Why Convergence? A Contrarian View and a Path to Convergence Enabling Specialization

Barney Maccabe
Computer Science and Mathematics Division
Oak Ridge National Laboratory
maccabeab@ornl.gov

A position paper submitted to the BDEC workshop on Pathways to Convergence.

Introduction

In the Hobbes OS/R project we are exploring mechanisms to support the composition of independently developed applications. These mechanisms employ light-weight, whole system virtualization to enable independent software environments. A unique aspect of Hobbes is our exploration of mechanisms to selectively break down the isolation imposed by whole system virtualization to enable interactions between the applications being composed. While our initial motivation was based on the challenges associated with developing large scale multi-physics/multi-scale applications like those found in CASL (Consortium for the Advanced Simulation of Light water reactors), it quickly became apparent that the mechanisms we were exploring could also be used to compose the data analytics portions of a scientific workflow with the simulation portions of this workflow. Because the approach taken in Hobbes can easily support the integration of applications that use fundamentally different software stacks, we believe that the mechanisms we are developing provide an alternate approach to the convergence of Big Data Analytics (BDA) and High Performance Computing (HPC). An approach that embraces the value of specialization while providing users with an integrated software ecosystem that can easily accommodate new hardware technologies and programming models as they are developed.

Being contrary: sharing is good, but specialization is also good

The Merriam-Webster dictionary provides the following definition for convergence (http://www.merriam-webster.com/dictionary/convergence): "the merging of distinct technologies, industries, or devices into a unified whole."

The *BDEC Pathways to Convergence* "Call for contributions" implies that shared infrastructure is required to achieve convergence. All of the value (Why should convergence be our goal?) is based on *sharing*: "sharing is required," "sharing is desirable," "sharing is efficient," and "sharing has impact." Moreover, all of the suggested pathways to convergence reinforce this perspective by focusing on *common* aspects of the converged infrastructure: "Common tools and technologies," "Common vision …," "and "Common ecosystem… ." It goes without saying that exploring convergence as defined by a common infrastructure based on shared components has high value. However, too much emphasis on sharing and common aspects will result in an infrastructure that doesn't do anything well and is brittle in the face of externally driven changes.

Dating back to the earliest MPP systems in the 1990's (and probably before that), large scale HPC systems have used partitioning to enable specialization within a single system. As an example, the Titan system deployed at Oak Ridge National Laboratory uses specialized hardware and software for nodes in the compute partition which is very different than the hardware and software used for the nodes in the system management or I/O partitions. While these different parts of the system do not share common hardware or software, the whole system functions as a unified whole.

Importantly, we should not strive to embrace sharing and commonality as an end, but instead, we should try to understand the costs and tradeoffs associated with building a unified, integrated system that supports specialization.

INCREMENTAL MIGRATION AND INTEROPERABILITY

There is great benefit in defining a fully converged and deeply integrated infrastructure. If nothing else, this exercise defines a target and we can gauge our progress with respect to this target. However, we must simultaneously build systems that support *incremental migration* toward the target and *interoperability* between alternatives. Finally, we must also recognize that full convergence is not likely.

Incremental migration is essential, due to the size of the code bases that need to be maintained and the costs associated with migrating these code bases to a new converged target. The general principle is that one needs to see improvements in performance or some other metric of value that are commensurate with the effort needed to advance the code toward the converged target. Notice that it is quite likely that the effort needed to advance the code will outweigh the benefits long before the converged target is fully reached.

Interoperability between alternatives recognizes that there will always be alternative approaches that need to be brought into the converged target. Here, we don't need to look any further than programming models/languages. Consider the example of MPI. MPI was developed as a convergence of all the different message passing systems available at the time. While MPI was the dominate programming model for two decades, other, specialized programming models (e.g., PGAS) persisted and, because the community did not focus on interoperability from the start, we now face challenges in integrating applications that use MPI with applications that are based on PGAS.

MECHANISMS: VIRTUALIZATION AND SELECTIVE ISOLATION

We believe that the mechanisms of virtualization and selective isolation can be effectively deployed to address the challenges of incremental migration and interoperability, especially in the context of BDA and HPC. Virtualization provides the isolation needed to run an arbitrary software stack for parts of a composed application while incrementally migrating other parts to the application to a converged target environment. While the Hobbes project has been focused on full system virtualization, we believe that other forms of virtualization, e.g., containers, might be able to support this goal. Selective isolation breaks down the natural isolation between virtual systems by providing mechanisms for interaction (e.g., shared memory, data streams, events, etc) and enables interoperability.