Application-Level Regression Testing Framework using Jenkins

Timothy A. Bouvet, NCSA, University of Illinois
Reuben D. Budiardja, ORNL, Oak Ridge, Tennessee
Galen W. Arnold, NCSA, University of Illinois
Gregory H. Bauer, NCSA, University of Illinois
Talk Outline

• Introduction and Motivation
• Implementation & Deployment
  • Basic Configuration
  • Connecting to multiple HPC systems
  • Security consideration, authentication, authorization
• Application Test Structure
• Actual Use Cases
Introduction & Motivation

- The complexities of monitoring an HPC system:
  - Layers and versions of software stack
  - System-level configurations
  - I/O components (memory, filesystem)
  - Diversity in user need and usage
  - Variable network traffic
  - Performance regression may occur over time
    - requires historical data to easily detect
  - **Application-level regression testing**
Goals

- Monitor usability and performance
- User-level experience
  - no privilege access, minimum ‘specialized’ tests
- Early detection of regressions
  - Tests are run during production period
- Automated and consistent test solution
- Store historical data
- **Solution: Jenkins Automation Server**
What is Jenkins?

- An “open source automation server”
- Automate user-level tasks
  - Execute arbitrary commands
  - Build software & submit jobs
  - Archive test results
  - Notify (email, text, …) on error
- “Never sleeps or takes vacation”
- Chosen for
  - versatility
  - multitudes of plugins (plot, ssh, … )
  - large community support
Implementation & Deployment
Basic Configuration

Jenkins (master) on CentOS VM (GUI, archive, ...)

HPC 1

HPC 2

HPC 3

application tests / scripts run on HPC systems

Jenkins "node"

slave.jar
daemon

SSH Plugin
Accessing (Remote) HPC Test Systems

Jenkins “Node”
- Requires running “slave.jar” daemon
- Daemon connects back to master on specific port
- Remote execution appears “local”
  - no need to copy files / data
  - can access lustre, /home as if they are local resources

With SSH Plugin
- Need to set up passwordless login with SSH keys
- Closer resemblance to user experience
- Have to manually manage data movement (file copying, getting results back) to

NICS
NCSA
Security Considerations

- Jenkins allows execution of arbitrary command
  - use Apache HTTPD server in front of Jenkins’, pass requests via `mod_proxy`
  - allow authenticated access only from institutional IPs via firewall rules
  - use RSA OTP for authentication via `pwauth` + custom session management (details in paper)
- Use LDAP for authorization with Jenkins “matrix-based” security
## Jenkins Global Security

### Authorization

- **Anyone can do anything**
- **Legacy mode**
- **Logged-in users can do anything**
- **Matrix-based security**

<table>
<thead>
<tr>
<th>User/group</th>
<th>Overall</th>
<th>Credentials</th>
<th>Agent</th>
<th>Job</th>
<th>Run</th>
<th>View</th>
<th>SCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub_UTK</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
</tr>
<tr>
<td>arnoldg</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
</tr>
<tr>
<td>bw_admin</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
</tr>
<tr>
<td>bw_seas</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
</tr>
<tr>
<td>rbudiard</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
</tr>
<tr>
<td>Anonymous</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
</tr>
</tbody>
</table>
Application Test Structure
An Application Test: Jenkins Project

- **Name and description** - what is tested, frequency, resources
- **Limit number of “builds” kept** - conserve server storage
- **Source code management** - subversion or git
- **Build trigger** - manual or automated schedule
- **Build commands** – shell commands to execute test
- **Post-build Actions** – generate plots, notifications on failure

*Project can be easily copied from an existing one*
Admin Gui Home Page

Add JYC tests that are considered done and production-ready here with "Edit View".

Select Project

Schedule a Build

Pass

Anonymous Only View

Disabled

Fail

Application-Level Regression Testing Framework using Jenkins
<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>module load autocorr</td>
</tr>
<tr>
<td>module load automake</td>
</tr>
<tr>
<td>~bsh/myenv.sh</td>
</tr>
<tr>
<td>rm -rf jenkins-ior</td>
</tr>
<tr>
<td>mkdir jenkins-ior</td>
</tr>
<tr>
<td>cd jenkins-ior</td>
</tr>
<tr>
<td>git clone <a href="https://github.com/LLNL/ior">https://github.com/LLNL/ior</a></td>
</tr>
<tr>
<td>cd iOR</td>
</tr>
<tr>
<td>./bootstrap</td>
</tr>
<tr>
<td>./configure --host=x86_64 CC=cc</td>
</tr>
<tr>
<td>make -j 4</td>
</tr>
<tr>
<td>file src/ior</td>
</tr>
<tr>
<td>ls -l src/ior</td>
</tr>
</tbody>
</table>

For the test script:

```
cp src/ior ~amoldg/ior/src/ior
# make clean; make
cat -n ~amoldg/ior/compare_stripe.pbs # there is a tor script input file for ior in the same directory
MYJOBID=`qsub -v ~amoldg/ior/compare_stripe.pbs`
MYJOBID=`echo MYJOBID=$MYJOBID`
~bsh/wait_for.sh $MYJOBID

echo "--- JOB RAN ON ---"
cat ~/scratch/stripes/ior/compute-nodes
if [ ! -f ~/scratch/stripes/ior/compute-nodes ]
then
  # it's all good
else
echo "ior compute-nodes not found, pbs job likely hit walltime"
fi
```

Save  Apply
Project View

Project JobLaunch-BW

Job Launcher Test to Compare Wrapped (XALT) Aprun to Real Aprun on Tests: environment, xalt, aprun

Permalinks

- Last build (#7718), 23 min ago
  - Last stable build (#7718), 23 min ago
  - Last successful build (#7718), 23 min ago
  - Last failed build (#7677), 1 day 17 hr ago
  - Last unsuccessful build (#7677), 1 day 17 hr ago
  - Last completed build (#7718), 23 min ago
Example Console Log

| Jenkins | BlueWaters | JobLaunch-BW | #7718 |

Console Output

Started by timer
Building on master in workspace /var/lib/jenkins/workspace/Jc
Updating https://github.com/reubendb/SWTools-BWCITrunk/job/
At revision 22

No changes for https://github.com/reubendb/SWTools-BWCITrunk
[JobLaunch-BW] $ /bin/sh -xe /tmp/jenkins125936332362860387.4 + rsync -az --delete -e ssh /var/lib/jenkins/workspace/JobLat/bwjenkins/tests/

Access by OTP or Two Factor Certificate Authority only.
Use myproxy-logon -s tfca.ncsa.illinois.edu -p 7512 for gsissh or ssh -o PreferredAuthentications=keyboard-interactive

Blue Waters Admin Team
executing script:
SW_CONFIG="/scratch/system/bwjenkins/sw_config"
PATH="/sw/tools/bin:/sw/tools/bin:$PATH"
PYTHONPATH="/sw/tools/bin:/sw/tools/bin:$PYTHONPATH"
SW_ROOT="/scratch/system/bwjenkins/tests"
Use Cases
Project Failure Email Notification

See <http://bwjenkins.ncsa.illinois.edu/job/Lustre_Check_OST_BW/7218/display/redirect>
------------------------------------------
Started by timer
Building on master in workspace <http://bwjenkins.ncsa.illinois.edu/job/Lustre_Check_OST_BW/ws/>
executing script:
SW_WORKDIR="/scratch/system/bwjenkins/sw_workdir"
JOB_NAME="Lustre_Check_OST_BW"

snx11003-OST010c-osc-ffff885fc0cd3c00: check error: Resource temporarily unavailable

[SSH] exit-status: 255
Build step 'Execute shell script on remote host using ssh' marked build as failure
Recording plot data
Saving plot series data from: <http://bwjenkins.ncsa.illinois.edu/job/Lustre_Check_OST_BW/ws/bw_active>
Slow JobLaunch-BW Plot
Slow Lustre Scratch from a login node
Slow IOR-login Test Example
IOR-BlueWaters test failures

Two Failed Tests that went over 10 minute wallclock. These failures indicate an issue with the filesystem.
Plot Usage: Wiki Filesystem dashboard

Jenkins Monitoring, see also: http://bwjenkins.ncsa.illinois.edu/view/BlueWaters/

**scratch**

- **IOX peak performance**
  - Higher is better.
  - If low only once, build might have used busy OSTS.

- **mdtest file metadata performance**
  - Higher is better.
  - Trouble if mdtest and IOR rates are both low.

---

**scratch and $HOME is test**

---

**home**

- No IOR testing:
  - We don't do large block I/O on $HOME.

- **mdtest file metadata performance**
  - Higher is better.
Conclusion

• Regression testing framework, enabling:
  • Reproducibility of tests
  • Regression testing
  • Rapid reaction ← more tests are added as needed

• Has been deployed at
  • Blue Waters at NCSA
  • NICS
  • Compute and Data Environment for Science (CADES) at ORNL (in progress)

• Future work: integrate with other change managers