Consistent Application Performance at Exascale

William Kramer and David Skinner

June 21, 2009

This white paper examines the future of application performance consistency on exascale parallel computing systems. By performance consistency we mean the regularity of wall clock times to complete a fixed amount of application progress. In particular we do not address consistency of results from applications. Correctness of results is an important topic as well and will be treated separately.

The design of high-performance computers concentrates on increasing computational performance for applications. Performance is often measured on an optimally configured, dedicated or near-dedicated system to show the best case in performance. In real environments, resources are seldom dedicated to a single task and systems run multiple tasks that may negatively influence each other. This more complex production context is arguably more important in setting user expectations of application performance. Managers of large HPC resources likewise depend on consistent application performance in order to allocate the resource to multiple science teams.

Large-scale systems running in production mode are particularly prone to performance fluctuation. By their nature they involve a large number of components servicing a varied workload. Resource contention, which results in performance degradation, can be caused by underprovisioned interconnects, topology mismatches, congestion aware messaging, assignment of memory, systems software layers, system management event timing (daemons running at particular times, aka "system jitter"), bugs in configurations, software and hardware and system management and configurations. Keeping all of these impediments in check so that users observe consistent performance is challenging at the terascale and petascale. It is therefore crucial that we consider how larger machines and larger applications can avoid the pitfalls encountered with today's machines.

What level of consistency is reasonable to expect at exascale? Inconsistency of parallel applications has implications for how much useful work can be produced by exascale systems. Performance inconsistency is caused by many factors but on well-managed HPC systems, the simple causes of inconsistency (multiple jobs running within a shared memory processor, simple configuration mistakes, etc.) are not the primary causes of inconsistency. Factors leading to changes in performance occur over multiple time scales and originate both from within systems, within applications and from external sources. As a result, variability in runtime performance is strongly tied to the hardware and software architecture. On today's terascale system, it has been shown that high degrees of consistency (CoV < 1%) are regularly achievable for most workloads (Kramer W. T., 2008).

The performance impact of inconsistency can be quite large, becoming the dominant impediment to parallel scaling in some cases. Consistency, or really the lack of it, will play an even larger role for the effectiveness of exascale architectures unless proactive steps are taken to address it. Inconsistency can result from a myriad of causes, including the hardware architecture.

Understanding the parallel scaling factors leading to performance inconsistency needs to be a chief concern of the design and use of exascale systems. Since the majority of testing and performance analysis is done on test systems much smaller than production machines, it is common to encounter variability-induced performance loss at scale that goes unseen on smaller machines.

The variability of performance has as much impact on users' ability to accomplish their goals as availability and mean time between failures. For example, the user's productivity is impacted at least as much when performance varies by a factor of two as when a system's availability is only half the expected time! In both cases, the amount of work done is only half what is expected of the system.

Multiple sources show inconsistency in runtimes leads to many negative impacts [(Figueira and Berman 1966), (Worley and Levesque 2004), (Zhang, Sivasubramaniam, Moreira, & Franke, 2001)], all of which reduce the value of HPC (and future exascale) systems. The first impact is less overall work done by the system. Runtime inconsistency is inherently bad for performance since variations in runtime proceed upward from some best case runtime, i.e., variation is seldom toward better than optimal performance. The longer a task takes, the more time it takes to get usable results for analysis. Since some applications have a strict order of processing steps (i.e. in climate studies, year one has to be simulated before year two can start), they cannot directly overcome this slowdown via, say, increased parallelism. Inconsistency can also introduce wider error margins for non-deterministic applications, leading to more difficulty verifying results.

Inconsistency decreases the efficiency of HPC parallel systems since cycles are lost to both job failure and complex job scheduling to mitigate the lack of consistency [(Srinivasan, et al. 2002), (Lee, et al. 2004)]. Jobs fail through incorrect estimation of the batch queue requirements. System scheduling becomes less effective because users must be overly conservative in requesting batch time. Most scheduling software relies on user-provided run estimates, or times assigned by default values, to schedule work effectively. When a cautious user overestimates runtime, the job scheduler operates on poor information and results in inefficient scheduling selections on systems. These all contribute to the loss of user productivity and decreased system impact.

Consistency is influenced by a number of factors.

- System configuration and management errors and bugs (Kramer and Ryan 2003). At exascale, with orders of magnitude more components, there will be increased likelihood that such artifacts are introduced.
- Hardware architectural features–including the network topology, size of computational nodes, hardware collective features (from vectors to distributed collectives), automated error recovery and hardware consistency features (e.g. global cocks) (Skinner and Kramer October 6-8, 2005). At exascale, the trade-offs of many more cores within an SMP/node or a much broader network will greatly influence the consistency of systems.
- Software architectural features–including message passing collectives, the OS footprint (microkernels, lightweight OS, full OS), synchronization primitives, automated error recovery and service provisioning (Kramer and Ryan, May 2003), The software layers, being less integrated than hardware design and limited by hardware features, prove the most challenging area to control inconsistency at the exascale.

- Application implementations–including ineffective use of resources, static workload allocation, I/O and application specific check pointing. At the exascale, in order to deal with the system challenges of resiliency, parallelism and performance, applications will have more responsibility for dynamic workload reassignment, adaptive behaviors (AMR) and recovery. This will lead to even more challenges for consistency unless there are well-planned interactions between the system components and the applications.
- Resource management–including scheduling applications that compete for resources, prioritization, quality of service, and coordination of services. Often this type of consistency challenge is the result of insufficient information for the scheduling agents and insufficient methods for applications to express their needs. At the exascale, power management will increase the need for dynamic interactions competing to serve different goals. For example, the exascale facility may wish to control power costs, or the system may do power control automatically, without taking into account the consistency needs of the applications nor the time researchers need results.

The challenge is how to maintain this level of consistency at the exascale. To date, once inconsistency is identified, it is possible, albeit not always easy, to restore consistency by making changes to parameters, fixing bugs and adjusting configurations and so on. It is not clear this will be the case at exascale unless consistency is a holistic design parameter. Key issues for assuring consistency at the exascale include

- Architectural and system design criteria that reflect consistency requirements
- Testing for consistency at scale
- Well-studied solutions and trade-offs for consistency
- Consistency metrics for exascale systems
- Understanding system architectural influences that can be explicitly linked to consistency
- Resource management that is too narrowly defined
- Ineffective methods to express performance and consistency needs up and down the software hierarchy

In order for exascale systems to exhibit the consistency that is required to make the applications and systems productive, new understanding of the causes and solutions to inconsistency are needed, along with new ways of measuring the impact that design, implementation and operational choices have on consistency. In order for applications to mitigate the effects that make systems inconsistent, new mechanisms for expressing consistency requirement and applications reactions are also required.

Figueira, S. M., & Berman, F. (1966). Modeling the Effects of Contention on the Performance of Heterogeneous Applications. *Proceedings of the High Performance Distributed Computing (HPDC '96)*, (p. 392).

Kramer, W. T. (2008). *PERCU: A Holistic Method for Evaluating High Performance Computing Systems.* University of California at Berkeley, Department of Electrical Engineering and Computer Science. Berkeley, CA: University of California.

Kramer, W., & Ryan, C. (May 2003). *Performance Variability of Highly Parallel Architectures.* Berkeley, CA: Lawrence Berkeley National Laboratory.

Kramer, W., & Ryan, C. (2003). Performance Variability on Highly Parallel Architectures. *International Conference on Computational Science 2003*. Melbourne Australia and St. Petersburg Russia.

Lee, C. B., Schwartzman, Y., Hardy, J., & Snavely, A. (2004). Are user runtime estimates inherently inaccurate? *10th Workshop on Job Scheduling Strategies for Parallel Processing.* New York, NY.

Skinner, D., & Kramer, W. (October 6-8, 2005). Understanding the Causes of Performance Variability in HPC Workloads. *2005 IEEE International Symposium on Workload Characterization (IISWC-2005).* Austin, TX.

Srinivasan, S., Kettimuthu, R., Subrarnani, V., & Sadayappan, P. (2002). Characterization of Backfilling Strategies for Parallel Job Scheduling. *nternational Conference on Parallel Processing Workshops (ICPPW'02)*, (p. 514).

Ujfalussy, B., Wang, X., Zhang, X., Nicholson, D. M., Shelton, W. A., Stocks, G. M., et al. (November, 1998). High performance first principles method for complex magnetic properties. *Proceedings of the ACM/IEEE SC98 Conference.* Orlando, FL: IEEE Computer Society, Los Alamitos, CA 90720-1264.

Worley, P., & Levesque, J. (2004). The Performance Evolution of the Parallel Ocean Program on the Cray X1. *Proceedings of the 46th Cray User Group Conference.*

Zhang, Y., Sivasubramaniam, A., Moreira, J., & Franke, H. (2001). Impact of Workload and System Parameters on Next Generation Cluster Scheduling Mechanisms. *IEEE Transactions on Parallel and Distributed Systems , 12* (9), 967-985.