Leveraging 3D PCRAM Technologies to Reduce Checkpoint Overhead for Future Exascale Systems

Xiangyu Dong[†‡], Naveen Muralimanohar[†], Norm Jouppi[†], Richard Kaufmann[†], Yuan Xie[‡]
[†]Hewlett-Packard Labs, [‡]Pennsylvania State University
[†]Email: {xiangyu.dong,naveen.muralimanohar,norm.jouppi,richard.kaufmann}@hp.com
[‡]Email: {xydong,yuanxie}@cse.psu.edu

ABSTRACT

The scalability of future massively parallel processing (MPP) systems is challenged by high failure rates. Current hard disk drive (HDD) checkpointing results in overhead of 25% or more at the petascale. With a direct correlation between checkpoint frequencies and node counts, novel techniques that can take more frequent checkpoints with minimum overhead are critical to implement a reliable exascale system. In this work, we leverage the upcoming *Phase-Change Random Access Memory* (PCRAM) technology and propose a hybrid local/global checkpointing mechanism after a thorough analysis of MPP systems failure rates and failure sources.

We propose three variants of PCRAM-based hybrid checkpointing schemes, *DIMM+HDD*, *DIMM+DIMM*, and *3D+3D*, to reduce the checkpoint overhead and offer a smooth transition from the conventional pure HDD checkpoint to the ideal 3D PCRAM mechanism. The proposed pure 3D PCRAM-based mechanism can ultimately take checkpoints with overhead less than 4% on a projected exascale system.[1]

1. INTRODUCTION

MPP systems are designed to solve complex mathematical problems that are highly compute intensive. These workloads are capable of exploiting the entire system processing power and typically take many days to complete. Although the individual nodes in MPP systems are designed to have a high Mean Time to Failure (MTTF), the reliability of the total system degrades significantly as the number of nodes increases. For example, the "ASCI Q" supercomputer at Los Alamos National Laboratories had an MTTF of less than 6.5 hours [1]. As a result, modern supercomputers encounter frequent crashes severely impacting the workload completion time. This problem will get worse as the exascale era approaches where the system will likely have five to ten times more processors compared to current generation systems. In addition, a study from Intel [2] shows that transient errors in processors and memories are likely to increase by 32X in the next ten years, which will further accelerate the failure rate growth in future systems.

[1]X. Dong and Y. Xie were supported in part by NSF grants 0702617, 0720659, 0903432 and SRC grants.

SC09 November 14-20, 2009, Portland, Oregon, USA

To tolerate the rising failure rate and reduce its impact on workload running time, modern MPP systems are equipped with a centralized non-volatile storage system (typically built with arrays of disks) that takes frequent synchronized checkpoints of every node in the system. However, the current approach has many serious limitations. First, the design of using a single centralized medium storing all checkpoints is inherently not scalable; second, as the number of compute nodes increases and the size of applications grow, the performance overhead of conventional techniques can reach an unacceptable level. A recent study by Oldfield *et al.* [3] showed a 1-petaFLOPS system can potentially take more than 50% performance hits because of frequent checkpointing operations. Therefore, with the current trends of increasing system size and decreasing system reliability, it is not feasible for future MPP systems to employ conventional checkpointing techniques.

The primary source of delay in conventional checkpointing is the time spent on writing checkpoints to storage due to the limited bandwidth provided by the network and the storage system. Since the state of each node has to be preserved during checkpointing, the entire MPP system is stalled until checkpointing completes. This causes severe degradation of workload performance. Any effort to save performance by reducing the checkpoint frequency will again negatively impact performance as the amount of useful work lost in the event of a failure is inversely proportional to the checkpoint frequency. Hence, an efficient approach that can take checkpoints at a high frequency with a minimum overhead is required to reap the performance benefits of future MPP systems.

A scalable solution to this problem is to take checkpoints in a local storage medium. Unlike global checkpoints that are accessible by all the nodes, local checkpoints are private to each node. Therefore, the local checkpoint cannot be reached in the event of node loss or other permanent hardware failures. To provide complete protection, it is necessary to take both global and local checkpoints with different frequencies. While this approach looks expensive, in this work we show that a significant checkpoint overhead reduction can be achieved by tuning the local and global checkpoint ratio. A model to identify the local and global checkpoint intervals that incur the least overhead is also derived in this work.

Local checkpointing can be done in many ways. While DRAM is relatively fast, its high leakage and volatile nature makes it an energy-expensive option. While NAND flash is non-volatile, its low write endurance significantly limits the checkpoint frequency. This work shows the emerging *Phase-Change Random Access Memory* (PCRAM) technology is an ideal choice for local checkpointing with unique characteristics such as non-volatility, zero standby leakage power, fast random read accesses, and significantly improved lifetime compared to NAND flash.

In general, this work is focused on how to use emerging memory technologies to extend the life of the checkpoint/restart mech-

anism. Fault detection and silent data corruption is another significant problem by itself in the supercomputing community, and it is out of the scope of this work. However, it is still reasonable to assume that the time required to detect a failure is much less than the checkpoint interval, even in this work the interval might be as fast as 0.1 seconds. Therefore, we neglect the overhead caused by failure detection when we evaluate the performance of our approaches.

2. CHECKPOINTING IN MPP SYSTEMS

Checkpointing is one of the most widely-used techniques to provide fault-tolerance for MPP systems. There are two main categories of checkpointing mechanisms: *coordinated* or *communication-induced*. Using coordinated checkpointing, all cooperating processes work together to establish a coherent checkpoint, and all the processes must arrive at consistent states before a checkpoint operation is performed. Using communication-induced checkpointing, each process checkpoints its own state independently whenever the state is exposed to other processes (e.g., when a remote process reads the page written to by the local process). For large-scale applications, coordinated checkpointing is more popular [3]. In this work, we examine *coordinated, application-directed, periodic checkpoints*.

2.1 Problems in HDD-based Checkpoint

The in-practice checkpoint storage device is HDD: several nodes in the MPP system are assigned to be the I/O nodes that are in charge of the HDD accesses. Thus, the checkpoints have to be moved from compute nodes to I/O nodes via network connections, and such data movements consume a large part of the system I/O bandwidth. Even with a high I/O bandwidth, this checkpointing operation is still limited by the poor HDD bandwidth.

Although a distributed file system, like *Lustre*, can aggregate the file system bandwidth to hundreds of GB/s, in such systems the checkpoint size also gets aggregated by the scale of nodes, nullifying the benefit.

Therefore, the sustained transfer rate of HDDs ($<$200MB/s [4]) is a serious bottleneck of HDD-based checkpointing. The significance of this problem is demonstrated by the fact that the I/O generated by HDD-based checkpointing consumes nearly 80% of the total file system usage even on today's MPP systems [3], and the checkpoint overhead accounts for over 25% of total application execution time in a petaFLOPS system [5].

2.2 Solution: Local/Global Hybrid Checkpoint

The main motivation of centralized global checkpointing is to cover a wide range of failures including the complete failure of a node (i.e., the global checkpoint can be used to start up a hot spare to resume the execution). However, a thorough analysis of failure rates of MPP systems shows that a majority of failures are transient in nature [6] and can be recovered by a simple reboot operation. As a result, a significant number of failures can be recovered by taking a local checkpoint private to each node.

Therefore, in addition to taking global checkpoints, we propose local checkpoints that periodically backup the state of each node in their own private memory. Every node has a dedicated local memory for storing its system state. Similar to its global counterpart, the checkpointing is done in a coordinated fashion. We assume that a global checkpoint is made from an existing local checkpoint. This two-level hybrid checkpointing gives us an opportunity to tune the local to global checkpoint ratio based on failure types. For example, a system with high transient failures can be protected by frequent local checkpoints and a limited number of expensive global checkpoints without losing performance. The proposed local/global checkpointing is also effective in handling failures during the checkpoint operation. Since the scheme does not allow concurrent local and global checkpointing, there will always be a stable state for the system to rollback even when a failure occurs during the checkpointing process. The only time the rollback operation is not possible is when a node fails completely in the middle of making a global checkpoint. While such failure events can be handled by maintaining multiple global copies, the probability of a global failure in the middle of a global checkpoint is less than 1%. Hence, we limit our proposal to a single copy of local and global checkpoint.

Table 1: The statistics of the failure root cause collected by LANL during 1996-2005 [7]

Cause	Occurrence	Percentage
Hardware	14341	60.4%
Software	5361	22.6%
Network	421	1.8%
Human	149	0.6%
Facilities	362	1.5%
Undetermined	3105	13.1%
Total	23739	100%

Whether the MPP system can be recovered using a local checkpoint after a failure depends on the failure type. In this work, all the system failures are divided into two categories:

- *Failures that can be recovered by local checkpoints:* In this case, the local checkpoint in the failure node is still accessible. If the system error is a transient one, (i.e., soft error, accidental human operation, or software bug), the MPP system can be simply recovered by rebooting the failure node using its local checkpoint. If the system error is due to a software bug or hot plug/unplug, the MPP system can also be recovered by simply rebooting or migrating the computation task from one node to another node using local checkpoints.
- *Failures that have to be recovered by global checkpoints:* In the event of some permanent failures, the local checkpoint in the failed node is not accessible any more. For example, if the CPU, the I/O controller, or the local storage itself fails to work, the local checkpoint information will be lost. This sort of failure has to be protected by a global checkpoint, which requires storing system state in either neighboring nodes or a global storage medium.

As a hierarchical approach, whenever the system fails, the system will first try to recover from local checkpoints. If one of the local checkpoints is not accessible, the system recovery mechanism will restart from the global checkpoint.

2.3 System Failure Category Analysis

In order to learn what percentage of the failures can be recovered by local checkpointing, we studied the failure events collected by the Los Alamos National Laboratory (LANL) during 1996-2005 [7]. The data covers 22 high-performance computing systems, including a total of 4,750 machines and 24,101 processors. The statistics of the failure root cause are shown in Table 1.

We conservatively assume that *undetermined failures* have to rely on global checkpoints for recovery, and assume that the failures caused by *software*, *network*, *human*, and *facilities* can be protected by local checkpoints:

- If nodes halt due to *software failures* or *human mal-manipulation*, we assume some mechanisms (i.e., timeout) can detect these failures and the failure node will be rebooted automatically.
- If nodes halt due to *network failures* (i.e., widely-spread network

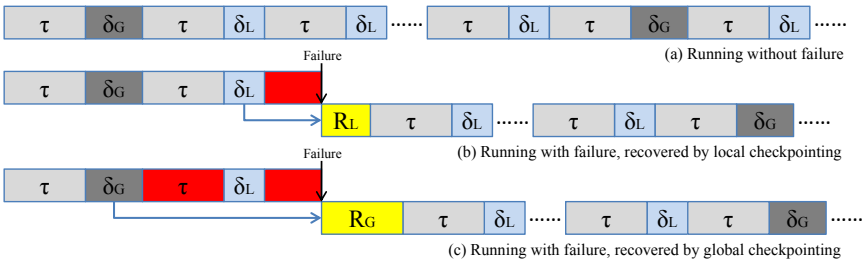

Figure 1: A conceptual view of execution time broken by the checkpoint interval: (a) an application running without failure; (b) an application running with a failure, where the system rewinds back to the most recent checkpoint, and it is recovered by the local checkpoint; (c) an application running with a failure that cannot be protected by the local checkpoint. Hence, the system rewinds back to the most recent global checkpoint. The red block shows the computation time wasted during the system recovery.

congestion) or *facilities downtime* (i.e. global power outrage), automatic recovery is impossible and manual diagnose/repair time is inevitable. However, after resolving the problem, the system can simply restart using local checkpointing.

The remaining *hardware failure* accounts to more than 60% of total failures. However, according to research on the fatal soft error rate of the "ASCI Q" system at LANL in 2004 [6], it is estimated that about 64% of the hardware failures are attributed to soft errors. Hence, observing the failure trace, we have the following statistics: $60.4\% \times 64\% = 38.7\%$ soft errors, and $60.4\% \times (1 - 64\%) = 21.7\%$ hard errors. Assuming that soft errors can be protected by local checkpoints but hard errors need global checkpoints, we can estimate that around 65.2% of failures can be corrected by local checkpoints and only 34.8% of failures need global checkpoints.

Further considering the soft error rate (SER) will greatly increase as the device size shrinks, we project that SER increased 4 times from 2004 to 2008. Therefore, we make a further estimation for the petaFLOPS system in 2008 that 83.9% of failures need local checkpoints and only 16.1% failures need global ones. This failure distribution provides a significant opportunity for the local/global hybrid checkpointing scheme to reduce the overhead.

Finally, since the soft error rate is so important to future exascale system reliability, a detailed sensitivity study on SER is demonstrated in Section 6.7.

3. A NEW MODEL FOR LOCAL/GLOBAL HYBRID CHECKPOINTING

In an MPP system with checkpointing, the optimal checkpoint frequency is a function of both failure rates and checkpoint overhead. A low checkpoint frequency reduces the impact of checkpoint overhead on performance but loses more useful work when failures take place, and vice versa. Young [8] and Daly [9] derived expressions to determine the optimal checkpoint frequency that strikes the right balance between the checkpoint overhead and the amount of useful work lost during failures. However, their models do not support local/global hybrid checkpointing. In this work, we extend Daly's work [9] and derive a new model to calculate the optimal checkpoint frequencies for both local and global checkpoints.

Let us consider a scenario with the following parameters as listed in Table 2 and divide the total execution time of a checkpointed workload, T_{total}, into four parts:

$$T_{total} = T_S + T_{dump} + T_{rollback,recovery} + T_{extra-rollback} \quad (1)$$

where T_S is the original computation time of a workload, T_{dump} is the time spent on checkpointing, $T_{rollback,recovery}$ is the recovery cost when a failure occurs (no matter it is local or global), and

Table 2: Local/Global Hybrid Checkpointing Parameters

T_S	The original computation time of a workload
p_L	The percentage of local checkpoints
p_G	$1 - p_L$, the percentage of global checkpoints
τ	The local checkpoint interval
δ_L	The local checkpoint overhead (dumping time)
δ_G	The global checkpoint overhead (dumping time)
δ_{eq}	the equivalent checkpoint overhead in general
R_L	The local checkpoint recovery time
R_G	The global checkpoint recovery time
R_{eq}	The equivalent checkpoint time in general
q_L	The percentage of failure covered by local checkpoints
q_G	$1 - q_L$, the percentage of failure that have to be covered by global checkpoints
$MTTF$	The system mean time to failure
T_{total}	The total execution time including all the overhead

$T_{extra-rollback}$ is the extra cost to discard more useful work when a global failure occurs.

The checkpoint dumping time is simply the product of the number of checkpoints, T_S/τ, and the equivalent dumping time per checkpoint, δ_{eq}, thus

$$T_{dump} = \frac{T_S}{\tau}(\delta_{eq}) \quad (2)$$

When failure occurs, at least one useful work slot has to be discarded as the red slot shown in Figure 1(b) and the second red slot shown in Figure 1(c). Together with the recovery time, this part of overhead can be modeled as follows with the approximation that the failure occurs half way through the compute interval on average,

$$T_{rollback,recovery} = \left(\frac{1}{2}(\tau + \delta_{eq}) + R_{eq}\right)\frac{T_{total}}{MTTF} \quad (3)$$

where $T_{total}/MTTF$ is the expected number of failures.

Additionally, if a failure has to rely on global checkpoints, more useful computation slots will be discarded as the first red slot shown in Figure 1(c). In this case, the number of wasted computation slots, on average, is approximated to $p_L/2p_G$. For example, if $p_L = 80\%$ and $p_G = 20\%$, $80\%/20\% = 4$ useful computation slots will be potentially wasted and the expected number of wasted computation slots is $p_L/2p_G = 2$. Hence, this extra rollback cost can be modeled as follows,

$$T_{extra-rollback} = \frac{p_L q_G}{2p_G}(\tau + \delta_L)\frac{T_{total}}{MTTF} \quad (4)$$

Eventually, after including all the overhead mentioned above, the

total execution time of a checkpointed workload is,

$$\begin{aligned} T_{total} &= T_S + \frac{T_S}{\tau}(\delta_{eq}) \\ &+ \left(\frac{1}{2}(\tau + \delta_{eq}) + R_{eq}\right)\frac{T_{total}}{MTTF} \\ &+ \frac{p_L q_G}{2p_G}(\tau + \delta_L)\frac{T_{toal}}{MTTF} \end{aligned} \quad (5)$$

where the two equivalent parameters, δ_{eq} and R_{eq}, can be calculated as follows,

$$\delta_{eq} = \delta_L \cdot p_L + \delta_G \cdot p_G \quad (6)$$

$$R_{eq} = R_L \cdot q_L + R_G \cdot q_G \quad (7)$$

It can be observed from the equation that a trade-off exists between the checkpoint frequency and the rollback time. Since many variables in the equation have strict lower bounds and can take only discrete values, we use MATLAB to optimize the two critical parameters, τ and p_L, using a numerical method. It is also feasible to derive closed-form expressions for τ and p_L to enable run-time adjustment for any changes of workload size and failure distribution, but they are out of the scope of this paper. A detailed analysis on checkpoint interval and local/global ratio under different MPP system configurations is discussed in Section 6.

4. PHASE-CHANGE MEMORIES

To implement the local/global hybrid checkpoint, fast and permanent local storage is required. While HDD is slow, DRAM is volatile, and NAND flash can only be written for about 10^5 times, the emerging phase-change memory is a good candidate. This section gives a brief introduction to the *Phase-Change Memory* technology.

4.1 PCRAM Backgournd

- **Phase-Change Mechanism:**
  Phase change memory is an emerging technology that fundamentally differs from other conventional memories. Unlike SRAM, DRAM or NAND flash technologies that use electrical charges, PCRAM changes the state of a Chalcogenide-based material, such as alloys of germanium, antimony, or tellurium ($GeSbTe$, or GST), to store a logical "0" or "1". For instance, GST can be switched between the crystalline phase (SET or "1" state) and the amorphous phase (RESET or "0" state) with the application of heat. The crystalline phase shows high optical reflectivity and low electrical resistivity, while the amorphous phase is characterized by low reflectivity and high resistivity. Due to these differences, phase-change materials can be used to build both memory chips and optical disks. As shown in Figure 2, every PCRAM cell contains one GST and one access transistor. This structure has a name of "1T1R" where T refers to the access transistor, and R stands for the GST resistor.
- **PCRAM Read Operation:**
  To read the data stored in a PCRAM cell, a small voltage is applied across the GST. Since the SET state and RESET state have a large variance on their equivalent resistances, data are sensed by measuring the pass-through current. The read voltage is set sufficiently high to invoke a sensible current but low enough to avoid write disturbance. Usually, the read voltage is clamped between $0.2V$ to $0.4V$ [11]. Similar to traditional memories, the word line connected to the gate of the access transistor is activated to read values from PCRAM cells.
- **PCRAM Write Operation:**
  The PCRAM write operation is characterized by its SET and RESET operations. As illustrated in Figure 3, the SET operation crystallizes GST by heating it above its crystallization temperature, and the RESET operation melt-quenches GST to make the material amorphous. The temperature during each operation is controlled by applying the appropriate current waveform. For SET operation, a moderate current pulse is applied for a longer duration to heat the cell above the GST crystallization temperature but below the melting temperature; for REST operation, a high power pulse heats the memory cell above the GST melting temperature. Recent PCRAM prototype chips demonstrate that the RESET latency can be as fast as $100ns$ and the peak SET current can be as low as $100\mu A$ [11, 12].
- **PCRAM Cell Size & Scalability:**
  The cell size of PCRAM is mainly constrained by the current driving ability of the NMOS access transistor. The achievable cell size can be as small as $10 - 40F^2$ [11, 12], where F is the feature size. When NMOS transistors are substituted by diodes, the PCRAM cell size can be reduced to $4F^2$ [13]. Related research [14] shows PCRAM has excellent scalability as the required SET current can be reduced with technology scaling. Although multi-bit cell is available recently [15], we use single-bit cell in this work for faster access.

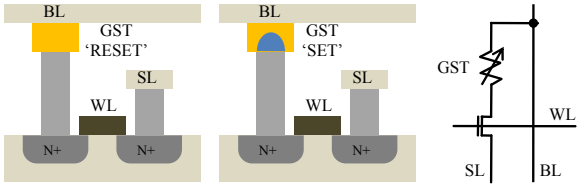

Figure 2: The schematic view of a PCRAM cell with NMOS access transistor (BL=Bitline, WL=Wordline, SL=Sourceline)

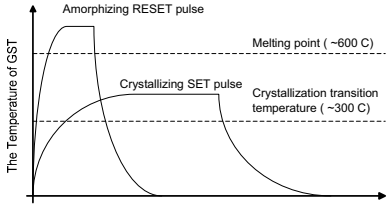

Figure 3: The temperature-time relationship during SET and RESET operations

4.2 Comparison

Comparing to other storage technologies, such as SRAM, DRAM, NAND flash, and HDD, PCRAM shows its relatively good properties in terms of density, speed, power, and non-volatility. As listed in Table 3, the PCRAM read speed is comparable to those of SRAM and DRAM. While its write operation is slower than SRAM and DRAM, it is still much faster than its non-volatile counterpart – NAND flash. More importantly, the PCRAM write endurance is within the feasible range for the checkpointing application. Pessimistically assuming the PCRAM write endurance of 10^8 and checkpoint interval of $10s$, the lifetime of the PCRAM checkpointing module can still be more than 30 years, while the lifetime of its NAND flash counterpart is less than 30 hours. We expect the PCRAM write endurance will be higher than 10^{10} in 2017, so that an even more aggressive checkpoint interval, i.e. $0.1s$, would not be a problem for PCRAM lifetime.

Table 3: Comparison among SRAM, DRAM, NAND flash, HDD, and PCRAM (Source: [4, 10])

	SRAM	DRAM	NAND flash	PCRAM	HDD
Cell size	$> 100F^2$	$6 - 8F^2$	$4 - 6F^2$	$4 - 40F^2$	-
Read time	$\sim 10ns$	$\sim 10ns$	$5\mu s - 50\mu s$	$10ns - 100ns$	$\sim 4ms$
Write time	$\sim 10ns$	$\sim 10ns$	$2 - 3ms$	$100 - 1000ns$	$\sim 4ms$
Standby power	Cell leakage	Refresh power	Zero	Zero	$\sim 1W$
Endurance	10^{18}	10^{15}	10^5	$10^8 - 10^{12}$	10^{15}
Non-volatility	No	No	Yes	Yes	Yes

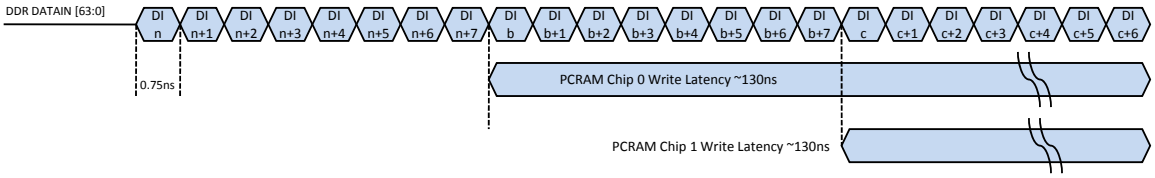

Figure 4: The consecutive WRITE waveform of the PCRAM DIMM on a DDR3-1333 bus: After loading 8 pieces of DDR data, the data latches in the selected PCRAM chip becomes full, and the PCRAM write operation is initiated; During the write operation of PCRAM chip 0, the next coming DDR data are directed to chip 1, and so forth.

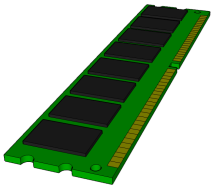

Figure 5: The schematic view of a DIMM (9 chips on each side, 18 chips in total)

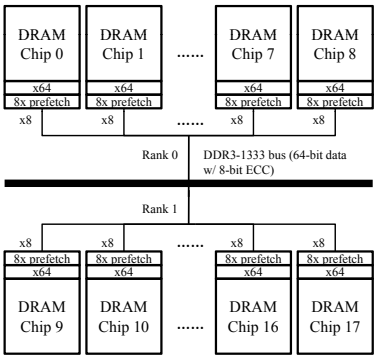

Figure 6: The organization of a DRAM DIMM

5. INTEGRATING PCRAM MODULES INTO MPP SYSTEMS

As mentioned in Section 4, PCRAM is a promising candidate for local checkpointing. In this section, two methods are proposed to introduce the PCRAM-based local checkpointing: PCRAM-DIMM and 3D-PCRAM. An in-house PCRAM simulation tool, called *PCRAM-sim* [16], is used during the design of these two approaches.

5.1 PCRAM-DIMM: Separate DIMMs for PCRAM

As an intermediate way to integrate PCRAM into future MPP systems, we first evaluate allocating PCRAM on separate Dual-Inline Memory Modules (DIMMs).

As shown in Figure 5, usually on each DIMM (with ECC protection) there are 18 memory chips (9 on each side). The DDR bus has a 64-bit data path (plus another 8-bit ECC path). If each memory chip provides 8 bits (x8 configuration) as shown in Figure 6, 9 chips are enough to provide a 72-bit DDR word, and the 18 chips in total can be separated into two ranks.

However, this DRAM memory organization cannot be directly adopted by our PCRAM-DIMM design. The key problem is that PCRAM has a much longer write latency ($\sim 100ns$). If we design the PCRAM DIMM using the same memory organization of a DRAM DIMM, the write bandwidth it can provide is only $0.32GB/s$, far below the DDR3-1333 bandwidth of $10.67GB/s$.

To solve the bandwidth mismatch between the DDR bus and the PCRAM, two modifications are introduced:

(a) As shown in Figure 7, the configuration of each PCRAM chip is changed to x72, while the 8x prefetching scheme is retained for compatibility with the DDR3 protocol. As a result, there are

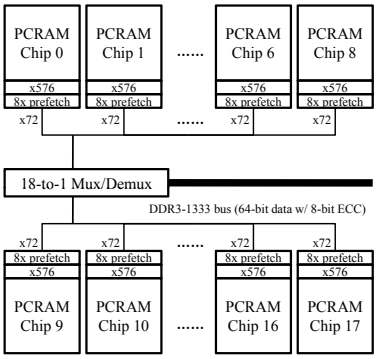

Figure 7: The organization of the proposed PCRAM DIMM

72×8 data latches in each PCRAM chip, and during each PCRAM write operation, 576 bits are written into the PCRAM cell array in parallel;

(b) The 18 chips on DIMMs are re-organized in an interleaved way. For each data transition, only one PCRAM chip is selected. A 18-to-1 data mux/demux is added on DIMMs to select the proper PCRAM chip for each DDR3 transition.

Consequently, as shown in Figure 4, the PCRAM write latency of each PCRAM chip can be overlapped. The overhead of this new DIMM organization includes: (1) one 1-to-18 data mux/demux; (2) 576 sets of data latches, sense amplifiers, and write drivers on each

Table 4: Different configurations of the PCRAM chips

Process	Capacity	# of Bank	Read/RESET/SET	Leakage	Die Area
65nm	512Mb	4	$27ns/55ns/115ns$	$64.8mW$	$109mm^2$
65nm	512Mb	8	$19ns/48ns/108ns$	$75.5mW$	$126mm^2$
45nm	1024Mb	4	$18ns/46ns/106ns$	$60.8mW$	$95mm^2$
45nm	1024Mb	8	$16ns/46ns/106ns$	$62.8mW$	$105mm^2$

PCRAM chip. The mux/demux can be implemented by a circuit that decodes the DDR3 address to 18 chip select signals (CS#). The overhead of data latches, sense amplifiers, and write drivers are evaluated using *PCRAMsim*.

Various configurations are evaluated by *PCRAMsim* and the results are listed in Table 4.

Based primarily on SET latency and area efficiency, we use the 45nm 1024Mb 4-bank PCRAM chip design as a guide, and all the PCRAM DIMM simulations in Section 6 are based on this configuration. Meanwhile, the write bandwidth of PCRAM-DIMM is $64bit \times 8 \times 18/106ns = 10.8GB/s$, which is compatible with the DDR3-1333 bandwidth $10.66GB/s$. In addition, according to our *PCRAMsim* power model, for each 576-bit RESET and SET operation, it consumes total dynamic energy of $31.5nJ$ and $19.6nJ$, respectively. Therefore, assuming that "0" and "1" are written uniformly, the average dynamic energy is $25.6nJ$ per 512 bits, and the 1024Mb PCRAM DIMM dynamic power under write operations is $25.6nJ/512b \times 10.8GB/s \approx 4.34W$. The leakage power of the 18-chip PCRAM DIMM is estimated to be $60.8mW \times 18 = 1.1W$.

5.2 3D-PCRAM: Deploying PCRAM atop DRAM

The PCRAM-DIMM scheme discussed above has limitations: copying from DRAM to PCRAM has to go through the processor and the DDR bus; it not only pollutes the on-chip cache but also has the DDR bandwidth constraint. Hence, in future exascale MPP systems, the PCRAM-DIMM local checkpointing may still require a non-trivial fraction of the total execution time.

As the ultimate way to integrate PCRAM in a more scalable way, we further propose the 3D-PCRAM scheme deploying PCRAM directly atop DRAM. By exploiting emerging 3D integration technology [17] to design the 3D PCRAM/DRAM chip, it becomes possible to dramatically accelerate the checkpoint latency and hence reduce the checkpoint overhead to the point where it is almost a negligible percentage of program execution.

For compatibility, the interface between DRAM chips and DIMMs is preserved. Our proposed modifications to the main memory are mainly constrained by four key requirements:

- The new model should incur minimum modifications to the DRAM die, while exploiting 3D integration to provide maximum bandwidth between PCRAM and DRAM.
- We need extra logic to trigger the data movement from DRAM to PCRAM only when the checkpoint operation is needed and only where the DRAM bits are dirty.
- We need a mechanism to provide the sharp rise in supply current during PCRAM checkpointing.
- There should be an effective way to transfer the contents of DRAM to PCRAM without exceeding the thermal envelope of the chip.

These four challenges are solved individually as follows:

(1) To reduce the complexity of the 3D stacked design, we use the same number of banks in the PCRAM and DRAM dies. Since the diode-accessed PCRAM cell size is similar to that of DRAM, we can model PCRAM banks of similar size to its DRAM counterpart. When making connections between dies, for the ultimate bandwidth, a cell-to-cell connection is desired. However, such a design needs very high density Through-Silicon-Vias (TSVs) and hence has low area efficiency. Thus, we opt for connections at the granularity of *mats*. A *mat* is a self-contained module with a set of memory cells and logic capable of storing or retrieving data (in *PCRAMsim*, a mat is composed of four sub-arrays). For the proposed 3D design, we make connections between the input bus of a mat in the DRAM to the corresponding mat in the PCRAM as shown in Figure 8. Assuming a typical bank has 16 mats, we calculate that the required TSV pitch is less than $74\mu m$. ITRS [18] shows the achievable TSV density is about $3.8\mu m$ that far exceeds our requirements. Table 5 shows the detailed specifications.

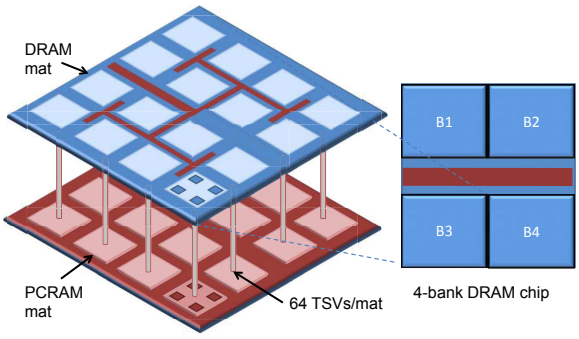

Figure 8: A conceptual view of 3D-PCRAM: the DRAM module is stacked on top of the PCRAM module.

Table 5: 3D stacked PCRAM/DRAM memory statistics and the comparison between 3D-PCRAM and PCRAM-DIMM

Bank size	$32MB$
Mat count	16
Required TSV pitch	$< 74\mu m$
ITRS TSV pitch projection for 2012	$3.8\mu m$
3D-PCRAM delay (independent of memory size)	$0.8ms$
PCRAM-DIMM delay (2GB memory)	$185ms$
3D-PCRAM bandwidth (2GB DIMM)	$2500GB/s$
PCRAM-DIMM bandwidth	$10.8GB/s$

(2) To control the data transfer from DRAM to PCRAM, we include an address generator circuit and a multiplexer for each DRAM mat. An address generator is essentially a counter which retrieves the contents of a DRAM mat and sends it to its PCRAM counterpart when triggered. To hide the high write penalty of PCRAM, we use the multiplexer to interleave the writes between four sub-arrays in the PCRAM mat. To employ an incremental checkpointing technique, dirty page management is required for every page in the DRAM. This only costs 1-bit of overhead for each page, and avoids unnecessary transfers from DRAM to PCRAM.

(3) Although high-density TSVs can provide ultra-wide bandwidth as high as 2.5TB/s in our demonstration, an ultra-high peak current is also needed for parallel PCRAM cell writes. In such a case, the transient power consumption can be as high as 700W. However, this peak power is only required within an extremely short interval of 0.8ms and the actual energy consumption is as low as 0.56J. To handle this short period of power consumption, we in-

Table 7: Bottleneck Factor of Different Checkpoint Schemes

	Local Medium	Local Bottleneck	Global Medium	Global Bottleneck
Pure-HDD	-	-	HDD on I/O nodes	HDD, Network BW
DIMM+HDD	Self's PCRAM DIMM	Memory BW	HDD on I/O nodes	HDD, Network BW
DIMM+DIMM	Self's PCRAM DIMM	Memory BW	Neighbor's PCRAM DIMM	Network BW
3D+3D	Self's 3D DIMM	3D BW	Neighbor's 3D DIMM	Network BW

Table 6: Temperature estimations of 3D-PCRAM modules

Scenario	Local checkpoint interval	Package temperature
DRAM Only	-	319.17K
1-Layer PCRAM stacked	1.00s	319.57K
1-Layer PCRAM stacked	0.10s	320.54K
1-Layer PCRAM stacked	0.01s	330.96K

clude a super capacitor (about 0.6F) on each 3D PCRAM/DRAM DIMM.

(4) To confirm that our 3D-PCRAM scheme will not cause thermal problems, we evaluated the impact of heat from 3D stacked PCRAM memory on the DRAM DIMMs. We obtain the estimated temperature listed in Table 6 using HotSpot [19]. Note that the increase in temperature is negligible as long as the checkpoint interval is longer than 0.1s. Hence, for all our experiments (Section 6), we set the lower bound of local checkpoint interval to be 0.1s.

6. EXPERIMENTAL RESULTS

The primary goal of this work is to improve the checkpoint efficiency and prevent checkpointing from becoming the bottleneck to MPP scalability. In this section, the analytical equations derived in Section 3 is mainly used to estimate the checkpoint overhead. In addition, simulations are also conducted to get the quantitative parameters such as the checkpoint size.

6.1 Checkpointing Scenarios

In order to show how the proposed local/global hybrid checkpoint using PCRAM can reduce the performance and power overhead of checkpoint operations, we study the following 4 scenarios:

- *Pure-HDD*: The conventional checkpoint approach that only stores checkpoints in HDD globally.
- *DIMM+HDD*: Store checkpoints in PCRAM DIMM locally and in HDD globally. In each node, the PCRAM DIMM capacity is equal to the DRAM DIMM capacity.
- *DIMM+DIMM*: Store local checkpoints in PCRAM DIMM and store neighbors' checkpoints in another in-node PCRAM DIMM as the global checkpoints. In each node, the PCRAM DIMM capacity is thrice as the DRAM DIMM capacity.
- *3D+3D*: Same as *DIMM+DIMM*, but deploy the PCRAM resource using 3D-PCRAM (described in Section 5) rather than PCRAM-DIMM.

The bottleneck of each scenario is listed in Table 7.

6.2 Scaling Methodology

We use the specification of the IBM Roadrunner Supercomputer [5], achieving a sustained performance of 1.026 petaFLOPS on LINPACK, to model the petaFLOPS baseline MPP system.

Socket Count: Roadrunner has a total of 19,872 processor sockets and achieves an average of 52 gigaFLOPS per socket. We assume that the future processors can scale their performance with future increases in transistor count to 10 teraFLOPS per socket by the year 2017 [20]. Hence, to cross the exaFLOPS barrier, it is necessary to increase the socket count by 5X (from 20,000 to 100,000).

Table 8: The specification of the baseline petascale system and the projected exascale System

	1 petaFLOPS	1 exaFLOPS
FLOPS	10^{15}	10^{18}
Year	2008	2017
# of sockets	20,000	100,000
Compute/IO node ratio	15:1	15:1
Memory per socket	4GB	210GB
Memory BW	10GB/s	32GB/s
Network BW	2GB/s	20GB/s
Aggregate file system BW	220GB/s	1100GB/s
Normalized SER	1	32
Transient error percentage	91.5%	99.7%

This implies that the number of failures in exascale MPP systems will increase by at least 5X even under the assumption that the future 10-teraFLOPS socket retains the same MTTF as today.

Memory per Socket: The memory requirement of future MPP systems is proportional to the computational capabilities of the projected processor. Typical MPP workloads that solve various nonlinear equations can adjust the scheduling granularity and thread size to suit the configuration of a processor. Therefore, as the computing power of a processor scales from 52 gigaFLOPS to 10 teraFLOPS, the application memory footprint in each processor will also increase. In general, the memory capacity required per socketis proportional to $(FLOPS)^{3/4}$ [2]. The current generation Roadrunner employs 4GB per Cell processor. Based on the above relation, a future socket with 10-teraFLOPS capability will require 210 GB of memory.

Memory Bandwidth: Both DRAM main memory access time and PCRAM DIMM checkpoint time are constrained by the memory bus bandwidth. The last decade has seen roughly a 3X increase in memory bandwidth because of the increased bus frequency and the prefetch depth. However, it is not clear whether similar improvements are possible in the next ten years. Preliminary DDR4 projections for the year 2012 show a peak bandwidth of 16GB/s. For our projected exaFLOPS system in 2017, we optimistically assume a memory bus bandwidth of 32GB/s. Nevertheless, note that the 3D-PCRAM checkpointing is not limited by memory bandwidth as mentioned in Section 5.2.

Network Bandwidth and Aggregate File System Bandwidth: We assume that their scaling trend will be similar to what we have seen in the past ten years. Hence, we scale the network bandwidth by 10X and file system bandwidth by 5X.

Soft Error Rate (SER) and System MTTF: The failure statistics of Roadrunner are not available yet in the literature, and the accurate projection of overall MTTF for future processors is beyond the scope of this paper. In this work, we simply assume the *hard error rate (HER)* and *other error (i.e. software bug) rate (OER)* remain constant, and only consider the scaling of *soft errors*. A

[2]Consider most MPP systems are used to solve differential equations and other numerical method problems, the required FLOPS scales up with 3 spacial dimensions and 1 temporal dimension, but the required memory size only scales up with 3 spacial dimensions.

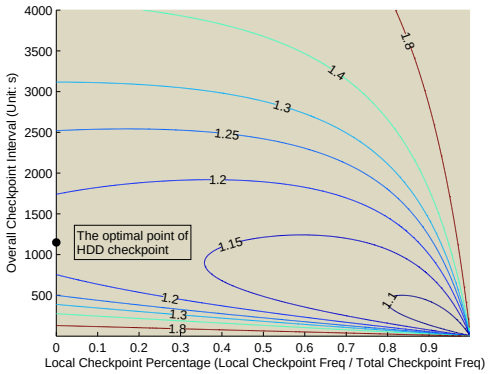

Figure 9: Effect of checkpoint interval and ratio on execution time of Pure-HDD (at the points where X-axis is 0)

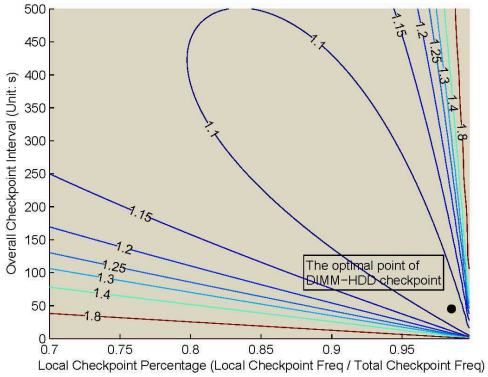

Figure 10: Effect of checkpoint interval and ratio on execution time of DIMM+HDD (a zoom-in version of Figure 9)

Table 9: Memory usage of NPB suite

Workload	Memory Usage	Workload	Memory Usage
BT.C	16.8%	CG.C	21.7%
DC.B	25.0%	EP.C	0.1%
FT.B	100%	IS.C	25.0%
LU.C	14.6%	MG.C	82.4%
SP.C	17.7%	UA.C	11.4%

study from Intel [2] shows that when moving from 90nm to 16nm technology the soft error rate will increase by 32X. Therefore, the total error rate (TER) of exaFLOPS system is modeled as,

$$\begin{aligned} TER_{EFLOPS} &= HER_{EFLOPS} + SER_{EFLOPS} \\ &\quad + OER_{EFLOPS} \\ &= HER_{PFLOPS} + 32 \times SER_{PFLOPS} \\ &\quad + OER_{PFLOPS} \end{aligned} \tag{8}$$

Checkpoint Size: To evaluate the checkpoint overhead for various system configurations, we need the average amount of data written by each node. Since it is hard to mimic the memory trace of a real supercomputer, we execute the NAS Parallel Benchmark (NPB) [21] on an actual system to determine the memory footprint of different workloads. The workloads are chosen from NPB CLASS-C working set size except for workloads DC and FT that employs CLASS-B working set since they are the most complex level that our environment can handle. Table 9 shows the memory usage of workloads that is projected for our baseline petaFLOPS system. We employ the same scaling rule applied for memory size to project the checkpoint size for future systems, thus the memory usage percentage remains the same.

Checkpoints can also be taken in an incremental fashion [22] to reduce checkpointing overhead. A detailed evaluation of this optimization is presented in Section 6.5. We used HP's COTSon simulator [23] to track the incremental size at page granularity for different checkpoint intervals. We employ the same incremental percentage size for all the projected systems.

Table 8 shows the MPP system configurations for a petaFLOPS and a projected exaFLOPS system. For the configurations between these two ends, we scale the specification values according to the time frame. For all our evaluations we conservatively assume that each checkpoint operation incurs an overhead of $1ms$ to initiate a coordinated checkpoint.

6.3 Performance Analysis

For all our evaluations, we employ the equations derived in Section 3 to determine the execution time of workloads in various systems and scenarios.

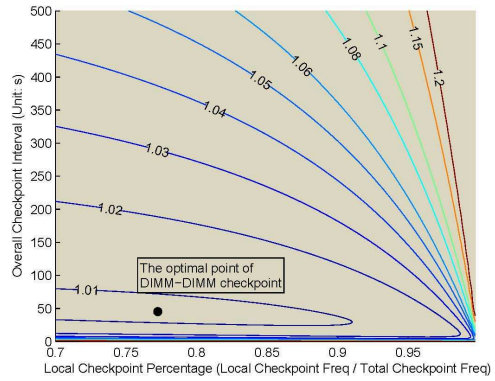

Figure 11: Effect of checkpoint interval and ratio on execution time of DIMM+DIMM

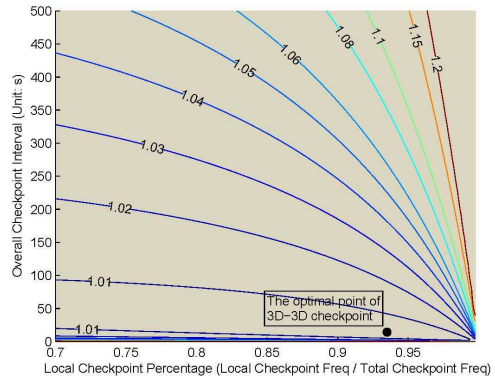

Figure 12: Effect of checkpoint interval and ratio on execution time of 3D+3D

For a given system, based on the system scale and the checkpoint size, the optimal checkpoint frequency can be decided. For this checkpoint frequency, an inherent trade-off exists between the proportion of local and global checkpoints. For example, as the fraction of local checkpoints increases, the overall checkpoint overhead drops, but the recovery time from global checkpoints rises; on the other hand, as the fraction of global checkpoints increases, the recovery time decreases, but the total execution time can take a hit because of the high checkpoint overhead. This trade-off is actually modeled by Eqn. 5 in Section 3, and the optimal values of the checkpoint interval (τ) and the percentage of local checkpointing (p_L) can be found.

This effect is illustrated in Figures 9-12 for the different scenar-

Table 10: The checkpoint overhead and system availability estimations

	Pure-HDD	DIMM+HDD	DIMM+DIMM	3D+3D
Checkpoint overhead (1 PFLOPS)	19.0%	7.4%	0.9%	0.6%
System availability (1 PFLOPS)	84.0%	93.1%	99.1%	99.4%
Checkpoint overhead (10 PFLOPS)	236.7%	23.7%	2.1%	1.0%
System availability (10 PFLOPS)	29.7%	80.8%	97.9%	99.0%
Checkpoint overhead (100 PFLOPS)	-	126.6%	7.2%	2.0%
System availability (100 PFLOPS)	0%	44.1%	93.3%	98.0%
Checkpoint overhead (1 EFLOPS)	-	-	37.0%	4.3%
System availability (1 EFLOPS)	0%	0%	73.0%	95.8%

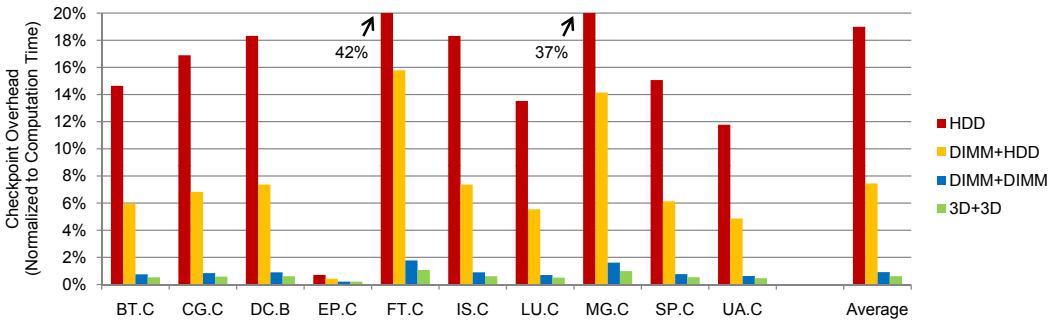

Figure 13: The checkpoint overhead comparison in a 1-petaFLOPS system (normalized to computation time).

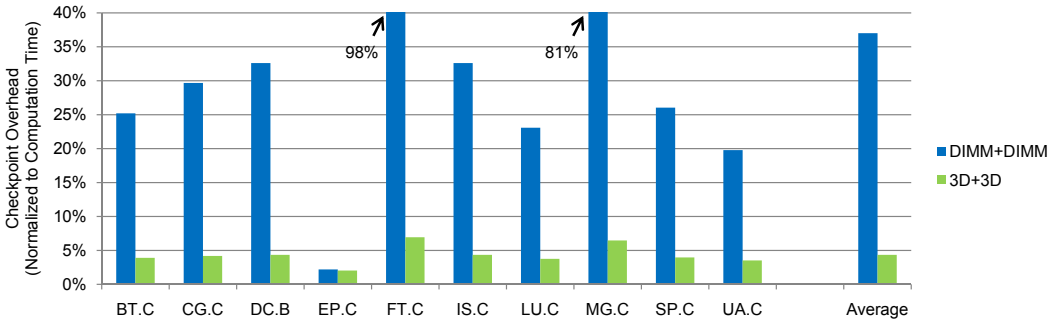

Figure 14: The checkpoint overhead comparison in a 1exaFLOPS system (normalized to computation time).

ios listed in Table 7 for a petaFLOPS system when the workload DC.B is simulated. Not surprisingly the *Pure-HDD* scheme, where all the checkpoints are performed globally using HDD (local checkpoint percentage is 0%), takes the maximum hit in performance. *DIMM+HDD*, including in-node PCRAM as local checkpointing storage, reduces the normalized checkpoint overhead from 18% to 7% with a local checkpointing percentage above 98%. As we change the global checkpointing medium from HDD to PCRAM-DIMM (*DIMM+DIMM*), the checkpoint overhead is dramatically reduced to 0.9% because HDD, the slowest device in the checkpoint scheme, is removed. In addition, since the overhead of global and local checkpoints are comparable in *DIMM+DIMM*, the optimal frequency for local checkpointing reduces to 77.5%. The *3D+3D* scheme that employs 3D DRAM/PCRAM hybrid memory has the least checkpoint overhead. We notice that the local checkpoint percentage in this case goes back to over 93% because the ultra-high 3D bandwidth enables a local checkpointing operation to finish almost instantly. Although the checkpoint overhead reduction achieved by *3D+3D* is similar to that of *DIMM+DIMM* in this case, we will see later that *3D+3D* does make a difference when future MPP systems reach the exascale.

Figure 13 shows the checkpoint overhead in a petascale system by using *pure-HDD*, *DIMM+HDD*, *DIMM+DIMM*, and *3D+3D*, respectively. In average, *DIMM+HDD* reduces the checkpoint overhead by 60% compared to *pure-HDD*. Moreover, the ideal "instant checkpoint" is almost achieved by implementing *DIMM+DIMM* and *3D+3D*. As listed in Table 10, the greatly reduced checkpoint overhead directly translates to the growth of effective computation time, or equivalent system availability.

The advantages of *DIMM+DIMM* and *3D+3D* are clear as the MPP system is scaled towards the exascale level where *pure-HDD* and *DIMM+DIMM* are not feasible any more; Figure 14 demonstrates the results. It can be found that both of *DIMM+DIMM* and *3D+3D* are still workable, and more importantly, the average overhead of *3D+3D* is still less than 5% even in the exascale system. The resulting system availability estimations are listed in Table 10. It shows that our intermediate PCRAM-DIMM and ultimate 3D-PCRAM checkpointing solutions can provide the failure resiliency required by future exascale systems with affordable overhead.

6.4 Power Analysis

Although the proposed techniques are targeted primarily to reduce the checkpoint overhead, they are useful for power reduction as well:

- Since PCRAM is a non-volatile memory technology, it does not consume any power when the system is not taking checkpoints. For example as shown in Table 10, using *3D+3D*

PCRAM checkpoints, during more than 95% of system running time the PCRAM modules can be turned off. Other approaches, i.e. battery-backed DRAM checkpointing, will inevitably leak power even when no checkpoints are being taken. Note that the nap power of a 2GB DRAM-DIMM is about 200mW [24], using battery-backed DRAM checkpointing in 1-petaFLOPS systems will inevitably waste about 20kW power. In contrast, our PCRAM checkpointing module does not consume any power during the computation time.

- With future supercomputers dissipating many mega watts, it is important to keep high system availability to ensure that the huge power budget is effectively spent on useful computation tasks. Our experiment shows that *DIMM+DIMM* can maintain the system availability above 73% and *3D+3D* can achieve near 96% system availability even on the exascale level.

6.5 Orthogonal Techniques

The PCRAM hybrid local/global checkpointing scheme can be combined with other orthogonal techniques to further reduce the checkpoint overhead. For example, *Remote Direct Memory Access* (RDMA) and *Incremental Checkpointing* are evaluated as two supplementary techniques in our experiment:

- *Remote Direct Memory Access*: Since there are two copies of checkpoints in the PCRAM hybrid local/global checkpointing scheme, the local checkpoint copy can be leveraged as the source of global checkpoints. Therefore, global checkpoints can be taken without halting computations, which means the global checkpoint can be overlapped with computation time as long as the global checkpoint latency is less than the local checkpoint interval.

- *Incremental Checkpoint*: After a full-size checkpoint taken at first, the full-size checkpoint can be followed by a sequence of incremental checkpoints, which only saves the dirty pages that have changed since the previous checkpoint [22]. When the system scale increases, more frequent checkpointing is required. The reduced checkpoint interval can lead to a much smaller incremental checkpoint size, and thus further alleviate the checkpoint overhead.

In our experiments, the effect of RDMA is simulated by the assumption that the global checkpoint latency can be totally hidden as long as it is smaller than the local checkpoint interval. In addition, HPąrs COTSon simulator [23] is used to track the incremental size at page granularity for different checkpoint intervals.

6.6 Scalability

Recall the motivation of the 3D PCRAM checkpointing is to maintain the checkpoint overhead under an acceptable level even when the MPP system reaches the exascale and the entire MPP system is highly unreliable. Hence we evaluate how different checkpointing schemes (as listed in Table 7) scale when the system scale goes up from today's petascale systems to future's exascale systems.

Figure 15 shows the effect of introducing local checkpointing on the total number of nodes in the system. It is clear that even with the incremental checkpointing optimization, the slow HDD checkpointing has trouble scaling beyond 2009 without taking a heavy hit in performance. Although the introduction of local PCRAM-DIMM checkpointing helps scale beyond 10 petaFLOPS, the poor scaling of HDD bandwidth hampers the benefit beyond 2011. The use of PCRAM-DIMM for both local and global checkpoints further raises the bar to a 500 petaFLOPS system. Beyond that, due to the increase in transient errors and poor scaling of memory buses, its overhead increases sharply. The proposed hybrid checkpointing combined together with the 3D PCRAM/DRAM memory shows excellent scalability properties and incurs less than 5% overhead even beyond exascale systems.

Moreover, observing the incremental checkpointing curves in Figure 15, it can be found that applying the incremental checkpoint in the conventional pure-HDD checkpoint does not extend the pure-HDD curve too much. However, when it is combined with PCRAM-based local/global hybrid checkpointing, this technique shows its great enhancement to the baseline schemes. That is because in our PCRAM hybrid checkpoint, the checkpoint interval can be set relatively low, and thus the number of dirty pages created during this interval or the incremental checkpoint size is dramatically reduced. This shows that when the 3D-PCRAM checkpoint is used together with the incremental checkpoint technique, the overall checkpoint overhead is only 3.4%, which can be translated into a MPP system availability of 96.7%. This negligible overhead makes the 3D-PCRAM checkpointing scheme an attractive method to provide reliability for future exascale systems.

6.7 SER Sensitivity Study

The effectiveness of the PCRAM-based local/global hybrid checkpointing depends on how many system failures can be recovered by local checkpoints. In our basic assumption, the soft error rate will increase by 32X in the exascale system. Combined with the 5X socket increase assumption, we find that the system MTTF almost degrades 116X. While our proposed PCRAM-based checkpointing is insensitive to this system MTTF degradation because over 99% of total failures are locally recoverable based on this assumption, the conventional HDD-based checkpointing is very sensitive to this change.

Although we believe aggressive soft error rate scaling is reasonable considering future "deep-nano" semiconductor processes, we cannot eliminate the possibility that the device unreliability can be hidden by some novel technologies in the future. In addition, our baseline setting, "ASCI Q", is widely considered as an unreliable system due to its non-ECC caches. Therefore, in order to avoid any exaggeration of the conventional checkpointing scalability issue, the scalability trend is re-evaluated with a new assumption that the soft error rate will remain at the same level as today's technology. Figure 16 shows another set of checkpoint overhead projection curves based on this new assumption.

As expected, the checkpoint overhead decreases as the number of soft errors is reduced. However, even with this new assumption, the conventional HDD-based technique (*pure-HDD*) still has trouble scaling beyond the 8-petaFLOPS scale. In contrast, the overhead of our PCRAM-based approach (*DIMM+DIMM* and *3D+3D*) is further reduced to less than 3% by utilizing orthogonal techniques such as incremental checkpointing and RDMA.

7. RELATED WORK

There have been many recent proposals [25–28] on improving checkpointing coverage and reducing its overhead in MPP systems. Chiueh and Deng [25] proposed a diskless checkpointing mechanism that employs volatile DRAM for storing both local and global checkpoints. Their idea is to split the DRAM memory in each node into four segments and employ three-fourths of the memory to make checkpoints. Here, every node's memory has three copies of backup - one in its own node and the remaining two in its neighboring nodes. When a node fails, every other node will use its local

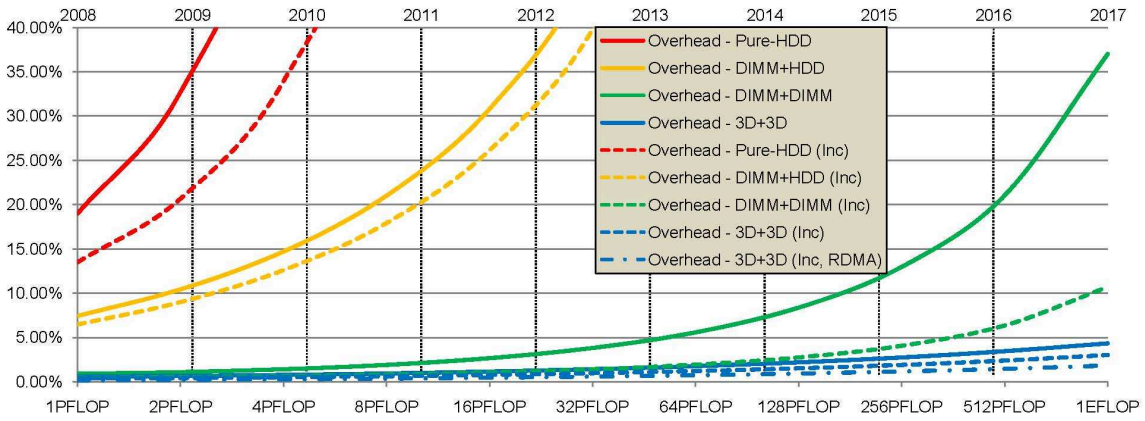

Figure 15: The average estimated checkpoint overhead from petascale systems to exascale systems (normalized to computation time).

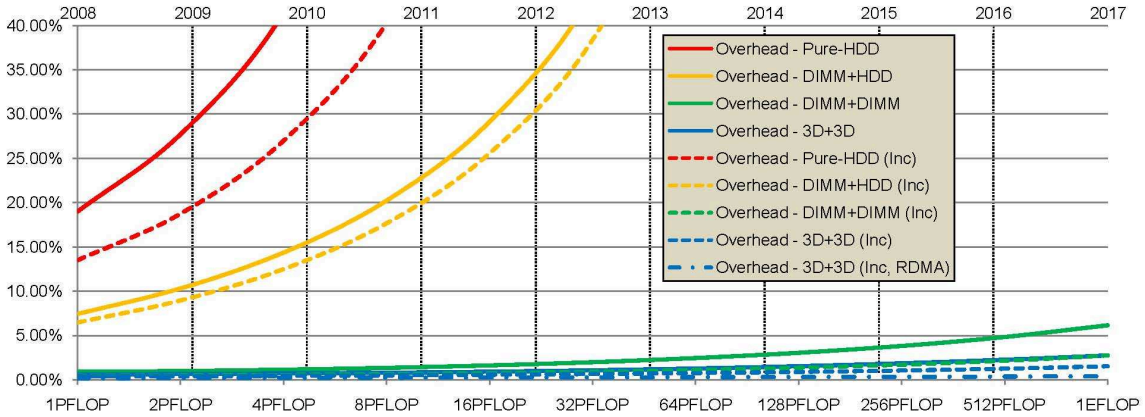

Figure 16: The new checkpoint overhead projection based on the assumption that SER remains constant from petascale to exascale (normalized to computation time).

backup to roll back to the checkpointed state. The content of the node that actually failed is recovered using one of the global copies stored in its neighboring nodes. The reason for maintaining two global copies is to handle failures during checkpointing. While this mechanism looks similar to the PCRAM-DIMM model evaluated in this work, it differs in many key aspects. First, in Chiueh's proposal both local and global checkpoints are synchronized and taken at the same time by stalling the program execution. This not only slows down the checkpoint process, but also increases the cost by making too many unnecessary global checkpoints. With the proposed multilevel checkpointing model, we show that the number of global checkpoints can be significantly reduced (to less than 10% of the local checkpoint count) without losing performance. We also show novel memory organizations are necessary to scale beyond 100 petaFLOPS systems.

Oliner, *et al.* [26] introduced a theory of *cooperative checkpointing* that uses global knowledge of the state and health of the machine to improve performance and reliability by dynamically initiating checkpoints. However, in order to reduce the checkpoint cost, the technique skips some scheduled checkpoints according to the risk of system failure. This decision depends on the accuracy of risk estimation. Unfortunately, an accurate failure prediction or risk estimation is a challenging problem.

Sobe [27] analyzed the overhead reduction by introducing the idea of local checkpoint storage and augmentation with parity, stored on another host. However, his research is still constrained in using HDD as the checkpoint storage. A recent work by Freitas and Wilcke [29] showed that HDD bandwidth is already at its limits in meeting the checkpointing needs of current generation systems. Most recently, Bronevetsky, *et al.* [28] presented a novel compiler analysis for optimizing automated checkpointing. Their work is a hybrid compiler/runtime approach, where the compiler optimizes certain portions of an otherwise runtime checkpointing solution, and then reduces the checkpoint size.

This previous research on checkpoint optimization reduces the checkpoint size, dynamically tunes the checkpoint interval, and sacrifices the system reliability by only supporting limited numbers of node failures. In contrast, our study in this paper shows how to take advantage of emerging PCRAM technology to dramatically improve the checkpoint dumping rate, and is complementary to other advanced checkpointing ideas.

8. CONCLUSION

Checkpointing has been an effective tool for providing reliable and available MPP systems. However, our analysis showed that current checkpointing mechanisms incur high performance penalties and are woefully inadequate in meeting future system demands. To improve the scalability of checkpointing, we proposed a hybrid checkpointing technique that takes checkpoints in both private and globally accessible memory. We then developed mathematical models based on failure rates and system configuration to identify the optimal local/global checkpoint interval that maximizes system performance. A thorough analysis of failure rates shows that a majority of failures are recoverable using local checkpoints, and local checkpoint overhead plays a critical role for MPP

scalability. To improve the efficiency of local checkpoints and maximize fault coverage we propose PCRAM-DIMM checkpointing. PCRAM-DIMM checkpointing enables MPP systems to scale up to 500 petaFlops with tolerable checkpoint overhead. To provide reliable systems beyond this scale, we leverage emerging 3D die stacking and propose 3D PCRAM/DRAM memory for checkpointing. Our 3D design incurs less than 6% overhead in an exascale system by making near instantaneous checkpoints. We also evaluate our mechanism on systems that can take incremental checkpoints and support RDMA. These optimizations further reduce the overhead to 2% compared to our simple 3D scheme. Thus, 3D-PCRAM checkpointing can provide the scalability needed to support future failure-resilient exascale systems.

9. REFERENCES

[1] D. Reed, "High-End Computing: The Challenge of Scale," Director's Colloquium, May 2004.

[2] S. Y. Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[3] R. A. Oldfield, S. Arunagiri, P. J. Teller *et al.*, "Modeling the Impact of Checkpoints on Next-Generation Systems," in *MSST '07. Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies*, 2007, pp. 30–46.

[4] Samsung, Hard Disk Drive, Apr 2009.

[5] G. Grider, J. Loncaric, and D. Limpart, "Roadrunner System Management Report," Los Alamos National Laboratory, Tech. Rep. LA-UR-07-7405, 2007.

[6] S. E. Michalak, K. W. Harris, N. W. Hengartner *et al.*, "Predicting the Number of Fatal Soft Errors in Los Alamos National Laboratory's ASCI Q Supercomputer," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 329–335, 2005.

[7] Los Alamos National Laboratory, Reliability Data Sets, http://institutes.lanl.gov/data/fdata/.

[8] J. W. Young, "A First Order Approximation to the Optimal Checkpoint Interval," *Communications of the ACM*, vol. 17, pp. 530–531, 1974.

[9] J. T. Daly, "A Higher Order Estimate of the Optimum Checkpoint Interval for Restart Dumps," *Future Generation Computer Systems*, vol. 22, no. 3, pp. 303–312, 2006.

[10] S. Y. Lee and K. Kim, "Prospects of Emerging New Memory Technologies," in *ICICDT '04. Proceedings of the 2004 International Conference on Integrated Circuit Design and Technology*, 2004, pp. 45–51.

[11] S. Hanzawa, N. Kitai, K. Osada *et al.*, "A 512kB Embedded Phase Change Memory with 416kB/s Write Throughput at 100μA Cell Write Current," in *ISSCC '07. Proceedings of the 2007 IEEE International Solid-State Circuits Conference*, 2007, pp. 474–616.

[12] F. Pellizzer, A. Pirovano, F. Ottogalli *et al.*, "Novel μTrench Phase-Change Memory Cell for Embedded and Stand-Alone Non-Volatile Memory Applications," in *Proceedings of the 2004 IEEE Symposium on VLSI Technology*, 2004, pp. 18–19.

[13] Y. Zhang, S.-B. Kim, J. P. McVittie *et al.*, "An Integrated Phase Change Memory Cell With Ge Nanowire Diode For Cross-Point Memory," in *Proceedings of the 2007 IEEE Symposium on VLSI Technology*, 2007, pp. 98–99.

[14] A. Pirovano, A. L. Lacaita, A. Benvenuti *et al.*, "Scaling Analysis of Phase-Change Memory Technology," in *IEDM '03. Proceedings of the 2003 IEEE International Electron Devices Meeting*, 2003, pp. 29.6.1–29.6.4.

[15] F. Bedeschi, R. Fackenthal, C. Resta, E. M. Donze, M. Jagasivamani *et al.*, "A Bipolar-Selected Phase Change Memory Featuring Multi-Level Cell Storage," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 217–227, 2009.

[16] X. Dong, N. Jouppi, and Y. Xie, "PCRAMsim: A System-Level Phase-Change RAM Simulator," in *ICCAD '09. Proceedings of the 2009 IEEE/ACM International Conference on Computer-Aided Design*, 2009.

[17] Y. Xie, G. H. Loh, B. Black, and K. Bernstein, "Design Space Exploration for 3D Architectures," *ACM Journal of Emerging Technologies in Computing Systems*, vol. 2, no. 2, pp. 65–103, 2006.

[18] International Technology Roadmap for Semiconductors, "Process Integration, Devices, and Structures 2007 Edition," http://www.itrs.net/.

[19] W. Huang, K. Sankaranarayanan, K. Skadron *et al.*, "Accurate, Pre-RTL Temperature-Aware Design Using a Parameterized, Geometric Thermal Model," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1277–1288, 2008.

[20] D. Vantrease, R. Schreiber, M. Monchiero *et al.*, "Corona: System Implications of Emerging Nanophotonic Technology," in *ISCA '08: Proceedings of the 35th International Symposium on Computer Architecture*, 2008, pp. 153–164.

[21] NASA, "NAS Parallel Benchmarks," http://www.nas.nasa.gov/Resources/Software/npb.html.

[22] J. C. Sancho, F. Petrini, G. Johnson, and E. Frachtenberg, "On the Feasibility of Incremental Checkpointing for Scientific Computing," in *IPDPS '04. Proceedings of the 18th International Parallel and Distributed Processing Symposium*, 2004, pp. 58–67.

[23] E. Argollo, A. Falcon, P. Faraboschi *et al.*, "COTSon: Infrastructure for Full System Simulation," HP Labs, Tech. Rep. HPL-2008-189, 2008.

[24] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating Server Idle Power," in *ASPLOS '09. Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2009, pp. 205–216.

[25] T.-C. Chiueh and P. Deng, "Evaluation of Checkpoint Mechanisms for Massively Parallel Machines," in *FTCS '96. Proceedings of the 26th Annual Symposium on Fault Tolerant Computing*, 1996, pp. 370–379.

[26] A. Oliner, L. Rudolph, and R. Sahoo, "Cooperative Checkpointing Theory," in *IPDPS '06. Proceedings of the 20th International Parallel and Distributed Processing Symposium*, 2006, pp. 14–23.

[27] P. Sobe, "Stable Checkpointing in Distributed Systems without Shared Disks," in *IPDPS '03. Proceedings of the 17th International Parallel and Distributed Processing Symposium*, 2003, pp. 214–223.

[28] G. Bronevetsky, D. J. Marques, K. K. Pingali *et al.*, "Compiler-Enhanced Incremental Checkpointing for OpenMP Applications," in *PPoPP '08. Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2008, pp. 275–276.

[29] R. F. Freitas and W. W. Wilcke, "Storage-Class Memory: The Next Storage System Technology," *IBM Journal of Research and Development*, vol. 52, no. 4/5, 2008.