Testing of HPC Scientific Software
Part 2

Presented at
Better Scientific Software tutorial

ISC18, Frankfurt, Germany

Anshu Dubey
Computer Scientist, Argonne National Laboratory
University of Chicago

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

License, citation and acknowledgements

License and Citation

- Requested citation: Anshu Dubey, Testing of HPC Scientific Software: Part 2, tutorial, in ISC High Performance 2018: DOI: 10.6084/m9.figshare.6453020.

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 the Argonne National Laboratory, which is managed by UChicago Argonne, LLC for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357.

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

How to evaluate project needs

And devise a testing regime

Why not always use the most stringent testing?

- Effort spent in devising tests and testing regime are a tax on team resources
- When the tax is too high…
  - Team cannot meet code-use objectives
- When is the tax is too low…
  - Necessary oversight not provided
  - Defects in code sneak through

Evaluating project needs

- Objectives: expected use of the code
- Team: size and degree of heterogeneity
- Lifecycle stage: new or production or refactoring
- Lifetime: one off or ongoing production
- Complexity: modules and their interactions

Commonalities

- Unit testing is always good
  - It is never sufficient
- Verification of expected behavior
- Understanding the range of validity and applicability is always important
  - Especially for individual solvers

Challenges with legacy codes

Checking for coverage

- Legacy codes can have many gotchas
  - Dead code
  - Redundant branches
- Interactions between sections of the code may be unknown
- Can be difficult to differentiate between just bad code, or bad code for a good reason
  - Nested conditionals

Code coverage tools are of limited help

CSE specific verification challenges

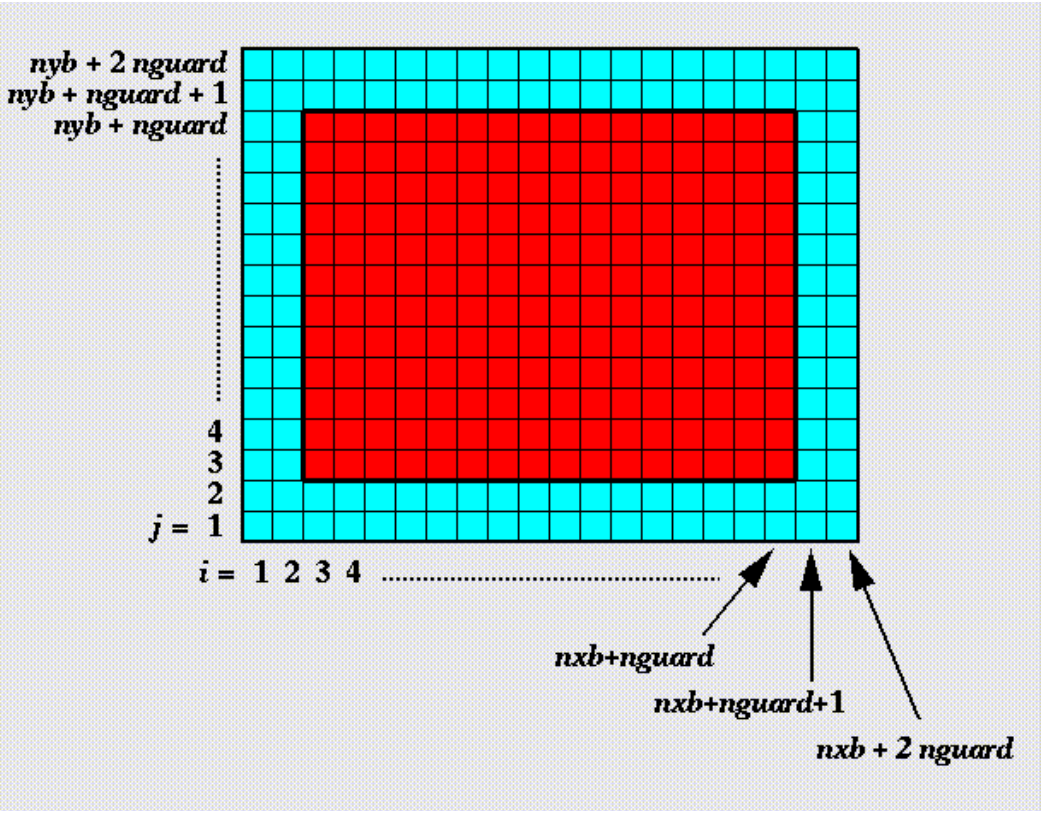
- Integration testing may have hierarchy too
- Particularly true of codes that allow composability in their configuration
- Codes may incorporate some legacy components
  - Its own set of challenges
    - No existing tests at any granularity
- Examples – multiphysics application codes that support multiple domains

Selection of tests

- Two purposes
  - Regression testing
    - May be long running
    - Provide comprehensive coverage
  - Continuous integration
    - Quick diagnosis of error
- A mix of different granularities works well
  - Unit tests for isolating component or sub-component level faults
  - Integration tests with simple to complex configuration and system level
  - Restart tests
- Rules of thumb
  - Simple
  - Enable quick pin-pointing

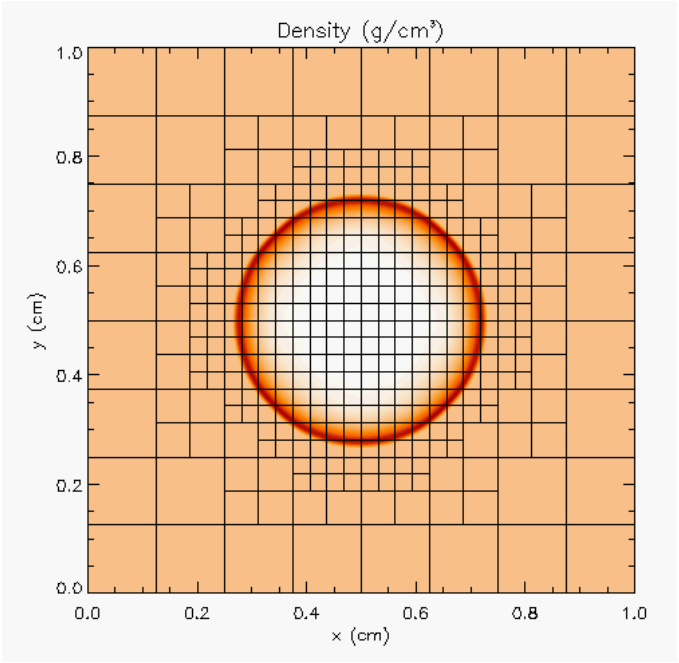
Example from Flash

- Verification of guard cell fill
- Use two variables A & B
- Initialize A including guard cells and B excluding them
- Apply guard cell fill to B
- Eos
  - Use initial conditions from a known problem
  - Apply eos in two different modes – at the end all variables should be consistent within tolerance



Against analytical solution

- Sedov blast wave
- High pressure at the center
- Shock moves out spherically
- FLASH with AMR and hydro
- Known analytical solution



Though it exercises mesh, hydro and eos, if mesh and eos are verified first, then this test verifies hydro

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Building confidence

- First two unit tests are stand-alone
- The third test depends on Grid and Eos
  - Not all of Grid functionality it uses is unit tested
    - Flux correction in AMR
- If Grid and Eos tests passed and Hydro failed
  - If UG version failed then fault is in hydro
  - If UG passed and AMR failed the fault is likely in flux correction

Approach for Test Selection

- Build a matrix
  - Physics along rows
  - Infrastructure along columns
  - Alternative implementations, dimensions, geometry
- Mark <i,j> if test covers corresponding features
- Follow the order
  - All unit tests – including full module tests
  - Tests representing ongoing productions
  - Tests sensitive to perturbations
  - Most stringent tests for solvers
  - Least complex test to cover remaining spots

Example

	Hydro	EOS	Gravity	Burn	Particles
AMR	CL	CL		CL	CL
UG	SV	SV			SV
Multigrid	WD	WD	WD	WD	
FFT			PT		

Tests	Symbol
Sedov	SV
Cellular	CL
Poisson	PT
White Dwarf	WD

- A test on the same row indicates interoperability between corresponding physics
- Similar logic would apply to tests on the same column for infrastructure
- More goes on, but this is the primary methodology

Refactoring

Testing needs during code refactor

Considerations

- Know bounds on acceptable behavior change
- Know your error bounds
  - Bitwise reproduction of results unlikely after transition
- Map from here to there
- Check for coverage provided by existing tests
- Develop new tests where there are gaps

Incorporate testing overheads into refactor cost estimates

An Approach

- Isolate a small area of the code
- Dump a useful state snapshot
- Build a test driver
  - Start with only the files in the area
  - Link in dependencies
    - Copy if any customizations needed
- Read in the state snapshot
- Verify correctness
  - Always inject errors to verify that the test is working

Methodology developed for the ACME project, proving to be very useful

Agenda

Time	Topic	Speaker
2:00pm-2:30pm	Why Effective Software Practices are Essential for CSE Projects	Anshu Dubey, ANL
2:30pm-3:00pm	Introduction to Software Licensing	Michael A. Heroux, SNL
3:00am-3:30pm	Better (small) Scientific Software Teams	Michael A. Heroux, SNL
3:30am-4:00pm	Improving Reproducibility Through Better Software Practices	Michael A. Heroux, SNL
4:00pm-4:30pm	*Break*	
4:30pm-5:00pm	Testing HPC Scientific Software – Part 1	Anshu Dubey, ANL
5:00pm-5:30pm	Testing HPC Scientific Software – Part 2	Anshu Dubey, ANL
5:30pm-6:00pm	Code Coverage Hands-on and CI Demo	Anshu Dubey, ANL