Advanced Testing for RELAP5-3D

Dr. George Mesina

RELAP5 International Users Seminar

Date: Oct 23-24, 2012

Introduction

- Goal of Testing
- Platforms
- Language
- Method
- Results

Goals of Testing

- The ideal for testing:
  To produce a bug-free computer code for use by the nuclear industry

- The achievable goal for testing:
  To find and fix every bug that INL's Standard Test Suite can reveal *before* releasing a RELAP5-3D product.
  - To make a better product, more tests are continually added
  - Testing reveals error that must be resolved.
  - The cycle of testing, debugging, fixing and retesting is time-consuming

- The Project Goal for testing: To create better testing methodology
  - So it takes less time. Then it can be done more often.
  - So it allows test suite expansion with little time increase.

RELAP5-3D Test Suites

Test Suite	Problems/Directories	Cases
Standard Installation	Normal, Athena, Other	104
Additional Feature	Pvm, Extra, FlexWall, MError, MStable	127
DOE Requested	Complex longer-running and DOE-specific cases	35
DA Set	Developmental Assessment cases	104
DTSTEP Test Matrix	PVM-DTSTEP interaction	> 2000

- The Standard Installation set is run before internal releases
- Except for some “Additional Feature” tests, all tests are run before a product release.

Computer Platforms

- Current testing methodology for RELAP5-3D
  - Run collection of test cases in serial mode on Linux or Windows workstations.
- Most platforms now have multiple cores
  - Running cases simultaneously on individual cores reduces testing time.
- Massively parallel platforms can run many test cases simultaneously.
  - This has already been implemented with the DTSTEP Test Matrix
  - 3.5 hours on workstation decreased to 3.5 minutes on 7 nodes.
    - Takes longer when fewer nodes available.

"Monty" Python Scripting Language

- Python is a powerful scripting language used on the INL clusters.
- It has many useful and powerful features (from their advertising):

Feature	Description
Software quality	Python's focus is readability, coherence, and software quality.
Developer productivity	It boosts developer productivity many times beyond compiled languages.
Program portability	Most Python programs run unchanged on all major platforms.
Support Libraries	It comes with its *standard library*, a large collection or pre-built & portable functionality.
Component integration	Python scripts can easily communicate with other parts of an application using a variety of integration mechanisms.

Parallel Method: High-level Description

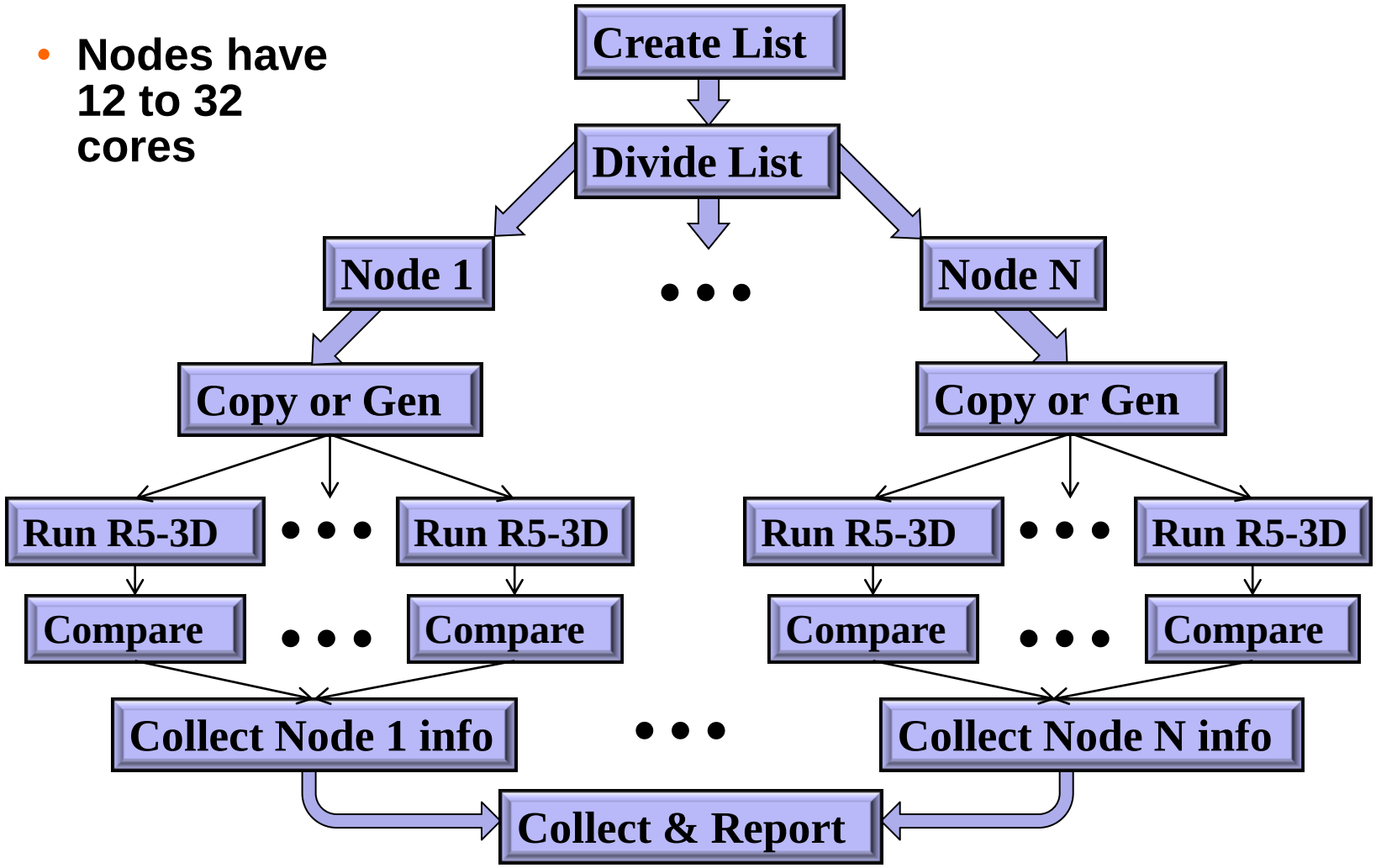
- User collects test cases into staging area

Python script does the following:

- Create a list of all test cases in staging area
- Divide list among compute nodes
- On each node, form RELAP5-3D execution commands for the tests
  - Fray: When a core of the node becomes free, it runs the next command
  - Run in temporary storage (its faster)
  - Handle failures properly
  - Compare to previous run (if available)
  - Collect execution statistics
- Collect information and report

Parallel Testing

- Nodes have 12 to 32 cores



How Testing Method Works

- Invoke massive parallelism through the *Portable Batch System* (PBS)
  - The forkmap feature creates the fray
- Python script BuildRunRep implements the method
  - Copies input files from staging area to a /tmp target directory
  - Also decompresses TGZ-files
- It handles base case and restart runs separately
  - Special handling is required for restart cases

Advantages of the New Method

- The method expands to any number of test cases
  - Merely place test case directories in the staging area
- Time required for entire test typically equals time for longest test case
  - The cases run in a fray.
  - Cores that run time-consuming cases seldom get second test case
- Tested BuildRunRep on INL clusters
  - In serial on Eos (Dell, 256GB, 3 nodes, 72 cores)
  - In parallel on Quark (Intel Xeon, 32 24-GB nodes, 12cores/node)
- Recent test on Quark
  - 2300 runs done in 9 minutes
  - Longest single run just over 8 minutes

Usage on Cluster

- Create “staging area” directory
  - Put in original copies of necessary files:
  - relap5.x, fluid files, subdirectories of input files, testing scripts
- Clean out temporary storage working area: /tmp/relap/
- Load all necessary enclave modules
  - Python, PBS, and PVM (if testing the coupling)
- Select the number of nodes for the test. Typically
  - 7 nodes for DTSTEP test matrix
  - 3 nodes for the rest
- Submit the run via PBS

Conclusions

- A new method for testing RELAP5-3D on numerous test cases has been devised
- The method will expand to any number of test cases
- The amount of time it takes to run the longest test case is typically the time required for the entire suite
- This has been implemented and tested on the INL cluster