The Under-Represented Resiliency Issues for Big Data and Extreme Scale Computing

2/25/14

Dr. William Kramer
Blue Waters Director
@Scale Program Office Director
National Center for Supercomputing Applications
University of Illinois Urbana-Champaign

Abstract

Many reliability investigations for large scale systems focus on hardware component issues for individual component failures. Significant progress in hardware design and resiliency make today's Petascale systems surprisingly effective and reliable. But, for more than a decade, system software has been shown to cause more failures than hardware on large scale systems. More importantly, software is the dominant cause for system wide outages that have widespread and more disruptive impacts on large scale system effectiveness. Chief among the causes for software failures is software related to storage and I/O, which will become more important in the BDEC era. This paper explores the past and current investigations into software failures on large scale systems and projects the impacts to exascale BDEC system. If this situation persists exascale systems reliability will be dominated by software failures to the point they may become unusable. Finally, the paper presents possible actions that could improve the current situation and give hope reliable exascale BDEC systems can be achieved.

Introduction

There is a large body of investigations and solution for improving reliability and resiliency for large scale systems[12345]. Much of the work concentrates on understanding the causes and frequency for hardware failures, and then remediation steps that can be effective to make failures more transparent to applications. The improvement in hardware design methods result in reliability is beyond what was expected and results in much improved mean time to interruption for systems and individual applications. New methods for hardware resiliency (RAID, Chipkill, ECC, dynamic routing, multiple connectivity, etc.) mean hardware can transparently fail and not impact applications in catastrophic ways.

However, the improvements in hardware reliability have now uncovered major and critical challenges for software reliability and resiliency that must be addressed to enable systems and applications to continue to improve at scale and scope. For the past decade, evidence that software is responsible for the majority of failures on @Scale systems[6] has been developed.

Examples of Software Error Impacts

The Blue Waters system[7], one of the largest Petascale systems in place, has a Computational and Analysis Sub-system with 22,640 XE6 compute nodes, 4,224 XK7 compute nodes, 582 Lustre Network I/O router nodes and 202 other service nodes. Each XE6 node has two AMD Interlagos processor modules and each XK7 node has one Interlagos processor s module and one NVIDIA K20X Kepler GPU. This equates to over 845,000 AMD integer cores and 4224 GPUs. The Blue Waters also system has over 17,000 2 TB disk drives in 36 Sonexion 1600 racks, creating three file systems /home and /scratch at 2.1 PB of usable storage and /scratch with 21 PB usable storage. Some characteristics of Blue Waters performance and reliability is presented in the table below.

During the last calendar quarter of 2013, the Blue Waters averaged only 1.7 node interrupts per day. A node interrupt is defined as a node failure that is, or is likely to be, apparent and disruptive to one or more applications. The actual node failure rate is somewhat higher since some node failures are detected by proactive testing, but these failures do not impact applications on the system and are not discussed here. The vast majority of single node failures are contained to a single node.

Blue Waters had 12 *"system wide outages"* to system hardware or software[i], which are defined as situations where outright failure or degraded services impact, or could impact, multiple applications. The system wide outage MTBF is 7.667 days for the last calendar quarter for 2013. Over a longer period of time for the first 10 months of initial operations, the system wide outage MTBF for Blue Waters was 6.7 days, including all failures[8]. In comparison, compared to projections there are a relatively small number of disk failures occurring, six (6) in Q4 2013 (a unit MTBF of hundreds of years), and all are transparent to applications.

Peak Floating Point Performance	14.1 Petaflops/s
Aggregate Memory	1.66 Petabytes
Sustained Floating Point Performance per Sustained Petascale Performance (SPP) benchmarks	1.31 Petaflops/s
Sustained I/O Bandwidth performance	1.185 Terabytes/s
/home - IOR-write performance	104.3 GB/s
/project - IOR-write performance	100.0 GB/s

[i] There were also two facility outages during this period. While these two failures

/scratch - IOR-write performance	980.7 GB/s
Mean Metadata performance	>25,000 creates/s >30,000 deletes/s >40,000 stat() calls/s
IOR performance for writing 1 GB files on /scratch	
10,000 clients	774.3 GB/s
15,000 clients	767.2 GB/s
Reliability Measures	
Q4 2013 Number of Node Interrupts	156
Q4 2013 Mean Node Interrupts per day	1.70 failures per day
Q4 2013 Number of System Wide Outages for Hardware or Software	12 per quarter
Q4 2013 System Wide Average MTBF per quarter for system hardware and software	7.667 days

Categorizing the *most likely cause*[ii] of failures over a somewhat longer period of time indicates software plays the most significant role for both node interrupts and system wide outages. For Blue Waters system wide outages[9], software causes approximately 75% of the outages, even when facilities and human error outages are considered. This pattern is similar to a decade worth of data from NERSC for several large scale systems[10]. Software reliability has improved over the past decades, but not at a sufficient rate to make up for the challenges for software due to hardware scale and complexity that Petascale systems represent.

The software caused outages are even more significant because software outages typically have a longer MTTR and also wider impacts during recover time, such as the entire system being unavailable. Furthermore, repeat failure events are more likely with software based errors until they are diagnosed and resolved. First time, successful resolution is relatively low compared to hardware failures. This observation is across multiple systems over the at least a decade.

For system wide outages caused by software, the most likely cause for more than ½ the outages is the parallel, shared filesystem[iii]. Here caution is needed since sometimes (often?) the file system is the obvious cause of failure but in reality it is reacting and/or recovering from lower level failures in the interconnect, load imbalance, etc. Other major software components that cause system wide failures are resource managers and interconnect software layers.

[ii] Most likely cause is either a confirmed root cause or the most likely cause as determined by successful corrective actions, partial root cause analysis, categorizing successful "bug fixes" of other methods. It is not feasible to determine root cause for all errors without significant cost and effort.

[iii] This seems independent of the type of filesystem – whether

Impact for @Scale computing

As systems scale, not only will the number of hardware components increase, the number of software instructions operating simultaneously will increase at a similar rates. Because hardware design and testing methods are far ahead of software development for reliability it is reasonable to expect software failures will dominant all reliability issues at Exascale unless steps are taken. When one considers firmware as software the rate of software failures will further compound. Similarly, if one considers performance degradation and inconsistency as a form of failure, the software contribution to failure rates will dwarf hardware contributions.

Furthermore, the common approach to exascale programs is to have experienced hardware vendors develop proprietary hardware that is multi-purpose (e.g. target other product markets and uses as well as exascale) and supported by vendor(s). Hence, it is in vendors' self interest to invest in reliable designs, integration and testing, and indeed an entire industry exists for tools to assist with such design and hardware vendors consider their design simulation and emulation processes as strong competitive advantages.

But exascale software plans are predominately based on open-source software solutions that in some form assume community-based support and a minimum of vendor supplied proprietary software. Furthermore, a significant part of the software may not have much other commercial use. The open source approach does eventually drive towards high quality software but is often more reactive than proactive in up front software design and methods, and often emphasized features and function over reliability.

This future does not even take into consideration errors and failures in applications or other software tools.

Proposed Actions

In order to make exascale BDEC systems usable, particular by a wide range of applications, software reliability and resiliency has to increase at a much faster rate than just keeping up with hardware scale. Furthermore, since BDEC will expand the resiliency requirement for data stores, file systems and data transfer, file system software failures have to exponentially improve.

Some steps that should be undertaken in creating reliable BDEC exascale systems include:

1. Expanding investigations of software reliability and failure analysis for large scale systems.
2. Better categorization of software failure modes and models for system software failure modes.

3. Improve error reporting through hierarchies for independently developed software component– both to applications and in system logs – to better enable root cause analysis.
4. Institute automated software design methods for critical system software, including the file systems, that enable analytical and emulation verification and validation of both software modules and complete software sub-systems. Most hardware developers use advanced simulation and emulation not just for common cases but also for extreme (margin) cases to uncover error modes for completing implementation. Software development should have the same rigor and sophisticated methods.
5. Study the virtual organizations that participate in the creation and evolution of open source software to determine ways to improve the human processes. Since many of these virtual organizations are without formal hierarchy, new social incentives may be beneficial to influence behaviors.
6. In addition to functional testing of software, institute proactive performance and reliability testing as part of the BDEC exascale software development processes. Performance consistency, even in the face of sharing resources, should be an important requirement for software resiliency.
7. Develop new, formal storage models (such as the IEEE Mass Storage Model) so that validation and verification can be performance against model and design requirements rather than just black box testing.
8. Often recovery software paths are seldom executed until failures occur. These code pathways are also not optimized for good performance, and may be developed as after fact. Institute "margin" testing for software.
9. Institute research and development projects for new theory and implementations for efficient reliability and recovery methods at scale.
10. In addition to funding the research and development of software, fund, at equivalent levels, reliability and performability testing of the software as part of the BDEC exascale software development methodology. As is the case in most large, commercial software development project, the development and the testing functions should be separated so for every software institute/team there should be a parallel testing institute/team.
11. Initiate reliability and performability testing and analysis simultaneously with research and development activities. In other words, do not make verification and validation of software reliability an after thought.

References

[1] Bianca Schroeder, Garth A. Gibson, Understanding Disk Failure Rates: What does

[2] Gonzalez, Antonio, Scott Mahlke, Shubu Makherjee, Resit Sendag, Derek Chiou and Joshua Yi, *Reliability: Fallacy or Reality*, IEEE Micro, November-December 2007, p36-45, 0272-17322/07

[3] Gainaru, Ana, Franck Cappello, Marc Snir, William Kramer - *Failure prediction for HPC systems and applications: current situation and open issues*, International Journal of High Performance Computing, 2013

[4] Nagaraja, Kiran, Xiaoyan Li, Ricardo Bianchini, Richard P. Martin, Thu D. Nguyen *Using Fault Injection and Modeling to Evaluate the Performability of Cluster-Based Services*, Usenix Symposium on Internet Technologies (USITS 2003), March 26-28, 2003, Seattle, WA

[5] E. Pinheir, W. D. Weber, and L.A. Barroso. *Failure Trend in a large Disk Drive Population*. Google Inc. In Proc. of the 5th USENIX conf. (FAST'07), Feb. 2007

[6] William Kramer, *PERCU: A Holistic Method for Evaluating High Performance Computing Systems*, A Dissertation, University of California Berkeley, October 2008. http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-143.pdf

[7] Jeffrey S. Vetter, editor, *Contemporary High Performance Computing From Petascale toward Exascale*, Chapman and Hall/CRC, 2013, pages 339–366, Print ISBN: 978-1-4665-6834-1, eBook ISBN: 978-1-4665-6835-8

[8] The system wide events data is from Cray, NCSA and UIUC reports.

[9] Catello Di Martino, Fabio Baccanico, Zbigniew Kalbarczyk, Ravishankar Iyer, Joseph Fullop, William Kramer, Lessons *Learned From the Failure Analysis of a Petascale System: the Case of Blue Waters*, paper accepted for publication at the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014), Atlanta Georgia, June 2014

[10] Makhtarani, Akbar, William Kramer, Jason Hick, *Reliability Results of NERSC Systems*, LBNL Technical Report number LBNL-934480, June 15, 2008, http://www.osti.gov/servlets/purl/934480-gG3q6y/