of application server/database/operating system
- Run smoke tests
- If smoke test fails, Jenkins automatically bypasses executing any of the subsequent downstream jobs related to that environment.
- If smoke test passes, run tests (Selenium) downstream

Liferay Smoke Tests

- builds the portal war
- zips it up
- deploys it into an app server
- starts the app server
- logs into the portal
- asserts that it was able to log in
- logs out
- asserts log out
- shuts down the app server

Liferay Smoke Tests

- If all the smoke test pass (e.g. – something wrong):
- Test every deployment w/ all tests (10,000) it takes about 7-10 hours after the initial build

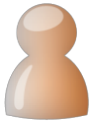
Growing Pains

- Liferay Core Development:

"Scotty, We need more power!"

- Liferay QA:

"I'm givin' her all she's got, Captain!



Growing Pains

- As Liferay supports more platforms and features, testing takes longer
- 50-100% utilization on each machine
- Jenkins, unclustered, has no master to manage, and creates inefficiency (i.e. – idle servers after testing is done)
- As Liferay's core code base and the number of Liferay plugins grow, there is more to test.

Liferay's Evolving Setup

- 100 isolated Jenkins nodes =

Bottlenecks & Inefficency

- When one node is finished it does not necessarily start another task
- Other tests do not start until they are manually started or the last one is finished

Liferay's Evolving Setup

- Jenkins Cluster with master and 250 slaves
- Continuous deploying of new VMs and building new environments while other nodes finish tasks/tests.

Liferay's Current Setup

- 250 physical servers at colo facility
- Jenkins 1.4 Cluster
- Selenium Grid
- 10,000 tests per **version**

Liferay's Future Plans

- 500 Servers by end of 2011
- 1000 Servers by mid-2012
- Expansion room for 2000 total servers
- Move from a colo to in-house managed facility

Liferay's Future Plans & Challenges

- How to deploy Jenkins en masse?
- How to deploy the configuration and plugins en masse?
- Currently have a homegrown framework where we utilize batch scripting and a NAS that's mapped to all of the servers, and remotely shutdown/restart/format/reset the servers and/or Jenkins instances

Liferay's Future Plans & Challenges

- Jenkins calls test scripts in Liferay that handle launching a Selenium server instance alongside a Liferay instance for testing
- **Jenkins Selenium Grid Cluster!!!**
- Remote configure Jenkins servers en masse as well using ant/batch scripting so we can bring down a server and repurpose it pretty quickly through utilizing the scripting framework
- Unexplored territory for Liferay
- Looking forward to Ryan Campbell & Kohsuke Kawaguchi's talk later ☺

Why Our Own Cloud vs. Cloud Service?

- Should we have used a cloud service setup (PaaS)?
- For Liferay, we found if you know your utilization, building our own was more cost-effective
- We calculated it to cost roughly 10-12x more to do it on the cloud than to house it internally even when you factor in all the other costs such as maintenance, electricity, bandwidth, etc.
- Cloud service is very useful when you don't know the demand or if is unpredictable. If you don't know whether you need 10 or 100 or 1000 servers, and you don't know how often they are actually being used, then a cloud service is cheaper because you have lower up front costs.
- For Liferay, the cloud service is definitely more expensive per unit cost (i.e. – cloud vendors have to make money some how). Since we know our usage (close to 90% because they're build servers), it's a LOT cheaper for us to do it in house.

Our Own Cloud vs. EC2?

- When we calculated for just 100 servers, based on our calculations, since we run 4 vms on each machine and we have utilization of at least 50%, that would be 300k in fees to Amazon a year. If we ran at 100%, we'd pay 600k a year to Amazon
- So for 100k for hardware (one time expense), plus 50k in colocation fees a year, plus X $ for salary of maintaining the servers per year, it's still cheaper than Amazon. **The reason it's cheap for us is because we actually USE it a lot and we have decent expertise.**

Our Own Cloud vs. EC2?

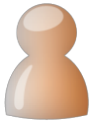
- Most people use it a fraction % of the day, because their tests finish relatively quickly. Our tests are **hours** at a time.
- For Liferay, the cloud service is definitely more expensive **per unit**. Since we know our usage (close to 90% because they're build servers), it's a huge savings for us to do it in house.

Our Own Cloud vs. EC2?

- With $1 million worth of servers, (one time cost), for the same power, we'd be paying Amazon $6 million in fees a year if we used it 100%. And the next year, we don't have to pay for the servers again! Just what's broken.
- Today, our internet fees at the colo (for 250 servers) is 96k a year. Before, for 100 servers, it was 50k a year. So to lower that, we're even going to move our colo in-house (i.e. we're building our own colo now)

PaaS Cloud Is Still A Great Option

- UNLESS, they happen to be in a situation similar to Liferay.
  - Have a huge need and a predictable utilization requirement
  - Have in-house expertise
- Most flexible option
- Weigh the one-time capitalization cost vs. yearly operating costs



Thank You To Our Sponsors

Platinum Sponsor	CloudBees
Gold Sponsor	LIFERAY
Silver Sponsor	redhat
Bronze Sponsors	New Relic CHARIOT SOLUTIONS Practical, Smart Software Development sauce LABS eXo

Coming Soon: **The CloudBees Newsletter for Jenkins**

✓ *Please complete the Jenkins survey to help us better serve the community (bonus: a chance to win an Apple TV!)*

Questions?

- James Min, Sr. Consultant / Sales Engineer
- james.min@liferay.com
- http://www.liferay.com/web/james.min/blog