SAND2013-3089C

A Simulation Infrastructure for Examining the Performance of Resilience Strategies at Scale

Kurt B. Ferreira*
Scalable System Software
Sandia National Laboratories
kbferre@sandia.gov

Scott Levy and Patrick G. Bridges
Department of Computer Science
University of New Mexico
{slevy|bridges}@cs.unm.edu

ABSTRACT

Fault-tolerance is a major challenge for many current and future extreme-scale systems, with many studies showing it to be the key limiter to application scalability. While there are a number of studies investigating the performance of various resilience mechanisms, these are typically limited to scales orders of magnitude smaller than expected for next-generation systems and simple benchmark problems. In this paper we show how, with very minor changes, a previously published and validated simulation framework for investigating application performance of OS noise can be used to simulate the overheads of various resilience mechanisms at scale. Using this framework, we compare the failure-free performance of this simulator against an analytic model to validate its performance and demonstrate its ability to simulate the performance of two popular rollback recovery methods on traces from real HPC workloads, showing how performance can vary dramatically both with scale and the communication behavior of the application.

1. INTRODUCTION AND BACKGROUND

Reliability is a key challenge in the design of future extreme-scale high performance computing (HPC) systems. As these systems continue to grow dramatically in size and complexity, they are becoming less reliable. In fact, failures are predicted to go from the current state of several failures per day [17, 27–29, 31] to multiple failures per hour [7].

Fault-tolerance for HPC systems and distributed systems in general has been studied for several decades. Although many techniques have been proposed, *checkpoint/restart* remains the most commonly used technique [6]. The prevalence of checkpoint/restart is due to a number of factors: failure is a relatively rare event, checkpointing is easy to integrate in an application's computation because applications are generally self-synchronizing, and application state can be saved and restored much more quickly than a given system's mean time to interrupt (MTTI). These factors have kept the overheads of traditional checkpoint/restart on current systems relatively low.

The assumptions on which checkpoint/restart rests are unlikely to be true in future generation systems, however [8,20, 27], prompting a wide range of recent research optimize rollback/recovery protocols. These optimizations range from memory-based coordinated checkpointng schemes [19,21,30] to asynchronous checkpointing schemes with message logging [1, 2, 11, 12, 24].

All of these techniques involve complex trade-offs, and understanding these trade-offs at the scales expected in next-generation systems is vital to evaluating their suitability. Unfortunately, it has been difficult to evaluate proposed resilience methods at the scales expected on future systems, and so their true overheads are not known. Additionally, evaluations have generally been limited to microbenchmarks rather than production workloads. Both of these are because system time on leadership-class machines is difficult to obtain, and future systems are expected to be much larger than any machine currently in existence.

In this work, we show how the performance of new resilience techniques can be simulated at large scale on production workloads through minor changes to an existing validated simulation framework. In particular, we show that novel resilience techniques can generally be accurately modeled as *detours* in a simulator originally designed for investigating the impact of OS noise (or *jitter*) on application performance. We demonstrate this approach by validating it against an analytic coordinated checkpointing model [3]. We also show how it can be used to evaluate the performance of two popular rollback/recovery methods in scenarios that were previously not easy to evaluate – showing that the application's communication behavior can have a dramatic impact on a method's efficiency. Due to space limitations, we focus our evaluation in this paper on failure-free performance. However, the framework is also capable of simulating performance in a faulty environment.

The organization of this paper is as follows: in the following section we describe the design and implementation of our contributions to this exiting simulation framework as

*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.

well as how we evaluate its performance. In Section 3 we validate the failure-free performance prediction capabilities of this framework against a simple analytic model. In this section we also evaluate the performance of two popular rollback/recovery methods on real production HPC workloads, showing how an application's communication properties can significantly influence efficiency. We discuss related work in Section 4. We conclude in Section 5, summarizing our contributions and discussing avenues of future research.

2. APPROACH

2.1 Overview

To evaluate the impact of resilience mechanisms on HPC applications at scale, we augment `LogGOPsim`, a previously published [15] and freely available [32] simulator. `LogGOPsim` is based on the popular LogP model. LogP and its variants have a long history of accurately predicting the performance of large-scale parallel applications and algorithms. The simulation framework consists of three major components: a trace collector, a schedule generator, and an optimized LogGOPS discrete-event simulator. The trace collector records the actual MPI communication of the target application. The schedule generator uses the MPI traces to generate a schedule that captures the control- and data-flow of the application while preserving the happens-before relationships within the application. The LogGOPS discrete-event simulator reads the generated schedule, performs a full LogGOPS simulation and reports the end times for each process.

This validated simulation framework was developed to simulate applications at scale. It has the ability to extrapolate traces collected on smaller scale systems. This allows for the simulation of platforms larger than those currently in existence while keeping the same communication characteristics (equivalent to weak-scaling of the application). This powerful framework has been shown to be capable of simulating a single collective operation over up to 10 million processes and applications on up to 64 thousand processes on a single CPU. It has been used to evaluate the performance of collective communications [16] and the impact of OS noise [14] on large-scale applications. A detailed study of the simulation framework and its functionality is presented in [15].

The key insight that allows us to use the `LogGOPsim` simulator is that resilience mechanisms (e.g., writing checkpoints, restart, rework) can be modeled as CPU *detours*: cycles used for something besides the application, similar to OS noise. One key difference between OS noise and these resilience detours that our work must address is that "noise" events must be replayed synchronously with the application communication/computation pattern rather than in the asynchronous manner of typical OS noise.

2.2 `Libsolipsis`

We model resilience in `LogGOPsim` using a new library, `libsolipsis` that generates CPU detours based on a specified resilience mechanism and the application's communication pattern. The library links to the application using the MPI profiling layer, intercepting all MPI calls. The output of this library is a per-process detour file that can be provided as input to `LogGOPsim`. The detour files contain the timestamp and the duration of each of the resilience detours. Detours generated by this library might include writing a checkpoint, writing a message log entry, or simulating a failure[1].

For the purpose of this work, we focus on the libraries' ability to emulate failure-free performance of two popular resilience mechanisms: coordinated checkpointing and asynchronous checkpointing with message logging [6]. We focus on these on these two methods because coordinated checkpoint/restart is currently the most popular approach and asynchronous checkpointing has been proposed as a low-overhead checkpoinint option for future extreme-scale systems.

For asynchronous checkpointing with message logging, our library writes the timestamp and the duration of the local checkpoints. In addition, the library must handle the logging of messages to stable storage. For pessimistic message logging [6], we modify the CPU overhead parameter (o in the LogGOPS model) for send operations (o_s) to account for the write to stable storage. For optimistic message logging, we write the time and duration of writing to the message log, if any, in the detour file. The `LogGOPsim` simulator uses a single detour file to simulate asynchronous checkpointing across all of the nodes in the system.

For coordinated checkpoint/restart, the library only writes the timestamp and the duration of each checkpoint taken by the application. When the simulation is run, we use the "`--noise-cosched`" option of the `LogGOPsim` simulator. This option ensures all detour files are co-scheduled on all processors, thereby simulating coordinated checkpoint/restart.

Although we only present failure-free performance of resilience mechanisms here, the library can also simulate node failure. To simulate failure, the library generates failure times for each node from a random distribution based on a per-node mean time between failure (MTBF). When a failure is generated, the library adds a detour event that includes the the time required to restart from the last checkpoint and the time required for rework (i.e., the time since the last checkpoint). The `LogGOPsim` simulator will ensure that all communication in the trace file that depends on the failed node will be delayed until the node has "recovered". The library can also simulate hybrid or hierarchical approaches (e.g., [12]) that apply one resilience approach within node clusters and a different resilience approach between the clusters.

2.3 Test Environment

To validate the performance of this simulation framework and to motivate the importance of this framework, we evaluated the performance of two widely used rollback/recovery based mechanisms. We present results for: two key HPC applications, CTH [5], a shock physics simulation code, and LAMMPS [22, 26], a molecular dynamics simulation code; and one microbechmark HPCCG [25], a conjugate gradient solver. All of these applications were developed at Sandia National Laboratories. They represent important HPC modeling and simulation workloads. They use different computational techniques, are frequently run at very large scale,

[1]In the case of a failure, the duration of the detour includes both the restart and rework time on the failed node.

sometimes for weeks at a time, and are key simulation applications for the US Department of Energy. These applications also contain easily-adaptable checkpoint mechanisms that will be used in this work.

The application and resilience traces for this work were collected by running these three applications on 128 nodes of a Cray XE6 machine. The simulations were run on the same platform. For the simulation runs, we use the `LogGOPS` parameters as measured using the `netgauge` benchmark [13,33] on the XE6 platform. In the case of coordinated checkpointing, we simulate an aggregate filesystem bandwidth of 256 GB/s. For the asynchronous case, each node can write to the filesystem at 2 GB/s. Finally, for all of the results in this paper, we simulate a low-noise environment (e.g., the BlueGene/L [4, 34] family of supercomputers) by only injecting detours arising from the resilience mechanisms.

3. RESULTS

We demonstrate this approach in two different ways: First, we validate the simulation framework described in the previous section against a simple analytic model of coordinated checkpoint performance. Second, we demonstrate its ability to simulate the impact of different resilience strategies on system performance in failure-free cases at scales and using workloads that were previously challenging to evaluate.

3.1 Validation Against an Analytic Model

We first validate the failure-free coordinated checkpointing performance of this simulation framework against a simple analytic model. Equation 1 presents a simple failure-free analytic model of application performance with coordinated checkpointing and shared stable storage. The total wall clock time is computed as the time required to solve the problem without interruption plus the time to take each of the required checkpoints.

$$T_w = T_s + \frac{T_s}{\tau} \times \delta \tag{1}$$

T_w is the wall clock time, in this case without failures, T_s is the solve time of the application without any resilience mechanism, τ is the checkpoint interval [3], and δ is the checkpoint commit time (time to write one checkpoint). For coordinated checkpointing to shared stable storage, we can express the checkpoint commit time as:

$$\delta = \frac{N * ||c_{avg}||}{\beta} \tag{2}$$

where N is the number of nodes, $||c_{avg}||$ is the average checkpoint size per node, and β is the aggregate write bandwidth to stable storage.

Figure 1 compares this simple model with the output of the simulator. The model and the simulator use identical values for T_s, τ, and δ. For the simulator, we use the LogGOPS parameters as measured on a Cray XE6 platform using `netgauge`. The simulation data is based on a communication trace from CTH. In the figure we see that the simulation framework wall clock time differs by approximately 10% for these parameters. Overall, however, the simulator accurately matches the shape of the performance curve of this simple model with relatively low error. This suggests that the simulation framework can be used to accurately predict application performance of resilience mechanisms at scale.

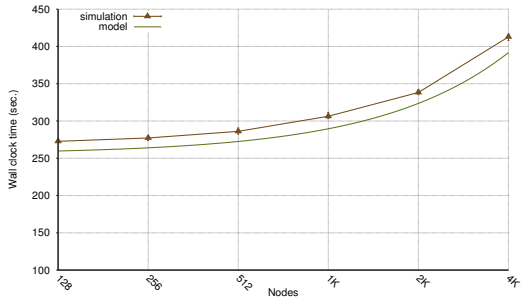


Figure 1: Comparison of the the analytic model described in Equation 1 with the simulator for coordinated checkpointing to stable storage in a failure-free environment. Both the model and the simulator use identical values for T_s, τ, and δ

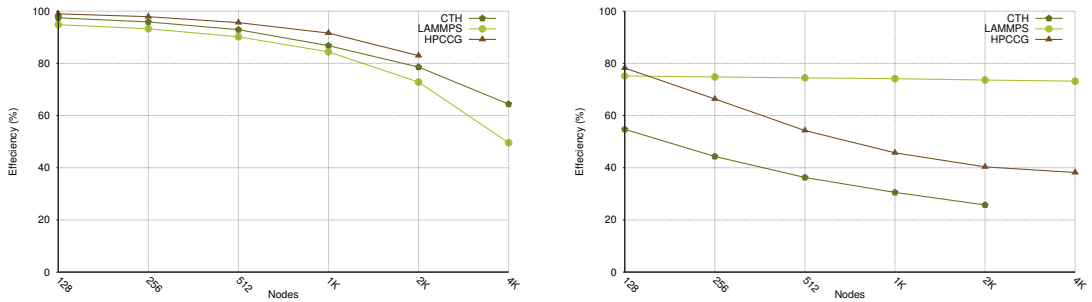
3.2 Rollback/Recovery Performance at Scale

Because resilience interacts with application communication and computation, different resilience mechanisms may have unexpected performance impacts on applications at scale. We note that this is similar to the previously unexpected impact of OS noise `LogGOPsim` was originally designed to study. In this section, we demonstrate the use of the modified simulation framework to analyze such interactions for two popular rollback/recovery mechanisms for real application traces at non-trivial scales.

Specifically, we measure the failure-free performance at scale of coordinated checkpointing to shared stable storage with uncoordinated checkpointing to node-local storage for LAMMPS, CTH, and HPCCG. The metric we use for this evaluation is application efficiency. Efficiency is defined as the percentage of time spent in the application performing computation for the problem. This excludes time spent on the resilience mechanism, rework, dealing with failures, etc. For example, suppose we have an application whose solve time is 90 time units without interruption. If the addition of a resilience mechanism causes the application's time-to-solution to increase to 100 time units, the efficiency of the application would be $\frac{90}{100} \times 100$, or 90%.

In our tests, we take significantly more checkpoints than would we expected at these scales. The reasoning behind this is that we intend to use the simulator to evaluate the influence of scale and communication patterns on application performance. This may give us some insight into application performance in a large-scale, failure-prone environment. We also keep the checkpoint interval constant independent of node count. Lastly, the coordinated checkpointing checkpoints are written to a shared store with an aggregate checkpoint commit bandwidth of 256GB/sec. For uncoordinated checkpointing we assume each node has a checkpoint commit bandwidth of 2GB/sec.

Figure 2 shows the efficiency of these two resilience techniques using production workloads LAMMPS and CTH, and the microbenchmark HPCCG. Each of the techniques in



(a) Coordinated Checkpointing (b) Uncoordinated Checkpointing

Figure 2: Coordinated and Uncoordinated checkpointing efficiency (the percent of time spent performing work for the application and and not the resilience mechanism) using the simulator for CTH, LAMMPS, and HPCCG. Each of these mechanisms is using the same checkpoint interval and checkpoint size per process. These simulations show performance an environment where checkpointing needs to be done frequently. For coordinated checkpointing checkpoints are written to a shared aggregate checkpoint commit bandwidth of 256GB/sec. For uncoordinated checkpointing we assume each node has a checkpoint commit bandwidth of 2GB/sec. We see from these simulations that coordinated checkpointing has a significant negative impact on application efficiency the grows as systems increase in size. We also see that asynchronous checkpointing has a significant negative impact on application efficiency for CTH and HPCCG. Although the trend suggests that the rate of decrease in efficiency tapers off for very large systems, there is a significant decrease in efficiency for systems that are larger than the small systems that are typically available for evaluation. In contrast, LAMMPS exhibits only nominal decreases in application efficiency.

these figures are using the same checkpoint interval and checkpoint size per process. For the coordinated checkpointing results in Figure 2(a), we assume all checkpoints are written to shared a shared stable storage which has an aggregate commit bandwidth of 256GB/sec. For the uncoordinated results in Figure 2(b), we assume each node has a checkpoint bandwidth of 2GB/sec. For the coordinated results, we see the predictable dip in efficiency at larger node counts. This dip occurs for all applications and is generally independent of an applications communication pattern. This dip in efficiency is due to the fact that the nodes write to shared storage. As node counts increase, so does data being concurrently written to the shared storage, thereforie performance is degraded. In ranges outside of those specified here, efficiency drops precipitously, rapidly approaching 0% efficiency.

The uncoordinated results in this figure are more interesting. First, the overheads of the uncoordinated approach are much higher at smaller node counts than coordinated. Also, the applications communication pattern greatly influences the efficiency of this technique. For example, CTH, which does a good deal more bulk data transfer and collective communication, sees much lower efficiency than HPCCG. And both of these workloads do much more communication and collective communication than LAMMPS, which sees nearly constant overheads from this uncoordinated approach over the range tested.

These results point directly to the importance of this simulation framework as a number of factors must be considered when determining the most efficient resilience technique. In previous work, scale is typically the only factor considered. As we can see, both scale and an application communication behavior must also be considered.

4. RELATED WORK

Significant effort has been devoted to developing strategies that reduce the overhead of traditional coordinated checkpointing as systems grow larger. However, evaluation of these approaches has been on systems that are significantly smaller than even today's largest systems. Our approach allows for a more thorough evaluation of how these proposed improvements scale as systems grow toward exascale.

Many approaches have been proposed that eliminate the coordination overhead by allowing processes to checkpoint independently. For example, Bosilca et al. [1] propose an elaborate solution using uncoordinated checkpointing and pessimistic message logging. They evaluate their approach on a 130-node system and present results for applications running on 9, 16 and 25 nodes. The authors extend this work to use sender-based message logging in [2]. Their evaluation is conducted on a 32-node cluster. Guermouche et al. [11] leverage send-determinism to reduce the number of messages that must be logged. They evaluate their approach on several systems, the largest of which is composed of 1024 nodes.

A common problem in coordinated checkpointing is contention for access to the global filesystem. Many solutions to reduce this contention have been proposed. Moody et al. [19] propose multi-level checkpoints. This approach stores the most recent checkpoints on the compute nodes. Older checkpoints are moved to global filesystem. They show that, in most cases, recovery from the failure of a single node can be accomplished with local checkpoint data. To evaluate the effectiveness of their approach they use several systems, the largest of which was composed of 1024 nodes.

Our approach builds upon efforts to understand the effects of system noise. Hoefler et al. [14] use the `LogGOPsim` simulator

to explore the effect of recorded system noise on the performance of several scientific applications. Ferreira et al. [9] inject synthetic noise in a low-noise environment to characterize the effect of frequency and duration of noise events on several production workloads. Although these efforts form the foundation of our contribution, we model a distinct set of behaviors in distributed systems.

5. CONCLUSIONS AND FUTURE WORK

In this work, we presented a novel method for simulating the impact of resilience techniques on production workloads running on large-scale systems. In particular, we showed that the behavior of resilience techniques can be accurately modeled as detours in a simulator originally designed for investigating the impact of OS noise on application performance. We validated the predictive ability of this simulation framework by comparing it to an analytic coordinated checkpointing model [3]. Finally, we used this framework to evaluate the performance of two popular rollback/recovery methods on production HPC workloads. Our results illustrate that the communication properties of an application can significantly alter the impact of resilience strategies on application efficiency. For example, the effect of using uncoordinated checkpointing on the efficiency of LAMMPS is significantly different than its impact on the efficiency of CTH.

While the results in this work are important, there are several avenues of promising future work. First, while the coordinated and uncoordinated checkpoint strategies considered in this work are important data points, there are a number of optimizations to these approaches that attempt to significantly improve application performance [8, 10–12, 18, 19, 23, 24]. Integrating these approaches into `libsolipsis` will be important for evaluating the efficiency of resilience mechanisms at scale. Second, in this work we focused solely on failure-free application performance. Evaluating performance in the presence of failures will be key to determining the proper resilience strategy for extreme-scale systems. Third, we need to evaluate performance for the larger scale systems expected in the future. Additionally, `LogGOPsim` allows us to investigate how various LogP model parameters can influence performance. This might provide insight on which architectural features would improve performance of future systems. Lastly, `libsolipsis` is in the process of public release at Sandia National Laboratories. Once this process is complete, we hope the `LogGOPsim` developers see the merit of this work and decide to distribute it with their simulation framework.

6. REFERENCES

[1] G. Bosilca, A. Bouteiller, F. Cappello, S. Djilali, G. Fedak, C. Germain, T. Herault, P. Lemarinier, O. Lodygensky, F. Magniette, V. Neri, and A. Selikhov. MPICH-V: Toward a scalable fault tolerant mpi for volatile nodes. In *Conference on High Performance Networking and Computing (SC2002)*, pages 1–18, Baltimore, MD, november 2002.

[2] A. Bouteiller, F. Cappello, T. Herault, G. Krawezik, P. Lemarinier, and F. Magniette. MPICH-V2: a fault tolerant MPI for volatile nodes based on pessimistic sender based message logging. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, New York, NY, USA, 2003. ACM.

[3] J. T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Gener. Comput. Syst.*, 22(3):303–312, 2006.

[4] K. Davis, A. Hoisie, G. Johnson, D. J. Kerbyson, M. Lang, S. Pakin, and F. Petrini. A performance and scalability analysis of the bluegene/l architecture. In *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 41. IEEE Computer Society, 2004.

[5] J. E. S. Hertel, R. L. Bell, M. G. Elrick, A. V. Farnsworth, G. I. Kerley, J. M. McGlaun, S. V. PetneY, S. A. Silling, P. A. Taylor, and L. Yarrington. CTH: A software family for multi-dimensional shock physics analysis. In *Proceedings of the 19th International Symposium on Shock Waves*, pages 377–382, July 1993.

[6] E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Comput. Surv.*, 34(3):375–408, 2002.

[7] K. B. et al. Exascale computing study: Technology challenges in achieving exascale systems. `http://www.science.energy.gov/ascr/Research/CS/DARPAexascale-hardware(2008).pdf`, Sept. 2008.

[8] K. Ferreira, R. Riesen, P. Bridges, D. Arnold, J. Stearley, J. H. L. III, R. Oldfield, K. Pedretti, and R. Brightwell. Evaluating the viability of process replication reliability for exascale systems. In S. Lathrop, J. Costa, and W. Kramer, editors, *SC*. ACM, Nov. 2011.

[9] K. B. Ferreira, P. Bridges, and R. Brightwell. Characterizing application sensitivity to os interference using kernel-level noise injection. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, page 19. IEEE Press, 2008.

[10] K. B. Ferreira, R. Riesen, R. Brightwell, P. G. Bridges, and D. Arnold. Libhashckpt: Hash-based incremental checkpointing using GPUs. In *Proceedings of the 18th EuroMPI Conference*, Santorini, Greece, September 2011 [to appear].

[11] A. Guermouche, T. Ropars, E. Brunet, M. Snir, and F. Cappello. Uncoordinated checkpointing without domino effect for send-deterministic MPI applications. In *International Parallel Distributed Processing Symposium (IPDPS)*, pages 989–1000, may 2011.

[12] A. Guermouche, T. Ropars, M. Snir, and F. Cappello. HydEE: Failure containment without event logging for large scale send-deterministic mpi applications. In *IPDPS*, pages 1216–1227. IEEE Computer Society, 2012.

[13] T. Hoefler, T. Mehlan, A. Lumsdaine, and W. Rehm. Netgauge: A Network Performance Measurement Framework. In *Proceedings of High Performance Computing and Communications, HPCC'07*, volume 4782, pages 659–671. Springer, Sep. 2007.

[14] T. Hoefler, T. Schneider, and A. Lumsdaine. Characterizing the Influence of System Noise on Large-Scale Applications by Simulation. In *International Conference for High Performance Computing, Networking, Storage and Analysis*

(SC'10), Nov. 2010.

[15] T. Hoefler, T. Schneider, and A. Lumsdaine. LogGOPSim - Simulating Large-Scale Applications in the LogGOPS Model. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 597–604. ACM, Jun. 2010.

[16] T. Hoefler, C. Siebert, and A. Lumsdaine. Group Operation Assembly Language - A Flexible Way to Express Collective Communication. In *ICPP-2009 - The 38th International Conference on Parallel Processing*. IEEE, Sep. 2009.

[17] A. A. Hwang, I. A. Stefanovici, and B. 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*, pages 111–122. ACM, 2012.

[18] D. Ibtesham, D. Arnold, P. G. Bridges, K. B. Ferreira, and R. Brightwell. On the viability of compression for reducing the overheads of checkpoint/restart-based fault tolerance. *2012 41st International Conference on Parallel Processing*, 0:148–157, 2012.

[19] A. Moody, G. Bronevetsky, K. Mohror, and B. R. de Supinski. Design, modeling, and evaluation of a scalable multi-level checkpointing system. In *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '10)*, pages 1–11, 2010.

[20] R. A. Oldfield, S. Arunagiri, P. J. Teller, S. Seelam, M. R. Varela, R. Riesen, and P. C. Roth. Modeling the impact of checkpoints on next-generation systems. In *24th IEEE Conference on Mass Storage Systems and Technologies*, pages 30–46, Sept. 2007.

[21] J. S. Plank, Y. B. Kim, and J. J. Dongarra. Algorithm-based diskless checkpointing for fault tolerant matrix operations. In *Twenty-Fifth International Symposium on Fault-Tolerant Computing. Digest of Papers*, pages 351–360, Pasadena, CA, USA, June 1995. Los Alamitos, CA, USA : IEEE Comput. Soc. Press, 1995.

[22] S. J. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal Computation Physics*, 117:1–19, 1995.

[23] R. Riesen, K. Ferreira, D. Da Silva, P. Lemarinier, D. Arnold, and P. G. Bridges. Alleviating scalability issues of checkpointing protocols. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 18:1–18:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[24] T. Ropars, A. Guermouche, B. Uçar, E. Meneses, L. V. Kalé, and F. Cappello. On the use of cluster-based partial message logging to improve fault tolerance for mpi hpc applications. In E. Jeannot, R. Namyst, and J. Roman, editors, *Euro-Par (1)*, volume 6852 of *Lecture Notes in Computer Science*, pages 567–578. Springer, 2011.

[25] Sandia National Laboratory. Mantevo project home page. `https://software.sandia.gov/mantevo`, Apr. 10 2010.

[26] Sandia National Laboratory. LAMMPS molecular dynamics simulator. `http://lammps.sandia.gov`, Apr. 10 2013.

[27] B. Schroeder and G. A. Gibson. A large-scale study of failures in high-performance computing systems. In *International Conference on Dependable Systems and Networks (DSN)*, June 2006.

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

[29] B. Schroeder, E. Pinheiro, and W.-D. Weber. Dram errors in the wild: a large-scale field study. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 193–204. ACM, 2009.

[30] L. M. Silva and J. G. Silva. An experimental study about diskless checkpointing. In *24th EUROMICRO Conference*, pages 395 – 402, Vasteras, Sweden, August 1998. IEEE Computer Society Press.

[31] V. Sridharan and D. Liberty. A study of dram failures in the field. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 76:1–76:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[32] T. Hoefler. LogGOPSim - A LogGOPS (LogP, LogGP, LogGPS) Simulator and Simulation Framework. `http://www.unixer.de/research/LogGOPSim/`, Apr. 10 2013.

[33] T. Hoefler. Netgauge - a network performance measurement toolkit. `http://www.unixer.de/research/netgauge/`, Apr. 10 2013.

[34] T. B. Team. An Overview of the BlueGene/L Supercomputer. In *2002 ACM/IEEE conference on Supercomputing (Supercomputing '02)*, pages 1–22. IEEE Computer Society Press, 2002.