Low Design-Risk Checkpointing Storage Solution for Exascale Supercomputers

Nilmini Abeyratne
University of Michigan
Ann Arbor, Michigan 48109
sabeyrat@umich.edu

Trevor Mudge
University of Michigan
Ann Arbor, Michigan 48109
tnm@umich.edu

ABSTRACT

This work presents a checkpointing solution for exascale supercomputers that employs commodity DRAM and SSD devices that pose a low design risk compared to solutions that use emerging non-volatile memories.

The proposed local checkpointing solution uses DRAM and SSD in tandem to provide both speed and reliability in checkpointing. A Checkpoint Location Controller (CLC) is implemented to monitor the endurance of the SSD and the performance loss of the application and to decide dynamically whether to checkpoint to the DRAM or the SSD. The CLC improves both SSD endurance and application slowdown; but the checkpoints in DRAM are exposed to device failures. To design a reliable exascale memory, a low latency ECC is added to correct all errors due to bit/pin/column/word faults and also detect errors due to chip failures, and a second Chipkill-Correct level ECC is added to protect the checkpoints residing in DRAM.

1. INTRODUCTION

Aggregate failure rates of millions of components along with software failures, environmental problems, and human errors will result in frequent failures in exascale supercomputers [1]. Overcoming these failures requires a fast and reliable checkpoint/restart framework. Usually, checkpoints are made to a non-volatile storage such as a hard disk, but increasingly, solid-state drives (SSDs) are replacing hard disks because they provide higher read/write bandwidth, lower power consumption, and better durability [2]. The question becomes whether SSDs are sufficient for storing checkpoints or if we should wait for emerging memory technologies.

Emerging non-volatile memory technologies such as phase change memory, memristors, and STT-RAM have been suggested to provide fast and reliable checkpointing [3, 4, 5]. These technologies are almost as fast as DRAM (10-300ns), yet reliable like storage. However, they are not dense or cost-efficient like flash. Although Intel's 3D XPoint is expected to cost half of DRAM [6], recent innovations in 3D NAND-flash such as stacking 48 layers will only cheapen flash. Furthermore, flash devices have well understood failure patterns and strong ECC codes to protect them [7]. Since the U.S. Department of Energy's Exascale Computing Initiative plans to deploy exascale computing platforms by 2023 [8], the designs for them will have to be finalized 3-4 years prior, similar to plans for Summit (2018) and Aurora (2018-2019) supercomputers that were completed by 2015. Commercial availability and maturity of both DRAM and NAND-flash prove them a low-risk option sufficient for at least the first generation of exascale systems, if used correctly.

2. PROBLEM STATEMENT

Local checkpointing to local storage have stemmed from a need to avoid the slowdown resulting from transferring checkpoints to the remote parallel file system (PFS). Existing local checkpointing solutions either checkpoints to the DRAM or to the SSD storage device. On one hand, DRAM is fast (50ns) but loses the checkpoint after a reboot. Furthermore, DRAM's limited capacity not only limits the size of the largest checkpoint that can be made but also limits the amount of usable memory for applications. On the other hand, SSDs are reliable and capacious but slow and have low endurance (about 10^4-10^5 program/erase cycles). SSD manufacturers employ various tricks such as DRAM buffers and sophisticated wear-leveling to extend lifetime. Currently, SSDs on the market are guaranteed a lifetime of 3-5 years with a cap on the total number of terabytes that can be written [9]. Nevertheless, writing gigabyte-sized checkpoints several times a day to the SSD can take a toll on its endurance.

When using SSD flash memory for checkpointing, reducing the checkpoint size or frequency remain the most effective ways to stretch its lifetime. To this end, we implement a system that selectively checkpoints to a DRAM in order to reduce the number of writes to the SSD thereby lengthening its useful lifetime. To accomplish this task, we implement a Checkpoint Location Controller that i) estimates SSD lifetime, ii) estimates application's performance loss, and iii) monitors checkpoint size. The CLC detects checkpointing frequencies that lead to SSD lifetime falling under the typical manufacturer's guarantee of 5 years, and reduces these frequencies by redirecting some checkpoints to the DRAM.

The hybrid DRAM-SSD solution merges the benefits of both DRAM and SSD: namely, speed and reliability. Furthermore, checkpointing to the DRAM helps to reduce SSD wearout. However, DRAM is prone to transient errors and checkpoints corrupted by them cannot be used for recovery. For that reason, a dual mode ECC memory system is proposed to protect regular application data with a normal ECC algorithm and checkpoint data with a strong ECC algorithm. The normal ECC, which is in the critical path of memory accesses, is designed to have small decoding latency and to correct frequently occurring small errors so as to avoid frequent recovery by checkpoints. The strong ECC is designed to strongly protect the checkpoint without modifying the DRAM devices. During restart, an attempt is always made to recover from the checkpoint in ramdisk. However, if the ECC logic signals an uncorrectable memory error, then the entire ramdisk checkpoint is discarded and the backup checkpoint file in the SSD is retrieved.

3. CONTRIBUTIONS

This work makes the following contributions:

- **A low-risk exascale memory system.** Commodity DRAMs and SSDs are used to create a low design-risk checkpointing solution and prove that system designers do not have to wait until newer non-volatile memory technologies are ready.
- **Hybrid DRAM-SSD checkpointing**. A hybrid mechanism for local checkpointing that uses both DRAM and SSD flash memory is implemented to achieve speed and reliability.
- **SSD-lifetime-aware checkpoint controller.** An intelligent Checkpoint Location Controller (CLC) is created to decide when to checkpoint to the SSD considering its endurance decay and performance degradation.
- **Dual-ECC memory.** A dual-mode ECC memory is implemented which has a normal mode to protect regular application data and a strong mode to protect the DRAM checkpoint. ECC-protected checkpoints ensure error-free restarts at recovery.

4. PROPOSED DESIGN

4.1 Hybrid DRAM-SSD Checkpointing

An overview of the hybrid DRAM-SSD checkpointing solution is presented in Figure 1. All compute nodes have main memory consisting of x4 ECC-DRAM devices and one SSD flash memory device. The hybrid system writes to both DRAM and SSD on the local node. It can exist within a hierarchical framework where global checkpoints are still written to the remote PFS.

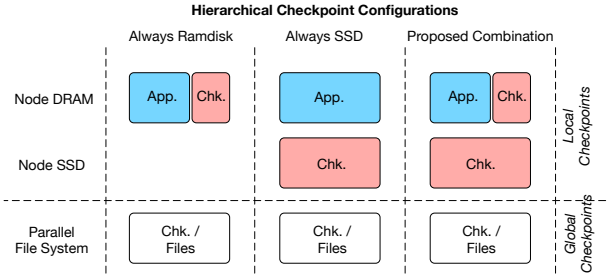


Figure 1: The proposed idea utilizes both commodity DRAM and commodity SSD for checkpoints.

4.2 Checkpoint Location Controller (CLC)

A controller, CLC, is introduced to monitor checkpointing behavior and direct checkpoints to their appropriate storage location. The CLC writes checkpoints to the ramdisk or to the SSD by setting the file path to point to either the ramdisk or the SSD.

The CLC makes its decision based on three parameters. **1) Lifetime Estimation:** The endurance of an SSD is described by bytes written (e.g. PBW–petabytes written), which is the total amount of writes that it can withstand without wearing out. To measure endurance decay, the CLC checks if the checkpoint bandwidth to the SSD by the application is too high such that the SSD will wear out in less than 5 years. If so, the checkpoint is re-directed to the ramdisk. **2) Performance Loss Estimation:** The CLC determines whether the dynamic performance loss due to checkpointing exceeds a specified bound (e.g. 10%), and if so, directs the next checkpoint to the ramdisk. **3) Checkpoint Size:** Finally, the CLC directs all large checkpoints too big to fit in ramdisk to the SSD. However, if this decision conflicts with the prior 'lifetime' and 'performance loss' decisions, then the checkpoint is skipped altogether.

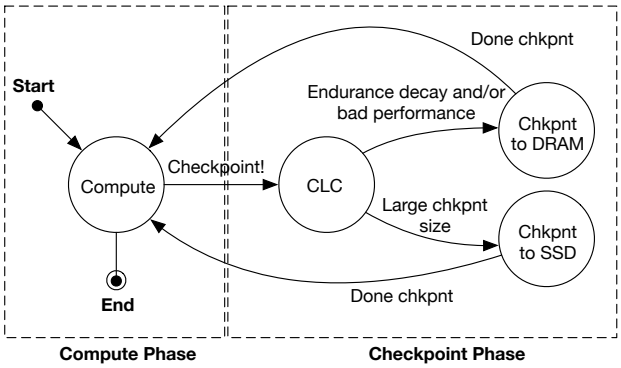


Figure 2: This state machine representing application execution shows how in the checkpoint phase the CLC dynamically decides the checkpoint location on each iteration.

Currently, the controller is written as a library that is added to the application's source code. Figure 2 illustrates how it is integrated into the application. It can interface with existing frameworks such as Scalable Checkpoint/Restart (SCR) [10].

4.3 ECC Design

A dual-mode ECC memory is proposed for conventional x4 DRAM modules containing 18 chips (16 data + 2 ECC). The first mode is to protect regular application data and the second mode is to protect checkpoint data. Both modes have the ability to detect an entire chip failure without implementing the conventional Chipkill-Correct. Unlike Chipkill-Correct which has to activate two memory ranks, the proposed ECC schemes only activates a single x4 DRAM rank. Reed-Solomon (RS) codes over Galois Field (2^8) are used for both ECC algorithms because they provide strong correction and detection capability.

Fault Model. Faults can be classified into small granularity faults (bit/column/pin/word) and large granularity faults (chip). Small granularity faults occur more frequently than large granularity faults and account for more than 70% among all DRAM faults [11]. A bit fault and a column fault lead to a single bit error in a data block. A pin fault results in 8 bit errors in the same data pin positions. A word failure corrupts 4 consecutive bit errors in a single beat. A whole chip failure leads to 32 bit errors (8 beats with 4 bits/beat) in a 512 bit data block.

Normal ECC. Normal ECC, specifically RS(36,32) over GF(2^8), provides error correction coverage for regular data accesses, similar to typical ECC DIMMs for servers. It is designed to meet the following requirements:

1. To correct frequent errors due to single-bit/column/word failures without triggering restart from a checkpoint.
2. To have small decoding latency of syndrome calculation since it is in the critical path of memory access.

3. To activate one rank per memory access and to have better timing/power/energy than Chipkill-Correct.

The decoder is set up to correct all errors due to small granularity (single bit/column/word) faults in a single chip, detect errors due to 1 chip failure, and have strong detection capability for 2 chip failures. It has a small decoding latency of 0.48ns and a detection capability of 99.9969% for double chip failures. Furthermore, since only 1 rank is activated in each memory access, it has better timing/power/energy performance than the traditional x4 Chipkill-Correct scheme.

Strong ECC. Strong ECC, specifically RS(19,16) over GF(2^8) with an embedded RS(18,16), protects the integrity of checkpoints in DRAM. It is designed to meet two requirements:

1. To provide Chipkill-Correct level reliability, which can correct all errors due to a single chip failure and detect all errors due to two chip failures. The strong error correction capability reduces the probability of accessing the SSD's checkpoint during restart.
2. To require minimal differences in hardware so as to be able to switch easily from and to normal ECC. Since ramdisk pages can be mapped anywhere in physical memory, the DRAM modules should be flexible in holding normal or checkpoint data without special modifications to the DRAM devices.

By embedding RS(18,16) within RS(19,16), we make it possible to implement Chipkill-Correct reliability without modifying the DRAM devices. It is possible to do this because the parity check matrix of RS(18,16) is embedded in the parity check matrix of RS(19,16) and thus these two codes can share the same decoding circuitry.

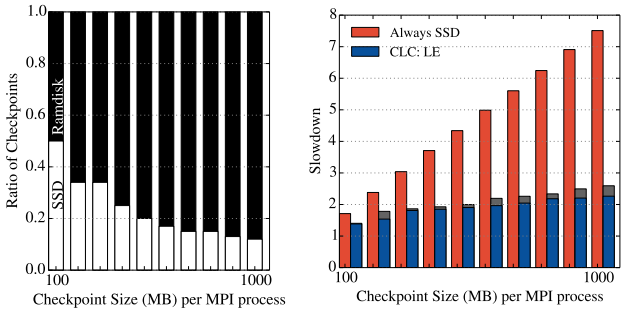
Modification to the Memory Controller. The strong ECC mode exists simultaneously with normal ECC that protects regular memory data; and only requires modification to the memory controller, not the DRAM devices. Regular data is routed via the normal encoder/decoder and ramdisk data is routed via the strong encoder/decoder. In order to identify ramdisk/checkpoint data, the page table can be marked with a special flag to indicate ramdisk pages.

5. MAJOR RESULTS

The test program is an MPI program written in C++ that can be tuned to test a wide variety of checkpoint sizes. It mainly consists of two phases: compute and checkpoint. The compute phase runs an algorithm which takes roughly 5 seconds to finish, and the checkpoint phase writes a file of a specified size to either the ramdisk or the SSD. There are 100 total iterations of the two phases and the checkpoint size is the same in all iterations. The test program was run with 64 MPI processes across 8 nodes.

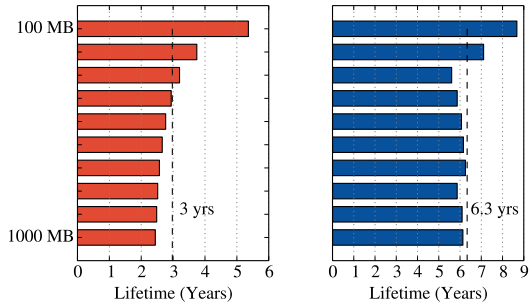
5.1 CLC Results

Figure 3a shows that when the Lifetime Estimation (LE) feature is enabled, fewer checkpoints are written to the SSD, especially at larger checkpoint sizes. At 1000MB per process, only 12% of checkpoints are written to the SSD. Advantageously, this leads to a performance improvement; the slowdown of the benchmark is considerably lessened to an average of only 1.9× (Figure 3b)—as opposed to the nearly 8× slowdown (4.6× on average) if always checkpointing to the SSD. Figure 4 shows the improvement in endurance gained. If checkpoints are only written to the SSD as in Figure 4a, then the SSD is estimated to last an average of 3 years across all the checkpoint sizes. On the other hand, the LE feature of the controller extends the SSD lifetime to an average of 6.3 years (Figure 4b), ensuring that users can get the guaranteed 5 years of life from their SSD.



(a) Checkpoint Location (b) Slowdown

Figure 3: CLC's lifetime estimation feature dynamically redirects checkpoints to ramdisk.



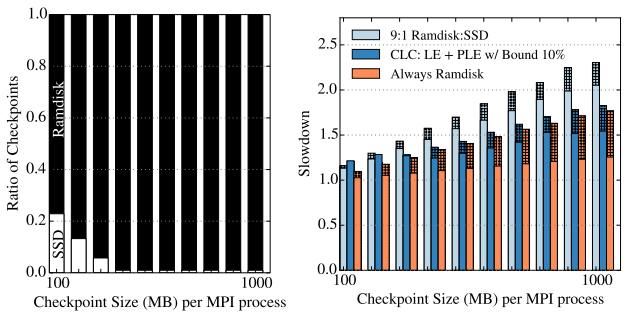
(a) Checkpoint to SSD Only (b) Checkpoint with CLC

Figure 4: CLC's lifetime estimation feature improves SSD lifetime from 3 years 6.3 years.

Figure 5a shows that enabling the Performance Loss Estimation feature writes even fewer checkpoints to SSD, further decreasing slowdown to 47% on average, which is much closer to the 'always-ramdisk' approach which achieved 42% slowdown on average (Figure 5b). The performance loss bound was set to 10% in this experiment. Performance is also compared to a naïve scheme where every 10th checkpoint is written to the SSD, labeled as "9:1 Ramdisk:SSD". This scheme performed better for small checkpoint sizes, indicating that a fixed scheme might be sufficient for applications with small checkpoint sizes that want to achieve a balance between performance and reliability. However, across all checkpointed sizes, it's average slowdown is 72%, that is 25% worse than CLC's PLE feature.

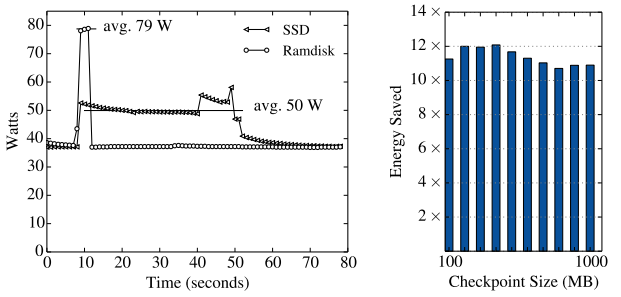
5.1.1 *Energy Results*

Energy saved from writing checkpoints to the DRAM is an additional benefit of our proposed hybrid method. The SSD which we used in our experiments consumed 50W of power on average during a 42-second checkpoint operation (Figure 6a). In contrast, the same checkpoint operation to the DRAM consumed 79W of power on average and took just 3 seconds. Even though DRAM's power consumption is higher, due to its speed advantage it can save overall checkpointing energy by 10× (Figure 6b).



(a) Checkpoint Location (b) Slowdown

Figure 5: (a) CLC's Performance loss estimation further reduces the slowdown. Shaded regions above each bar represent worst-case overheads from strong ECC encoding.



(a) Power Profile (b) Energy Savings

Figure 6: (a) The SSD consumes 50W during a write operation, whereas the DRAM consumes 79W. (b) However, due to DRAM's faster write bandwidth, re-directing some checkpoints to the DRAM saves overall checkpoint energy.

5.2 ECC Results

5.2.1 *Synthesis Results*

Decoding units of RS(36,32), RS(18,16) and RS(19,16) codes over GF(2^8) were synthesized using 28nm industry library. For normal ECC, the detection latency is 0.48ns followed by 0.47ns for single symbol correction. For strong ECC, the detection latency is 0.41ns followed by an additional 0.88ns for single chip failure correction. For both schemes the decoding/detection latency is less than one memory cycle (1.25ns if the DRAM frequency is 800MHz).

5.2.2 *Error Coverage*

The reliability of four ECC schemes, namely, RS(36,32) for normal ECC, RS(18,16) and RS(19,16) for strong ECC, and x4 Chipkill-Correct was evaluated with 10 million Monte Carlo simulations for single bit, pin, word, and chip failure events. Each fault type was injected into a single chip or two chips. For each type of error event, the detectability, correctability, and the silent data corruption rates were calculated and are given in Table 1.

6. CONCLUSION

Exascale supercomputers have millions of components that can fail. A 100 petabyte memory system—100× larger than ORNL Titan supercomputer's 1 petabyte memory system—alone consists of millions of DDR4 DRAM devices backed by hundreds of thousands of SSD flash devices. Resilience to failing components must be achieved by creating a fast and reliable checkpoint/restart framework.

This work has proposed a hybrid DRAM-SSD checkpointing solution to achieve speed and reliability for local checkpointing while also reducing the endurance decay of SSDs. The CLC that we implemented monitors SSD endurance, performance degradation, and checkpoint size to dynamically determine the best checkpoint location. The CLC improved SSD lifetime from 3 years to 6.3 years on average. Furthermore, our normal ECC provides low-latency correction for errors due to bit/pin/column/word faults and our strong ECC provides Chipkill-Correct capability to DRAM checkpoints to reduce the overheads of rollback.

The presented solution demonstrates that it is in fact possible to build an exascale memory system using commodity DRAM and SSD and gain both speed and reliability without relying on emerging memory technologies.

Table 1: The error protection capability

Failure Mode	Normal ECC RS(36,32)	Strong ECC RS(18,16)	Strong ECC RS(19,16)	Chipkill-Correct
Single Chip Failures				
1 bit, 1 word	100% correctable	100% detectable	100% correctable	100% correctable
1 pin, 1 chip	100% detectable	100% detectable	100% correctable	100% correctable
Double Chip Failures				
1 bit + {1 bit, 1 word, 1 pin, or 1 chip}	100% detectable	100% detectable	100% detectable	100% detectable
1 word + {1 bit, 1 word, 1 pin, or 1 chip}	100% detectable	100% detectable	100% detectable	100% detectable
1 pin + 1 pin	99.9999% detect, 0.0001% silent,	100% detectable	100% detectable	100% detectable
1 chip + {1 pin, 1 chip}	99.9969% detect, 0.0031% silent,	100% detectable	100% detectable	100% detectable

References

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

[2] Xiangyong Ouyang, S. Marcarelli, and D.K. Panda. Enhancing checkpoint performance with staging io and ssd. SNAPI 2010.

[3] Xiangyu Dong, Naveen Muralimanohar, Norm Jouppi, Richard Kaufmann, and Yuan Xie. Leveraging 3D PCRAM Technologies to Reduce Checkpoint Overhead for Future Exascale Systems. SC 2009.

[4] Ping Chi, Cong Xu, Tao Zhang, Xiangyu Dong, and Yuan Xie. Using Multi-level Cell STT-RAM for Fast and Energy-efficient Local Checkpointing. ICCAD 2014.

[5] S. Kannan, A. Gavrilovska, K. Schwan, and D. Milojicic. Optimizing Checkpoints Using NVM as Virtual Memory. IPDPS 2013.

[6] Objective Analysis. A Close Look At The Micron/Intel 3D XPoint Memory, September 2015.

[7] Fei Sun, Ken Rose, and Tong Zhang. On the use of strong bch codes for improving multilevel nand flash memory storage capacity. In *IEEE Workshop on Signal Processing Systems (SiPS): Design and Implementation*, 2006.

[8] U.S Department of Energy Office of Science and National Nuclear Security Administration. Preliminary Conceptual Design for an Exascale Computing Initiative, November 2014.

[9] Intel Cooperation. Intel Solid-State Drive DC S3700 specification, October 2012.

[10] Adam Moody, Greg Bronevetsky, Kathryn Mohror, and Bronis R. de Supinski. Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System. SC 2010.

[11] Vilas Sridharan and Dean Liberty. A Study of DRAM Failures in the Field. SC 2012.