EPJ Web of Conferences **245**, 07010 (2020) https://doi.org/10.1051/epjconf/202024507010
CHEP 2019

Standalone containers with ATLAS offline software

Marcelo Vogel[1,*], *Mikhail* Borodin[2], *Alessandra* Forti[3], and *Lukas* Heinrich[4]

[1]Faculty of Mathematics and Natural Sciences, University of Wuppertal, Gausstrasse 20, 42119 Wuppertal, Germany

[2]Department of Physics and Astronomy, University of Iowa, 203 Van Allen Hall, Iowa City, IA 52242-1479, USA

[3]School of Physics and Astronomy, University of Manchester, Oxford Road, Manchester M13 9PL, UK

[4]European Organization for Nuclear Research (CERN), CH-1211 Geneve 23, Switzerland

Abstract. This paper describes the deployment of the offline software of the ATLAS experiment at LHC in containers for use in production workflows such as simulation and reconstruction. To achieve this goal we are using Docker and Singularity, which are both lightweight virtualization technologies that can encapsulate software packages inside complete file systems. The deployment of offline releases via containers removes the interdependence between the runtime environment needed for job execution and the configuration of the computing nodes at the sites. Docker or Singularity would provide a uniform runtime environment for the grid, HPCs and for a variety of opportunistic resources. Additionally, releases may be supplemented with a detector's conditions data, thus removing the need for network connectivity at computing nodes, which is normally quite restricted for HPCs. In preparation to achieve this goal, we have built Docker and Singularity images containing single full releases of ATLAS software for running detector simulation and reconstruction jobs in runtime environments without a network connection. Unlike similar efforts to produce containers by packing all possible dependencies of every possible workflow into heavy images (≈200GB), our approach is to include only what is needed for specific workflows and to manage dependencies efficiently via software package managers. This approach leads to more stable packaged releases where the dependencies are clear and the resulting images have more portable sizes (≈16GB). In an effort to cover a wider variety of workflows, we are deploying images that can be used in raw data reconstruction. This is particularly challenging due to the high database resource consumption during the access to the experiment's conditions payload when processing data. We describe here a prototype pipeline in which images are provisioned only with the conditions payload necessary to satisfy the jobs' requirements. This database-on-demand approach would keep images slim, portable and capable of supporting various workflows in a standalone fashion in environments with no network connectivity.

*e-mail: marcelo.vogel@cern.ch

1 Introduction

Docker is a light weight, software based virtualization technology [1], which encapsulates a piece of software inside a complete file system containing everything that the applications will need to run: code, runtime, system tools, system libraries, etc. (see Figure 1, and 2). Processes run natively in the host yet isolated in containers, differentiating Docker from virtual machines, which simulate both hardware and software. This isolation guarantees that the software will always run in the same way regardless of the host's configuration, thus providing a uniform runtime environment ideal for distributed computing. The ATLAS Computing Group of the ATLAS experiment [2] at the Large Hadron Collider (LHC) is actively engaged in the deployment of containers for use in grid computing.

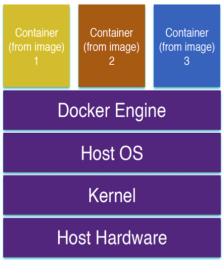

Figure 1. Docker will share the host OS kernel, but isolate different environments or 'containers' with respect to data, code, networking, etc.

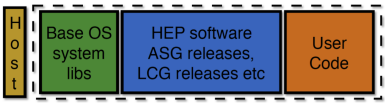

Figure 2. Docker images can be thought of as archived, executable file system snapshots, consisting of layer bundles (shown in color) corresponding to different software stacks, each one added sequentially on top of previously built ones.

2 ATLAS software in Docker images

The ATLAS Offline Software Group currently hosts a fairly comprehensive body of documentation with detailed instructions for building, running and managing containers with ATLAS software [3]. The documentation discusses the use of Dockerfiles, which are configuration files used by the Docker engine to build images in an automatic way. In a nutshell, a Dockerfile consists of a set of sequential instructions with each one generally adding a new layer to the image (see Figure 3). Dockerfiles facilitate image building by providing a clean and standardized way of installing and configuring software on top of more basic images and/or prebuilt layers. The end result is complete runtimes packaged in containers, which can in principle run on any minimally configured host.

In the ATLAS Gitlab docker repository we currently maintain Dockerfiles for the creation of images with full production software releases. A Dockerfile can be used as part of a pipeline in which a single software release is installed on a Scientific Linux CERN 6 [4] base

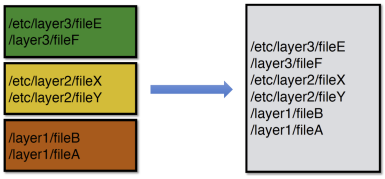

Figure 3. Docker images are built from layers.

image first, and conditions and geometry databases are installed and configured last. This process currently supports the creation of images that can be used in detector simulation in fully standalone mode, that is, with no need for a network connection. In addition to the single release layer, the pipeline installs a light database package that is sufficient to support most simulation workflows (left side of Figure 4).

The ATLAS Gitlab docker repository can be downloaded using the 'git clone' command:

```
git clone https://:@gitlab.cern.ch:8443/atlas-sit/docker.git
```

In addition to the standalone images for simulation we build images for digitization and reconstruction workflows. In this case the conditions database is prepared by copying COOL folders [5] from a central conditions Oracle database to a SQLite file, which is then packaged in the images. This SQLite database file has the minimal payload necessary to execute a reconstruction job, specified by performing a folder dump of the COOL folders that are required to process the input data based on its metadata, such as the run number range and/or the time interval, which specify data taking periods. We currently build images for MC digitization and reconstruction workflows that run in fully standalone mode (right side of Figure 4). A similar procedure is followed to build images for raw data reconstruction; however, these images are not completely standalone as there are currently no ATLAS tools available for copying from the online trigger menu database to an SQLite file. For this reason, performance tests on images used for data reconstruction require a network connection to this Oracle database.

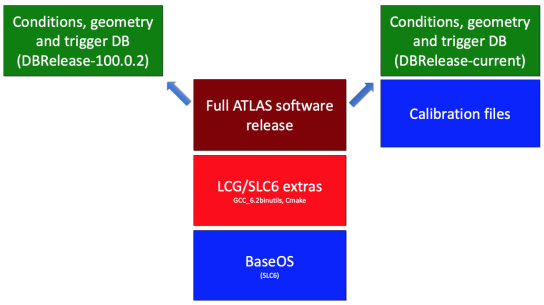

Figure 4. Image production pipeline. Left side for simulation and right side for reconstruction.

We currently provide examples of all three types of images in a public registry at Docker Hub [6]. To download the images, execute:

```
docker pull atlas/athena:21.0.15_100.0.2 # detector simulation
docker pull atlas/athena:21.0.23_DBRelease-200.0.1 # digi+reconstruction
docker pull atlas/athena:21.0.50_200.0.1 # data reconstruction
```

3 Performance of containers running ATLAS software

We document here a basic comparison of the performance in time of typical production workflows executed in images with full installations of ATLAS releases, and in 'bare metal' mode on the same host where releases are served via CVMFS [7] and the databases via connections to Frontier servers [8]. For the purpose of profiling the images thus far produced, we compare jobs run in Docker images versus bare metal in the following workflows:

- Simulation: simulation of particle-material interactions
- Digitization: simulation of detector read-out
- Reconstruction: clustering, tracking and reconstruction of physics objects in MC and data

The metrics measured are the wall-time and the CPU-time of the various production jobs. For each type of workflow the job is executed first using the appropriate full release image, where the container is only allowed to connect to the network for the trigger menu database in real data reconstruction, and then on bare metal where ATLAS software is provided through CVMFS and the databases are accessed through the network. The general technical specifications of the host machine used in the tests are listed in Table 1.

Table 1. Technical specifications of the testing host machine.

Test Machine	Description
`Linux desktop`	Intel Core i5-6500 CPU, 3.20GHz, Memory 7.8GB CentOS Linux 7, Docker 19.03.5, CVMFS 2.5.2

3.1 MC simulation, digitization and reconstruction

The two images built for processing Monte Carlo events were tested in two important production workflows: detector simulation and reconstruction. A full-simulation test job was run on 2 generated events (Randall-Sundrum Graviton→WW→lνqq) in the image built for simulation, and on bare metal. Figure 5 shows the performance in time of the container and on bare metal for the simulation job. Similarly, digitization and reconstruction jobs were run in the image built for reconstruction and on bare metal for 5 (HITS) events. Figure 6, Figure 7 and Figure 8 show the performance in time of the container compared to bare metal in the digitization, reconstruction and Analysis Object Data (AOD) production jobs respectively. Table 2 summarizes the timing measurements for the containers and bare metal in the simulation, digitization, reconstruction and AOD production jobs. All measurements on bare metal were taken with a 'warm' CVMFS cache, containing some of the necessary requirements for the job staged from previous runs.

Table 2. Timing comparison for MC workflows (times in minutes).

	Container		Bare-metal	
System	wall-time (σ)	CPU-time (σ)	wall-time (σ)	CPU-time (σ)
Simulation	9.64 (0.02)	9.58 (0.01)	9.14 (0.11)	8.15 (0.05)
Digitization	3.74 (0.27)	3.56 (0.07)	3.66 (0.02)	2.77 (0.01)
Reconstruction	7.07 (0.42)	6.99 (0.10)	8.53 (0.25)	6.06 (0.05)
AOD	1.34 (0.02)	1.27 (0.01)	1.60 (0.01)	0.80 (0.004)

σ is the sample standard deviation

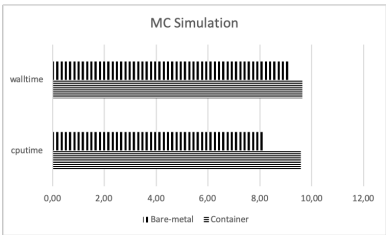

Figure 5. Timing comparison for simulation test.

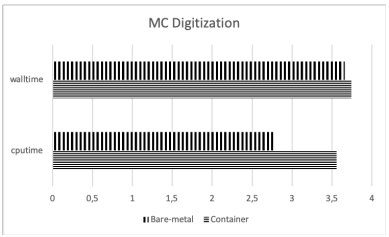

Figure 6. Timing comparison for digitization test.

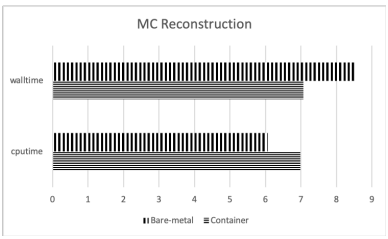

Figure 7. Timing comparison for reconstruction test.

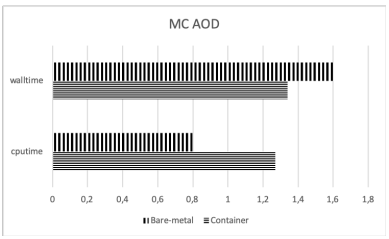

Figure 8. Timing comparison for AOD making test.

3.2 Real data reconstruction

A reconstruction test job was run on 25 real data events in the image built for reconstruction of raw data. Figure 9 shows the performance in time of the container compared to bare metal for the reconstruction job. Table 3 summarizes the timing measurements in the container and on bare metal for the reconstruction job. All measurements on bare metal were taken with a 'warm' CVMFS cache.

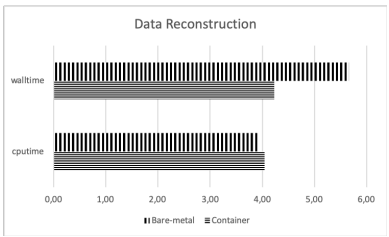

Figure 9. Timing comparison for reconstruction test.

Table 3. Timing comparison for real data workflows (times in minutes).

	Container		Bare-metal	
System	wall-time (σ)	CPU-time (σ)	wall-time (σ)	CPU-time (σ)
Reconstruction	4.23 (0.18)	4.04 (0.04)	5.65 (0.14)	3.92 (0.03)

σ is the sample standard deviation

4 Conclusions

The preliminary tests carried out in this study within the ATLAS experiment at LHC showed that typical production workflows running in containers with ATLAS software performed generally better in basic timing tests than the same jobs running on bare metal, where resources are obtained via connections to external file systems (e.g., CVMFS) and databases. We observe a reduction in the wall-time of reconstruction jobs executed in containers of about 20% when compared to the wall-times of jobs executed on bare metal. These results are expected since containers have all the resources packaged locally in the image to be consumed by jobs: software releases, calibration files, geometry and conditions databases and the POOL file payloads referenced in them. Therefore, we observe shorter wall-times using containers in workflows with heavy consumption of conditions databases, such as reconstruction of data and Monte Carlo events. On the other hand, containers performed similarly in time measurements to bare metal in workflows with low database requirements, such as detector simulation and digitization. We conclude that packaging ATLAS releases and databases in containers not only reduces the execution time of common production workflows, but also removes the requirement that sites provide access to these resources over the network. This is of particular importance when attempting to expand typical production workflows into opportunistic computing resources and sites with restricted network connectivity.

References

[1] Docker: https://www.docker.com
[2] ATLAS Collaboration (2008), *J. Inst. 3* S08003.
[3] How to use Docker containers in ATLAS: https://atlassoftwaredocs.web.cern.ch/athena/intro-docker
[4] SLC6: Scientific Linux CERN 6: https://linux.web.cern.ch/scientific6/
[5] A Valassi et al (2011), LCG Persistency Framework (CORAL, COOL, POOL): Status and Outlook. J. Phys.: Conf. Ser. 331 042043
[6] ATLAS Docker Hub image repository: https://hub.docker.com/u/atlas/dashboard
[7] CernVM File System: http://cernvm.cern.ch/portal/filesystem
[8] Welcome to Frontier: http://frontier.cern.ch/