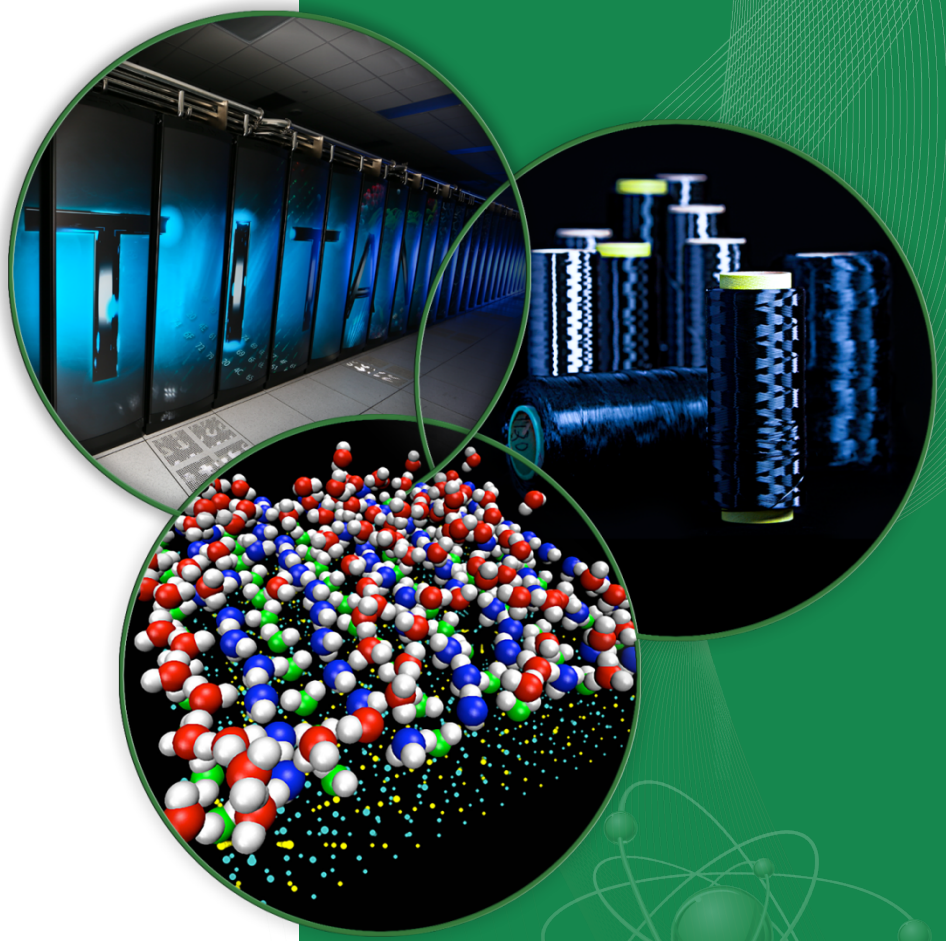
Use of Continuous Integration Tools for Application Performance Monitoring



Verónica G. Vergara Larrea
Wayne Joubert
Chris Fuson

CUG2015
April 26 - 30, 2015
Chicago, IL

ORNL is managed by UT-Battelle
for the US Department of Energy

OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

Overview

- **Part 1:** Background & Motivation
- **Part 2:** Application Performance Monitoring
- **Part 3:** Environment Monitoring

Part 1:

Background

Background & Motivation

- High performance computing systems are increasingly complex
  - Hardware (HW) and software (SW) stack
- How to measure impact of HW/SW changes?
  - Performance of applications, user environment stability
- **Proposed solution:** Application level monitoring system
  - *Often:* develop a tool from scratch
  - *Another option:* use/repurpose well-supported existing tools (no need to reinvent the wheel!)

Background & Motivation

- Many tools available from a system administration perspective
  - Nagios, Ganglia, Cray's Node Health Checker, etc
  - Help detect failures with systems and services
- Sustained system performance monitoring
  - NERSC's SSP metric, DoD's monitoring system
  - Detect performance degradation using benchmarks and applications
- In-house tools often developed
  - Significant center resources needed to maintain
- CI tools already provide most of the features needed to monitor performance and stability

CI vs. Monitoring workflow

CI workflow

- *Target:* software project
- Test, test, test
- After every commit:
  - Build software
  - Run set of tests
- Test on a regular schedule
- Alert when failures occur

Monitoring workflow

- *Many targets:* scientific application, benchmark, environment test
- After system software upgrade:
  - Build application
  - Submit job
- Run regularly to track performance over time
- Alert when failures occur
- Alert when performance degradation observed

Requirements

Tool

- Open source
- Freely available
- Recent release and/or bug fixes available
- Well-supported
- Flexible
- Portable
- Easy to deploy
- Minimal amount of customization needed to fit the workflow

Monitoring Workflow

- Graphical user interface
- Full job control
- Interactive dashboard
- Configurable analysis and plotting
- Reporting capabilities
- Customizable notifications
- Security features available
- Archiving capabilities
- Resilient to failures

CI Tools Evaluated

- Started with popular CI tools
- Grouped tools into:
  - commercial vs. freely available
  - closed vs. open source
  - hosted vs. deployable
- Top FOSS contenders: Jenkins, Buildbot, Continuum, Go
- Commercial/paid: TeamCity, Bamboo, Travis CI

CI Tools Evaluated (cont'd)

Tool	License	Latest Release	Dashboard	Features	Plugins	Plotting
Buildbot	GPL	Dec 2014	✓	Launch, report	10s	✗
Continuum	Apache 2.0	Jun 2014	✓	Full-control	10s	✗
CruiseControl	BSD-style	Sep 2010	✓	Launch, report	10s	✗
Go	Apache 2.0	Jan 2015	✓	Launch, report	10s	✓
Jenkins	CC & MIT	Jan 2015	✓	Full-control	> 1,000	✓
Travis CI	MIT	Oct 2014	✓	Report		
TeamCity	Proprietary	Jan 2015	✓	Full-control		✓
Bamboo	Proprietary	Nov 2014	✓	Full-control		✓

Why Jenkins?

- Large user community behind it
  - Over 100,000 users
- Provides full job control and management
  - Interactive dashboard
  - Flexible job scheduling
- Extensible with over a thousand plugins
- Plotting and reporting capabilities
- Customizable views
- Robust notification capabilities

Part 2:

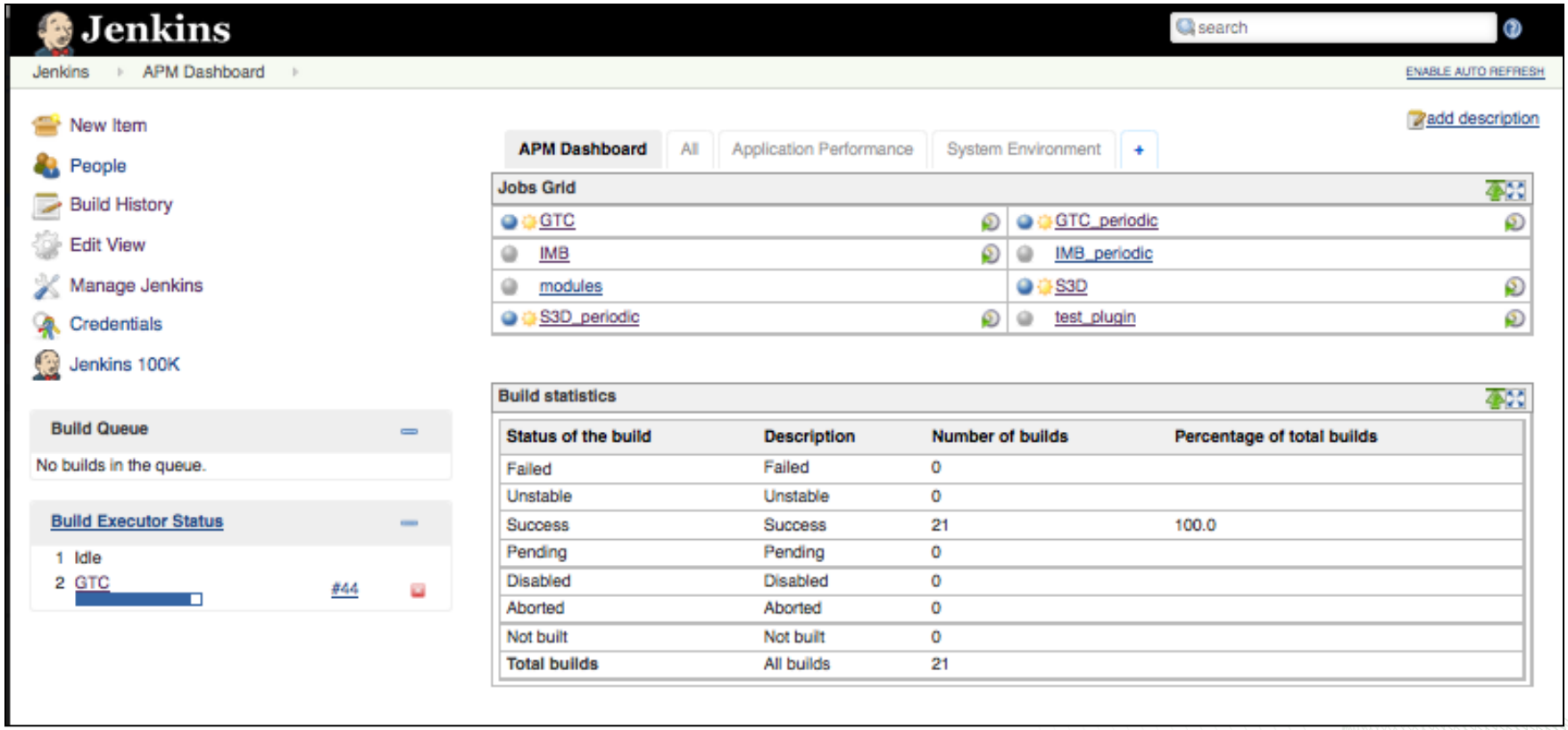
Application Performance Monitoring

Jenkins initial setup

- The Jenkins application is easy to download, launch with Java
- User can access Jenkins through a web browser interface
- Installed Plot plugin, Parametrized Trigger plugin, LDAP plugin, Dashboard View plugin, AnchorChain plugin
- Set up as two directories:
  - Jenkins install directory
  - Supporting scripts directory
- Scripts directory is stored in git repo
- Jenkins configure files for defining dashboards and build targets stored also in scripts directory for version control

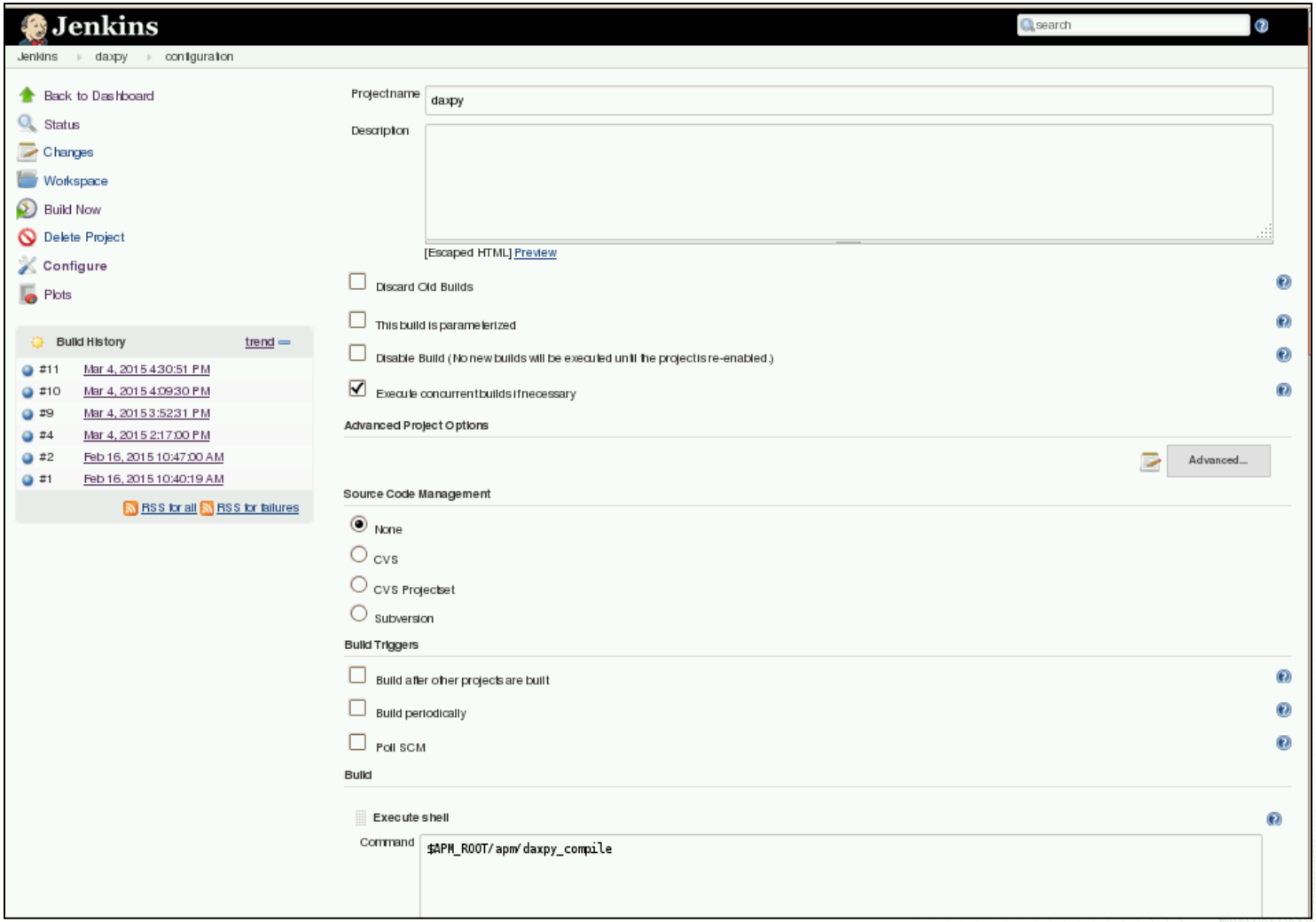
Jenkins dashboard

- Main dashboard presents list of executable build targets on right
- Left panel shows status of builds currently being executed
- “build” here can be a software build or any executable operation



Jenkins build targets

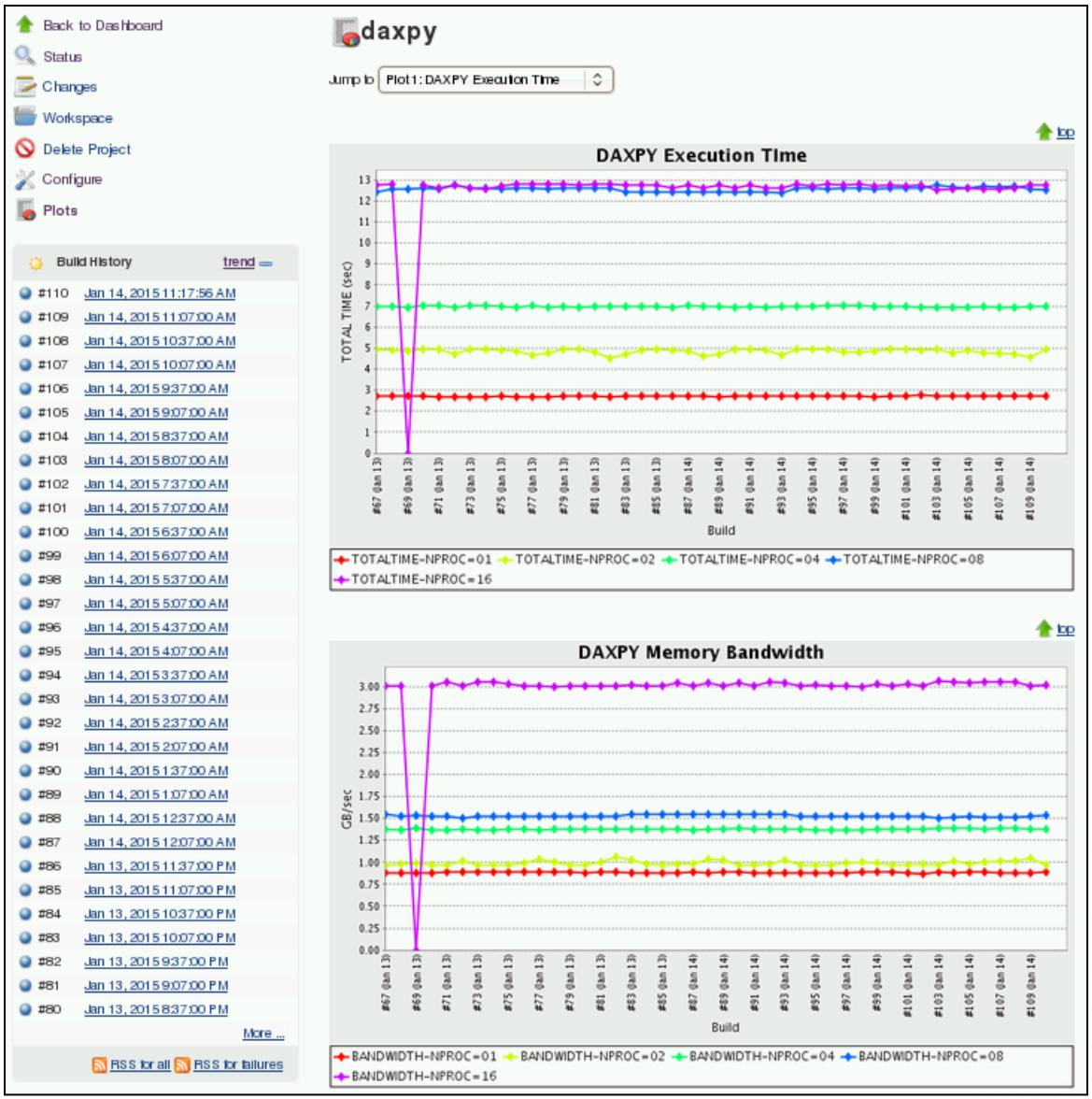
- Configure page used to configure the operations to perform a build



Example: DAXPY

- Test case: execute DAXPY kernel on 1-16 cores of a single node
- Shell script 1: create and compile executable
- Shell script 2: create and submit PBS scripts to run executable and collect timing results to files; spin loop to wait for PBS jobs to complete
- Settings on Jenkins DAXPY build configure page:
  - Execute two scripts
  - Run Plot plugin on results
  - Set up to run periodically with cron-like syntax
- Then can launch single DAXPY run or start periodic runs from the DAXPY dashboard
- For any build instance, user can view execution status on main page; can view job console output in real time if desired; can later access build artifacts for further inspection

DAXPY results

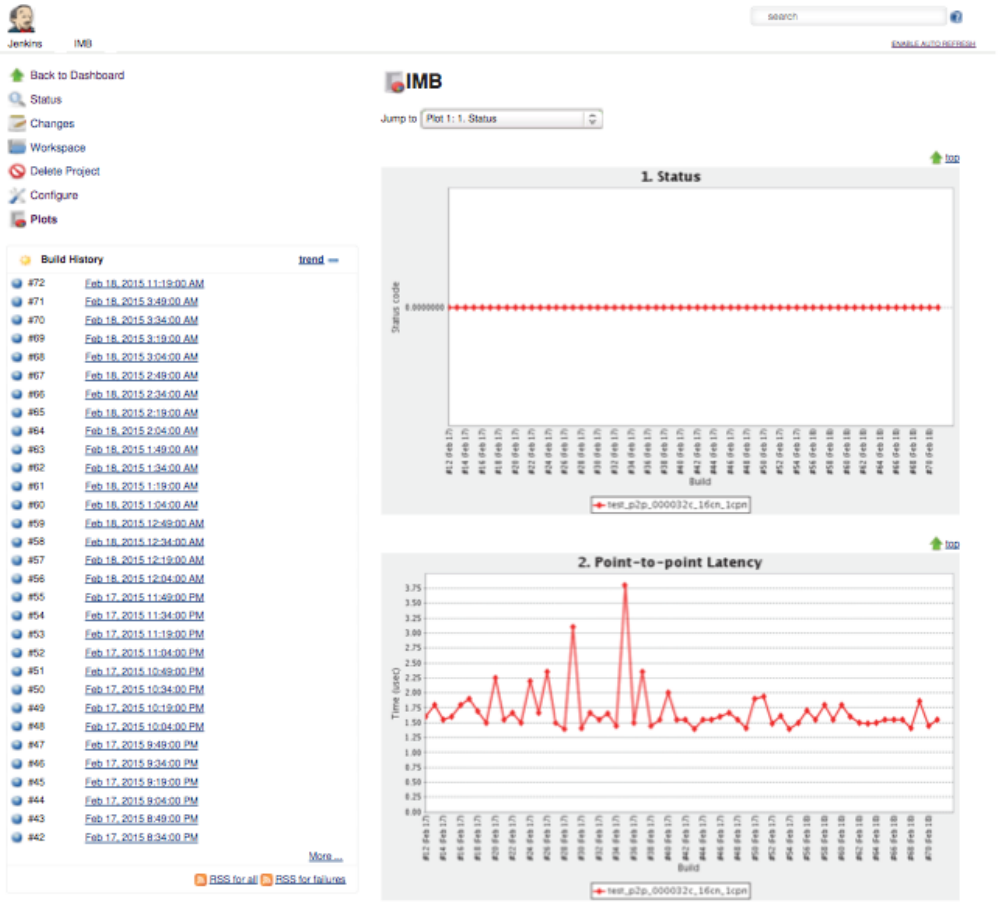


Example: application harness code

- OLCF has an application test harness, used for acceptance testing and IO system testing
- Has two parts:
  - Harness code proper – schedules and monitors builds and runs of applications
  - Harness applications – defines build and execution procedures for selected applications
- Each harness application instance has several scripts:
  - build_executable.x – compile the code
  - submit_executable.x – create PBS script and submit
  - check_executable.x – check run results for correctness
  - report_executable.x (NEW) – extract run metrics of interest
- Two “coupling” scripts were written to interface the already-existing large set of harness applications to Jenkins:
  - harness_build – build the code
  - harness_submit – submit the run, spin loop until completion, collect results

Harness example: IMB

- Simple run of Intel MPI Benchmarks suite to perform ping-pong test
- Used existing harness code as-is, only needed to add report_executable.x file to extract latency/bandwidth metrics from IMB output file



An alternative: Splunk

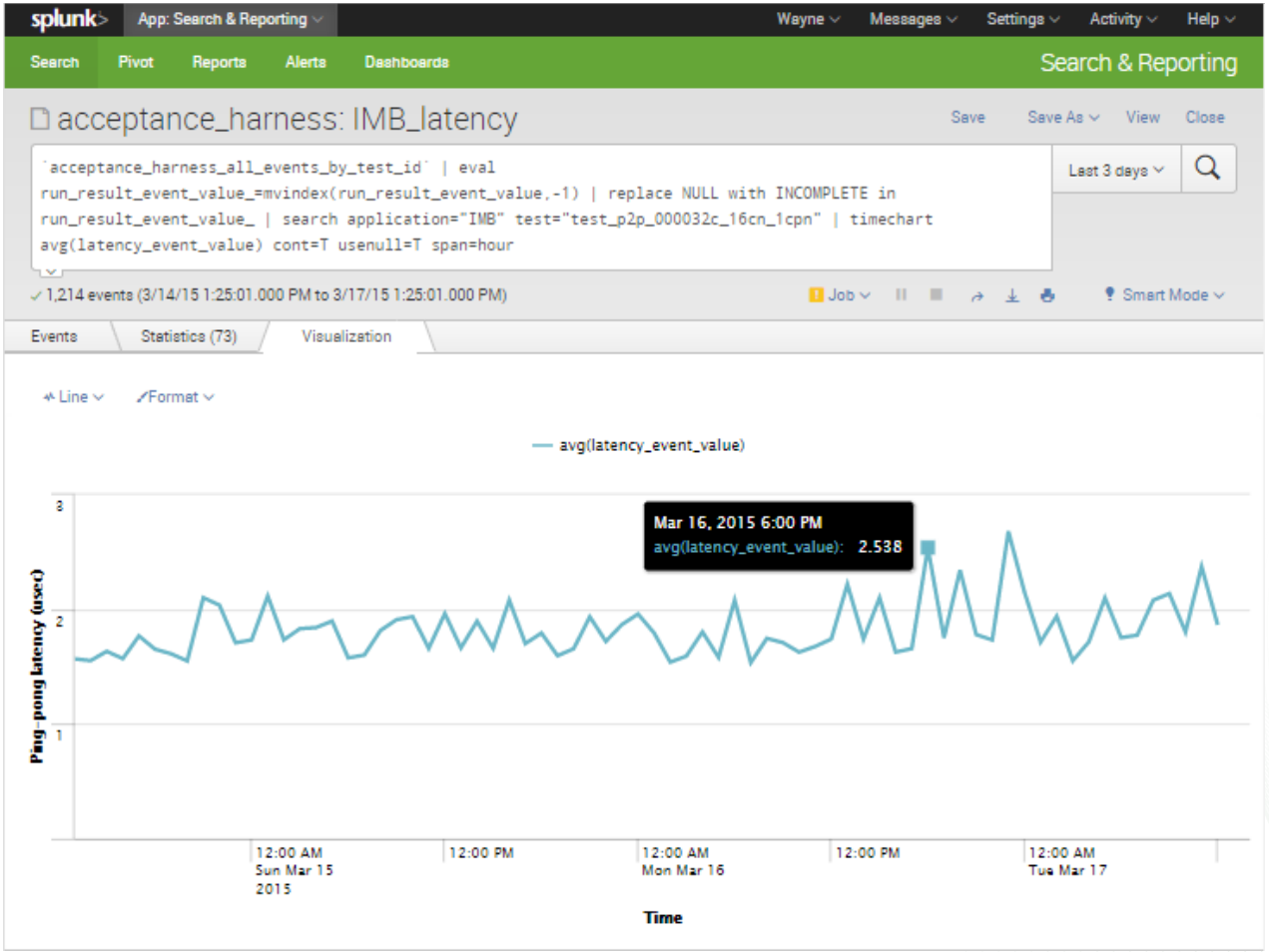
- Our experience with Jenkins indicated the plot capabilities were not as flexible as would be desired
- We also wanted to be able to visualize results from runs initiated by the existing harness, which is not straightforward for Jenkins which wants to own the build process
- Investigated the Splunk log monitoring tool as an alternative
- Does not satisfy all our original requirements, e.g., regarding job management, but has superior reporting capablities

Splunk: Implementation

- Used the install of Splunk already in use in the center
- Made small changes to the harness to write key events to system log, e.g., build/submit/run begin/end events
- Modified report_executable.x for harness applications to write metrics of interest to the system log, appropriately tagged
- Wrote Splunk code to collect these events, group by application run instance
- Splunk reports and dashboard implmented to present results of application runs over time

Splunk example: IMB

- Used the same IMB example described previously
- Ran in the harness, used Splunk to collect and display results



Part 3:

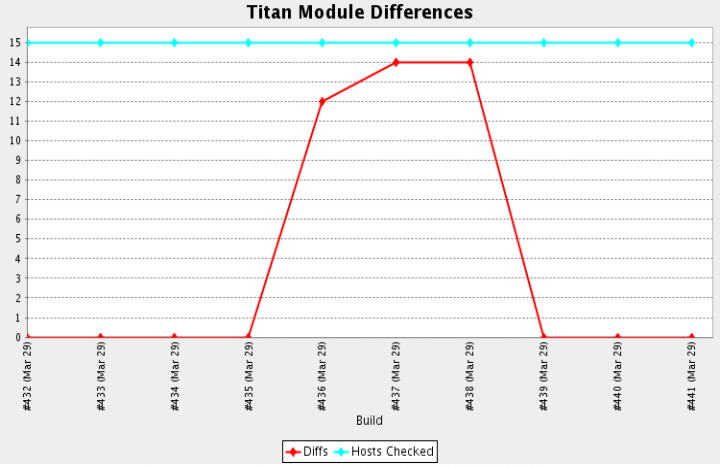
Environment Monitoring

Environment Monitoring

- Large number of environment variables, modules, tools that can impact system use
- Environment changes can be just as impactful as performance changes
- Many environment triggers are behind scenes and not known to most users
- Staff members have ability to change environment
  - Small change: *should not impact anyone*
  - Software installs become defaults
- Verify consistent standard environments between systems, login nodes, and over time
- Examples:
  - Ensure modules same across login nodes
  - Find changes in a system's default modules over time
  - Ensure batch job submissions follow system's batch policies
  - Ensure pre-defined set of tools and environment variables exist on each system
  - Verify ability to send/retrieve data between the HPSS and Data Transfer Nodes

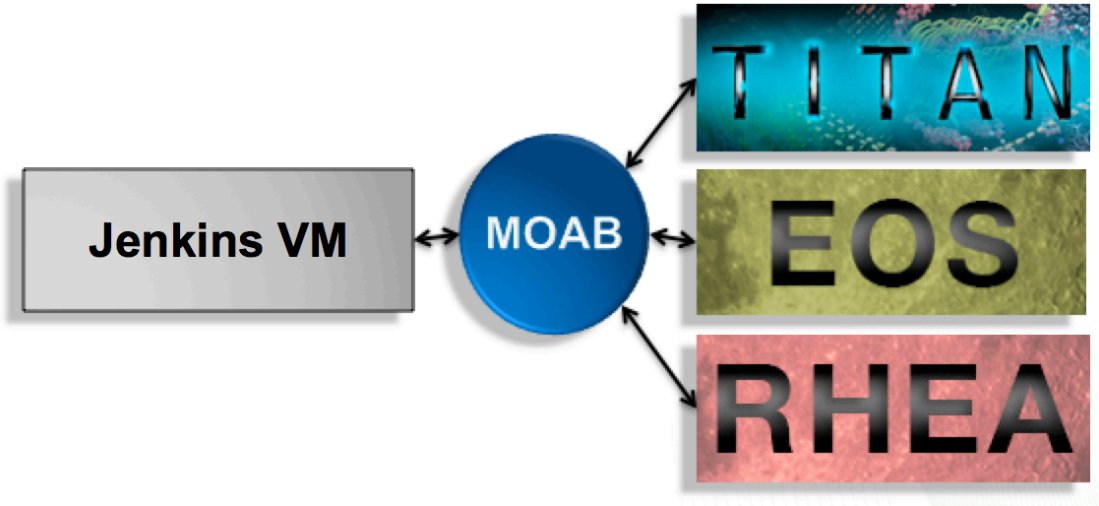
Module Monitor Example

```
See <http://localhost:8080/job/common_modules_within_system/438/>
----------------------------------------
Started by user anonymous
Building in workspace <common_modules_within_system/ws/>
[titan] $ /bin/sh -xe /tmp/hudson3903209286972211677.sh common_modules_within_system/bin/run_test_titan
[[10:47:41 3-29][submit_batch:main:202]] Checking for lockfile: <common_modules_within_system/ws/lockfile>
[[10:47:41 3-29][submit_batch:main:546]] Creating file: <common_modules_within_system/ws/lockfile>
[[10:47:41 3-29][submit_batch:main:528]] Creating batch file
[[10:47:41 3-29][submit_batch:main:546]] Creating file: <common_modules_within_system.pbs>
[[10:47:41 3-29][submit_batch:main:614]] Removing <common_modules_within_system/ws/batch-started>
[[10:47:41 3-29][submit_batch:main:614]] Removing <common_modules_within_system/ws/batch-completed>
[[10:47:41 3-29][submit_batch:main:335]] Submitting batch job
[[10:47:41 3-29][submit_batch:main:344]] Job submitted: 2305631
[[10:47:41 3-29][submit_batch:main:370]] Waiting on batch job to start
[[10:48:41 3-29][submit_batch:main:377]] Batch job started
[[10:48:41 3-29][submit_batch:main:413]] Waiting on batch job to complete
[[10:51:41 3-29][submit_batch:main:420]] Batch job completed
[[10:51:41 3-29][submit_batch:main:614]] Removing <common_modules_within_system/ws/lockfile>
[[10:51:41 3-29][process_results:main:27]] Processing results: Begin
[[10:51:41 3-29][process_results:main:56]]
HostsTested,15
ModuleDiffs,14
ModulesTested,225
ModuleVersionsTested,369
Miss Details:
VirtualGL(3):ERROR:106: missing on titan-ext4 titan-ext5 titan-ext1 titan-ext6 titan-ext3 titan-ext2
titan-ext7 Found on titan-login8 titan-login5 titan-login6 titan-login7 titan-login2 titan-login4
titan-login3 titan-login1
...
[ERROR][[10:51:41 3-29][process_results:main:73]] Module differences found: 14
Died at common_modules_within_system/bin/process_results line 289.
Build step 'Execute shell' marked build as failure
Recording plot data
```



Throwing a wrench in the works – Two factor authentication

- Center requirement
  - System Access
  - Prevents automated ssh
  - Limits plug-ins
- BioUno's PBS plug-in
- Center-wide batch system
  - Already in place to enable user workflows
  - Provides ability to submit batch jobs between OLCF user systems
    - qsub between Titan, Data Transfer System, Pre/Post Processing and Analysis
    - Jenkins can also use to access OLCF user facing systems
  - Replace ssh with qsub
  - All work must go through batch system
    - Compute jobs, Compiles, Environment checks



Center-wide Batch Access

- Must manage batch jobs
  - ssh returns once task completes or fails
  - qsub returns immediately
    - Given task may not start for minutes, hours, days
  - Do not want to load queue with multiples of same test
    - Often batch system accepts jobs when target system unavailable
- Methods to track batch job progress
  - Testers need ability to submit work to batch system and wait until batch job completes
  - Provide language independent functionality to testers (script, plugin)
  - Two methods tested:

1. Poll Queue

- qstat, showq
- Straightforward
- Queue polling issues?
  - Load
  - Communication timeouts

2. Monitor Files

- Utilize center-wide filesystems
- Reduce batch polling load
- Provides additional insight into job progress
- Control time allowed for each step

Conclusions

- Cron concern, but tool is more than cron
- Allows us to visually organize and list tests
  - Can see big picture and holes in testing
- Known tool
  - Security, infrastructure, staff already familiar
  - Reduce barrier for others to contribute
- Not Jenkins experts, still plugins/features to investigate
- Jenkins has already proven to be beneficial; we will continue to add tests and investigate additional plugins

Acknowledgements

- Many people contributed to this effort: Don Maxwell, Arnold Tharrington, Clay England, Rich Ray, Jason Kincl, Ashley Barker, Adam Carlyle, Robert French, Tony DiGirolamo, Ryan Adamson.

- This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Questions?

Thank you!