Using Docker for High Energy Physics

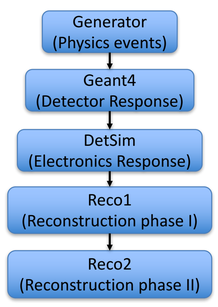
Jim Kowalkowski, Adam Lyon, and Marc Paterno / Fermilab

Abstract

Docker is a container technology that provides a way to "wrap up a piece of software in a complete filesystem that contains everything it needs to run" [1]. We have experimented with Docker to investigate its utility in three broad realms: (1) allowing existing complex software to run in very different environments from that in which the software was built (such as Cori, NERSC's newest supercomputer), (2) as a means of delivering the same development environment to multiple operating systems (including laptops), and allowing the use of tools from both the host and container system to their best advantage, and (3) as away of encapsulating entire software suites (in particular, a popular cosmology-based Markov Chain Monte Carlo parameter estimation system), allowing them to be supported for use on multiple operating systems without additional effort.

[1] "What is Docker?", https://www.docker.com/what-docker.

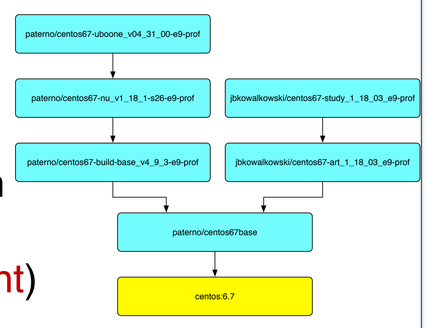
Running Existing Code



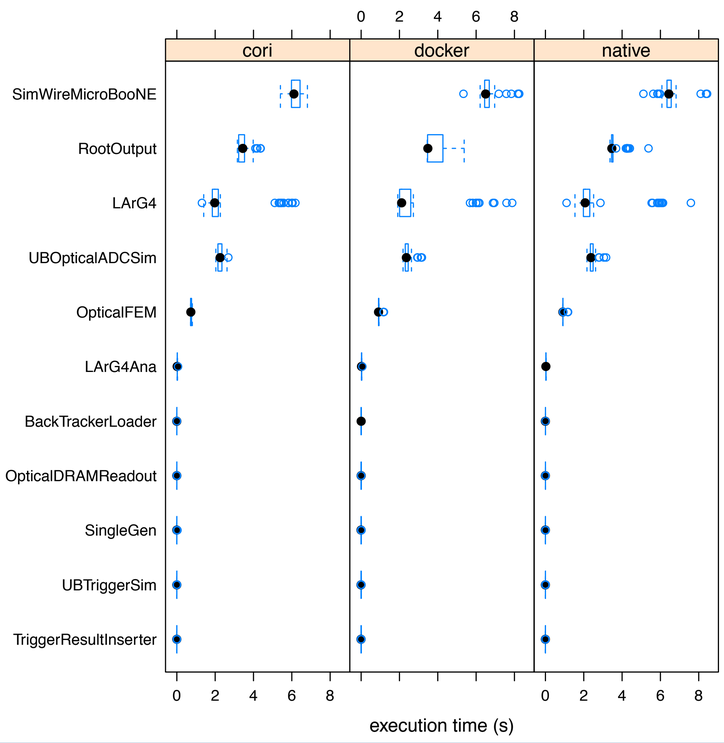
MicroBooNE experiment simulation:

- 11 framework modules (experiment-specific libraries of code)
- includes use of Geant4 detector simulation software and ROOT i/o
- same code as run in their continuous integration tests

- Code natively built on SLF6
- Build images on SLF6; upload to DockerHub
- Install on our local resources with docker pull; on Cori with shifterimg pull
- Run containers using Docker's (and Shifter's) ability to mount host filesystem in running container through command-line options (--volume=$SCRATCH/test.out:/mnt)



- Performance comparison between processors
  - Native: Intel E52680v2, 2.8 GHz, Ivy Bridge
  - Docker: Intel E52680v2, 2.8 GHz, Ivy Bridge
  - Cori: Intel E52698v3, 2.3 GHz, Haswell
- Median time per event about 0.5% slower with the docker container

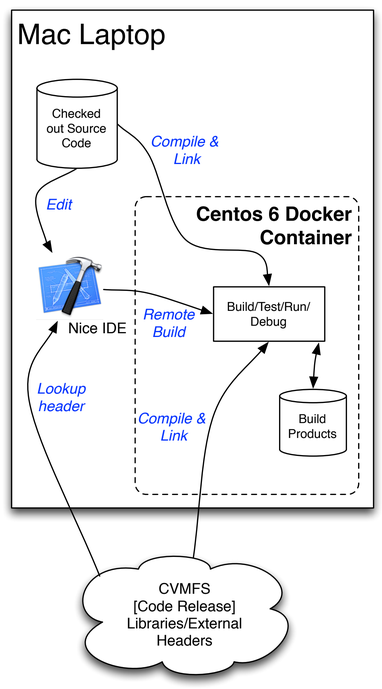


Hybrid development environment

- Make releases for OSX, Scientific Linux (Centos) 6 & 7, and Ubuntu 14 & 16
- Jenkins is great for multiplatform builds, but sometimes need to build and debug on particular platform
- Docker containers are excellent solutions to develop, build, test, run, debug on platform that does not match personal machine
- Docker for Mac makes Docker easy to install, run and maintain on Macs
- Container monitoring useful for performance/memory debugging and optimization

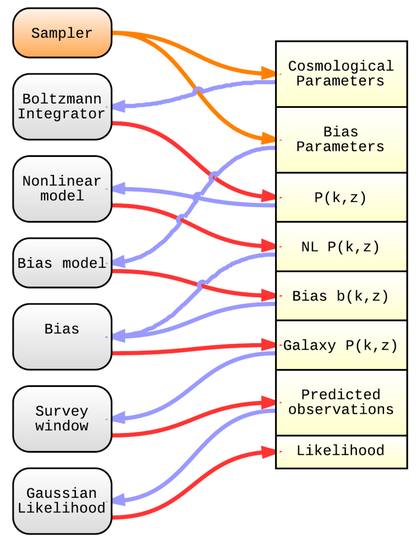
Example: Mac laptop with Centos 6 Docker container

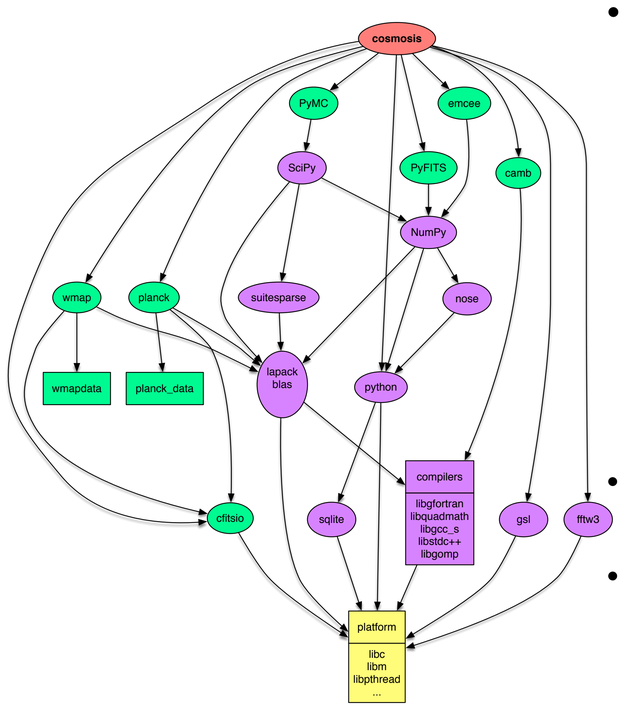
- Note use of host volumes
- Nice IDE on Host may be used to develop code (many can trigger a remote build)
- Docker commands are complicated: need scripts



Migrating to a native docker distribution

- Parameter estimation package with focus on modularity
- Brings together & connects existing code wrapped into modules
- Plug in architecture - easy replacement/addition of code
- Multi-language modules
- Choice of physics & likelihood modules, samplers





- Requirement: must build software systems with modern C++ language features
  - Old way: Satisfied by building everything with a version of compiler that we build and deliver
  - New way: construct a container with the system compiler that we pick, available in a standard system location. The system installer now works out-of-the box! (yum, apt, even Python's pip)
- Works with everything from laptops to supercomputers
- 3rd party software integration is now easy because system tools are used directly