Continuous Integration Project

Vito Di Benedetto for CI Project Team

FIFE Workshop

20th-21st June 2016

The Problem

- Bad habits in code development can break your code:

The Problem

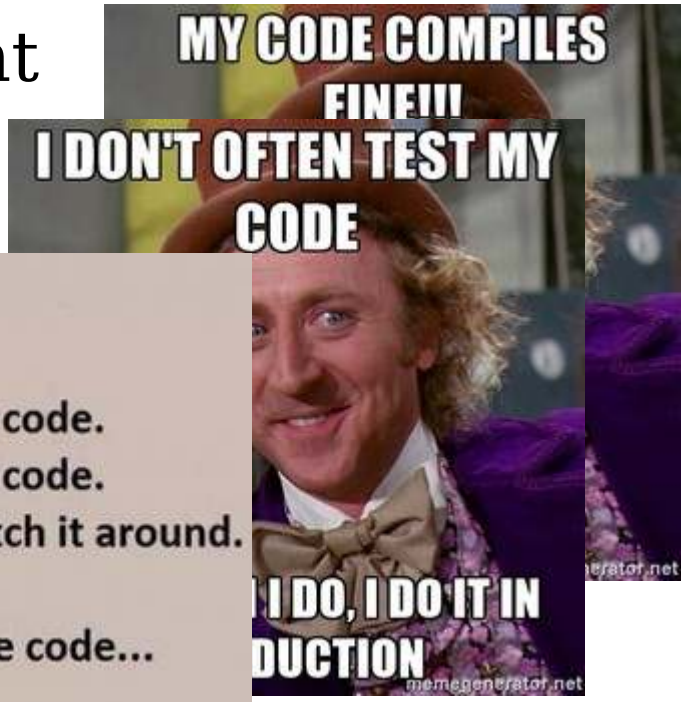
- Bad habits in code development can break your code:

The Problem

- Bad habits in code development can break your code:

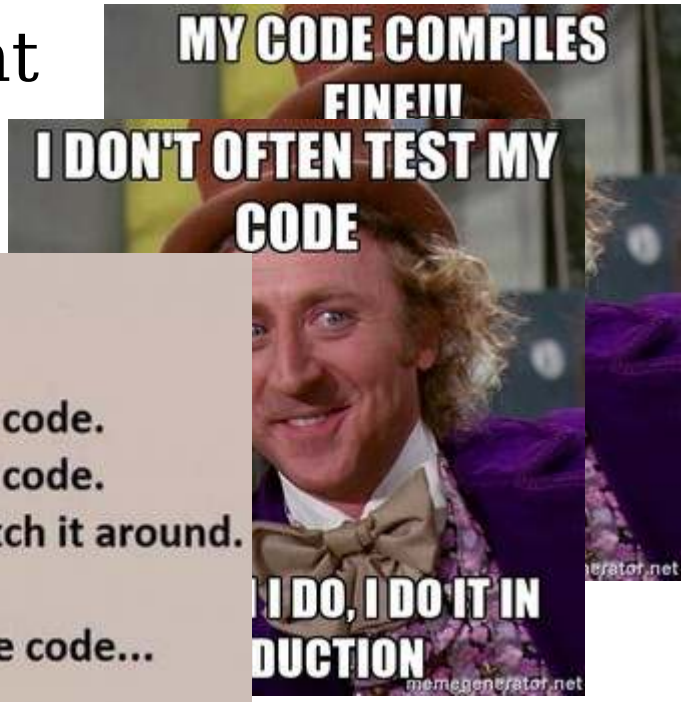
The Problem

- Bad habits in code development can break your code:



The Problem

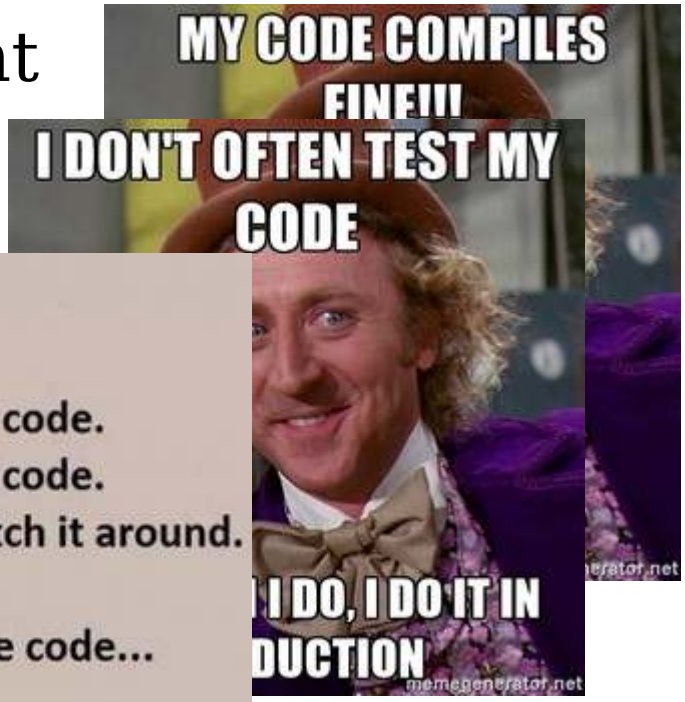
- Bad habits in code development can break your code:



- Sometime also good practice in code development can overlook some hidden bug…

The Problem

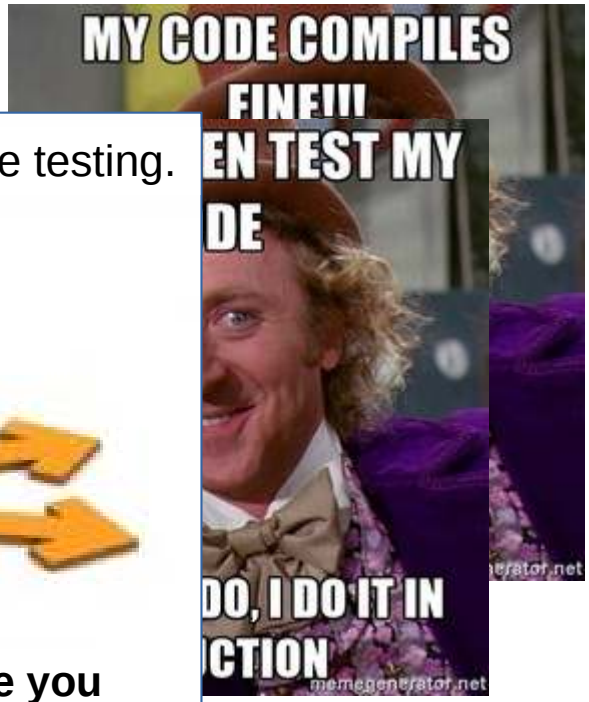
- Bad habits in code development can break your code:



- Sometime also good practice in code development can overlook some hidden bug...

The Problem

- Bad habits in code development can break y

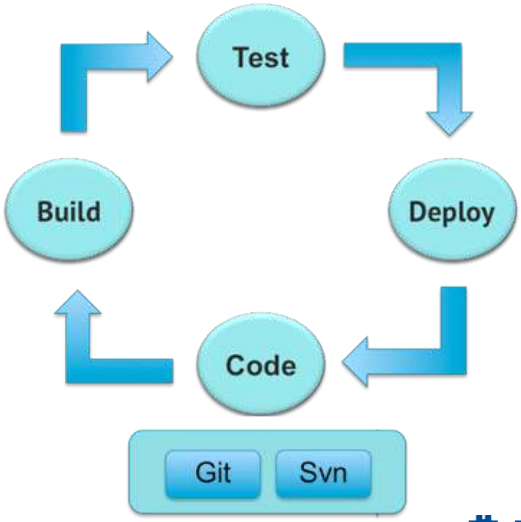


Keep on a straight path with proper code testing.

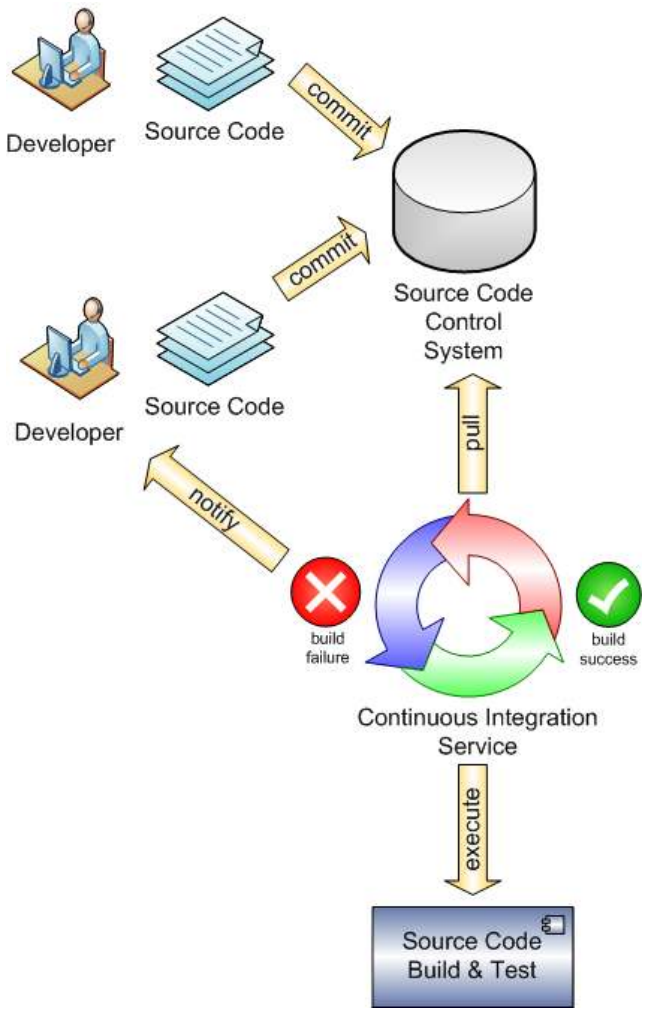
- Sometime a
  developmen
  bug...

Continuous Integration

- Continuous Integration (CI) is a development practice that requires developers to integrate their code into a shared repository.
- Each “commit” is verified by an automated build procedure that tests the code and allows teams to detect problems early, hopefully before the code goes in production.



Standard CI build job

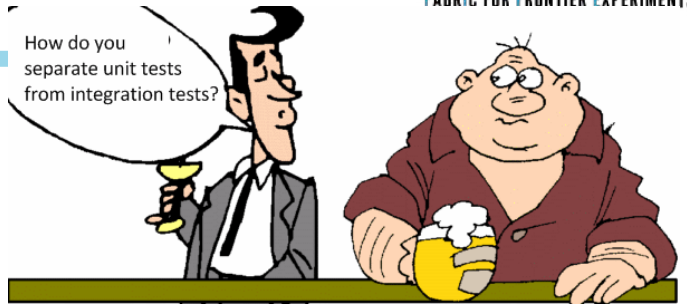


- Developers commit new code implementing bug fix, new feature, …
  - CI build job is triggered.
    - Pull the code from the repository.
    - Build the code.
    - Run unit tests.
    - Install the code.
    - Run CI tests.
  - Report the status of the CI build.
  - Notify developers in case of failure in the CI build caused by their commits.

CI build requirements

- A set of instructions to:
  - setup the build environment;
  - checkout the code;
  - build the code;
  - run unit tests;
  - install the code;
  - run integration tests.
- Recommended storage:
  - all package dependencies should live on CVMFS (this also allows to run the code on OSG sites);
  - all data files required by the CI build job should live in dCache;
  - eventually some package dependency can live in /grid/fermiapp (not recommended).

Unit and Integration tests



- Unit test
  - A unit test is an automated piece of code that invokes a single logical unit of the system and checks a single assumption about the behavior of that logical unit.
- Integration test
  - Integrates/combines the unit tested modules and tests the behavior as a combined unit;
  - its goal is to test the interfaces among the units/modules;
  - verifies that the (major) parts of a system work well together.
- Test features:
  - trustworthy (you don't need to debug the test code just to be sure of its results);
  - able to be fully automated (you want to run the test in an automatic procedure);
  - run fast (you want quick feedback);
  - independent (you want be able to run the tests in parallel, running concurrently as many test as the number of CPU cores available on the build node);
  - ...

CI test categories

- Regression test:
  - runs existing tests against modified code;
  - checks whether code changes break anything that worked prior to the change.
- Reproducibility test:
  - make sure that running the code using the same input, will “always” generate the same output.
- Back-compatibility test:
  - make sure that new code is able to access data files produced with a previous code release.
- Validation test:
  - make sure that new code produces meaningful results.
- ...

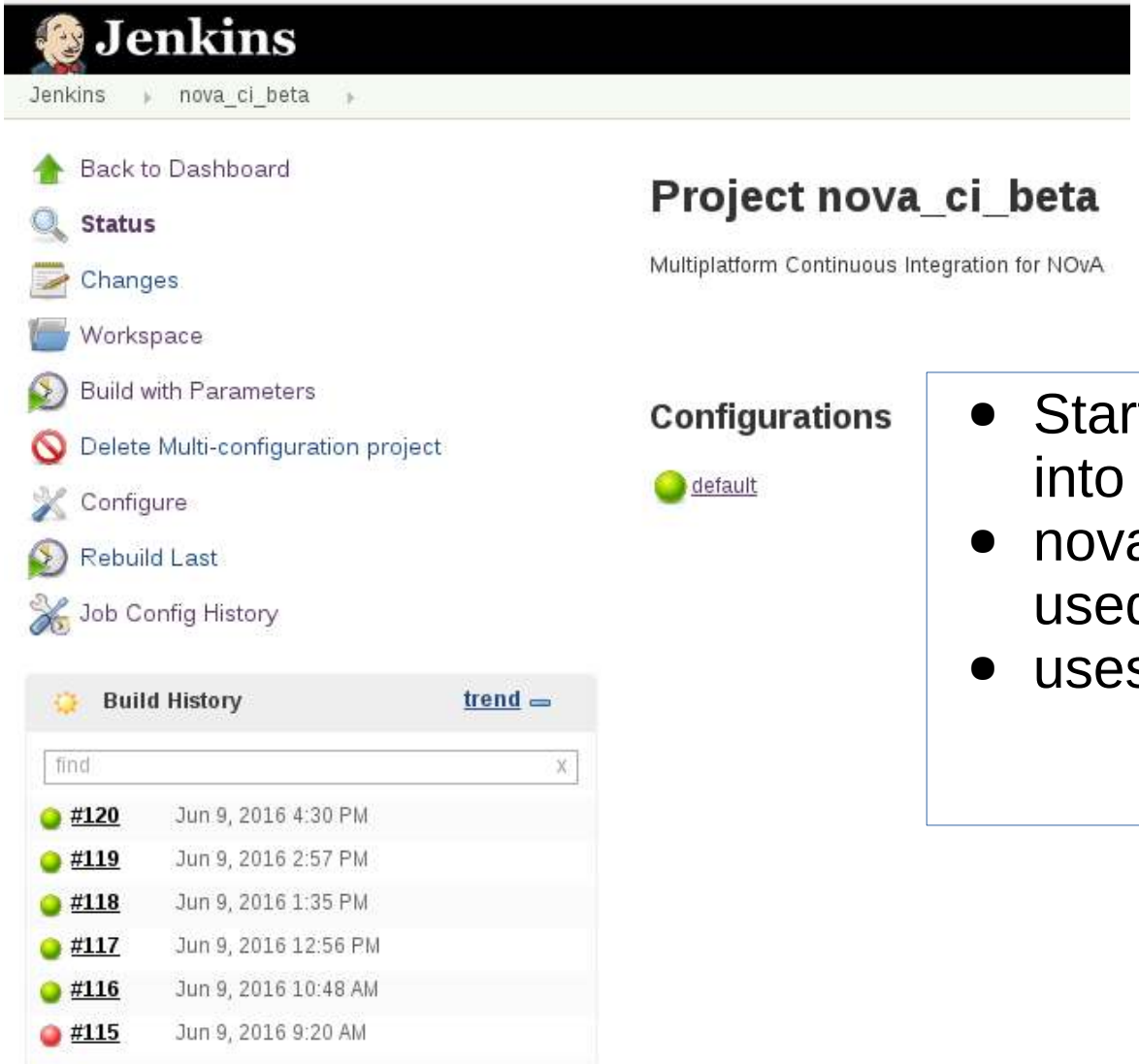
The CI Project

- Fermilab has already applied the Continuous practice to the LArSoft-based experiments. Experiments on-boarded in LAr CI are: μBooNE, DUNE, LArIAT and ArgoNeuT.
- The aim of the CI Project is to improve the existing tools and extend the CI service also to other IF experiments;
- the CI Project can help to have a healthy code at all times.

The CI Project

- Provides:
  - Jenkins project associated to the CI build;
  - repository with general script to handle CI builds;
  - repository for the experiment CI configuration files.
  - Web application for monitoring;
  - DB to collect/store statistics (build time, memory usage, ...);

The Jenkins project for NOvA CI



- Started to on-board NOvA into the CI Project;
- nova_ci_beta Jenkins project used for NOvA CI builds;
- uses SLF6 build slaves.

Work flow configuration example

cfg/workflow.cfg excerpt

```
[default]
personality=mrb
workflow=NOvA_CI
notify_email=vito@fnal.gov,mfattoru@fnal.gov
notify_success=true
notify_blame=false
build_db_uri=http://dbweb6.fnal.gov:8080/NovaCI/app
...
[mrb]
# define what the stages do:
# eval_n: setup the code environment
_eval1 = source /cvmfs/nova.opensciencegrid.org/externals/setups
_eval2 = . `ups setup ups`
_eval3 = setup mrb
...
#checkout: instruction to checkout the code
...
#build: instruction to build the code
...
# make_test:instruction to run unit tests
...
#install: instruction to install the code
...
#ci_tests: instruction to run the CI tests
...
[NOvA_CI]
stages  = _eval_n checkout_x_modules build make_test install ci_tests
modules = novaart.pkgs.svn
```

default configuration

"personality" configuration using MRB as build tool

workflow configuration

- The "***default*** configuration" selects the ***personality*** and the ***workflow*** to use.
- The "***personality*** configuration" defines the stages using a particular build tool.
- The "***workflow*** configuration" selects the stages to execute in the CI build, and the list of code modules to process.
- In the current implementation the NOvA CI workflow runs 6 stages: **_eval_n; checkout; build; make_test (unit tests); install and ci_tests** (disclaimer: unit test are not available for NOvA).

Test configuration example

test/ci_tests.cfg excerpt

```
[DEFAULT]
STEPS_NOVASOFT = raw raw2root reco calib
NOVA_REFERENCE_VERSION_NOVASOFT=default
BASEFILENAME_NOVASOFT=neardet_r00011552_s05
EXPCODE_NOVASOFT=novasoft
EXPSCRIPT_NOVASOFT=ci_regression_test_novasoft.sh
INPUTFILEDIR_NOVASOFT=/pnfs/nova/persistent/users/vito/ci_tests_inputfiles
```

Experiment specific section

```
[test ci_raw2root_regression_test_novasoft]
script=${NOVA_CI_DIR}/test/%(EXPSCRIPT_NOVASOFT)s
STEP=1
NEVENTS=1
TRIGGER=t00
SOURCE_STEP=raw
STREAM_LABEL=none
args=%(NEVENTS)s %(STEP)s %(NOVA_REFERENCE_VERSION_NOVASOFT)s %(BASEFILENAME_NOVASOFT)s_%(TRIGGER)s
%(EXPCODE_NOVASOFT)s %(SOURCE_STEP)s %(STREAM_LABEL)s %(STEPS_NOVASOFT)s
inputfiles=%(INPUTFILEDIR_NOVASOFT)s/%(BASEFILENAME_NOVASOFT)s_%(TRIGGER)s_Reference_%(SOURCE_STEP)s_
%(NOVA_REFERENCE_VERSION_NOVASOFT)s.raw %(INPUTFILEDIR_NOVASOFT)s/testmask.txt
%(INPUTFILEDIR_NOVASOFT)s/ci_test_raw2root_novasoft.fcl
...
```

CI test section

```
[suite default]
testlist=ci_raw2root_regression_test_novasoft ci_reco_regression_test_novasoft ci_calib_regression_test_novasoft
```

CI test suite section

- The “experiment specific section” initializes a set of variables required to initialize the script that runs the CI tests.
- The “CI test section” sets additional variables for the specific CI test.
- The “CI test suite section” collects a list of tests to run all together.
- In the current implementation there are **3 regression tests** running the **Raw2Root, Reconstruction** and **Calibration** NOvA processing.

Web application for CI monitoring

Useful to monitor past and current CI build status

[http://dbweb6.fnal.gov:8080/NovaCI/app/view_builds/index]

- shows the status of each stage of the CI workflow;
- shows also the status for individual CI tests using a tool-tip;
- the status of each CI stage and CI test is identified by a color code;
- each bullet in the matrix provides a link to the logs;
- the Web pulls information from the NOvA CI DB.

Multiplatform continuous integration for NOvA

Select builds:

From build:

# of builds:

Select platforms:

Linux 2.6.32-573.12.1.el6.x86_64

Linux 2.6.32-573.26.1.el6.x86_64

Select test suites:

default

Select date range:

From:

To:

Update

Build	Start Time	Platform	checkout	build	make_test	install	ci_tests
nova_ci_beta/110	2016-06-08 14:24:11.175166	Linux 2.6.32-573.12.1.el6.x86_64					
nova_ci_beta/109	2016-06-08 14:22:22.172627	Linux 2.6.32-573.12.1.el6.x86_64					
nova_ci_beta/108	2016-06-08 13:08:53.245479	Linux 2.6.32-573.12.1.el6.x86_64					
nova_ci_beta/107	2016-06-08 12:27:52.444923	Linux 2.6.32-573.12.1.el6.x86_64					
nova_ci_beta/106	2016-06-08 12:24:03.189076	Linux 2.6.32-573.26.1.el6.x86_64					
nova_ci_beta/105	2016-06-08 12:19:43.822078	Linux 2.6.32-573.26.1.el6.x86_64					
nova_ci_beta/104	2016-06-08 12:19:43.537685	Linux 2.6.32-573.12.1.el6.x86_64					
nova_ci_beta/103	2016-06-08 12:19:36.996948	Linux 2.6.32-573.12.1.el6.x86_64					
nova_ci_beta/102	2016-06-08 11:47:26.775068	Linux 2.6.32-573.12.1.el6.x86_64					
nova_ci_beta/101	2016-06-08 11:19:18.729286	Linux 2.6.32-573.12.1.el6.x86_64					

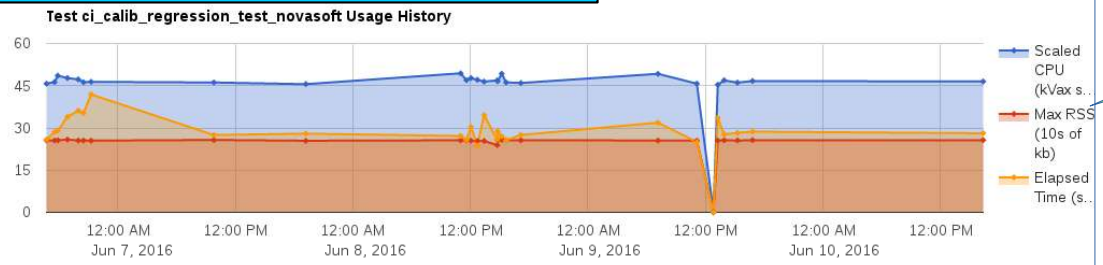
Progress Legend: Running, Pending, Succeeded, Failed, Skipped

CI monitoring statistics

Statistics from the NOvA CI "Calibration test"

ci_calib_regression_test_novasoft on EL6
Build nova_ci_beta/121
Trigger: trigger script by vito@novagpvm12.fnal.gov
Failed
Success

Memory usage



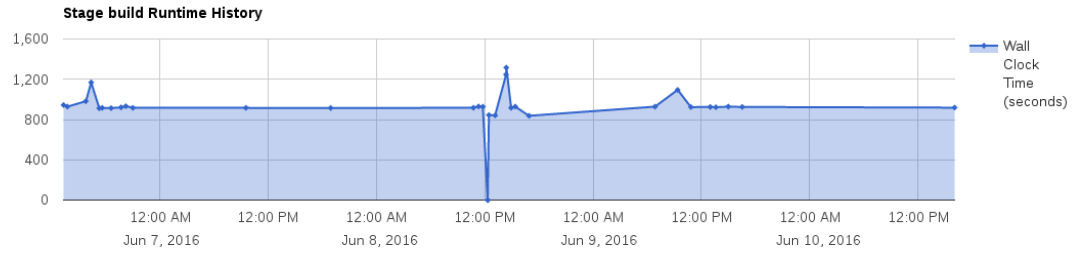
Test: ci_calib_regression_test_novasoft

stdout
stderr

Link to the logs

Registered 2016-06-10 16:20:27.469562	
Started 2016-06-10 16:20:27.832907	
exitcode	0
rusage_user_cpu	10.110000
rusage_scaled_user_cpu	46.506607
rusage_system_cpu	0.610000
rusage_scaled_system_cpu	2.806037
rusage_elapsed	28.090000
rusage_%cpu	38.000000
rusage_avgtext	0.000000
rusage_avgdata	0.000000
rusage_maxrss	262256.000000
rusage_inputs	65368.000000
rusage_outputs	1424.000000
rusage_major_faults	436.000000
rusage_minor_faults	50025.000000
rusage_swaps	0.000000
valerrs	0
success	True
Finished 2016-06-10 16:20:56.768274 exit code: 0.0	

build for EL6
Build nova_ci_beta/121, Trigger: trigger script by vito@novagpvm12.fnal.gov on EL6
status:
Success



Phase: build

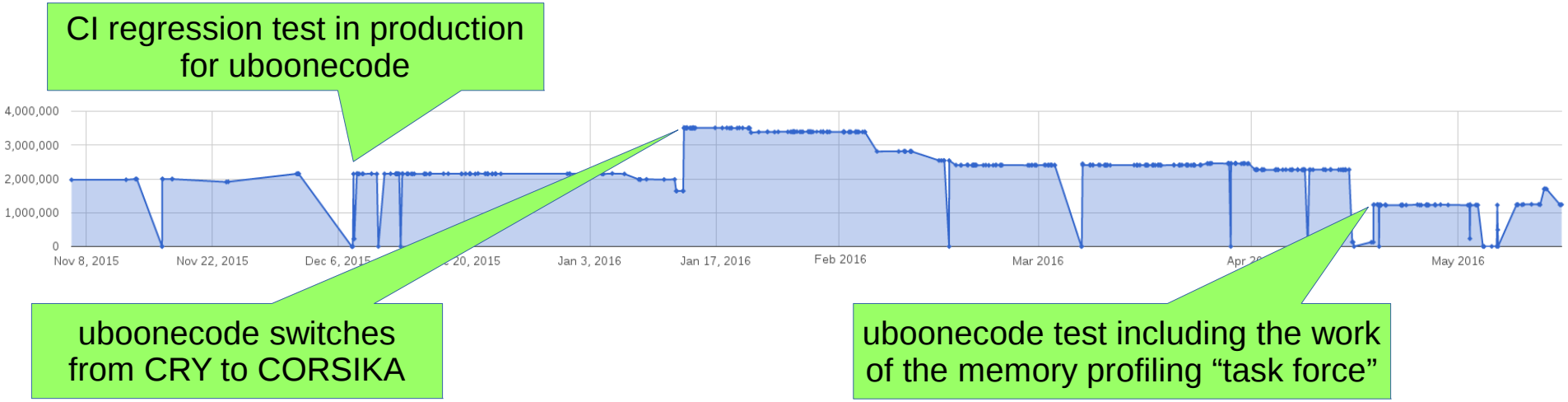
build
Started 2016-06-10 16:04:33.750498

Finished 2016-06-10 16:19:51.991690
exit code: 0

Statistics from the NOvA CI "build stage"

CI monitoring statistics - µBooNE

- Memory usage history plot: uboonecode geant4 stage as an example.



- Memory usage for 1 event.
- Using CORSIKA as cosmic shower generator, memory usage jumped from ~2Gb to ~3.5Gb.
- Over the time the memory usage got some improvement.
- After the intervention of a memory profiling “task force” the memory usage went down to ~1.2Gb.

Continuous Integration highlights

- CI will help you to have a healthy code at all times;
- CI workflow can handle code in ***git*** or ***svn*** repositories;
- CI workflow can build and test a list of interdependent modules altogether;
- user can test any desired branch/tag of the code;
- user can run CI tests locally using her/his own code just built;
- users can add/implement their own CI tests.
- Experiments will be the stakeholder;
- References:
  - the CI Project wiki

Are you interested in the CI service?

Experiments can require the CI service through SNOW:
Scientific Computing Services / Scientific Production Processing / Continuous Integration Service (working in progress)

- Basic requirements for the experiment code:
  - have a well defined and documented build chain;
    - possibly using MRB/cmake;
  - have all software dependencies as UPS products;
  - have all software dependencies available on CVMFS;
  - have all needed accessory files (flux files, ...) on dCache.

Future plans

- The CI Project Team is glad to provide the CI service to IF experiments.
- The CI practice has already been successfully adopted by LArSoft-based experiments;
- NovA, the first not LArSoft-based experiments, has just been on-boarded;
- the plan is to on-board all IF experiments;
  - CI service will provide a software facility to constantly monitor the status of the experiment code;
    - will help to maintain a healthy code;
    - will help to monitor resource usage;
    - will help to monitor code performances;

Acknowledgment

- Thanks to Brian Rebel
  - for fixing issues found by NOvA CI build in the NOvA code;
- and to Pengfei Ding
  - for pointing out the documentation for the build of NovA code.

Thank you!