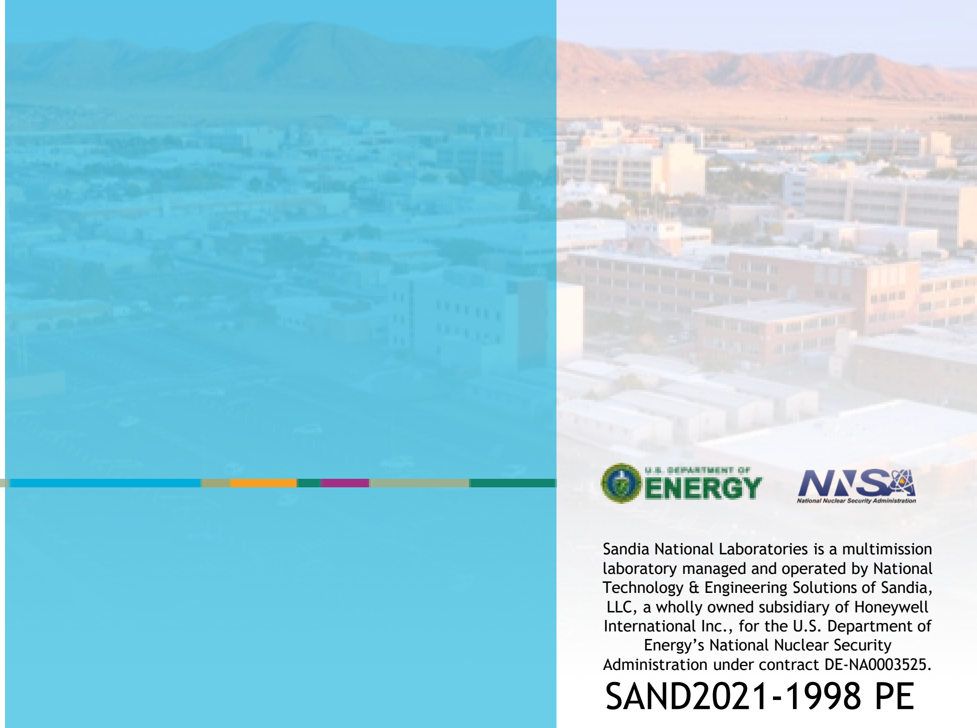
LDMS Testing on Target Systems



Benjamin Allan

1/5/2021

SAND2021-1998 PE

Machinery

By “target systems” we mean various Sandia HPC clusters with divergent hardware and software stacks, on which the building and use of LDMSD depends in peculiar ways.

By “testing” we mean the tests that can be performed using the code and utilities included in the github ovis-hpc/ovis source tree, with minimal support scripts for robots contained in gitlab-ex/baallan/ovis-jenkins/:

- [pll-]ldms-static-test.sh
- make check
- make installcheck
- jenkins

Outline: Testing goals & methods

What

Where

How

Why this way(s)

Driver scripts

What we want to make sure always works (just ldmsd, not DB & web)

Most of the following have many variants ([list]) and combinations (cross-product x of lists) to be covered.

- **repository clone** ([no permutations except tag/branch name])
- **generate build files** (autogen.sh) ([production environments])
- **configure** ([multiple option sets])
- **build rules** ([option sets] x [production environment])
- **install rules** ([option sets] x [production environment])
- **package** ([configuration] x [rpm build environment] x [spec file])
- **daemon startup** ([systemd scripts] x [site ldmsd configurations])
- **sampler => store pipelines** ([multidaemon config] x [production environment] x [audit tool])
- **scaled-up pipelines** ([simulated node count] x [queue system] x [audit tool] x [production environment])

Where we want it to be tested and work (SNL)

- Github, for as much simple stuff as possible with Github Actions
- Current platforms:
  - CTS1/omnipath/mlx5 (cts1x, eclipse, attaway, manzano)
  - Cray XC (mutrino/voltrino)
- Legacy platforms:
  - TLCC2/mlx4 (skybridge, chama, hazel)
  - IBM/Power-# (testbeds)
- Server platforms
  - RHEL8/mlx6 (bobbyd, bitzer)
  - RHEL7 (testbeds (bobbyd), mon1, desktops)
  - Centos (vm desktops, servers)

How

Driven as part of any build (make check or installcheck) anywhere

- Complaining remote builder/user should be able to run related tests and report

Driven by github processes

- Specific public containers (Licensing issues w/redhat)

Driven by Jenkins @ SNL

- Scripts managed in a public repo, but not expected to be widely portable

Driven by X replacement for Jenkins harness

Coded in the language appropriate to the feature tested

- python drivers to test python features
- bash scripts to demonstrate bash-driven deployment environments (systemd & user sbatch scripts)
- C to unit test and the like; these must be drivable by some standard(s), e.g. make check

Reporting results in some way that can eventually be integrated in a dashboard

Why test as above?

Complementary to container-based testing: need both

In-tree test code is most easily kept in sync with the code tested

In-tree test code can be easily examined/run by concerned customers to validate their build

Tests also serve to create verified & up-to-date examples of daemon configuration files for users to read/modify on their installation.

Single-node testing (in-tree)

ldms-static-test.sh

- Bash framework provided for defining a group of daemons and verifying function
- Man page documents the operations (start, stop, ls, collect FD list, etc)
- One 'case' per test script
  - Test scripts are "high level" bash, with all the LDMS config input boiler-plate automated.
  - "Capture everything":
    - daemon logs, file input, cmd line input, store output, audit tool logs,
    - stdout from case script/slurm, open file descriptor list (if enabled).
    - Can be tarred and shared for debugging.
  - Audit tools can be enabled/disabled per ls or per ldmsd execution
- Driver ***ldms-run-static-tests.sh*** runs all configure-enabled cases from make check/installcheck
  - Yields case pass/fail/xfail
  - Wrapper of ldms-static-test.sh invocations
- Currently used audit tools: valgrind, gdb
- Error handling is "fail fast"; commands after a failed command are ignored, except for summary reporting.
- Root privilege is required to test certain plugins; root execution requires extra environment value 'allowroot=1' be given.

Eg: "ldms-static-test.sh meminfo"

Multi-node testing (in-tree)

pll-ldms-static-test.sh

- SPMD bash version of ldms-static-test.sh, with automatic round-robin daemon location, port number management.
- Man page documents the parallel handling
- One 'case' per test script
  - Each case has one or more separate, resource manager specific, submission files
    - Slurm input is currently supplied
  - Automated replication (where needed) of similar input files from templates
- Intended to be compatible with Jenkins-like tools that understand slurm.
- Error handling is more warning oriented, as node failure may be an expected case

Eg. "NFLAP=20 sbatch -n 16 -p batch --time=30 --nodes=4 --account=xxxx $(pll-ldms-static-test.sh -b cluster)"

Build/package/deployment testing

Jenkins

- Minimal configuration done in Jenkins (lists of string values for controlling test option grid combinations)
  - Options:
    - Code configuration
    - “build” step tested (autogen, configure, compile, install, check, package)
    - Platform (Jenkins slave instance)
  - Repo of bash scripts keeps the real work out of Jenkins, because we want to replace Jenkins.
- Jenkins iterates the combinations

Open Suse Build Server

- Stria/Astra ARM package builder (LDMS on ARM-flavored TOSS3)

Automated deployment cannot be done on production clusters (policy) but could on containers (s.t. container policy on production cluster).

- Relocatable TOSS rpms allow testing of ‘deployment’ without root in production environments or desktops (but user-level systemd remains a sticky wicket in RHEL7).