# On the Trend of Resilience for GPU-Dense Systems

Kyushick Lee
University of Texas at Austin
*kyushick@utexas.edu*

Michael B. Sullivan, Siva Kumar Sastry Hari
Timothy Tsai, Stephen W. Keckler
NVIDIA
{*misullivan,shari,timothyt,skeckler*}*@nvidia.com*

Mattan Erez
University of Texas at Austin
*mattan.erez@utexas.edu*

***Abstract*—Emerging high-performance computing (HPC) systems show a tendency towards heterogeneous nodes that are dense with accelerators such as GPUs. They offer higher computational power at lower energy and cost than homogeneous CPU-only nodes. While an accelerator-rich machine reduces the total number of compute nodes required to achieve a performance target, a single node becomes susceptible to accelerator failures as well as sharing intra-node resources with many accelerators. Such failures must be recovered by end-to-end resilience schemes such as checkpoint-restart. However, preserving a large amount of local state within accelerators for checkpointing incurs significant overhead. This trend reveals a new challenge for the resilience in accelerator-dense systems. We study its impact in multi-level checkpointing systems and with burst buffers. We quantify the system-level efficiency for resilience, sweeping the failure rate, system scale, and GPU density. Our multi-level checkpoint-restart model shows that the efficiency begins to drop at a 16:1 GPU-to-CPU ratio in a 3.6 EFLOP system and a ratio of 64:1 degrades overall system efficiency by 5%. Furthermore, we quantify the system-level impact of possible design considerations for the resilience in GPU-dense systems to mitigate this challenge.**

## I. Introduction

Future HPC systems may face scalability challenges because of resilience. Current end-to-end resilience schemes for HPC rely on checkpoint-restart, which periodically preserves a snapshot of application and/or system state such that any fault can be tolerated by restarting the computation from a previous snapshot (checkpoint). However, because global I/O bandwidth (which is necessary for reliably preserving and restoring checkpoints) scales more slowly than compute, taking a checkpoint will require an increasing amount of time as machines scale. At the same time, hardware faults are generally proportional to system scale, necessitating more frequent checkpoints in the future. This dual challenge hinders efficient computing in time and energy at scale.

Multi-level checkpointing systems mitigate the exhaustive global I/O activity due to checkpointing [1]. It exploits different storage media to create checkpoints; the rollback point depends on the type of each specific failure. For example, a checkpoint written to node-local main memory recovers detectable but uncorrectable memory errors (DUEs) which are relatively frequent. Meanwhile, rare but severe system-wide power failures are handled by rolling back from more reliable media such as the global file system. Multi-level checkpointing systems, therefore, enable multi-tier checkpoints to have different optimal intervals depending on the likelihood of failures for resilience. As a result, multi-level checkpoint reduces the frequency of expensive global checkpoints, and it can avoid I/O checkpointing burstiness.

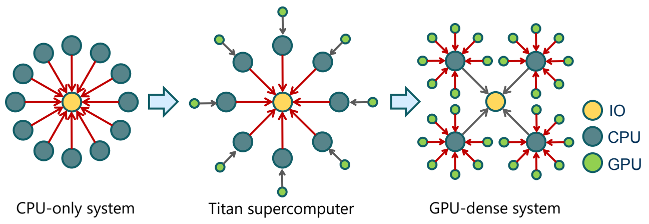

Fig. 1. Illustration of bandwidth hot spots due to check-pointing in the systems with different GPU density.

Emerging HPC systems tend to be heterogeneous with dense accelerators such as GPUs. A single device offers multi-teraflop computation at lower energy and cost than CPU as an effective accelerator for compute-intensive applications. Systems with a high ratio of accelerators to CPUs offer multi-exaflop performance at low node counts [2]–[4], however, the large aggregated memory capacity of GPUs as well as host-side state must be preserved as a part of global state.

Optimizing node-local resilience is important yet challenging with the trend of increasing heterogeneity in HPC systems. GPUs are the most common example of such accelerators in HPC. The high memory capacity of GPUs must be preserved to the host along with the host-side state. However, bursty GPU preservation traffic can slow down this node checkpoint time.

We identify that accelerators introduce a severe problem for the multi-level checkpointing systems at scale. The burstiness of checkpointing traffic within an accelerator-dense node can degrade the overall system performance significantly. At a checkpoint, application state including all accelerators within the node must be preserved, causing a burst of preservation traffic such that every accelerator can utilize only a fraction of the shared intra-node system bus and host memory. Figure 1 illustrates the organization of an accelerator-dense system and highlights the intra-node preservation bandwidth as the primary performance limiter. Unlike with current node-level checkpoint mechanisms, burstiness due to preserving accelerator-local state cannot be resolved by an intra-accelerator copy because this device-local memory is typically heavily utilized. This leads to the host-accelerator preservation bandwidth as being a key parameter for the efficiency of a resilient accelerator-dense system.

We quantify the effect of the bursty GPU preservation traffic on future multi-exaflop, multi-level checkpointing systems and with burst buffers while scaling GPU density. A multi-level checkpoint model shows that a high ratio of GPUs per CPU within a node will degrade throughput by 5—10% when projecting current HPC trends. The failure rates and system hierarchy in such a system at scale limit the high system-level efficiency for resilience in multi-level checkpointing systems. For example, we show that a projected multi-exaflop GPU-dense system with more than 8 GPUs per CPU will perform 10% worse than a system that does not face the burstiness problem.

We discuss possible design considerations to mitigate the bursty system bus accesses for preserving the accelerator state. First, a high-speed interconnect such as NVLink reduces the checkpoint traffic burst. Second, enhancing the reliability of the accelerator device allows for longer checkpoint intervals. We quantify and discuss these system design options for GPU-dense systems.

## II. Background

### A. Accelerators in Supercomputers

The total cost of ownership (TCO) is a critical concern in terms of acquiring and maintaining large-scale systems. Accelerator-dense systems generally require fewer nodes to meet performance goals, offering high energy efficiency and low acquisition costs that considerably improve TCO. Table I shows the energy efficiency of accelerators adopted by the current top 10 supercomputers. It demonstrates that it is more cost-effective to use accelerators to compose a machine with a certain performance target than a CPU-only system. Accordingly, many current supercomputers are heterogeneous, and the ratio of accelerators to CPUs continues to grow. For example, each compute node of Coral Summit system includes 3 GPUs per CPU.

TABLE I
Comparison of performance and power consumption of accelerators.

| Computer Type | Xeon E5 | KNL | V100 | PEZY |
|---|---|---|---|---|
| Performance [TFLOP] | 0.211 | 3.05 | 7.83 | 4.1 |
| Power [Watt] | 145 | 215 | 300 | 130 |
| Perf/Watt [GFLOP/W] | 1.46 | 9.53 | 26.1 | 31.54 |

### B. Detectable Uncorrectable Errors (DUEs) in GPU Memory

GPU memory uses weaker memory protection than CPU memory in order to meet the demand of high memory bandwidth needed for the massive thread-level parallelism. GPU memory such as HBM2 has many channels where each channel performs 32B accesses to a DRAM die. Because a single die participates in a memory transfer, chipkill-like error correction codes (ECC) are not currently feasible for GPU memory; therefore, current GPU memory is protected with single-bit error correcting and double-bit error detecting (SECDED) ECC. As GPU density increases, GPU failure (primarily due to memory DUEs) is by far the most frequent failure mode as projected in Figure 2. Due to their smaller total system size, GPU-dense systems tend to have lower overall failure rates. However, as we show later, the limited and shared intra-node system bus and host memory can potentially limit system efficiency in highly GPU-dense systems.

### C. Globally Coordinated Checkpoint-Restart

Globally coordinated checkpoint-restart is the most practical resilience mechanism in HPC systems. All distributed nodes coordinate to identify globally consistent state across the application; typically at a global collective operation such as a barrier. The global coordination ensures no in-flight messages that potentially cause an inconsistent state while checkpointing, and the snapshot of the global state is preserved to reliable storage. When any failure occurs, the application restarts from the last preserved state (checkpoint) such that it can continue to execute correctly to completion, or until the next failure. Global checkpoint restart is simple but effective resilience scheme in the system where the mean time between failures (MTBF) is much longer than the checkpoint time.

MTBF and checkpoint time determines the optimal checkpoint interval and system-level efficiency for resilience. However, MTBF decreases as system scale grows, requiring more frequent checkpoints to reduce rollback loss. At the same time, when a large-scale application checkpoints, a large number of nodes may attempt to simultaneously write their large state to a bandwidth-constrained global file system. Such checkpointing traffic bursts expand the checkpoint time and degrade system utilization.

### D. Burst Buffers for Checkpointing Systems

A burst buffer is a network-attached I/O appliance that lessens the concern of bursty disk accesses in large HPC systems [5]. It leverages an intermediate I/O layer between the compute and I/O nodes to better manage global disk bandwidth. This new tier of storage operates as a temporary buffer which absorbs surges in global I/O activity. Burst buffers allow compute nodes to continue executing while files written to the burst buffer are asynchronously flushed to the global file system. The burst buffer scales with compute nodes and transparently accelerates global I/O. Alternatively, instead of a remotely shared burst buffer, some HPC clusters, such as Coral Summit, utilize node-local disks as a burst buffer [6]. Furthermore, Agrawal et al. propose to exploit node-local non-volatile memory with a hardware controller that is capable of trickling memory-level checkpoints from node-local cores to global disk [7].

### E. Multi-Level Checkpoint and Restart

Multi-level checkpoint and restart [8] mitigates frequent bursty I/O accesses by efficiently utilizing the bandwidth at each level of the storage hierarchy. Checkpointing to node-local storage such as local disk or memory is fast and it can tolerate intra-node failures but it is not capable of handling rare-yet-severe global failures. Multi-level checkpoint-restart

TABLE II
Configuration of a single node in the base system organization.

| Base Node Configurations | |
|---|---|
| Perf/CPU [TFLOP] | 0.384 |
| Perf/GPU [TFLOP] | 7.8 |
| CPU/Node | 2 |
| Core/CPU | 24 |
| Memory/CPU [GB] | 256 |
| Core/GPU | 5120 |
| Memory/GPU [GB] | 32 |
| SSD BW/Node [GB/s] | 2.15 |
| Total PFS BW [GB/s] | 2500 |
| PCIe/CPU [GB/s, unidirectional] | 64 |
| Node/Cabinet | 18 |

improves overall system-level efficiency by optimizing the checkpoint interval for each level of storage based on the bandwidth and failure rate associated with that level. We quantify the importance of memory-level checkpointing to the performance efficiency of the checkpointing system in Section V.

## III. Study of Failure Rates at Scale

### A. Projecting System Configuration

We project our base system configuration inspired by Coral Summit, a leadership-class computing system at Oak Ridge National Laboratory (ORNL) [2], [9]. The system organization details such as CPU, GPU and GPU density are projected for the Power9 CPU, V100 GPU, and different GPU densities as summarized in Table II. Note that our baseline system is not exactly the same as Coral Summit—for example, Summit has 3 GPUs per CPU, 16GB of GPU memory and NVLink2 between GPUs and CPU.

We scale the GPU density along with the number of cabinets to grow the system. We limit each CPU to 32 PCIe 4.0 lanes, with GPU density scaling provided by switches, if necessary. Figure 3 shows the trend of storage bandwidth and node count in two different system configurations at scale. The aggregated

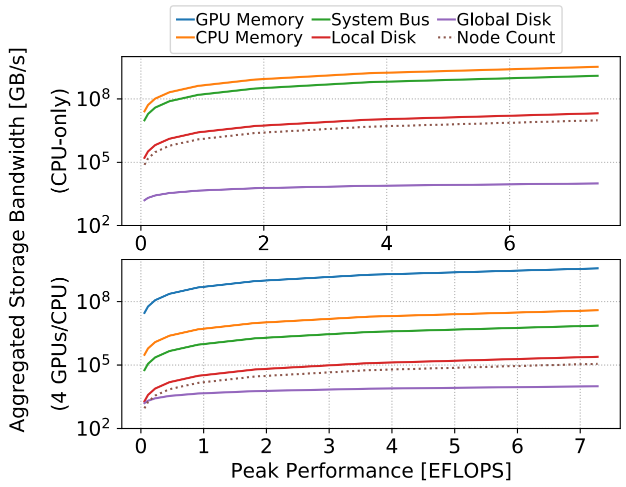

Fig. 3. Storage bandwidth trends for homogeneous and heterogeneous systems with increasing system scales.

CPU, system bus and local disk bandwidths are significantly higher in the CPU-only system than in the system with 4 GPUs per CPU because of node count. To achieve the same compute power, the system without GPUs requires 84X more nodes than the system with 4 GPUs per CPU. We also scale the global file system bandwidth by a factor of 1.3 for every doubling of aggregate system performance, based on rough trends from planned large-scale system upgrades (Edison-to-Cori at NERSC [10], [11], Titan-to-Summit at OLCF [9], and MIRA-to-Aurora at ALCF [12]).

### B. Failures of Heterogeneous Systems at Scale

Our system performance model accounts for not only for the multi-level checkpoint latency but also for the rollback loss associated with failures. We therefore estimate and project failure rates for the GPU-dense nodes and other system components. We use published information on large-scale HPC systems to derive scaling and baseline reliability parameters for our model. Specifically, we use statistics gathered from 261 days on the Blue Waters supercomputer [13], projecting the same per-component failure rates for our Summit-inspired

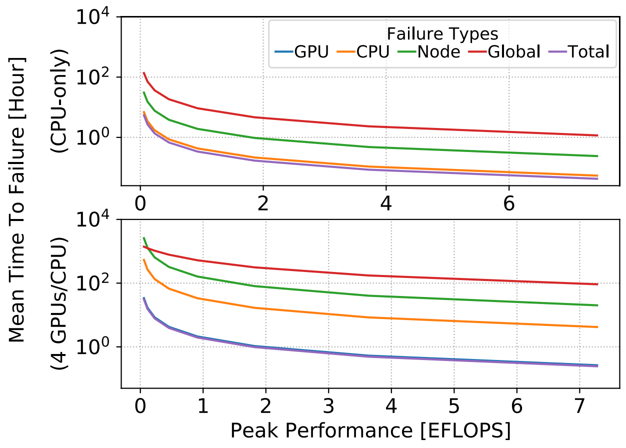

Fig. 2. Mean time to failure (MTTF) of homogeneous and heterogeneous systems with GPUs as the system scale increases.

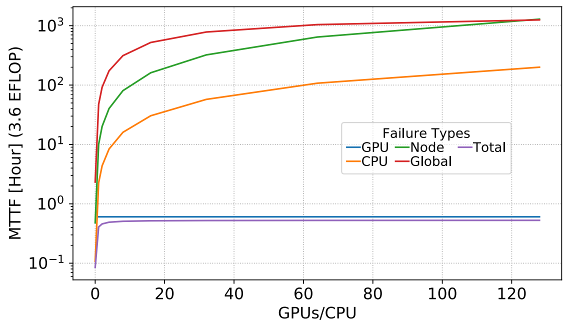

Fig. 4. MTTF at the scale of GPU density at 3.6 EFLOP system.

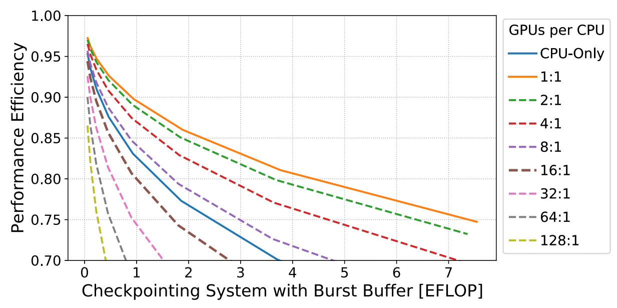

Fig. 5. Performance efficiency vs. system scale and GPU density in the single-level global checkpointing system with burst buffers. The lack of memory-level checkpoint hinders efficient resilience at scale.

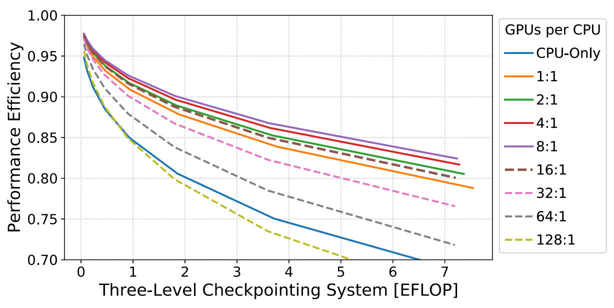

Fig. 6. Performance efficiency of the three-level checkpointing system; the lower effective per-accelerator checkpointing bandwidth degrades efficiency as accelerator density (lines) and node count (x-axis) increase.

systems as measured on Blue Waters. We also combine per-chip memory failure rates based on field measurement study in Titan [14]. For GPU failure statistics, we use 144, 178 and 98 hours for the MTTFs of double-bit error, ECC page retirement, and off-the-bus errors, respectively. The field study of failures in Titan reports that identifying bad GPU cards and resolving their integration issue significantly lowers the off-the-bus failures. Note that we use the off-the-bus MTTF number after resolving faulty GPU devices.

These GPU failure rates are projected to our baseline system inspired by Coral Summit. The system-level failure rate in our study is lower than in prior work. For example, the total failure rate of our 0.9 EFLOP machine is 1.34E-04 (2-hour mean time between interrupts), while that of the 1 EFLOP machine in the previous work [15] is 5.56E-04–4.15 times higher. The main reason for our lower failure rates is the more realistic GPU-dense machine organization we consider: Summit-like nodes with tens or hundreds of teraflops per node allow exascale systems with fewer than $\frac{1}{27}$ the number of nodes used in prior work [15], [16].

In the CPU-only system, the large number of compute nodes leads to frequent node failures. Multi-cabinet failures are considered as system-wide global failures which need to be recovered from global disk while they still scale with the number of nodes. The global failure rate also includes environmental failures such as power failures which do not scale along with the node count. Heterogeneous systems with GPUs increase the MTTF by reducing the node count for a given performance target. GPU failures dominate the overall failures in heterogeneous systems. However, the reduced number of compute nodes with GPUs improves total failure rates at scale.

Figure 4 shows the MTTF of each system component at different GPU-to-CPU ratios in a 3.6 EFLOP system. We scale the GPUs per CPU from 0 to 128. The CPU-only system at 3.6 EFLOP (zero GPU per CPU) shows very low MTTF because of the required large node count. While the GPU MTTF remains the same at different GPU densities in the system, the MTTFs of the other failure types gradually increase along with GPU density. This is because the total GPU count in the system remains constant with scaling the density of GPUs which mainly contribute to the performance target (3.6 EFLOP). As GPU density increases, however, the total cabinet count decreases for the same EFLOP, which reduces the node and/or system-wide failures.

## IV. Study of Performance Efficiency at Scale

Figure 5 and Figure 6 demonstrate the problem of GPU checkpoint traffic bursts within the node when modeling state-of-the-art multi-level checkpointing for GPU-dense exascale systems based on the future Coral Summit supercomputer at ORNL [17]. We estimate the performance efficiency—the ratio between the execution time of the application without any resilience mechanisms without failures and the total time including checkpoint-restart—based on two kinds of system: a single-level global checkpointing with burst buffers and a three-level checkpointing system.

### A. *Checkpointing GPU-Dense System with Burst Buffers*

Though the system with burst buffers utilizes node-local or remote non-volatile storage like the second storage tier of the three-level checkpointing system, global checkpoints are written to the burst buffer, then asynchronously trickled down to the global file system. Since there is not yet any public burst buffer component failure data, we estimate the efficiency with the same analytical model assuming no burst buffer failures. Even though the efficiency of the system with burst buffers is optimistically estimated with this assumption, such a system still suffers from significant efficiency drop as shown in Figure 5. First, this is because the node checkpoint time that scales along with the node counts is not still sufficiently fast for high failure rates at scale, which eventually increases rollback loss at optimal checkpoint intervals. Second, a high GPU:CPU ratio reduces the node counts at the same performance target. However, high GPU density increases node checkpoint time due to limited node-local storage bandwidth (2.15 GB/s in our target system). Note that the aggregated burst buffer bandwidth is proportional to the node count. Thus, the checkpointing system without memory-level checkpoint increases checkpoint time and degrades the system-level efficiency significantly at scale.

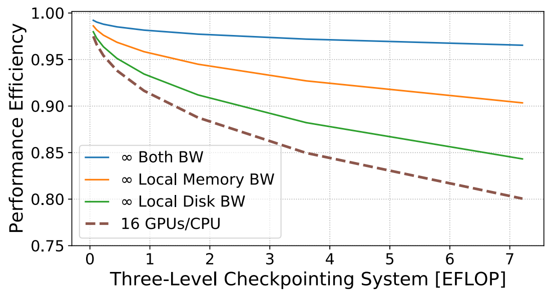

Fig. 7. Limits on the performance efficiency of accelerator-dense multi-level checkpointing. Improving the memory-level checkpoint bandwidth is the most effective option.

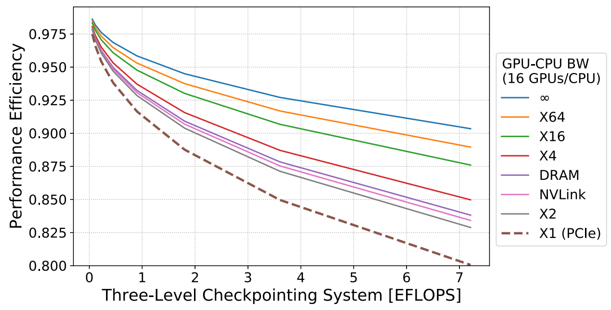

Fig. 8. The system-level efficiency as the system bus bandwidth grows. X1 represents the system bus bandwidth of our baseline system.

## B. *Multi-Level Checkpointing System with GPUs*

Figure 6 demonstrates the effect of a memory-level checkpoint congestion due to GPU checkpoint traffic bursts within the node of the three-level checkpointing system. This organization resembles Scalable Checkpoint Restart [1] on a scaled up version of Summit, which is likely to be the highest performing checkpointing solution supported on that machine [9]. Our second checkpoint tier is located in node-local NVMe SSD storage [18]. We find multi-level checkpointing to be promising yet insufficient to meet the checkpoint-restart efficiency goals. The overhead of checkpointing the GPUs significantly degrades overall system performance at GPU densities greater than 8:1, in particular for very large-scale systems.

Using optimal checkpoint intervals from a multi-level checkpoint model [17], the fast memory-level checkpoint can have a short interval to reduce rollback distance on memory-level errors. However multi-level checkpoints, alone, are insufficient for providing high efficiency for large-scale GPU-rich systems. Figure 7 shows the efficiency of a three-level checkpoint in a 16 GPUs/CPU node with infinite intra-node preservation bandwidth, infinite local disk bandwidth, and both. This analysis shows that quickly storing GPU data to host memory is by far the most effective approach for regaining system performance.

## V. Mitigating Bursty GPU Preservation

### A. *High-Speed GPU–CPU Interconnect.*

All accelerators within the node share the host memory and interconnect bandwidth. Because the maximum GPU—CPU transfer rate in our target system configuration depends on the limited interconnect bandwidth, a high-speed link such as NVLink2 or a wide PCIe bus can reduce the memory-level checkpoint time to improve the overall system-level efficiency in the multi-level checkpointing systems. We study the effect of interconnect bandwidth between GPUs and CPU to overall system performance efficiency.

Figure 8 shows the projected system-level efficiency at different system bus bandwidths at scale. NVLink offers 150GB/s for each CPU[1], improving the performance efficiency by 1.4% at 3.6 EFLOP. Higher GPU–CPU copy bandwidth fundamentally resolves the congestion due to bursty interconnect activity for preserving GPU state, but scaling past host DRAM bandwidth speeds will make the system memory the bottleneck unless both are improved. We show that the 170 GB/s DRAM bandwidth of our Summit-like node offers only slightly better efficiency than NVLink interconnect speeds.

### B. *Sensitivity Study of GPU failure rate.*

Reliable GPU devices enable longer memory-level checkpoint intervals, improving the overall system-level efficiency. We study the impact of GPU failure rates on the resilience of the three-level checkpointing system. Figure 9 shows the performance efficiency of a three-level checkpointing system at 3.6 exaflops with 4, 16, and 64 GPUs per CPU. The efficiency of three-level checkpointing with 64 GPUs per CPU is sensitive to GPU failure rates, decreasing to less than 35% at 50X rates. However, 4 GPUs per CPU sustain 60% in performance efficiency at highly-scaled GPU failure rates. On the other hand, 0.1X GPU failure rates allows multi-level checkpoint to achieve 90% efficiency for the 3.6 exaflop system. This implies that reliable GPU devices allows the use

[1]We use the same NVLink configuration as an IBM Power9 CPU—6 bricks of NVLink2 at 25GB/s unidirectional bandwidth per brick [19].

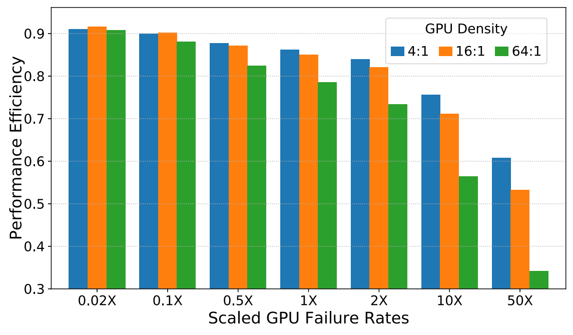

Fig. 9. GPU FIT scaling and efficiency in multi-level check-pointing system with 3.6 EFLOP, 16 GPUs per CPU configuration.

of high GPU density for the performance efficiency target. This is desirable, since a high GPU density considerably reduces the node count and system TCO.

## VI. Related Work

Checkpointing accelerator state is becoming important as accelerators increase in popularity for HPC. Recent accelerators such as GPUs have large memories and retain data in those memories for long periods without saving it to the CPU host. Therefore, including GPU memory state in the globally-coordinated checkpoints is essential. With the prevalence of GPUs, checkpointing systems for CUDA applications are being explored [20]–[22]. Checkpoint-restart for GPUs preserves the state of a CUDA program which is comprised of two component. The first component is a host-side state, such as driver memory, and open files and sockets, which the host checkpointing system preserves. The second component is the state within the GPU device memory that is required to correctly restart the application. The prototypes for GPU checkpoint-restart demonstrate both the feasibility of checkpointing GPU state and they identify the major checkpoint overhead, which is transferring data from the GPU to the host. We further characterize this overhead across a broad range of future system organizations and failure rates.

## VII. Conclusion

We find the potential problem of multi-level checkpoint and restart model to be employed in GPU-dense system and quantify the effect of bursty GPU preservation on the system-level efficiency. We find aggressive checkpoint-restart strategies such as burst buffers or multi-level checkpoint to be promising yet insufficient to meet the checkpoint-restart efficiency goals for GPU-dense systems. The overheads of checkpointing the GPUs significantly degrade overall system performance at the densities greater than 8:1, in particular at very large system scales. Our analysis, discussed in Section V, shows that quickly storing GPU checkpointing data to its host CPU memory and enhancing GPU reliability can effectively regain any lost system performance.

## References

[1] A. Moody, G. Bronevetsky, K. Mohror, and B. R. d. Supinski, "Design, modeling, and evaluation of a scalable multi-level checkpointing system," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1–11, IEEE Computer Society, 2010.

[2] F. Foertter, "Preparing GPU-accelerated applications for the Summit supercomputer." http://on-demand.gputechconf.com/gtc/2017/presentation/s7642-fernanda-foertter-preparing-gpu-accelerated-app.pdf, May 2017. GPU Technology Conference (GTC).

[3] Roy Kim, "NVIDIA DGX SATURNV ranked world's most efficient supercomputer by wide margin." https://blogs.nvidia.com/blog/2016/11/14/dgx-saturnv/, 2016. NVIDIA Blog.

[4] Tiffany Trader, "TSUBAME3.0 points to future HPE Pascal-NVLink-OPA server." https://www.hpcwire.com/2017/02/17/tsubame3-0-points-future-hpe-pascal-nvlink-opa-server/, 2017. HPC Wire.

[5] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn, "On the role of burst buffers in leadership-class storage systems," in *Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on*, pp. 1–11, IEEE, 2012.

[6] C. Zimmer, "Summit burst buffer." https://www.olcf.ornl.gov/wp-content/uploads/2018/05/Intro_Summit_Burst-Buffer-Webinar.pdf, 2018.

[7] A. Agrawal, G. H. Loh, and J. Tuck, "Leveraging near data processing for high performance checkpoint/restart," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, IEEE, 2017.

[8] A. Moody, G. Bronevetsky, K. Mohror, and B. R. d. Supinski, "Design, modeling, and evaluation of a scalable multi-level checkpointing system," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, (Washington, DC, USA), IEEE Computer Society, 2010.

[9] S. S. Vazhkudai, B. R. de Supinski, A. S. Bland, A. Geist, J. Sexton, J. Kahle, C. J. Zimmer, S. Atchley, S. Oral, D. E. Maxwell, *et al.*, "The design, deployment, and evaluation of the CORAL pre-exascale systems," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, p. 52, Nov 2018.

[10] NERSC, "Edison storage and IO." http://www.nersc.gov/users/computational-systems/edison/file-storage-and-i-o/.

[11] NERSC, "Cori storage and IO." http://www.nersc.gov/users/computational-systems/cori/file-storage-and-i-o/.

[12] Intel, "Ushering in a new era." https://www.intel.com/content/dam/www/public/us/en/documents/presentation/intel-argonne-aurora-announcement-presentation.pdf, April 2015.

[13] C. Di Martino, Z. Kalbarczyk, R. K. Iyer, F. Baccanico, J. Fullop, and W. Kramer, "Lessons learned from the analysis of system failures at Petascale: The case of Blue Waters," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2014.

[14] D. Tiwari, S. Gupta, *et al.*, "Understanding GPU errors on large-scale HPC systems and the implications for system design and operation," in *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, pp. 331–342, IEEE, 2015.

[15] J. T. A. Agrawal, G. Loh, "Leveraging near data processing for high-performance checkpoint/restart," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2017.

[16] Y. Chen, "Towards scalable I/O architecture for exascale systems," in *Proceedings of the Workshop on Many Task Computing on Grids and Supercomputers (MTAGS)*, pp. 43–48, 2011.

[17] S. Di, M. S. Bouguerra, L. Bautista-Gomez, and F. Cappello, "Optimization of multi-level checkpoint model for large scale HPC applications," in *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pp. 1181–1190, IEEE, 2014.

[18] Samsung, "Samsung PM1725a NVMe SSD." https://www.samsung.com/semiconductor/global.semi.static/Samsung_PM1725a_NVMe_SSD-0.pdf.

[19] A. Caldeira, "IBM Power System AC922 Introduction and Technical Overview." https://www.redbooks.ibm.com/redpapers/pdfs/redp5472.pdf, March 2018.

[20] H. Takizawa, K. Sato, K. Komatsu, and H. Kobayashi, "CheCUDA: A checkpoint/restart tool for CUDA applications," in *Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp. 408–413, IEEE, 2009.

[21] A. Nukada, H. Takizawa, and S. Matsuoka, "NVCR: A transparent checkpoint-restart library for NVIDIA CUDA," in *Proceedings of the International Symposium on Parallel and Distributed Processing (IPDPS) Workshops*, pp. 104–113, IEEE, 2011.

[22] R. Garg, A. Mohan, M. Sullivan, and G. Cooperman, "CRUM: Checkpoint-restart support for CUDA's unified memory," in *Proceedings of International Conference on Cluster Computing (CLUSTER)*, 2018.