Shifter: Containers for HPC

Lisa Gerhardt\textsuperscript{1}, Wahid Bhimji\textsuperscript{1}, Shane Canon\textsuperscript{1}, Markus Fasel\textsuperscript{2}, Doug Jacobsen\textsuperscript{1}, Mustafa Mustafa\textsuperscript{1}, Jeff Porter\textsuperscript{1}, Vakho Tsulaia\textsuperscript{1}

\textsuperscript{1}Lawrence Berkeley National Laboratory, Berkeley, CA, 94720, USA
\textsuperscript{2}Oak Ridge National Laboratory, Oak Ridge, TN, 37831, USA
E-mail: lgerhardt@lbl.gov

Abstract. Bringing HEP computing to HPC can be difficult. Software stacks are often very complicated with numerous dependencies that are difficult to get installed on an HPC system. To address this issue, NERSC has created Shifter, a framework that delivers Docker-like functionality to HPC. It works by extracting images from native formats and converting them to a common format that is optimally tuned for the HPC environment. We have used Shifter to deliver the CVMFS software stack for ALICE, ATLAS, and STAR on the supercomputers at NERSC. As well as enabling the distribution multi-TB sized CVMFS stacks to HPC, this approach also offers performance advantages. Software startup times are significantly reduced and load times scale with minimal variation to 1000s of nodes. We profile several successful examples of scientists using Shifter to make scientific analysis easily customizable and scalable. We will describe the Shifter framework and several efforts in HEP and NP to use Shifter to deliver their software on the Cori HPC system.

1. Introduction

High performance computing centers are becoming an increasingly valuable resource for HEP and NP experiments. Projected needs for the High Luminosity phase of the LHC vary, but the order of magnitude increase in luminosity will drive a similar increase in demand for computing resources. HEP and NP experiments are using HPC centers to provide a fraction of their computing, with ATLAS using 100M core-hours annually at US HPC centers and we expect this to increase.

NERSC, the primary scientific high performance computing facility for the Office of Science in the U.S. Department of Energy, is one of the HPC facilities that is working with HEP and NP scientists to run their workloads. Working with NERSC offers many advantages, as one of the largest facilities in the world devoted to providing computational resources and expertise for basic scientific research, NERSC is a world leader in accelerating scientific discovery through computation. NERSC is used by more than 6,000 scientists to run hundreds of different codes. Cori, NERSC’s newest supercomputer is a dual-partition system intended to support both data-intensive and HPC-style computing. Comprised of a 2,04-node Haswell partition and a 9,300-node Intel Xeon Phi “Knight’s Landing” partition, it will deliver roughly 5.5B core-hours per year. In addition to a wealth of CPU hours, Cori users also have access to a 28 PB Lustre scratch file system and a high-speed NVRAM Burst Buffer with an aggregate I/O speed of up to 1.7 TB/s \cite{1} as well as a 7 PB Spectrum Scale file system and 100 PB tape archive for long term storage.
However, running and installing codes on supercomputers like Cori can often be problematic, especially for HEP and NP workloads. Compute nodes lack local disk which causes problems for many traditional programs. The OS on a compute node is very minimal and designed to accelerate parallel jobs and reduce jitter. This often means that many of the standard Linux tools are not available on the compute nodes. Often it can be difficult to get the dependencies for software tools installed (e.g. installing FUSE which is required by CVMFS).

2. Shifter
NERSC has deployed Shifter [2] to address some of the challenges of using HPC resources. Shifter is a framework for deploying user-created images at large scale. Shifter can support Docker images as well as several other standard image formats (vmware, ext4, squashfs, etc.) and is tied into the batch system at NERSC. This allows users to use Docker, or some other image creation framework, to install complicated software stacks. When creating images, users can leverage system-level install tools like yum or rpm to easily install any needed dependencies. Users can also choose their operating system in these images, for example, HEP workloads could use the more familiar Scientific Linux operating system if desired.

In addition to offering a user-customized environment, Shifter offers some performance enhancements. On Cori, dynamic library files are brokered through an extra IO forwarding layer and extensive searches of many library paths can be slow. By bundling all shared libraries together into Shifter, library load times are improved by roughly a factor of four compared to the Lustre file system. Load times can be further improved over a RAM file system by directly adding the library paths to ldconfig cache (something that is only possible by leveraging root level functionality when creating Shifter images).

On Cori, Shifter is integrated with Slurm so that XFS files can be created at the beginning of a job and mounted into the image. These images are writable and can serve as a repository for temporary data created during a job, replacing some of the functionality of local disk. These XFS files are located on the Lustre file system, but because all metadata operations for the XFS file are local to the node, file operations are very fast. This has the added effect of delivering good performance even for “bad” IO like many small writes (Fig. 1).

![Figure 1. Bandwidth and IOPS results for Shifter XFS file IO.](image)

Shifter is open source software released under a BSD license (https://github.com/NERSC/shifter) and has been successfully installed at several sites ranging from IBM Blue Gene/Q centers to more conventional Linux clusters.
Figure 2. Load times in seconds for ATLAS simulation software stack as a function of the number of Cori nodes for Shifter (red), Lustre (blue) and the Burst Buffer (green).

3. Shifter and HEP/NP
3.1. Delivering CVMFS at Scale
As a proof of concept, Shifter was used to construct images that mimic the CVMFS functionality for ATLAS, ALICE, and CMS. This was done using python code called uncvmfs (https://github.com/ic-hep/uncvmfs) to deduplicate the entire CVMFS software stack and copy it into an ext4 image. The resulting images were quite large: about 3.5 TB with more than 50 million files and directories for the ATLAS CVMFS repository. The ext4 image was then compressed into a 300 GB squashfs image. This image was used for ATLAS Geant4 simulations and showed excellent scaling in start up time out to 500 nodes on Cori. Fig. 2 shows the startup time for Shifter (red), Lustre (blue) and the Burst Buffer (green).

While this test was quite promising, it was difficult to deliver CVMFS content in a timely manner. Images of this size take roughly a day to compress and copy onto Cori, leading to turn around times for new image releases longer than the few hours it takes for new information to propagate through the entire CVMFS framework. This has led other groups to investigate other means of accessing CVMFS with Shifter.

3.2. Shifter parrot Scaling Tests
If the CVMFS stack can be installed and updated on the Lustre file system, some of the overheads of creating monolithic CMVFS Shifter image can be avoided. To investigate this possibility, the ALICE group at NERSC performed several scaling tests using Shifter and parrot (https://cernvm.cern.ch/portal/filesystem/parrot) to access their CVMFS repository. They measured the startup times for four configurations out to concurrencies of 1600 processes: the ALICE CVMFS software stack contained in a Shifter image and unpacked into the image via uncvmfs, the CVMFS stack installed inside a Shifter image with cvmfs_preload and accessed via parrot, the CVMFS stack installed on the Lustre file system with cvmfs_preload and accessed via parrot, and parrot reading the CVMFS software stack via remote squid server with a global, prefilled cache on the Lustre file system (Fig. 3). As in the ATLAS case, the fastest load time were measured with the Shifter image. Using parrot to access CVMFS led to roughly a twofold increase in load time at low concurrencies. Accessing the CVMFS stack on the Lustre file system was about 30% slower than from Shifter. However, despite their relative differences, all load times were still under a minute for low concurrencies. At higher concurrencies, the delays from the Lustre file system become non-negligible compared to a purely Shifter-based
solution. Loading software directly from the squid server was slow enough to be untenable at all concurrencies.

All of the parrot based configurations offer the advantage of near-immediate software change propagation. This is currently still challenging for the monolithic Shifter approach with image build times of several hours to days. ALICE has a software release policy that requires a dynamic software distribution approach with small latencies that make using parrot to read from Lustre appealing. For the near future, ALICE will run at a low enough concurrency that the difference in load times between Shifter and parrot is negligible, but they are exploring a more Shifter-based solution for higher concurrency runs.

3.3. Speeding STAR with Shifter

STAR was also able to leverage Shifter for data production [3]. The dependencies of the STAR software stack are particularly complex, and it is often difficult to get the needed 32-bit libraries installed on systems. Using the Shifter framework, STAR scientists were able to create fully functional images relatively easily with a few clever tricks. By choosing a base OS that was the same as an existing system, they could simply tar up the compiled STAR software stack, copy it to image, untar it and have everything work as expected. This allowed them to avoid a lengthy compilation and installation that usually takes half a day.

Each STAR job needs to access detector condition information contained in a MySQL database. A typical job will access this database tens of thousands of times during analysis, so it is important that this step is not a bottleneck. For scalability reasons on HPC, STAR has opted for a setup where a local MySQL server with a fresh cache is launched at the start of the job to serve all the data processing threads on the compute node. Initial attempts to query this server with the payload on the Lustre file system took more than 30 minutes to complete. By copying the entire payload (about 30 GB for the entire STAR DB) to a XFS file mounted in the

Figure 3. Load times in seconds for ALICE CVMFS software stack as a function of the number of Cori cores for various configurations. See Section 3.2 for details.
Shifter image, the load times became negligible. Using the XFS file system allowed for trivial scaling of the concurrency of STAR jobs with minimal changes in the production framework.

4. Conclusion
A number of HEP and NP groups are already investigating Shifter at NERSC. In addition to the ATLAS, ALICE, and STAR cases described here, CMS, Dayabay, and LZ as well as several other groups from different science areas are also experimenting with Shifter. Shifter is making scientific analysis easier at NERSC by allowing users to build a custom environment. Shifter images offer scaling and better performance without substantial reworking of pipelines. Since Shifter images are portable, they can be used at any site that supports containers, facilitating reproducible scientific computing and portability.

5. Acknowledgements
This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References
[1] Bhimji W et al., 2016, 22nd International Conference in High Energy and Nuclear Physics, ‘Extreme I/O on HPC for HEP using the Burst Buffer at NERSC’.
[2] Canon R and Jacobsen D, 2016, Proceedings of Cray Users Group, ‘Shifter: Containers for HPC’ 103.
[3] Mustafa M et al., 2016, 22nd International Conference in High Energy and Nuclear Physics, ‘STAR Data Production at NERSC/Cori, an adaptable Docker container approach for HPC’.