Exploring Container Technologies for Large Scientific Libraries: Docker and Trilinos

Michael A. Heroux, Sandia National Laboratories & St. John's University, James M. Willenbring, Sandia National Laboratories, Sean J. Deal, St. John's University

The Trilinos Project produces, maintains and distributes a large collection of reusable, parallel scientific libraries. Docker provides container technologies that support compilation, packaging, distribution and execution of software on Linux, Mac OS and Windows systems[1], with emerging support for Cray platforms. In this abstract we describe recent efforts to explore the potential for using Docker in a variety of settings to enhance several Trilinos Project workflows. The technical foundation for this article is presented in an Honors thesis of one of the authors[2].

Trilinos Software Ecosystem

The Trilinos project produces and supports a large and growing collection of reusable software packages. The majority of packages (approximately 60) are part of a single Git repository hosted on GitHub.com[3]. The core Trilinos user base is composed of application developers running on parallel distributed memory systems with multicore, manycore and GPU devices. This user base is accustomed to forking from the Trilinos repository or downloading one of our compressed source tarfiles (released approximately quarterly), and then building the libraries, tests and examples on their native system. While this process works for many users, it is daunting, especially for new users.

Docker Possibilities

The Trilinos Project has used virtual machine environments such as VirtualBox and VMWare for many years, but there have always been technical (performance) or access (license) barriers to promoting these virtual environments. Docker promises to be different by preserving performance, both in execution time and storage size, and being widely available without restrictions.

Trilinos Tutorial via Docker. We explored Docker first as a way to package and distribute the web-based Trilinos tutorial (WebTrilinos). Prior to using Docker, using the tutorial required installation on a networked workstation with password controlled access. Docker is attractive because installing the tutorial was very challenging, requiring careful configuration of the web portal, special security privileges, and access monitoring. Using the Docker container, we can now have individual installations of WebTrilinos through a simple two-step process: Install Docker, then install the WebTrilinos container[4].

Reference Development and User Environment. We are presently exploring the use of Docker to provide a portable reference Trilinos development and user environment. While the Trilinos project tries to support as many computing platforms, operating systems and compiler environments as possible, there is tremendous value in having a universally accessible reference environment. Such an environment permits better collaboration on detecting and correcting software errors and can also be built to contain the many third party libraries that can be used via Trilinos, pre-built so that a developer can simply use them. This environment is also very useful as the foundation for a pre-check testing platform.

Standard Binary Distributions: One of the most obvious values of a pre-built version of the Trilinos libraries is its use by applications as the primary binary for linking. We anticipate that many of our users who currently rely upon a particular numbered Trilinos release (e.g., Trilinos 12.6.1), could instead rely upon a Docker container with a pre-built instance of the same version.

---

[1] Linux support is native. Native MacOS and Windows support is available in pre-release form.
[2] http://digitalcommons.csbsju.edu/cgi/viewcontent.cgi?article=1011&context=honors_thesis
[3] https://github.com/trilinos/Trilinos
[4] https://hub.docker.com/r/sjdeal/webtrilinos

Specialized Containers. Trilinos can be used to solve many kinds of problems, but to understand the available capabilities requires a significant effort. One must learn about many packages, parameters and data structures in order to be able to construct a solution strategy for their specific problems. For example, if someone wants to compute a truncated singular value decomposition (SVD) of a large matrix, they must understand that Anasazi is the Trilinos eigensolver package, that it can be used to compute a truncated SVD, and then must form a distributed matrix, or provide a function for "matrix-free" application of the linear operator to a vector. Using Docker and a modest driver program, we can provide a turn-key solver for large-scale SVD computations, and package it in a Docker container. These specialized containers can be valuable to our traditional CSE user community, but should be of particular value to data sciences communities who are, as we understand, particularly accustomed to accessing software capabilities in this way.

Portable High Performance. As demonstrated in Deal's thesis and with Cray's intent to support Docker, there is a realistic possibility that Docker containers can provide performance portability on many platforms, even those with GPUs and other advanced computing devices. The value of this potential cannot be overstated, and could truly revolutionize the packaging and distribution of scientific and engineering software, enabling qualitatively better computing workflows. While realizing the full potential of this approach is still some time away, there has been tremendous progress, even within the past year. The Trilinos project will continue to pursue this capability aggressively.

Trilinos Docker Effort Status

The Trilinos project has already seen benefits from using Docker for its WebTrilinos distribution, and we have measured excellent performance on a Linux cluster. We will continue these explorations.

Our present challenges with Docker include:

1. **Native Windows and MacOS support.** Native Windows and MacOS support is still in the beta testing phase. While we have had some success on MacOS (we have not tried the native Windows version), installation is still not seamless.
2. **Using Docker with a VPN.** Early experiences trying to use Docker while connected to a virtual private network (VPN) have not been successful.
3. **Docker container composition.** We can vertically stack Docker containers. But it would be useful if there was more support for generally combining containers. Not in a service sense, but in the sense of being able to deliver several software packages via one container. This can probably be scripted, but the ability to do this flexibly would be useful. There may be a solution to this issue of which we are unaware.
4. **Docker stability, maturity and support.** While we have a lot of excitement about Docker, we have found it to be surprisingly unstable. We have experienced the website being completely unavailable, and have had updated (non-beta) versions of Docker not work at all. We have had trouble with the beta version too, but expected that. It is also surprisingly difficult to find information when encountering problems. Basic info is well organized, but we had anticipated more discussion help threads directly applicable to problems we have had.

Summary

The Trilinos Project has experimented with Docker as a packaging and distribution technology in several settings. Despite some instabilities, the emerging features and community support are very promising. If Docker fulfills its full potential, we believe its use in CSE can qualitative improve scientific productivity. Furthermore, its use can provide a much-needed bridge between the CSE and data sciences communities, providing access to the scalable computational capabilities that have long been a part of the CSE software ecosystem to the data sciences.