A Case For Investing in Software Architectures and Framework Research

A. Dubey

January 12, 2015

In the last workshop I made the case that the workflow of large scale scientific data analysis resembles that for the computations of multi-physics phenomena in that there are multiple stages and components, that don't necessarily interoperate with one another very well. And therefore using the insights and lessons learned from multi-component multi-physics codes could prove to be valuable for the scientific data analysis, whether that data comes from simulations, observations, or experiments. While that continues to be true in general, the multi-physics, multi-component codes are facing a crisis of their own which should be urgently addressed. Therefore, this time I would like to focus on the growing chasm between where the extreme scale simulation or data analysis codes need to be and where they are today. In particular, I would like to draw attention to the lack of understanding about what should the software architecture (or framework) of a multi-component code look like in order to leverage the investment being made in the systems software and programming abstractions for future machines.

The software architects of today are faced with a bewildering array of design choices with very little to inform them about the possible implications of these choices. The successful large scale scientific codes of today did not grow out of vacuum. In the late 90's investment was made in the design of frameworks. Though some people mistakenly consider that effort to have largely failed, almost all the successful multi-physics codes of today with any degree of composability owe their infrastructural design to the ideas from successful and failed frameworks from that effort. The frameworks were designed and documented, and as a result software architects targeting the fat-node distributed memory paradigm had rich literature to draw upon, and had many examples to follow.

For the forthcoming exascale era the design space is bigger than ever, with more programming abstractions to be incorporated into the software architecture. There has been investment in research on programming models and abstractions that might make it easier to code for the increasingly heterogeneous platforms of the future. There has also been some investment in algorithms that might better deal with the new challenges such as decreasing memory per processing unit and need to minimize data movement. There is even some focus on increasing the software productivity in the scientific process. However, it is my belief that without some concrete experimentation with the interplay among various programming abstractions, we will not have the know-how to either develop new codes or refactor the current codes into nimble, robust, reliable and portable codes that will be needed for achieving

the scientific goals using the exascale resources. Also, such experimentation can inform the community about the tool-chain that can help various codes transition from their current infrastructure, which is designed primarily for distributed memory bulk synchronous parallelism to the infrastructure for more complex parallelism and memory models expected in future machines.

It is urgent that a few of the multi-physics capable frameworks be funded to be early adopters of the multiple emerging programming abstractions within their frameworks. These projects should also be encouraged to investigate the techniques for automating the transition process and translate those techniques into tool-sets for use by others in the community. Additionally, these projects should carefully document the methodologies and software process developed during the course of the adoption of abstractions and transition of the code base to the new framework. The advantages of such projects that combine research with production grade development will be two-fold. One is that the current CS research will be confronted with real world application problems, and if there are any gaps in their adoptability by the scientific codes they will be systematically identified. This will give an opportunity to shorten the time-to-adoption for the key technologies being developed under the various CS research programs targeting exascale computing. The second is that instead of hero-programmers rewriting and optimizing one code at a time with very little reusability, the proposed approach would create tangible artifacts that can be exploited by others facing similar transitions.