LArSoft Continuous Integration Infrastructure

Erica Snider

on behalf of
Mark Dykstra
Lynn Garren
Patrick Gartung
Gianluca Petrillo

June 17, 2014

Outline

- Continuous integration (CI)
- LArSoft goals and requirements
- The CI workshop
- CI system overview
- LArSoft testing infrastructure

Continuous integration

- What is continuous integration (CI)?
  - A software development practice in which team members integrate their work into the main development branch frequently, usually at least daily.
  - Each integration is tested by an automated build and test system designed to detect integration errors as quickly as possible
- Primary benefits
  - Catch integration issues quickly
  - Maintain a more stable main-line development branch
  - Can create a release with known properties at all times
  - Low cost
- Widely used technique
  - Similar methods (e.g., “Test Driven Development”), shown to increase productivity, despite overhead of writing appropriate tests

Continuous integration

- Testing protocol
  - Unit testing
    - Tests of smallest elements of the software to ensure they work as intended
    - Will hear more about what, why and how in Jim Amundson's talk later today
  - Integration testing
    - Tests of multiple software elements that are designed to work together
    - Most tests that use the art framework are really integration tests

LArSoft CI goals

- Maintain a test suite with broad range of tests
  - Cover all levels of the software, all experiments
- Maintain a flexible test framework
  - Support unit and integration testing
  - Support tiers/groups of tests to allow running tests to suit the situation.
- Operate a CI system for LArSoft + experiment software
  - Trigger test workflows based on specific events
    - Automated and manual trigger events
  - Provide rapid feedback to developers, managers
- Maintain a develop branch that always builds and runs

LArSoft CI workshop goals

- Present and discuss:
  - Current requirements for the LArSoft CI system
  - Vision for the fully implemented LArSoft CI system (LArCIC?)
- Obtain input from experiments on needs
- Discuss testing more generally
- Demonstrate the operation and basic features of the existing LArSoft CI system
- Provide guidance on how to create and integrate tests into the system
- Work with attendees on specifying, creating or integrating tests, new or existing, into the CI test suite for LArSoft and experiment software

LArSoft CI workshop agenda

Tuesday, 17 June 2014

14:00 - 14:30	Welcome, Goals, Scope of the Workshop *30'* Speaker: Dr. Robert Roser (Fermilab)
14:30 - 15:00	The CI Infrastructure *30'* Speaker: Erica Snider (Fermilab)
15:00 - 15:20	MicroBoone Goals, Status, Needs *20'* Speakers: Dr. Eric Church (Yale), Dr. Herbert Greenlee (Fermilab)
15:20 - 15:40	LBNE Goals, Status, Needs *20'* Speakers: Dr. Thomas Junk (Fermilab), Elizabeth Sexton-Kennedy (FNAL)
15:40 - 15:55	Unit Testing - practice and experience *15'* Speaker: Dr. James Amundson (Fermilab)
16:00 - 16:15	Break
16:15 - 17:45	Demonstration of end-to-end use of CI system and how to code/integrate new tests *1h30'*

LArSoft CI workshop agenda

Wednesday, June 18, 2014

09:00 - 11:30	Coding of New Test/Validation modules *2h30'*
11:30 - 12:00	Pandora - algorithms and interface with LArSoft Discussion *30'* Speaker: Mark Thomson (University of Cambridge)
12:00 - 13:00	Lunch *1h0'*
13:00 - 13:30	Data products - Needs? Ideas? *30'* Speaker: Dr. Brian Rebel (Fermilab)
13:30 - 14:00	Topic 2: NIM Paper *30'* *Go through outline and sections and authors for the LArSoft NIM paper*
14:15 - 14:45	Topic 3 *30'*
14:45 - 15:45	Back to Coding *1h0'*

LArSoft CI system requirements

- Basic operational capabilities
  - Automatically run standard set of tests for each push to 'develop'
    - Rapid tests covering basic workflows for all experiments
  - Automatic tests run for each nightly build, integration release
    - More time, so run more complex tests
  - Production release testing
    - Still more time, so could run higher statistics tests
    - Operational testing very important for these
  - (Desired) Allow “pre-testing” of local repository changes
    - Test local changes prior to committing to central repository
    - Have the technical capability for this, but need to explore implications
      - Cannot support lots of people doing this often
  - All results available via web interface
  - Email to individual(s) initiating trigger

Draft LArSoft CI system requirements

- Draft system requirements
  - Builds on requirements for central service
    - https://cd-docdb.fnal.gov:440/cgi-bin/ShowDocument?docid=5319
- Summary of all additional LArSoft requirements

1) A step in a CI workflow has access to results of previous steps in a workflow
2) Trigger event can specify configuration parameter values, eg, input data configuration
3) Input data configuration must be archived and versioned
4) Each run of a workflow must have a unique ID
5) Test development should not interfere with production tests
6) Must trigger on pushes to 'develop' and 'master', successful nightly build, successful integration and production release builds if automated with CI system
7) Configurable delay between a trigger event and launch of triggered workflow
8) Manual triggers must allow specification of branches, tags, or commits to use on the central repository

LArSoft CI system requirements

- Summary of all requirements (cont'd)

  9) Trigger configuration data should be propagated to any subsequent triggers generated by the workflow

  10) Workflows that build software must support specification of branch, tag, commit

  11) The system should validate input parameters before initiating workflows

  12) (Desirable) Visualization of build-step results

  13) CI system configuration must support versioning that can map to LArSoft versions

  14) Workflows must be executable independently of the CI system framework

CI infrastructure overview

- Three major components...
  - CI server and integration system
    - Using the Central Build (Integration) Service
  - A CI application
    - Schedules, runs test workflows, reports results
    - Runs on the CI server and integration system
  - Software testing framework
    - Provides structure and features to facilitate development, integration and running of tests
      - e.g., discovery of test scripts, implementing test tiers / groups, printing pretty summaries / web pages, etc
    - Operates within the CI application
- ...plus the tests themselves and some useful utilities
  - Standard art log file scanning, histogram comparison macros, etc.

CI infrastructure overview

- Central build (integration) system

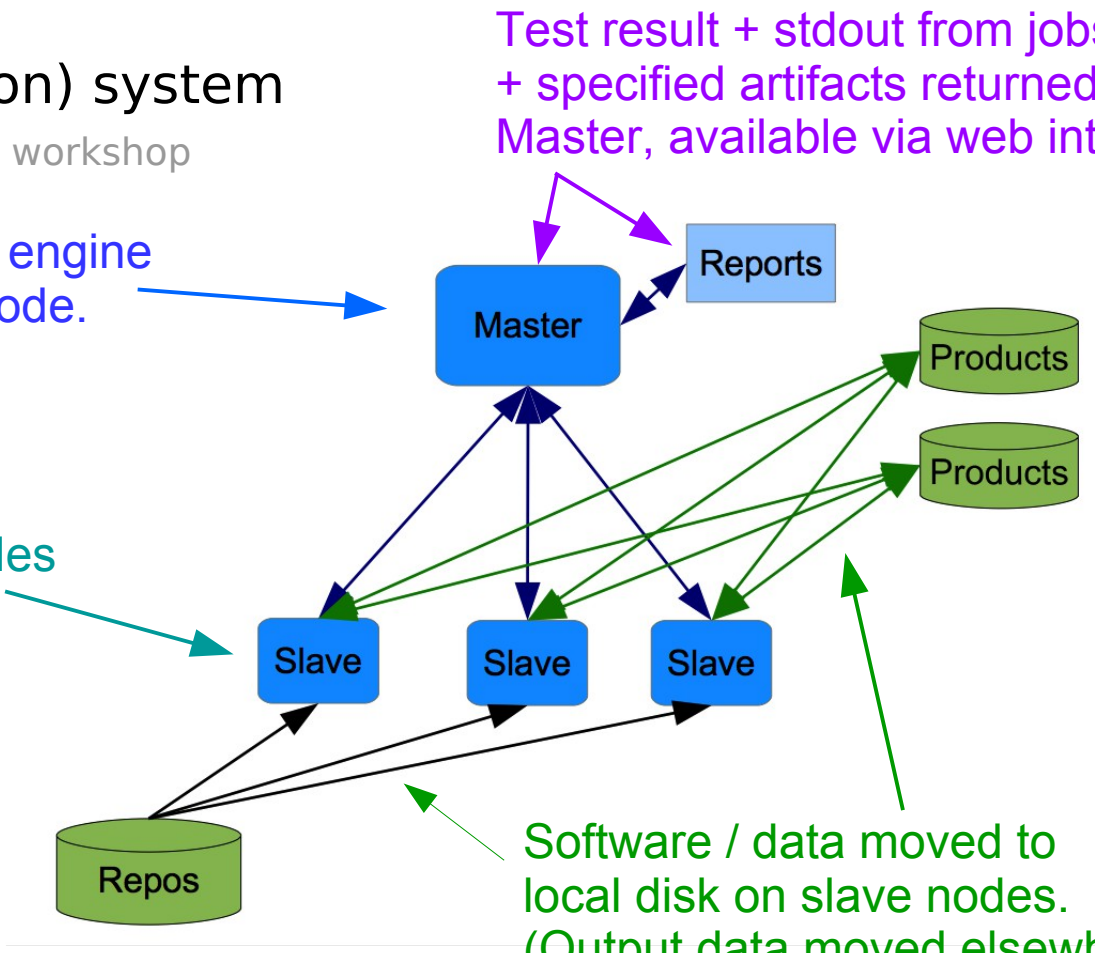
  From Glenn Cooper's talk at FIFE workshop

CI application / automation engine
(Jenkins) runs on Master node.

CI / build jobs / workflows
run on multi-core slave nodes
with local disk.

Rapid execution
of builds and tests.

Can be located anywhere



CI infrastructure overview

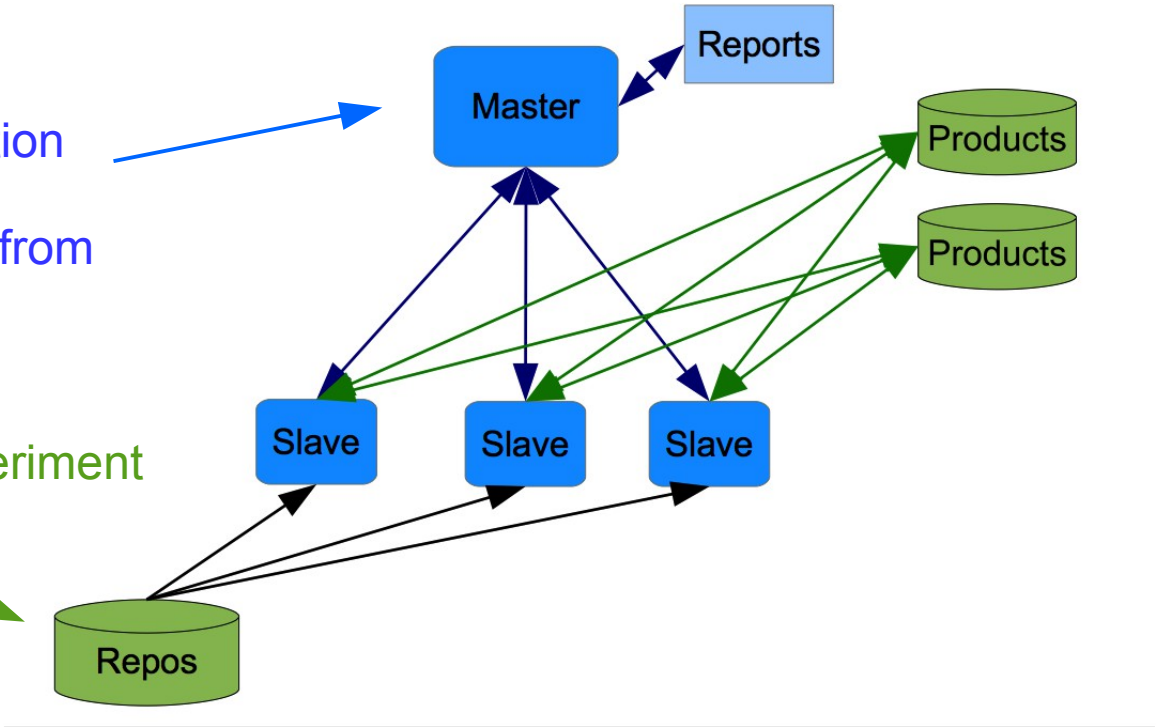
- CI workflow configuration and tests

CI job / workflow configuration lives within CI application. Maintained by “CI admins” from LArSoft + experiments.

.

Tests live in LArSoft + experiment repositories.

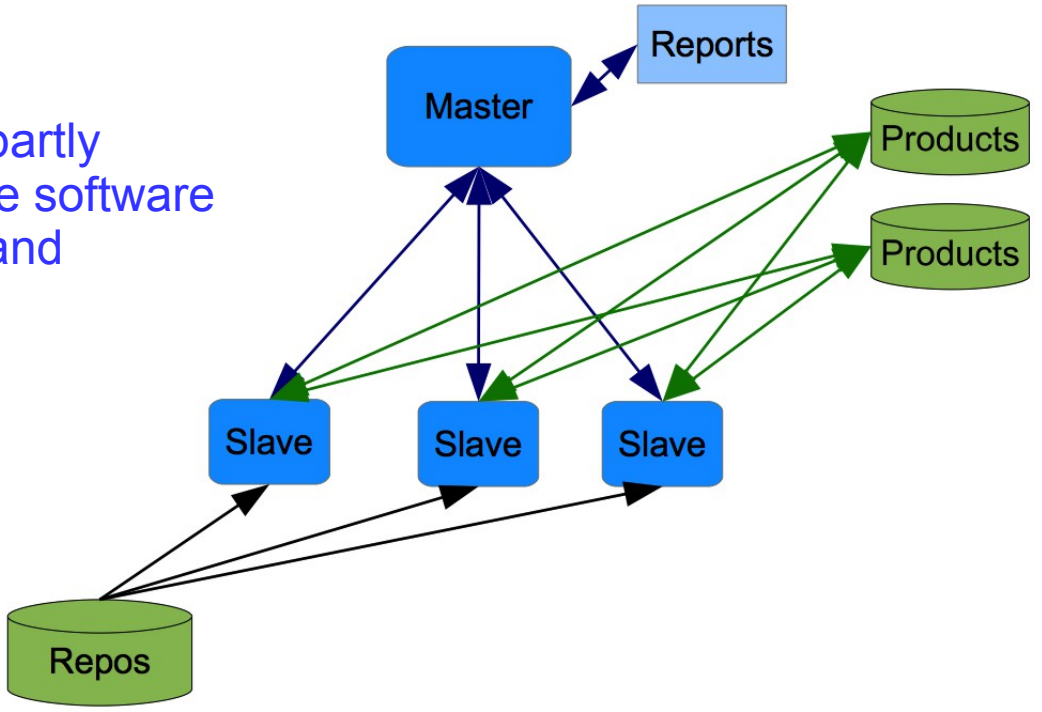
Written, maintained by LArSoft + experiment developers



CI infrastructure overview

- Testing framework

Testing framework provided partly by CI application, partly by the software build system (cetbuildtools), and partly by code maintained by CI admins.



LArSoft testing infrastructure overview

- Primary design goal
  - Make is very easy to add and administer tests
- Two basic test frameworks
  - mrb test
    - Aimed primarily at unit tests, but can run anything
    - Run tests out of the “build” area
    - Provides a ready-made test framework
  - Stand-alone test scripts
    - Aimed primarily at integration tests
    - Runs tests out of installed products
      - Allows testing of install procedure
    - Test framework implemented within Jenkins

mrb test

- Basic features
  - Runs 'buildtool –test' and 'make test' underneath
    - See https://cdcvs.fnal.gov/redmine/projects/cet-is-public/wiki/Buildtool_script
  - Utilizes '`cet_test( <test_script> )`' macro in CMakeList.txt files
    - Accommodates defining “test groups” to collect tests into suites
    - Many other capabilities
      - See documentation in cet_test macro file
  - Exit status of `<test_script>` communicates “success” / “failure” of test
  - Generates summary of tests and results after all tests run
- Adding a test
  - Add relevant script
  - Add a cet_test(…) line to CMakeLists.txt

Stand-alone test framework

- Basic features
  - Framework written to operate within Jenkins CI system
    - We own this part of the infrastructure
  - Finds test scripts based upon file naming convention
    - citest_<test name>.sh
    - These are installed into product bin areas during 'mrb install'
  - A test-tier map file specifies which test scripts belong to which tier / group
  - Exit status of test script communicates “success” / “failure” of test
  - Summary of tests saved in Jenkins log
    - stdout of scripts saved to separate log files
- Adding tests
  - Add a citest_* file
  - Add corresponding line to test-tier map file

What goes into a test?

- Tests can be executables
  - From lardata/tests/CMakeLists.txt:

    cet_test(sparse_vector_test)
    - Where 'sparse_vector_test' is an executable built out of the test area
- Can also be commands
  - From uboonecode/test/EventGenerator/CMakeLists.txt:

```
# use the general prodsingle_uboone.fcl file
# because OPTIONAL_GROUPS is defined, this test will not be run by default
# use mrb t --test-all to run all the tests
cet_test(prodsingle_uboone HANDBUILT
  TEST_EXEC lar
  TEST_ARGS --rethrow-all --config prodsingle_uboone.fcl
  OPTIONAL_GROUPS RELEASE
)
```

- Within a test, can perform comparison of results with a reference
  - Want to provide some standard tools for this

What goes into a test?

- Tests can be executables
  - From lardata/tests/CMakeLists.txt:

    cet_test(sparse_vector_test)
    - Where 'sparse_vector_test' is an executable built out of the test area
- Can also be commands
  - From uboonecode/test/EventGenerator/CMakeLists.txt:

```
# use the general prodsingle_uboone.fcl file
# because OPTIONAL_GROUPS is defined, this test will not be run by default
# use mrb t --test-all to run all the tests
cet_test(prodsingle_uboone HANDBUILT
  TEST_EXEC lar
  TEST_ARGS --rethrow-all --config prodsingle_uboone.fcl
  OPTIONAL_GROUPS RELEASE
)
```

Execute only when testing production release

- Within a test, can perform comparison of results with a reference
  - Want to provide some standard tools for this

The test code and scripts

- All tests live in the LArSoft / experiment repositories
  - For mrb test
    - <repository>/test/<package>/CMakeLists.txt
    - <repository>/test/<package>/<test script>
  - For stand-alone tests
    - <repository>/test/citest_<test name>.sh
    - <repository>/test/<package>/citest_<test name>.sh

Test script design rules

- Test scripts need to conform to the following rules
  - No arguments allowed for test script in CI system
    - Scripts can take arguments if appropriate defaults are defined
  - Cannot change the test environment (eg, by deleting outside the assigned test script area)
  - Exit status determines how test result is reported (pass / fail)
  - Provide descriptive output in case of failure
  - Tests should not rely on Jenkins
    - Must be able to run tests outside the CI system
  - Test should be capable of running on off-site machines
  - Stand-alone tests script names must be unique within a repository

Test script design rules

- Tests should assume
  - All setup's for code being tested (including patches) are already performed
  - Test script is running out of a standard sub-directory of working area
    - E.g., <working area>/test/<product>/<test name>
  - The environment contains information about the test workflow being run
  - Stdout will be shipped back to the server in a file <test name>.log.bz2
  - The environment will contain the location of local data areas
    - The Jenkins project will define this when run in the CI system
    - User must set this when run outside the CI system

Existing infrastructure and tests

- Have implemented only basic demonstrations
  - Use cases for mrb test
  - prodsingle_uboone.fcl as a stand-alone test
  - CPU times per module
    - Summary for a given run of the test
    - Storing history for comparison with previous tests

Currently all implemented in a single Jenkins project

- May not follow that convention for workflows in the final system

Will hear more about this during Mark's demo

Credits

- Central build / integration system
  - Build System Group
- Jenkins setup and configuration
  - Patrick Gartung + Build System Group
- Test framework and other test infrastructure
  - Mark Dykstra
  - Gianluca Petrillo
- Design
  - Mark Dykstra
  - Lynn Garren
  - Gianluca Petrillo
  - Erica Snider