Testing of HPC Scientific Software Introduction

Presented at

Better Scientific Software tutorial

SC17, Denver, Colorado

Alicia Klinvex
Sandia National Laboratory

NNSA National Nuclear Security Administration
U.S. DEPARTMENT OF ENERGY | Office of Science
IDEAS productivity

License, citation and acknowledgements

License and Citation

- This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
- Requested citation: Alicia Klinvex, Testing of HPC Scientific Software: Introduction, tutorial, in SC '17: International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, Colorado, 2017. DOI: 10.6084/m9.figshare.5593342.

Acknowledgements

- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This work was performed in part at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND NO SAND2017-5474 PE

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Outline

3

- Introduction
- Scientific software verification
- How to evaluate needs of a project and devise a testing regime
- Demo - Code coverage and continuous integration

Introduction

Why is testing important?

Definitions

Benefits of testing

- Promotes high-quality software that delivers correct results and improves confidence
- Increases quality and speed of development, reducing development and maintenance costs
- Maintains portability to a variety of systems and compilers
- Helps in refactoring
  - Avoid introducing new errors when adding new features
  - Avoid reintroducing old errors

How common are bugs?

Programs do not acquire bugs as people acquire germs, by hanging around other buggy programs. Programmers must insert them.
- Harlan Mills

- Bugs per 1000 lines of code (KLOC)
- Industry average for delivered software
  - 1-25 errors
- Microsoft Applications Division
  - 10-20 defects during in-house testing
  - 0.5 in released product

Code Complete (Steven McConnell)

Why testing is important: the protein structures of Geoffrey Chang

- Some inherited code flipped two columns of data, inverting an electron-density map
- Resulted in an incorrect protein structure
- Retracted 5 publications
  - One was cited 364 times
- Many papers and grant applications conflicting with his results were rejected

Why testing is important: the 40 second flight of the Ariane 5

- Ariane 5: a European orbital launch vehicle meant to lift 20 tons into low Earth orbit
- Initial rocket went off course, started to disintegrate, then self-destructed less than a minute after launch
- Seven variables were at risk of leading to an Operand Error (due to conversion of floating point to integer)
  - Four were protected
- Investigation concluded insufficient test coverage as one of the causes for this accident
- Resulted in a loss of $370,000,000.

Why testing is important: the Therac-25 accidents

- Therac-25: a computer-controlled radiation therapy machine
- Minimal software testing
- Race condition in the code went undetected
- Unlucky patients were struck with approximately 100 times the intended dose of radiation, ~ 15,000 rads
- Error code indicated that no dose of radiation was given, so operator instructed machine to proceed
- Recalled after six accidents resulting in death and serious injuries

Definitions

- Unit tests
  - Test individual functions or classes
- Integration tests
  - Test interaction, build complex hierarchy
- System level tests
  - At the user interaction level

Definitions

- Restart tests
  - Code starts transparently from a checkpoint
- Regression (no-change) tests
  - Compare current observable output to a gold standard
- Performance tests
  - Focus on the runtime and resource utilization

Policies on testing practices

- Avoid regression suites consisting of system-level no-change tests
  - Tests often need to be re-baselined
    - Often done without verification of new gold-standard
  - Hard to maintain across multiple platforms
  - Loose tolerances can allow subtle defects to appear

Policies on testing practices

- Must have consistent policy on dealing with failed tests
  - Issue tracking
    - How quickly does it need to be fixed?
    - Who is responsible for fixing it?
  - Add regression test afterwards (to avoid reintroducing issue later)
- Someone needs to be in charge of watching the test suite

Policies on testing practices

- When refactoring or adding new features, run a regression suite before checkin
  - Be sure to add new regression tests for the new features
- Require a code review before releasing test suite
  - Another person may spot issues you didn't
  - Incredibly cost-effective

Example: Trilinos checkin test script

- Detects which packages were modified by your commits
- Determines which packages you potentially broke
- Configures, builds, and tests those packages
  - On success, pushes to repo
  - On failure, reports why it failed
- Useful for ensuring your changes don't break another package
- May take a while, but many people run it overnight

Maintenance of a test suite

- Testing regime is only useful if it is
  - Maintained
    - Tests and benchmarks periodically updated
  - Monitored regularly
    - Can be automated
  - Has rapid response to failure
    - Tests should pass most of the time

Use of test harnesses

- Essential for large code
  - Set up and run tests
  - Evaluate test results
- Easy to execute a logical subset of tests
  - Pre-push
  - Nightly
- Automation of test harness is critical for
  - Long-running test suites
  - Projects that support many platforms

Jenkins
C-dash
Custom
(FlashTest)

Example: Trilinos automated testing

Login All Dashboards — Monday, June 06 2016 08:58:08 MDT

Trilinos

Dashboard | Calendar | Previous | Current | Project

Project

Project	Configure			Build			Test		
	Error	Warning	Pass	Error	Warning	Pass	Not Run	Fail	Pass
Trilinos	1	531	530	0	272	257	0	14	3976

SubProjects

Project	Configure			Build			Test		
	Error	Warning	Pass	Error	Warning	Pass	Not Run	Fail	Pass
Teuchos	0	21	21	0	12	9	0	0	227
ThreadPool	0	1	1	0	0	1			
Sacado	0	2	2	0	2	0	0	0	564
RTOp	0	20	20	0	0	20			
Kokkos	0	19	19	0	0	19	0	0	9
Epetra	0	21	21	0	12	9	0	1	244
Zoltan	0	21	21	0	13	8	0	0	135
Shards	0	1	1	0	0	1			
GlobiPack	0	1	1	0	0	1			

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Agenda

Tutorial evaluation form: http://bit.ly/sc17-eval

Time	Topic	Speaker
8:30am-8:45am	Why effective software practices are essential for CSE projects	David E. Bernholdt, ORNL
8:45am-9:15am	Introduction to software licensing	David E. Bernholdt, ORNL
9:15am-9:45am	Better (small) scientific software teams	Michael A. Heroux, SNL
9:45am-10:00am	Improving Reproducibility Through Better Software Practices	Michael A. Heroux, SNL
10:00am-10:30am	*Break*	
10:30am-10:45am	Testing of HPC Scientific Software: Introduction	Alicia M. Klinvex, SNL
10:45am-11:15am	Verification	Anshu Dubey, ANL
11:15am-11:45am	Evaluating project testing needs	Anshu Dubey, ANL
11:45am-12:00pm	Code coverage demo and CI demo	Alicia M. Klinvex, SNL

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT