An Exascale Approach to Software and Hardware Design

William Kramer and David Skinner

June 21, 2009

The demands of exascale require a complete rethinking of the software and hardware development process. For the past 10-15 years, horizontal layers of software and hardware design and development have been the de facto standard of creating HPC software. This situation is due in part to the influences of funding methods, research incentives, software methods, the open source movement and commercial outsourcing and specialization. This horizontal design approach leads to the development of discrete components in the software stack and independent hardware components – all developed with different methods, requirements and quality. Past generations of system software (from the earliest operating systems through to the original community source movement with Unix) and hardware, showed some degree of top-to-bottom vertical design. The last 10-15 years, however, have been dominated by plug-and-play componentization that is focused on horizontal functionality and portability.

The horizontal design approach has notable successes like the Linux kernel, MPI and a variety of job schedulers. It also has many challenges that are inhibiting progress and making even petascale systems challenging to fully exploit. As many who field terascale clusters know, every cluster is now unique with different horizontal components (often in name, at least in version). Currently, at the tera- and petascale, there is only one company that produces a system software stack that is vertically designed from top to bottom and two other companies that is providing a scaled, vertically tested stack that has specifically designed components added to the horizontal components.

To reach petascale, the HPC community has mitigated many of the issues in the horizontal design method such as:

- relying on vendors to do vertical testing and integration,
- standing up extra test bed resources for integration testing and error correction,
- taking excessive time from the few large-scale production systems to do integration, testing, diagnosis and correction, or
- living with inefficient and error-prone systems.

The current horizontal design method presents a number of insurmountable challenges to reach exascale. Yet economics and cost effectiveness will not let us return to the days of completely proprietary vertical methods. Nor can one organization alone, be it government or private, afford to pioneer exascale and make it a success.

There is some hope! The HPC community needs to change the horizontal approach of developing essentially isolated software components that have a narrow function and role in the system. Instead, the community must organize hardware and software development activities with component cross cutting principles. These principles should define the requirements, function, interfaces, integrations and performance needs for each horizontal component. Instead of thinking of integration as the final step in defining and developing an exascale system, it will be the first step at exascale.

The cross cutting requirements for the vertical design approach were identified to first order at the first IESP workshop. They include: resilience (reliability and fault tolerance); performance; programmability; computational models; I/O; consistency and verification; resource management; and power management/total cost of ownership.

There are limited proofs of existence that hint that the vertical design approach yields an effective and long-lived, yet flexible, solution to this conundrum. Some examples include:

- The DOE SciDAC program. SciDAC introduced the concept of software application development teams and software infrastructure teams that are linked in an iterative approach to developing applications that rely on increasingly more effective software infrastructure.
- Scientific “framework” development for large-scale experiments and long-lasting infrastructure. High energy physics regularly uses a formal process of vertical architecture definition, software development and testing often incorporating thousands of funded and unfunded contributors. The processes here are notable for progressive demonstrations of integrated milestones (e.g. Data Challenges) and timely delivery of equipment that is being co-developed.
- The methods used to produce community-based software such as the High Performance Storage System, which follows formal methods and shares development and support across multiple organizations via formal agreements.
- Commercial operating system development methods such as those at IBM, Sun and Cray.
- Formal testing methods that are used in the verification and validation of network protocol change proposals.

One factor motivating a renewed emphasis on vertical integration is the dominance of software failures as the causative factors in large-scale system lack of availability. Failure at scale of system software such as filesystems, batch schedulers and even authentication mechanisms such as LDAP is a major problem for HPC resource managers. In many cases, vendors leverage software that works well at smaller scales, but they place too much reliance on the ability of the software to integrate seamlessly at all levels. Some studies (Kramer W. T., 2008) indicate that on large systems, across vendors and architectures, software accounts for the majority of systemwide failures on HPC systems.

User experience can also suffer when developers pay insufficient attention to end-to-end software functionality. The usability of tools such as debuggers and performance profilers can diminish significantly when they are used beyond the scale that software vendors are capable of testing. Usability and thus the value of the HPC resource to science can be improved by a more goal-oriented vertical approach, one that builds in expectations of usability at scale.

The vertical approach is not at odds with previous approaches to HPC software development. A tightly coupled vertical design can still produce software that is of lateral use, but attention to vertical integration diminishes risks of software failure when relying upon “off the shelf” generic software. Vertical integration does not replace these software components but improves them for HPC.

In summary, we will not achieve exascale without a tightly coupled vertical design and integration process. The methods that got the HPC community to early petascale will not stretch to exascale. The vertical approach does not diminish community contributions, flexibility or openness. Rather, it makes investments by people and organizations more likely to have impact.