Code Coverage and Continuous Integration

Presented to
ATPESC 2017 Participants

Alicia Klinvex
Sandia National Laboratories

Q Center, St. Charles, IL (USA)
Date 08/09/2017

Code Coverage

How do we determine what other tests are needed?

- Code coverage tools
  - Expose parts of the code that aren't being tested
  - gcov
    - standard utility with the GNU compiler collection suite
    - counts the number of times each statement is executed
  - lcov
    - a graphical front-end for gcov
    - available at http://ltp.sourceforge.net/coverage/lcov.php

How to use gcov/lcov

- Compile and link your code with --coverage flag
  - It's a good idea to disable optimization
- Run your test suite
- Collect coverage data using gcov/lcov
- Optional: generate html output using genhtml

A hands-on gcov tutorial

- https://amklinv.github.io/morpheus/index.html

But I don’t use C++!

- gcov also works for C and Fortran
- Other tools exist for other languages
  - JCov for Java
  - Coverage.py for python
  - Devel::Cover for perl
  - profile for MATLAB
  - etc

Continuous integration

Continuous integration (CI): a master branch that always works

- Code changes trigger automated builds/tests on target platforms
- Builds/tests finish *in a reasonable amount of time*, providing useful feedback when it's most needed
- Immensely helpful!
- Requires some work, though:
  - A reasonably automated build system
  - An automated test system with significant test coverage
  - A set of systems on which tests will be run, and a controller

Continuous integration (CI): a master branch that always works

- Has existed for some time
- Adoption has been slow
  - Setting up and maintaining CI systems is difficult, labor-intensive (typically requires a dedicated staff member)
  - *You have to be doing a lot of things right to even consider CI*

Cloud-based CI is available as a service on GitHub

- Automated builds/tests can be triggered via pull requests
- Builds/tests can be run on cloud systems – no server in your closet. *Great use of the cloud!*
- Test results are reported on the pull request page (with links to detailed logs)
- Already being used successfully by scientific computing projects, with noticeable benefits to productivity
- Not perfect, but *far* better than not doing CI

Travis CI is a great choice for HPC

- Integrates easily with GitHub
- *Free* for Open Source projects
- Supports environments with C/C++/Fortran compilers (GNU, Clang, Intel[?])
- Linux, Mac platforms available
- *Relatively* simple, *reasonably* flexible configuration file
  - Documentation is sparse, but we now have working examples

Travis CI live demo

- https://github.com/amklinv/morpheus