An Operating System Resilient to DRAM Failures

Kurt B. Ferreira*[1], Kevin T. Pedretti[1], Ron Brightwell[1], Patrick Bridges[2], David Fiala[3], and Frank Mueller[3]

[1] Scalable System Software , Sandia National Laboratories†
[2] Department of Computer Science , University of New Mexico
[3] Department of Computer Science , North Carolina State University

Motivation

Concern is growing in the high-performance computing (HPC) community on the reliability of future extreme scale systems. With systems continuing to grow dramatically in node count and individual nodes also increasing in component count and complexity, large-scale systems are becoming less reliable. In fact, experts are predicting that failure rates may go from the current state of a handful a day [26, 25] to multiple failures an hour [2]. Recent studies have shown soft errors in main memory to be the source of many of these failures [13, 15]. With the predicted increase of memory density on future exascale systems [28] and expected power optimizations such as near-threshold supply voltages, the number of these failures is expected to dramatically increase.

Several methods have been developed to address these errors. Approaches include hardware-based techniques, such as single-bit error correction and double-bit detection (SEC-DED) and chipkill codes [9], as well as algorithm-based mechanisms that encode the correction mechanics directly into the application [12, 8, 6]. These mechanisms may, however, be insufficient at the elevated failure rates predicted for exascale systems, and most importantly, they may not protect the most important software running on a node - the operating system.

An operating system (OS) resilient to soft errors in memory is key to the scalability of exascale systems for a number of reasons. First, current operating systems are unable to recover from the vast majority of failures. Second, though the typical operating system only occupies a small portion of a system's total physical memory footprint, recent studies show substantially more errors in this region than the remainder of a system's memory [13]. Lastly, future HPC system software will need to continue running in the presence of memory failures if current application-based, forward error recovery mechanisms are to be successful. These forward-error recovery methods are theorized to have lower overheads and less wasted computation than current rollback/recovery mechanisms.

Our Position

In this paper we are advocating for augmenting the Kitten lightweight kernel [24] to make it resilient to DRAM failures. Kitten is a special-purpose, limited-functionality OS developed at Sandia National Laboratories that is designed for use on the compute nodes of massively parallel supercomputers. Its code base is derived from Linux, but is modified to minimize kernel-level functionality to only that needed for a set of mission-critical HPC applications and moves as much as possible into user-space. Kitten is similar to previous lightweight kernels (LWK) [23] such as SUNMOS [16, 22], Puma [5, 30], Cougar [4], Catamount [7, 29], and IBM's CNK [1]. Kitten, however, distinguishes itself from these prior LWKs by providing a combination of a Linux-compatible user environment [1], a more modern and extensible code base, and a virtual machine monitor capability via the Palacios virtual machine monitor [17] which allows full-featured guest operating systems to be loaded on-demand and executed with low overhead [14].

Kitten is an ideal target for this research; its focus on deterministic runtime behavior and simple data structures allow us to more easily reconstruct lost system state in many cases [10]. Also, as mentioned

*corresponding author: kbferre@sandia.gov

†Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

previously, Kitten allows us to leverage the Palacios high-performance virtual machine monitor which delivers machine-level virtualization functionality. As an example we consider errors in page-table memory. Like many OSs, errors in Kitten's page table memory are fatal, either killing the affected application or the entire node. However, Kitten's deterministic mapping of virtual to physical addresses would make it straightforward to determine corruption had occurred and simple to recreate the corrupted page table memory contents from the base physical address and length information stored in the address space region object. This could be difficult in a general purpose OS like Linux due to its demand paging scheme, where unpredictable physical addresses are assigned to virtual addresses at runtime. Extra redundant state would need to be stored, and furthermore it may be difficult or impossible to tell which page table values have become corrupted if hardware notification is not provided.

Related work

Resiliency and fault-tolerance has been identified by the Department of Energy and Department of Defense as one of the key fundamental challenges of extreme-scale computing. The majority of the work in this active research area has focused solely on the application and ignored the operating and runtime systems, which is the focus of this work. Essentially all of these approaches attempt to improve the performance of checkpoint/restart as it is the most widely used mechanism for fault-tolerance today. To the best of our knowledge, the only work similar to our resilient operating and runtime systems work is in the context of reliability for hostile environments, such as outer space and high radiation environments [27, 20, 18, 21, 19]. These methods typically have high runtime overheads [11] and it is unclear if they are appropriate for HPC.

In addition to the HPC application-based methods, a handful of researchers have been focusing on designing fault-tolerant userspace libraries for HPC systems that applications can use to construct algorithm-based resilience. An underlying assumption in these libraries is that the operating and runtime systems are resilient to failures or if not, an expensive restart of the OS must be done.

Assessment

We believe our approach in hardening the OS to DRAM failures will be critical for the scalability of future exascale-class systems. Additionally, the ability of the system software on a node to make progress in the presence of failures is required for emerging algorithm based fault-tolerant methods. How best to have applications take advantage of this resilient system software is still an open research question, though initial results appear promising [3].

Challenges addressed: This approach addresses the resilience challenge for exascale class systems identified by the Department of Energy and Department of Defense as one of the key fundamental challenges for extreme-scale computing. The ability of OSs and applications to continue in the presence of faults will be critical to the scalability of these systems.

Maturity: Lightweight kernels have been in production use on some of the largest systems in the world for nearly twenty years. The Kitten LWK is the latest in a long line of these LWKs and has been shown to be scalable on current parallel architectures [14].

Uniqueness: The fault-tolerance challenge addressed in this paper is unique to the post-petascale class of systems due to the unprecedented scale and power concerns inherent in these platforms

Novelty: The majority of the work in the area of fault tolerance for HPC has focused solely on the application and ignored the operating and runtime systems, the focus of this work. Essentially all current approaches attempt to improve the performance of checkpoint/restart, as it is the most widely used mechanism for fault-tolerance today.

Applicability: The results of this work would be applicable to any research area in which DRAM failures are common and protecting against them important.

Effort: The Kitten LWK is the latest in a long line of these LWKs and has been shown to be scalable on current scalable architectures [14]. Augmenting Kitten to handle failures will require several FTE's to target vulnerable OS subsystems, develop failure mitigation strategies and evaluate these strategies with actual capability workloads. Additionally, involvement of application developers to take advantage of these resilient systems will be required to ensure success.

References

[1] N.R. Adiga and et al. An overview of the BlueGene/L supercomputer. In *Supercomputing, ACM/IEEE 2002 Conference*, page 60, nov. 2002.

[2] Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Kerry Hill, Jon Hiller, Sherman Karp, Stephen Keckler, Dean Klein, Peter Kogge, Robert Lucas, Mark Richards, Al Scarpelli, Steven Scott, Allan Snavely, Thomas Sterling, R. Stanley Williams, and Katherine Yelick. Exascale computing study: Technology challenges in achieving exascale systems. `http://www.science.energy.gov/ascr/Research/CS/DARPAexascale-hardware(2008).pdf`, September 2008.

[3] Patrick Bridges, Mark Hoemmen, Kurt B Ferreira, Michel A Heroux, Philip Soltero, and Ron Brightwell. Cooperative application/os dram fault recovery. In: Fourth Workshop on Resiliency in High Performance Computing - in conjunction with the 17th International European Conference on Parallel and Distributed Computing, 2011.

[4] Ron Brightwell, Rolf Riesen, Keith Underwood, Trammell B Hudson, Patrick Bridges, and Arthur B Maccabe. A performance comparison of linux and a lightweight kernel. In *Proceedings 2003 IEEE International Conference on Cluster Computing*, pages 251–258, Hong Kong, China, December 2003.

[5] Ron Brightwell and Lance Shuler. Design and implementation of mpi on puma portals. In *Proceedings of the Second MPI Developer's Conference*, pages 18–25, South Bend, Indiana, July 1996.

[6] Greg Bronevetsky and Bronis de Supinski. Soft error vulnerability of iterative linear algebra methods. In *Proceedings of the 22nd Annual International Conference on Supercomputing*, ICS '08, pages 155–164, New York, NY, USA, 2008. ACM.

[7] William J. Camp and James L. Tomkins. The red storm computer architecture and its implementation. In *The Conference on High-Speed Computing: LANL/LLNL/SNL*, Salishan Lodge, Gleneden Beach, Oregon, April 2003.

[8] Zizhong Chen and J. Dongarra. Algorithm-based checkpoint-free fault tolerance for parallel matrix computations on volatile resources. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, April 2006.

[9] Timothy J. Dell. A white paper on the benefits of chipkill-correct ECC for PC server main memory. IBM Microelectronics Division, Nov. 1997.

[10] Kurt Ferreira, Kevin Pedretti, Patrick Bridges, Ron Brightwell, David Fiala, and Frank Mueller. Evaluating operating system vulnerability to memory errors. in Proceedings of the International Workshop on Runtime and Operating Systems for Supercomputers, 2012.

[11] David Fiala, Kurt B. Ferreira, Frank Mueller, and Christian Engelmann. A tunable, software-based DRAM error detection and correction library for HPC. In *Lecture Notes in Computer Science: Proceedings of the European Conference on Parallel and Distributed Computing (Euro-Par) 2011: Workshop on Resiliency in High Performance Computing (Resilience) in Clusters, Clouds, and Grids*, Bordeaux, France, Aug 2011. Springer Verlag, Berlin, Germany.

[12] Kuang-Hua Huang and Jacob A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Transactions on Computers*, C-33(6), June 1984.

[13] Andy A. Hwang, Ioan A. Stefanovici, and Bianca Schroeder. Cosmic rays don't strike twice: understanding the nature of DRAM errors and the implications for system design. In *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '12, pages 111–122, New York, NY, USA, 2012. ACM.

[14] John R. Lange, Kevin T. Pedretti, Trammell Hudson, Peter A. Dinda, Zheng Cui, Lei Xia, Patrick G. Bridges, Andy Gocke, Steven Jaconette, Michael Levenhagen, and Ron Brightwell. Palacios and kitten: New high performance operating systems for scalable virtualized and native supercomputing. In *IPDPS'10*, pages 1–12, 2010.

[15] Sheng Li, Ke Chen, Ming-Yu Hsieh, Naveen Muralimanohar, Chad D. Kersey, Jay B. Brockman, Arun F. Rodrigues, and Norman P. Jouppi. System implications of memory reliability in exascale computing. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 46:1–46:12, New York, NY, USA, 2011. ACM.

[16] Arthur B. Maccabe, Kevin S. McCurley, Rolf Riesen, and Stephen R. Wheat. SUNMOS for the Intel Paragon: A brief user's guide. In *Proceedings of the Intel Supercomputer Users' Group. 1994 Annual North America Users' Conference*, pages 245–251, June 1994.

[17] Northwestern University. Palacios: An os independent embeddable vmm. `http://v3vee.org/palacios`, March 10 2012.

[18] N. Oh, P.P. Shirvani, and E J McCluskey. Error detection by duplicated instructions in super-scalar processors. *Reliability, IEEE Transactions on*, 51(1):63–75, mar 2002.

[19] N. Oh, P.P. Shirvani, and E.J. McCluskey. Control-flow checking by software signatures. *Reliability, IEEE Transactions on*, 51(1):111–122, mar 2002.

[20] M. Rebaudengo, M.S. Reorda, M. Violante, and M. Torchiano. A source-to-source compiler for generating dependable software. In *Source Code Analysis and Manipulation, 2001. Proceedings. First IEEE International Workshop on*, pages 33–42, 2001.

[21] George A. Reis, Jonathan Chang, Neil Vachharajani, Ram Rangan, and David I. August. SWIFt: Software implemented fault tolerance. In *Proceedings of the international symposium on Code generation and optimization*, CGO'05, pages 243–254, Washington, DC, USA, 2005. IEEE Computer Society.

[22] Rolf Riesen, , Lee Ann Fisk, Chu Jong, Barney Maccabe, Kevin McCurley, Lance Shuler, Mack Stallcup, and David van Dresser. Sunmos, 1996.

[23] Rolf Riesen, Ron Brightwell, Patrick G. Bridges, Trammell Hudson, Arthur B. Maccabe, Patrick M. Widener, and Kurt Ferreira. Designing and implementing lightweight kernels for capability computing. *Concurrency and Computation: Practice and Experience*, 21(6):793–817, April 2009.

[24] Sandia National Laboratory. Kitten lightweight kernel. `https://software.sandia.gov/trac/kitten`, March 10 2012.

[25] Bianca Schroeder and Garth A. Gibson. A large-scale study of failures in high-performance computing systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN2006)*, June 2006.

[26] Bianca Schroeder and Garth A Gibson. Understanding failures in petascale computers. *Journal of Physics: Conference Series*, 78(1):012022, 2007.

[27] P.P. Shirvani, N.R. Saxena, and E.J. McCluskey. Software-implemented EDAC protection against SEUs. *Reliability, IEEE Transactions on*, 49(3):273 –284, sep 2000.

[28] Horst Simon. Exascale challenges for the computational science community. Technical report, Lawrence Berkeley National Laboratory and UC Berkeley, Oct. 2010.

[29] James L Tomkins, Ron Brightwell, William J Camp, Sudip Dosanjh, Suzanne M Kelly, Paul T Lin, Courtenay T Vaughan, John Levesque, and Vinod Tipparaju. The red storm architecture and early experiences with multi-core processors. *International Journal of Distributed Systems and Technologies*, 1(2):74–93, May 2010.

[30] Stephen R. Wheat, Arthur B. Maccabe, Rolf Riesen, David W. van Dresser, and T. Mack Stallcup. PUMA: An operating system for massively parallel systems. *Scientific Programming*, 3:275–288, 1994.