Using containers with ATLAS offline software

Marcelo Vogel
on behalf of the ATLAS Collaboration

- Why Docker? Efficiently ship & quickly launch applications on cloud computing resources
- What is Docker? Software-based environment and resource isolation technology
- How does it work? Processes run natively (but isolated) on minimally configured hosts

Docker is not a virtual Machine

Docker is not a Virtual Machine, which simulates both hardware and software. Docker will share the OS kernel, but isolate different environments ("containers") with respect to data, code, networking, etc.

Docker images: archived, executable file system snapshots

Images are portable, self-contained, reproducible software environments. Includes full dependencies (OS, libraries, user code)

Docker images are built from layers

A container is an instance of an image, which starts as a single process in the environment packaged in the image

Dockerfile: automatic image building

Images can be built automatically by Docker if provided with a Dockerfile, which is basically a set of instructions to be executed in command-line fashion (Maintained at gitlab.cern.ch)

ATLAS software in Docker images

The new repository for ATLAS computing documentation has a new section for working with Docker containers

Relative performance of containers running ATLAS software

Simple tests comparing Wall-time and CPU-time in reconstruction jobs using the light and full-release images. Docker is tested in two platforms frequently used by developers: macOS and Ubuntu

Latest version of Docker for Mac, version 17.06.0 (stable), Docker version 17.05.0 in Ubuntu.

Ubuntu 14.04 desktop, Intel Core 2 DUO CPU E7500, 2.93GHz (CVMFS 2.3.5). macOS Sierra 10.12.6, MacBook Pro (Retina, 15-inch, Mid 2014), i7 2.2 GHz (CVMFS 2.3.3, FUSE 3.5.4)

Reconstruction test job (q431, raw to esd) on 25 data events (data16_13TeV, physics_Main) in Athena, 21.0.33. A full-simulation test job is also used (5 events)

Consistency in bind mounts

"Benchmark" script: testMergeCont.C, a tutorial ROOT macro from ROOT 6.08.02, is used to evaluate the "speed" of different setups. It creates random histograms on a canvas, stores them in a ROOT file and reads them back

Comparing "speeds" in Reconstruction and Simulation

General remarks

Significant degradation of performance in Docker for Mac due to slow I/O, specially when all software is accessed through cvmfs

Slow I/O in Docker for Mac is largely mitigated by relaxing consistency checks in the bind mount, and by using full-release images

The performance of general reconstruction jobs is comparable to that achieved in SLC6 systems on raw metal