Case study of Kohsuke Kawaguchi

Machines

The master Hudson lives in SunBlade 2500with 2GB memory. It's behind Apache, so that Hudson gets its own JVM. This allows me to restart Hudson without bringing down other services in my system.

We currently have 14 or so slaves hooked up to this server. There are a variety of architectures — Solaris sparc, solaris x86, and some Linux. Some of them are 3 time zones away, but most of them are single/dual CPU workstations that developers own in their offices. Despite the number of slaves, it's not too uncommon to see all the slaves utilized, especially when we are getting close to a release.

All the machines belong to the corporate intranet, and so they share the common authentication (via NIS). A special user 'hudson' is created on all the machines so that files can be accessed between the master and the slave.

Monitoring and maintaining Hudson

Our team increasingly relies more on Hudson, so it's getting more important to keep it up and running. Server is launched with JPDA support enabled (so I can go in with the debugger any time if something goes wrong.) It's hooked up to visualgc, which provides a basic telemetry of the system (and helps me debug the memory related issues.) I'm considering to also launch it with a remote profiler enabled.

We run `rsync` to make sure that all the slaves have the right set of build tools.

What we run

This hudson primarily supports the JAXB, JAX-WS, and WSIT effort. Rougly speaking, each component has the main build (which builds against a fixed set of dependencies), the bleeding-edge build (which builds against the tips of the dependencies to detect any incompatibility), a whole bunch of test jobs, and a performance job (via japex.)

The main build is hooked to the CVS/Subversion change notification e-mail, so every time someone makes a change, it will be built. Once that is complete, it triggers the test jobs and the bleeding edge builds of its downstream jobs.

Some of our tests use a different test harness, so we developed a few plugins to display them in the same way Hudson renders JUnit results. Performance job runs on a dedicated machine in isolated fashion so that we can compare the results from one run to another, to detect performance regressions quickly.

Dependency management

All the components carry their own dependency jars inside their SCMs, so that they can be built independently anywhere. There's usually Ant targets that picks up the new version of dependency jars from Hudson, so when two teams agree that a build is good, we run this goal manually and commit the new file to SCM.

We rely on fingerprint feature extensively, so that we know what builds are tested and integrated. Because of this, when we integrate WSIT to Glassfish, we know exactly which builds of SAAJ, JAXB, FastInfoset, JAX-WS are in it. We can similarly associate any regression in any of our tests to the exact set of changes in the upstream projects. This makes it easy to identify who's to blame.