SC20

Everywhere we are | more than hpc.

Continuous Regression Testing of the Sustained Petascale Blue Waters Infrastructure

Nov.2020 • NCSA/University of Illinois

Topics

- Introduction & Motivation
- Deployed Jenkins Framework
- Classes of Tests
- Test Structure and Execution
- Use Cases
  - Post Power event checkout
  - A project team reported a build problem
  - A project team noted a recurring issue
  - mdtest and ls: filesystem functionality
  - IOR: filesystem performance
  - Data transfer nodes & Globus Online endpoint status
  - Vendor driver and software updates
- Conclusions

Introduction & Motivation

Monitoring a large HPC system is a complex task:

- Several layers and versions of software stack
- Multiple components (compute, I/O, network, etc)
- Diversity of user needs and usage
- Variability in environmental conditions (network traffic, etc)

Performance or functionality regression may occur at any moment

- Due to incorrect configuration of updated components
- Due to failing components
- Due to unexpected external events affecting system or users

Testing must be a continuous activity:

- Multi-level: from hardware pieces to full applications
- Multi-viewpoint: from system-admin tools to user programs

On Demand Testing

Testing at Boot Time on Blue Waters before Return to Service:

- IOR – I/O assessment
  - *mdtest* occasionally, if anything justifies it
- *test_nodes* (Cray) – tests each node with dense linear solver; both CPU and GPU nodes
- *test_links* (Cray) – tests each Gemini link in the system
- ~20 minutes total for tests

Testing after each *aprun* failure:

- run Node-Health Checker (NHC) on affected nodes

Continuous Regression Testing

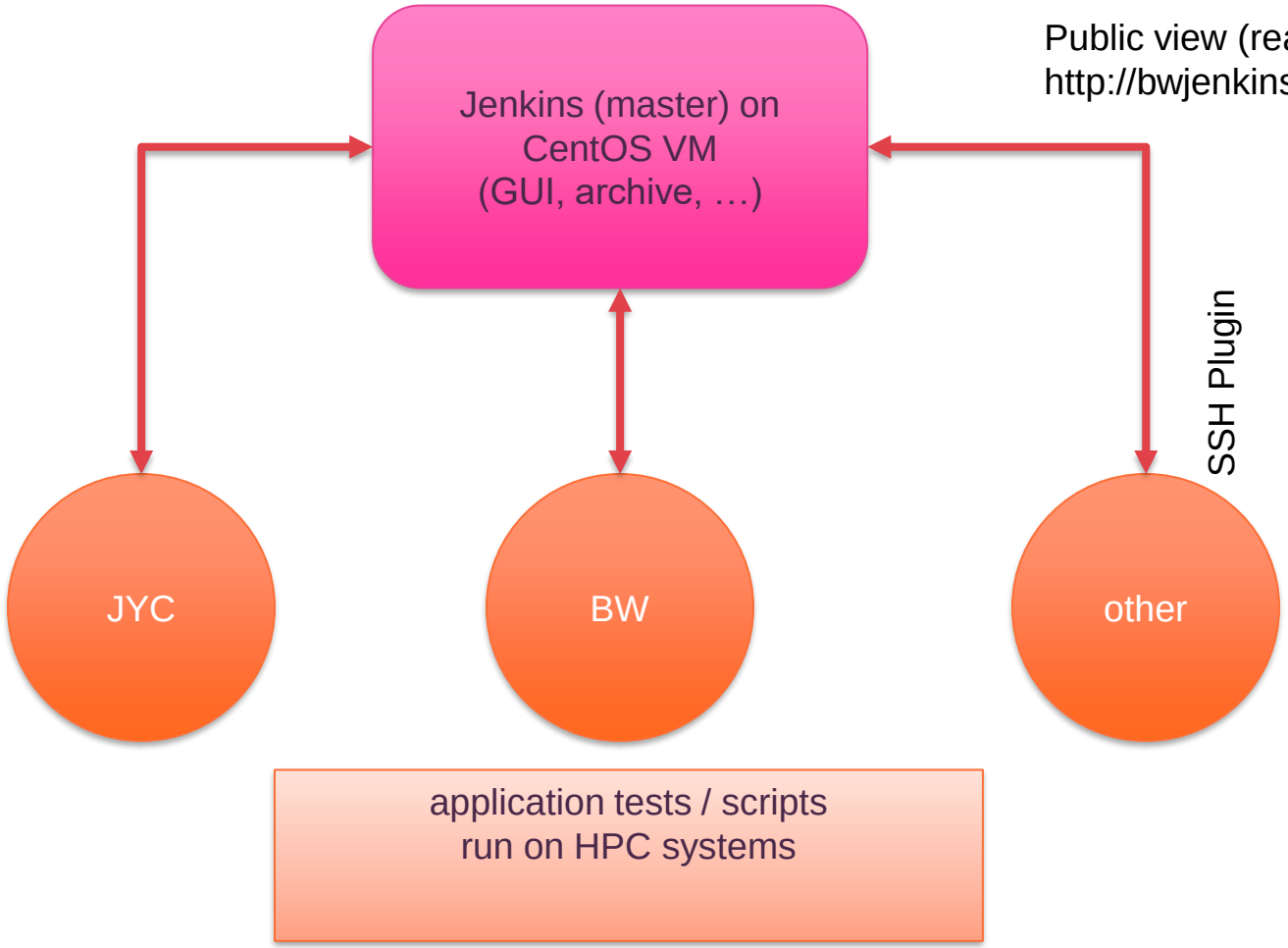
Requirements:

- Ability to monitor functionality and performance
- Monitor at user-level – no privileged access
- Automated and consistent test scheme
- Store historical data / results
- Run during normal operations, with minimal perturbation

Solution on Blue Waters: Jenkins Automation Server

- Open source
- Chosen for versatility, available plugins, community support
- Tests results can be stored, generate notifications

Deployed Jenkins Framework at NCSA



Systems: Blue Waters compute nodes, Test system, login nodes, disks, etc.

Jenkins Global Security

Very flexible configuration of capabilities:

Authorization

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

User/group	Overall					Credentials					Agent							Job								Run		View				SCM
	Administer	ConfigureUpdateCenter	Read	RunScripts	UploadPlugins	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Provision	Build	Cancel	Configure	Create	Delete	Discover	Read	Workspace	Delete	Update	Configure	Create	Delete	Read	Tag
Sub_UTK	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
arnoldg	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐	☑	☑	☑	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑	☑
bw_admin	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
bw_seas	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐	☑	☑	☑	☑	☑	☑	☐	☑	☑	☑	☑	☐	☑	☑	☑	☐	☑	☑	☑	☐	☑	☑
rbudiard	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
Anonymous	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☑	☐	☐	☐	☐	☐	☐	☑	☐

Classes of Tests

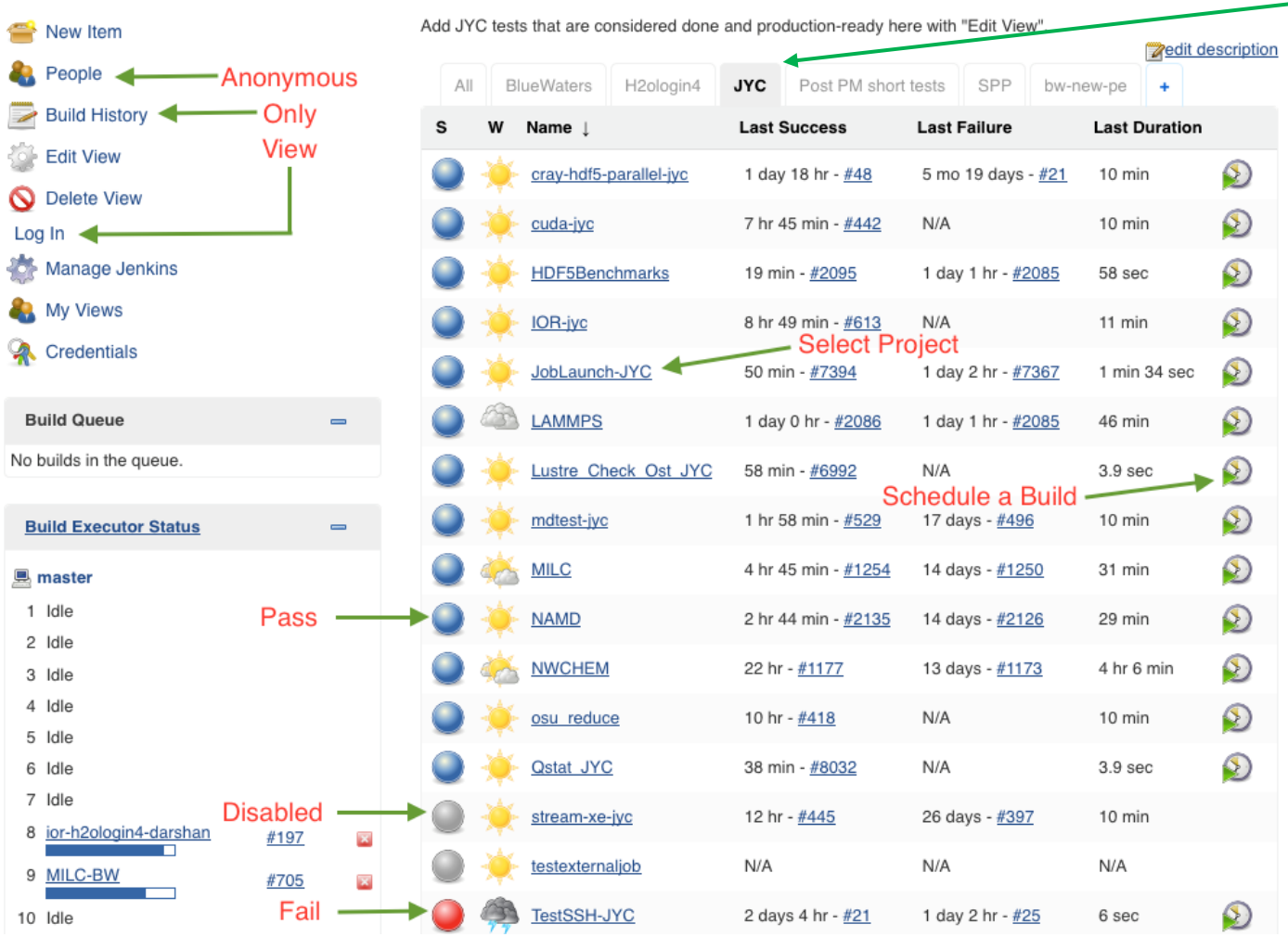
- **Utilities and full applications**: tests executed in compute nodes of Blue Waters
- **Login nodes**: tests of serial utilities and application builds that are typically conducted by users on the login nodes
- **ISC-related**: tests that mostly assess Quality-of-Service on the three filesystems of Blue Waters
- **JYC**: tests executed in our Test-and-Development System (TDS), a one-cabinet Cray machine with the same hardware and software as in Blue Waters
- **Lustre backup**: tests related to backup, or to purging old files, in some of the filesystems
- **Post-maintenance**: short tests that are executed after the system undergoes maintenance, mostly targeting a safe return to regular operations

Classes of Tests (cont.)

- **Power check**: tests for critical components of the system that must be assessed after a power event.
- **SPP applications**: tests with sustained petascale benchmarks, which are full scientific applications at scale.
- **Job scheduler**: tests to evaluate behavior of current version of the job scheduler, executed on TDS.

Test Structure and Execution

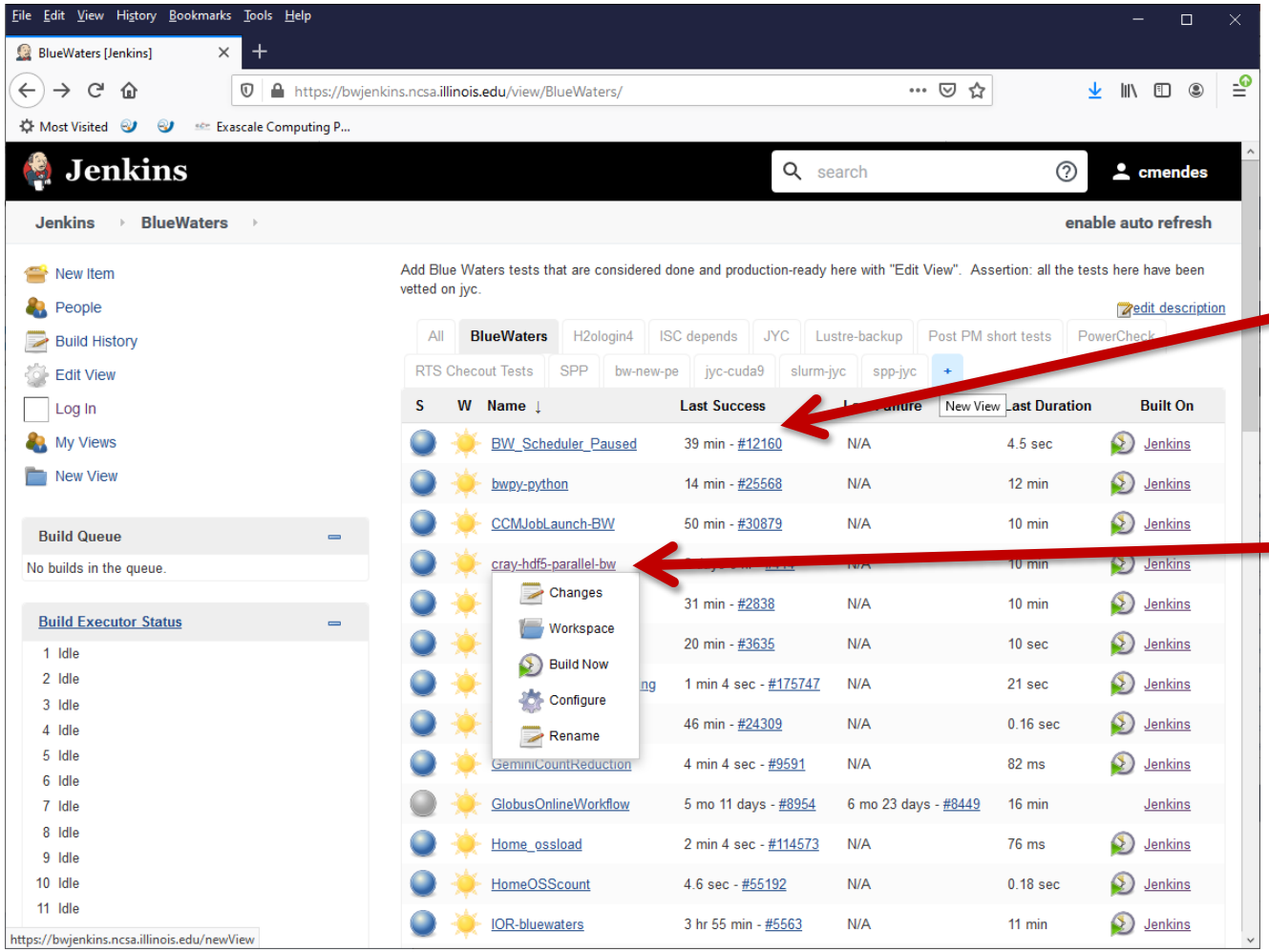
Jenkins GUI Home Page:



Accessible inside NCSA's network only !

Test Structure and Execution (cont.)

Tests on Blue Waters compute nodes:



Click to see output from the test execution

Click to build and/or configure test

Test Structure and Execution (cont.)

What tests are performed: depend on the sub-system tested

- Blue Waters tests:
  - Related to execution of programs on Blue Waters compute nodes
  - Classes of tests:
    - Scientific applications – NAMD, MILC, LAMMPS, …
    - Benchmarks – IOR, stream, …
    - Libraries – HDF5, CUDA, …
    - System support – Python, job scheduler, Shifter, mdtest, power
    - System components – mom nodes, mdtest, Gemini routers
- Test and Development System (JYC) tests:
  - Subset of Blue Waters tests, run on JYC
- Login-Node tests:
  - Executed on Blue Waters login-nodes – short app builds, libraries
- Post-Maintenance tests:
  - Applied prior to returning to regular operation

Test Structure and Execution (cont.)

How long and how many nodes…

- Job-less , typically 1 node, short time
  - Login node (via ssh)
  - Connect to a server/service and retrieve database row
  - Scrape a log
- Batch job tests, can be multi-node, sufficient time and scope for test
  - Programming environment (builds of benchmarks and apps)
  - Petascale performance tests (full applications)
  - Lustre : mdtest, ior
- Node counts, times, and frequency are reviewed by at least 2 people
  - Following internal best practices document in our wiki
  - Peer review of tests prior to production status
  - Less than couple hours wall time
  - Less than hundred nodes
  - Tests are owned and signed: email notification list

Test Structure and Execution (cont.)

How long and how many nodes – a few examples:

- Application test: NAMD
  - Frequency: every 12 hours; Duration: 1 hour max, 42 minutes actual
  - Number of nodes: 1; Number of processors: 32
- Application test: MILC
  - Frequency: every 12 hours; Duration: 1 hour max, 32 minutes actual
  - Number of nodes: 8; Number f processors: 8 × 32 = 256
- Benchmark test: IOR
  - Frequency: every 6 hours; Duration: 10 minutes max, 10 minutes actual
  - Number of nodes: 16; Number of processors: 16 × 7 = 112
- Library test: HDF5 parallel
  - Frequency: every 5 days; Duration: 10 minutes max, 10 minutes real
  - Number of nodes: 1; Number of processors: 1 × 16 = 16
- Accelerator test: CUDA bandwidth and matrix-multiply
  - Frequency: every 12 hours; Duration: 5 minutes max, 5 minutes real
  - Number of nodes: 1; Number of GPUs: 1

Use Cases – Post-Power event checkout

Jenkins

search

Greg B

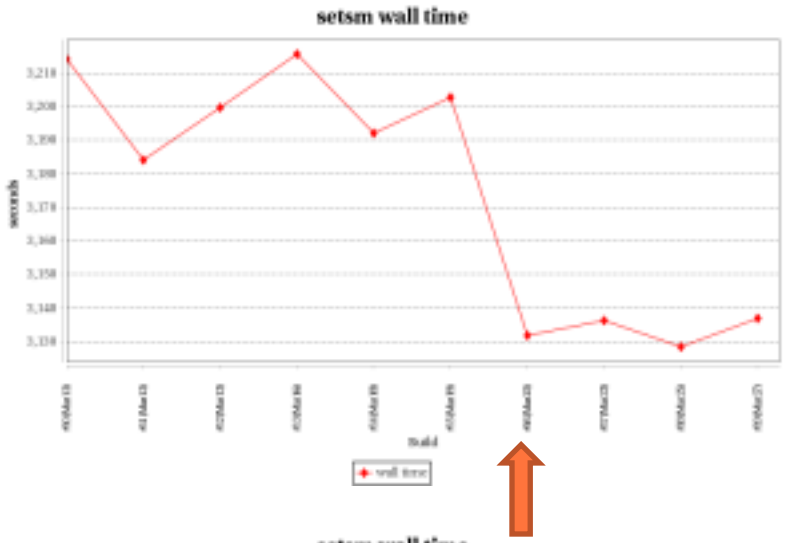
Jenkins › PowerCheck › Power_Fluctuation_Check › Full Stage View

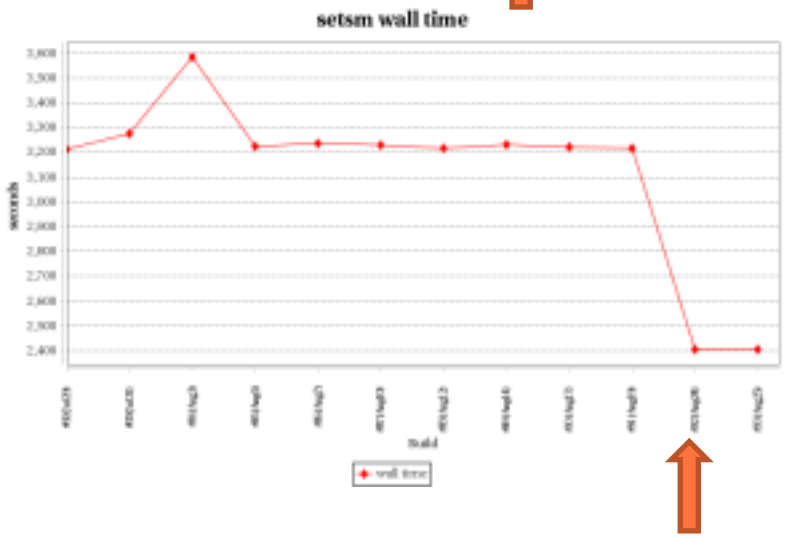
Power_Fluctuation_Check - Stage View

	Testing SSH access to bw.ncsa.illinois.edu via TestSSH-BW job	OST check via Lustre_Check_OST_BW job	Checking uptime for bw.ncsa.illinois.edu via Check_Uptime_Login_Node job	Checking number of down nodes via Check_Alps_for_Number_of_Down_Nodes job	Checking number of down nodes via Check_Torque_for_Number_of_Down_Nodes job	Checking Lustre availability via Check_Lustre_Availability job	Checking mom availability via Check_if_Mom_Down job	Checking LDAP availability via Check_LDAP job	Checking home QOS availability via QOS_home_check	Checking scratch QOS availability via QOS_scratch_check	Checking projects QOS availability via QOS_projects_check
Average stage times: (Average full run time: ~7min 4s)	4min 18s	16s	11s	19s	12s	11s	12s	17s	7s	10s	9s
#86 Sep 17 11:45 No Changes	20s	29s	12s	13s	12s	9s	12s	18s	10s	10s	9s
#85 Aug 19 08:34 No Changes	17s	14s	11s	14s	11s	14s	12s	17s	6s	10s	9s
#84 Aug 10 13:52 No Changes	15s	15s	11s	13s	12s	9s	11s	17s	6s	10s	9s
#83 Aug 02 09:57 No Changes	1min 41s failed										
#82 Aug 02 09:46 No Changes	10min 46s failed										

- Pipelined set of tests to run post-power event.
- Currently serialized but some tests could run concurrently.
- Order based on dependencies.

Use Cases – A project team reported a build problem



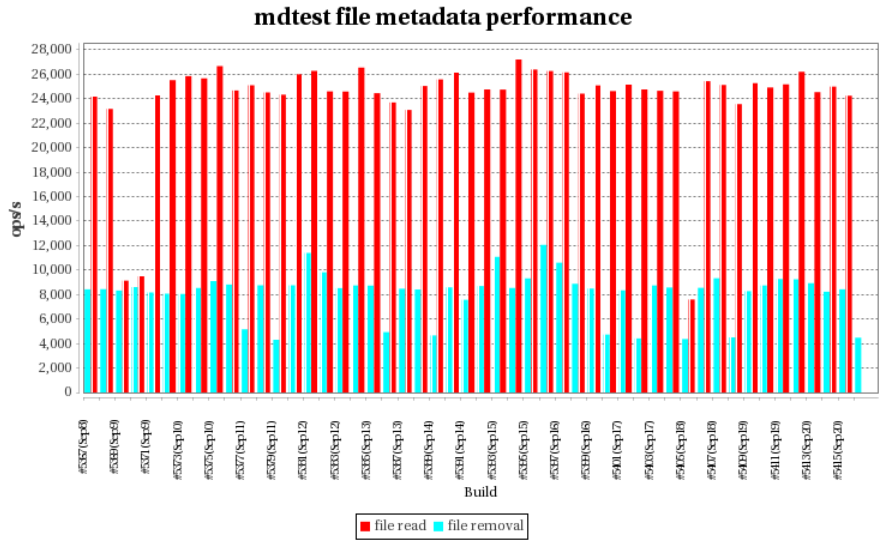


- A key project on the system reported a broken build.
  - Triaged by support staff (Makefile repaired)
  - Support staff automated the build with a script
    - Builds dependent libraries
    - Pulls project code from their github
    - Retrieves the git version
  - Deployed Jenkins build test (rebuilds any newer version)
  - Deployed Jenkins performance test (triggered by a good build)
- Support staff has provided feedback about performance change (improvements so far, no regressions) .
  - March 2020 (top left)
  - Aug. 2020 (lower left)

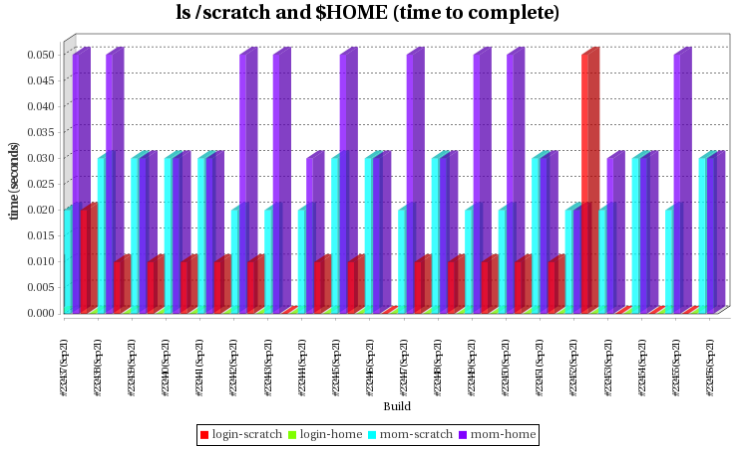
Use Cases – A project team noted a recurring issue

- A team running python jobs associated with their workflow reported : **python is down**
  - Jira issue created
  - Admin and support staff resolved the issue
  - The issue recurred **->** test case added to Jenkins for monitoring and notification
  - Problem recurred a couple times since (related to Lustre locks) and was **quickly resolved** due to prompt notification

Use Cases – mdtest and ls : filesystem functionality



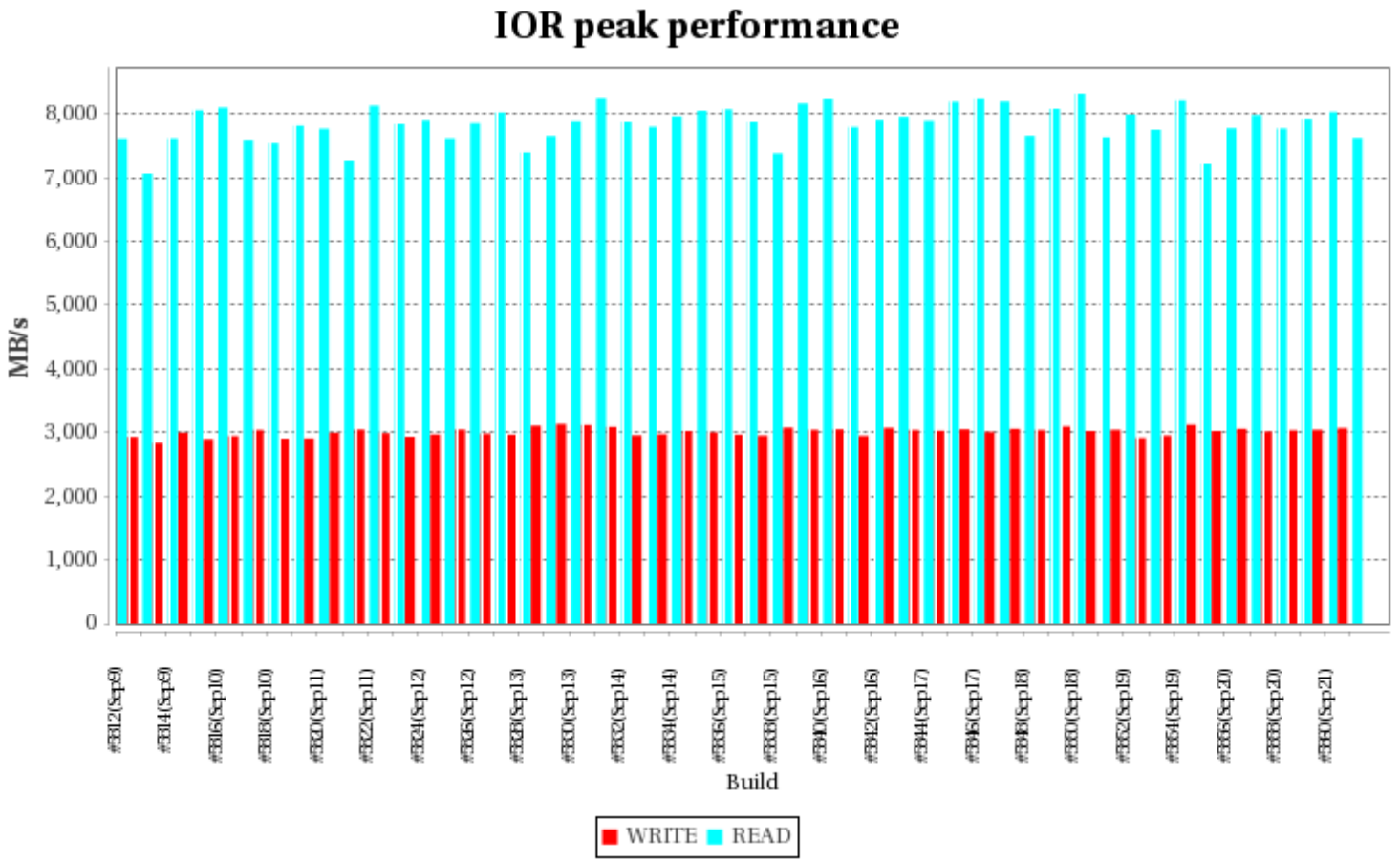
- mdtest (directories and metadata)
  - 4 compute nodes * 6 tasks MPI mdtest with 48k total files
  - git pull of the code and build with default programming environment



- ls ($HOME and scratch)
  - Runs on login nodes
  - 5 minute frequency: **early issue detection**

Use Cases – IOR: filesystem performance

- 16 nodes * 7 cores/node MPI IOR
  - Sized to match its counterpart on our test rack
  - Runs in a striped directory (lfs setstripe -c 8 .)
  - Does a fresh build of IOR using the default PrgEnv-cray
- Tests: automake, autoconf, and default build environment, Lustre striping and throughput for a small job using scratch filesystem.



Use Cases – Data transfer nodes and GO endpoint status

- With Globus Online (GO), an endpoint is up if any of its servers are up.
- This test checks all servers of an endpoint.
  - System admin. team wanted more granularity with 25 servers in our endpoint.
  - Jenkins test runs a python script using the GO API
    - query each server in an endpoint
    - setup with refresh credentials per GO API

```
[SSH] executing...
h2ologin4

real      0m5.977s
user      0m0.924s
sys       0m0.292s
--> ncsa#BlueWaters d59900ef-6d04-11e5-ba46-22000b92c6ec
   ie27.ncsa.illinois.edu :gsiftp service UP
   ...
   ie06.ncsa.illinois.edu :gsiftp service UP

[SSH] completed
[SSH] exit-status: 0

Finished: SUCCESS
```

Use Case – Vendor driver and software updates

- Mid 2018 we migrated from CUDA 7.5 -> 9.1
  - The change required testing functionality and performance
    - Existing codes
    - Recompiled codes
  - Multiple changes in the software stack (gcc version, gsl module)
  - Jenkins leveraged existing tests and historical performance

Recompiled binary tests

test name	CUDA 7.5 performance	CUDA 9.1 performance	Notes
CUDA matrixMul	277 GFlops/s	287 GFlops/s	3-4% improvement
AWP-ODC SPP test	6 minutes	5 minutes	> 15 %

Conclusions

Monitoring framework for Blue Waters: based on Jenkins

- Good versatility, easy to configure and use
- Tests can be run periodically or manually triggered
- Broad diversity in tests, from low-level to full applications
- Test results remain stored for historical analysis

Many concrete use-cases so far:

- Automatic detection/notification of incorrect system behavior
- Tests have been properly targeted to critical system components
- Big savings in diagnosis time were observed when problems occurred

Allowed testing to be a **continuous** activity:

- More tests can be easily added for critical components
- Existing test infrastructure can be extended to other (new) sub-systems

Acknowledgements

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) the State of Illinois, and as of December, 2019, the National Geospatial-Intelligence Agency. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

We thank the rest of the Blue Waters team at NCSA and the Cray site staff, for keeping Blue Waters healthy and productive during the last eight years.

More information on Blue Waters is available at https://bluewaters.ncsa.illinois.edu/