Evaluating Runtime and Power Requirements of Multilevel Checkpointing MPI Applications on Four Different Parallel Architectures: An Empirical Study

Xingfu Wu, Valerie Taylor
Argonne National Laboratory
University of Chicago
Email: {xingfu.wu, vtaylor}@anl.gov

Zhiling Lan
Department of Computer Science
Illinois Institute of Technology
Email: lan@iit.edu

Abstract—**While reducing execution time is still the major objective for high performance computing, future systems and applications will have additional power and resilience requirements that represent a multidimensional tuning challenge. In this paper we present an empirical study to evaluate the runtime and power requirements of multilevel checkpointing MPI applications using the FTI (Fault Tolerance Interface) library. We develop the FTI version of Intel MPI benchmarks to evaluate how FTI affects MPI communication. We then conduct experiments with two applications — an MPI heat distribution application code (HDC), which is compute intensive, and the benchmark STREAM, which is memory intensive — on four parallel systems: Cray XC40, IBM BG/Q, Intel Haswell, and AMD Kaveri. We evaluate how checkpointing and bit-flip failure injection affect the application runtime and power requirements. The experimental results indicate that the runtime and power consumption for both applications vary across the different architectures. Both Cray XC40 and AMD Kaveri with dynamic power management exhibited the smallest impact, whereas Intel Haswell without dynamic power management manifested the largest impact. Bit-flip failure injections with and without different bit positions for FTI had little impact on runtime and power. This provides us a good start to understand the tradeoffs among runtime, power and resilience on these architectures.**

1. Introduction

The insatiable demand for computational power continues to drive the deployment of ever-growing parallel systems. Production systems with hundreds of thousands of components are being designed and deployed. Future parallel systems are expected to have millions of processors and hundreds of millions of cores. These systems are increasingly complex, with hierarchically configured many-core processors and accelerators, together with a deep and complex memory hierarchy. As a result of the complexity, applications face an enormous challenge in exploiting the necessary parameters for efficient execution [19]. While reducing execution time is still the major objective for high performance computing, future systems and applications will have additional power and resilience requirements that represent a multidimensional tuning challenge. To embrace these key challenges, we must understand the complicated tradeoffs among runtime, power, and resilience. In this paper we present an empirical study to evaluate runtime and power requirements for multilevel checkpointing on four different parallel architectures.

Real-world scientific applications take a long time to execute on supercomputers, thereby relying on resilience techniques to successfully finish the long executions. A number of resilience methods have been developed for preventing or mitigating failure impact. Existing resilience strategies can be broadly classified into four approaches: checkpoint based, redundancy based, algorithm based, and proactive methods. Checkpoint/restart is a long-standing fault tolerance technique to alleviate the impact of system failures, in which the applications save their state periodically, then restart from the last saved checkpoint in the event of a failure. Multilevel checkpointing is the state-of-the-art design of checkpointing, and focuses on reducing checkpoint overhead to improve checkpoint efficiency. Such checkpointing libraries include FTI (Fault Tolerance Interface) [3] [17], SCR (Scalable Checkpoint/Restart) [34] [28], and diskless checkpointing [31]. Redundancy approaches improve resilience by replicating data or computation [14] [15] [16]. Algorithm-based fault tolerance methods maintain a coded, global consistent state of the computation in memory by modifying applications to operate on encoded data [20] [7] [5]. Proactive methods take preventive actions before failures, such as software rejuvenation and process or object migration [29].

While fault tolerance methods and power capping techniques continue to evolve, tradeoffs among execution time, power efficiency, and resilience strategies are still not well understood. Fault tolerance studies focus mainly on the tradeoffs between execution time, fault tolerance overhead, and resiliency, whereas most power management studies focus on the tradeoffs between execution time and power. Understanding the tradeoffs among all of these factors is crucial because future machines will be built under both reliability and power constraints. Recent research has focused on a theoretical analysis of energy and runtime for fault tolerance protocols [2] [1] [26] [12] [13]. Because of the potential impacts of various architectures, experiments are essential in order to fully understand how fault tolerant MPI applications impact both power and runtime on the different architectures.

Currently, the FTI library [17] [3] provides the means to perform fast and efficient application-level checkpointing. FTI leverages local storage, along with data replication and erasure codes, to provide several levels of reliability and performance. Di et al. [10] presented an analytical multilevel periodic checkpoint model based on various types of location-unpredictable failures and proposed an iterative method to find optimal checkpoint intervals using FTI on BG/Q. Balaprakash et al. [2] developed the analytical models of energy and runtime for multilevel checkpointing and analyzed the performance-energy tradeoffs for FTI on BG/Q. Their empirical results (with one fixed checkpointing frequency) indicated that the relative energy overhead due to the adoption of FTI was small on the studied applications and the tradeoffs between the runtime and energy were not significant because the difference between power consumption during computation and multilevel checkpointing was minor. This work motivated us to explore energy and power impacts of other FTI applications with various checkpointing configurations on different architectures.

In this work we present an empirical study to evaluate runtime and power requirements for multilevel checkpointing FTI on four different architectures: Cray XC40, IBM BG/Q, Intel Haswell, and AMD Kaveri. First, we develop the FTI version of the Intel MPI benchmarks (IMB) [21], and use these benchmarks to evaluate how FTI affects MPI communication. We then conduct experiments to evaluate how using FTI with various checkpointing frequencies and bit-flip failure injections at different bit positions affects the runtime and power requirements for an MPI heat distribution application code (HDC) [17] and the memory benchmark STREAM (MPI version) [35]. The experimental results show that the runtime and power consumption for both applications vary across the different architectures. Both Cray XC40 and AMD Kaveri with dynamic power management exhibited the smallest impact, whereas Intel Haswell without dynamic power management manifested the largest impact. Bit flip failure injection with and without different bit positions for FTI had little impact on runtime and power.

The remainder of this paper is organized as follows. Section 2 discusses the four different architectures and their power profiling environments. Section 3 describes multilevel checkpointing and FTI, and evaluates how FTI affects the MPI communication. Section 4 presents detailed experimental results to evaluate how FTI affects power and energy consumptions on the four architectures. Section 5 summarizes this work and discusses future work.

TABLE 1. Specifications of four different architectures

System Name	Cray XC40 Theta	IBM BG/Q Mira	Linux Cluster Shepard	Linux Cluster Cooper
Architecture	Intel KNL	IBM BG/Q	Intel Haswell	AMD Kaveri
Number of nodes	3,624	49,152	36	36
CPU cores per node	64	16	32	4
Sockets per node	1	1	2	1
CPU type and speed	Xeon Phi KNL 7230 1.30GHz	PowerPC A2 1.6GHz	Xeon(R) E5-2698 V3 2.3GHz	AMD A10-7850K 3.7GHz
L1 cache per core	D:32KB/I:32KB	D:16KB/I:16KB	D:32KB/I:32KB	D:16KB/I:96KB
L2 cache per socket	32MB (shared)	32MB (shared)	256KB (per core)	2MB (shared)
L3 cache per socket	None	None	40MB (shared)	None
Memory per node	16GB/192GB	16GB	128GB	16GB
Network	Cray Aries Dragonfly	5D Torus	Mellanox FDR InfiniBand	Mellanox FDR InfiniBand
Power tools	CapMC/PoLiMEr	EMON/MonEQ	PowerInsight	PowerInsight
TDP per socket	215W	55W	135W	65W
Power Management	Yes	No	No	Yes
File System	Lustre PFS	GPFS	Regular NFS	Regular NFS

2. Four Architectures and Environments

We conduct our experiments on four parallel systems with different architectures: Cray XC40 [36] and IBM BG/Q [27] at Argonne National Laboratory and the Intel Haswell [33] and AMD Kaveri [11] at Sandia National Laboratories. Details about each system are given in Table 1. Each Cray XC40 node has 64 compute cores (one Intel Phi Knight Landing (KNL) 7230 with the thermal design power (TDP) of 215 W), shared L2 cache of 32 MB (1MB L2 cache shared by two cores), 16 GB high-bandwidth in-package memory (MCDRAM), 192 GB DDR4 RAM, and a 128 GB SSD. We conduct our experiments with the cache-quad mode to utilize the MCDRAM as a cache on Cray XC40. Each BG/Q node has 16 compute cores (one BG/Q PowerPC A2 1.6 GHz chip with the TDP of 55 W [4]), shared L2 cache of 32 MB and 16 GB memory. Each Haswell node has 32 CPU cores (two Xeon E5-2698 V3 2.3 GHz chips with the TDP of 135 W per chip), shared L3 cache of 40 MB and 128 GB memory. Each Kaveri node has 4 CPU cores and 8 GPU cores (one A10-7850K 3.7 GHz chip with the TDP of 65 W), shared L2 cache of 2 MB and 16 GB memory.

The Cray XC40 system uses Cray Aries Dragonfly network with user access to a Lustre parallel file system with 10 PB of capacity and 210 GB/s bandwidth [36]. The BG/Q system uses a 5D torus network with user access to a GPFS file system [27]. Both the Haswell and Kaveri clusters use a Mellanox fourteen data rate InfiniBand network with regular NFS file systems [33]. Although the AMD Kaveri has GPUs, we use only CPU cores in order to provide consistency among the four systems while exploring differences.

Several general power measurement tools exist, such as PowerPack [18], PowerMon2 [23], and PowerInsight [22], as well as vendor-specific power management tools such as Cray's CapMC and out-of-band and in-band power monitoring capabilities [25], IBM EMON API on BG/Q [4], Intel RAPL [32], and NVIDIA's power management library [30]. In this work, we use PoLiMEr [24] to measure power consumption for the node, CPU and memory at node level on the Cray XC40 system, use PowerInsight to measure the power consumption for the node, CPU, memory, and hard disk at the node level on the Haswell and Kaveri systems, and we use MonEQ [37], an application-level power profiling tool based on IBM EMON API, to collect power profiling data on the BG/Q system.

Cray XC40 [25] [9] provides power management to operate more efficiently by monitoring, profiling, and limiting power usage in order to increase system stability by reducing heat dissipation, reduce utility costs by minimizing power usage when rates are the highest and calculate the actual power cost for individual users and/or jobs. PoLiMEr uses Cray's CapMC to obtain power and energy measurements of the node, CPU and memory. The power sampling rate used is approximately 2 samples per second (default).

PowerInsight provides the measurement for 10 power rails for CPU, memory, disk, and motherboard on the Haswell system and for 7 power rails for CPU, memory, disk, and motherboard on the Kaveri system. The power sampling rate used is 1 sample per second (default). The AMD Turbo CORE [6] on the Kaveri system provides a performance boost technology that maximizes processor core performance in the system's TDP while balancing the power budget between processor and graphics cores that share the same cooling solution.

On BG/Q, EMON API [4] provides 7 power domains to measure the power consumption for the node, CPU, memory, and network at the node-card level. The power sampling rate used is approximately 2 samples per second (default). Each node-card consists of 32 nodes. To obtain the power consumption at the node level, we calculate the average power by dividing by 32. We conduct our experiments on 32 nodes (a node-card) to obtain the power profiling data.

3. Impact of FTI on MPI Communication

Multilevel checkpointing allows applications to take both frequent inexpensive checkpoints and less frequent, more resilient checkpoints in a strategic way, resulting in better efficiency and reduced load on the parallel file system (PFS) [17] [34]. This is achieved by using local storage coupled with data replication in the compute nodes. In this work, we use the multilevel checkpointing library FTI [17] [3] to conduct our experiments.

FTI is a fault tolerance interface that adds a highly reliable layer between the operating system and the application. It provides five application-level subroutines: FTI_Init(), FTI_Protect(), FTI_Snapshot(), FTI_BitFlip(), and FTI _Finalize. The FTI_Init() reads a FTI configuration file before the application starts the real execution, delegates one process per node as FT-manager, and creates two MPI communicators: one for the FT-managers and another for the application processes. The MPI communicator created by FTI for the application processes is called FTI_COMM_WORLD, which replaces the global communicator MPI_COMM_ WORLD. FTI_Protect() records the information about variables later checkpointed and/or restored upon a failure. FTI_Snapshot() checks whether a checkpoint needs to be taken. Each time this function is called, a counter is incremented; and when that counter reaches the value set in the FTI configuration file, the checkpoint is taken. FTI_BitFlip() entails injecting bit-flip failures. FTI_Finalize() checks that all the FT-managers have finished their jobs and frees all the resources. Applications can benefit from these FTI subroutines by simply linking to the FTI library.

TABLE 2. PERCENTAGES ON CRAY XC40

# Cores	MPI_Sendrecv	MPI_Reduce	MPI_Allreduce	MPI_Gather
2	7.39%	5.80%	1.39%	1.87%
16	12.70%	5.96%	5.12%	0.99%
32	9.72%	6.20%	4.85%	1.41%
64	7.35%	5.57%	1.20%	1.05%
128	15.74%	6.91%	7.04%	1.44%
256	20.16%	6.11%	4.92%	4.66%
512	21.15%	5.64%	6.81%	2.32%
1024	13.08%	6.21%	13.19%	2.83%

FTI provides the following features for the initial configuration: four-level checkpointing (local write (L1), Partner copy (L2), Reed-Solomon coding (L3), and PFS write (L4)); checkpointing frequency; synchronous (default) or asynchronous at L2, L3, and L4; number of bit-flip failure injections; injection bit position; and injection frequency. The default four-level checkpointing configuration is (3, 5, 7, 11), corresponding to 3 minutes for L1, 5 minutes for L2, 7 minutes for L3, and 11 minutes for L4. The four checkpointing levels correspond to coping with the four types of failures: no hardware failure (software failure), single-node failure, multiple-node failure, and all other failures the lower levels cannot take care of, respectively [10]. FTI applications can perform checkpoints with various frequencies and bit-flip failure injections at different bit positions and frequencies.

To evaluate how FTI affects MPI communication, we developed an FTI version of IMB [21] and used it with the default checkpointing configuration (3,5,7,11) to quantify the overhead of FTI on the three architectures. Because the Kaveri system has only 4 cores per node and a total of 144 cores, we focus mainly on the Cray XC40, BG/Q and Haswell systems for scalability on up to 1024 cores.

The IMB performs a set of MPI performance measurements for point-to-point and global communication operations for a range of message sizes (default from 1 byte to 4 MB). We conducted our experiments for four important MPI subroutines common to the two applications: MPI_Sendrecv (bandwidth), MPI_Reduce (time), MPI_Allreduce (time), and MPI_Gather (time). To quantify the overhead of FTI given in Tables 2, 3 and 4, we use Equation 1. For a given number of cores and a given MPI subroutine, assume that the n message sizes $M_1, M_2, ..., M_n$ are used to measure the performance of the MPI subroutine. For each message size $M_i (i = 1, 2, ..., n)$, we denote its original performance by $V_O(M_i)$ and its performance under FTI by $V_F(M_i)$.

TABLE 3. PERCENTAGES ON BG/Q

# Cores	MPI_Sendrecv	MPI_Reduce	MPI_Allreduce	MPI_Gather
2	3.67%	4.15%	2.49%	0.57%
16	6.11%	1.59%	4.23%	3.14%
32	2.89%	1.83%	3.34%	3.99%
64	3.71%	1.79%	2.92%	3.70%
128	3.18%	1.68%	3.34%	3.58%
256	3.36%	1.83%	2.25%	1.26%
512	3.14%	2.23%	2.45%	1.16%
1024	1.73%	5.57%	3.81%	1.01%

TABLE 4. PERCENTAGES ON HASWELL

# Cores	MPI_Sendrecv	MPI_Reduce	MPI_Allreduce	MPI_Gather
2	2.80%	3.16%	5.26%	2.92%
16	1.56%	2.50%	1.72%	8.74%
32	3.26%	5.26%	1.52%	9.73%
64	1.88%	3.05%	1.08%	9.13%
128	5.07%	2.16%	0.73%	6.32%
256	3.09%	5.02%	1.80%	8.55%
512	10.62%	1.41%	2.99%	5.64%
1024	11.40%	6.27%	5.01%	8.40%

$$\frac{\sum_{i=1}^{n} \left| \frac{V_F(M_i) - V_O(M_i)}{V_O(M_i)} \right|}{n} * 100 \qquad (1)$$

The percentage in Equation 1 is defined as the mean of all absolute ratios of the performance difference between FTI IMB and the original IMB to the original IMB for each message size.

The data in Table 2 shows the impact of FTI on up to 1024 cores on Cray XC40. The percentages for MPI_Sendrecv are between 7.39% and 21.15%. For the two cases when the overhead is beyond 20% (corresponding to 256 and 512 cores), the cause was due to the shared network resources on Cray XC40 [8]. For MPI_Reduce, MPI_Allreduce and MPI_Gather, the percentages are less than 6.91% in most cases except 13.19% for MPI_Allreduce on 1024 cores.

The data in Table 3 indicates that the maximum percentage of the four MPI subroutines is 6.11%. Hence, FTI has little impact on MPI communication on BG/Q. Table 4 shows the impact of FTI on Intel Haswell. Some percentages for MPI_Sendrecv and MPI_Gather are in the range of 10%. In particular, for MPI_Sendrecv, the percentages are 5.07% or less on up to 256 cores. The overhead, however, is beyond 10% for the experiments with 512 and 1024 cores. For MPI_Reduce and MPI_Allreduce, the percentages are 6.27% or less on up to 1024 cores.

4. Experimental Results: Runtime and Power Requirements

In this section, we use two applications to conduct the empirical study. The first application is an FTI version of MPI heat distribution benchmark code (HDC) [17], which computes the heat distribution over time based on a set of initial heat sources. The checkpointing file size is 32 MB per MPI process. HDC is a compute-intensive, weak scaling application. The second application is STREAM (MPI version) [35], a memory-intensive, strong scaling benchmark. An FTI application can perform checkpoints with various frequencies and bit-flip failure injections at different bit positions. We explore the following types of configurations:

1) Original: the original application without checkpointing or failure injection.
2) ckp(3,5,7,11): the application with four-level checkpointing configuration (3, 5, 7, 11) (default) and synchronous at each level (default), where the checkpointing frequency is 3 minutes for L1, 5 minutes for L2, 7 minutes for L3, and 11 minutes for L4.
3) ckp(3,5,7,11)/e1: the application with four-level checkpointing configuration ckp(3, 5, 7, 11) and synchronous at each level, and one bit-flip failure injection, injection bit position 1 (the first bit) and injection frequency (10 seconds), where e1 stands for the bit position 1.

Overall, we explore 10 checkpointing configurations for four-level checkpointing and 7 checkpointing configurations with one bit-flip failure injection and 5 different bit positions.

We use Multiple Metrics Modeling Infrastructure (MuMMI) [38] [39] with the support of PoLiMEr, PowerInsight and MonEQ to collect performance data, power data, and performance counters on a single node of Cray XC40, the Haswell and Kaveri systems, and on 32 nodes of the BG/Q system. For a given application run, we execute the application 13 times on Cray XC40 and BG/Q and 14 times on the Haswell and Kaveri systems to ensure the consistency of the results, while collecting different sets of performance counters for a total of 40 performance counters for each architecture. These performance counters, however, are not used in this paper, but are valuable for other work. We found that the variation of the application runtime is very small (less than 1%), so we use the performance metrics corresponding to the smallest runtime for the empirical study.

The percentage is defined in Equation 2 as the ratio of the difference between one value and the baseline value to the baseline value:

$$\frac{Value - Baseline}{Baseline} * 100. \qquad (2)$$

We use Equation 2 to calculate the percentages in the remainder of this paper.

TABLE 5. PERCENTAGE FOR RUNTIME (S), AVERAGE NODE POWER (W) AND ENERGY (J) FOR HDC ON CRAY XC40

Configuration	Runtime	Node Power	Energy
Original (baseline)	1076	298.37	321046.12
ckp(1,2,3,4)	1.67%	-0.59%	1.08%
ckp(1,3,5,7)	1.58%	-0.54%	1.04%
ckp(2,3,4,5)	1.40%	-0.54%	0.85%
ckp(2,4,6,8)	1.12%	-0.32%	0.79%
ckp(3,4,5,6)	1.21%	-0.71%	0.49%
ckp(3,5,7,11)	1.02%	-0.42%	0.60%
ckp(3,5,7.9)	0.84%	-0.44%	0.40%
ckp(4,5,6,7)	1.12%	-1.17%	-0.06%
ckp(6,7,8,9)	1.12%	-1.00%	0.11%
ckp(8,9,10,11)	1.02%	-0.97%	0.05%

TABLE 6. COMPARISON OF RUNTIME (S), AVERAGE POWER (W), AND ENERGY (J) FOR HDC ON CRAY XC40

Configuration	Runtime	Node Power	CPU Power	Memory Power	Energy
Original (baseline)	1076	298.37	209.64	12.27	321046.12
ckp(1,2,3,4)	1.67%	-0.59%	-0.77%	1.96%	1.08%
ckp(4,5,6,7)	1.12%	-1.17%	-0.22%	-2.36%	-0.06%

4.1. Case I: MPI Heat Distribution Code

4.1.1. Cray XC40. Table 5 provides the percentage for runtime, average node power, and energy per node for 10 checkpointing configurations using the original as the baseline. Compared with the original application, the multilevel checkpointing FTI causes an increase in runtime by up to 1.67%, a decrease in node power consumption by up to 1.17%, and an increase in energy by up to 1.08%. Because of the synchronization at the checkpointing levels, simultaneously writing the checkpointing file of 32 MB per MPI process (a total of 64 processes) on a node (64 cores) during the checkpoints has little impact in the runtime, node power, and energy consumption because of the use of high speed I/O and 16 GB MCDRAM. Among the checkpointing configurations, the percentage range is very little, from -1.17% to 1.67% in runtime, node power, and energy. Notice that node power decreases for all checkpoints because of the dynamic power management of KNL. The checkpointing ckp(4,5,6,7) results in the lowest energy consumption because of the relative small runtime and the smallest node power.

As shown in Table 5, the ckp(4,5,6,7) results in the minimum energy consumption and the ckp(1,2,3,4) results in the maximum energy consumption. Table 6 summarizes the runtime, average power for different components (node, CPU, memory), and energy with and without resilience.

TABLE 7. PERCENTAGE FOR RUNTIME (S), AVERAGE NODE POWER (W) AND ENERGY (J) FOR VARIOUS CHECKPOINTING CONFIGURATIONS WITH 1 ERROR INJECTION FOR HDC ON CRAY XC40

Configuration	Runtime	Node Power	Energy
ckp(3,5,7,11)/e1 (baseline)	1087	297.15	323002.05
ckp(1,2,3,4)/e1	0.74%	-0.31%	0.42%
ckp(2,3,4,5)/e1	0.46%	-0.12%	0.33%
ckp(2,4,6,8)/e1	0	-0.58%	-0.58%
ckp(4,5,6,7)/e1	-0.18%	-0.19%	-0.37%
ckp(6,7,8,9)/e1	-0.37%	-0.18%	-0.55%
ckp(8,9,10,11)/e1	-0.55%	-0.04%	-0.60%

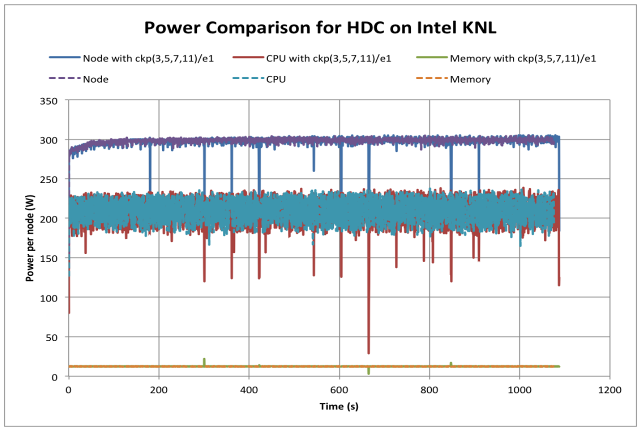

Figure 1. Power consumption over time for HDC on Cray XC40

As expected, the runtime increases with the complexity of the resilience strategy. However, the average power consumption for node and CPU decreases with respect to the various resilience strategies because of the dynamic power management of KNL during the checkpoints. The memory power increases by 1.96% for the ckp(1,2,3,4), the node power decreases by just 0.59%, however, the memory power decreases by 2.36% for the ckp(4,5,6,7), this results in the node power decrease by 1.17%. Therefore, the ckp(4,5,6,7) results in the minimum energy consumption.

In Table 7, we use seven checkpointing configurations from Table 5 corresponding to high frequency and low frequency configurations. The bit-flip injection occurs with bit position 1 (the first bit) and failure injection frequency of 10 seconds. The baseline corresponds to the default checkpoint configuration – ckp(3,5,7,11)/e1. The ckp(8,9,10,11)/e1 resulted in the smallest runtime and the lowest energy because of the fewest number of checkpoints. Overall, the bit-flip failure injection for FTI has little impact on the runtime and power on Cray XC40. Notice that among these checkpointing configurations, the percentage range is very small, from -0.60% to 0.74% in runtime, node power, and energy.

In Figure 1, we compare the power consumptions (for the node, CPU, memory) over time for the application with

TABLE 8. Comparison of runtime (s), average power (W), and energy (J) for bit flip fault injection with different bit positions for HDC on Cray XC40

Configuration	Runtime	Node Power	Energy
ckp(3,5,7,11)/e1 (baseline)	1087	297.15	323002.05
ckp(3,5,7,11)/e8	-0.18%	-0.57%	-0.75%
ckp(3,5,7,11)/e16	-0.18%	-0.64%	-0.82%
ckp(3,5,7,11)/e24	-0.18%	-0.51%	-0.69%
ckp(3,5,7,11)/e31	-0.09%	-0.47%	-0.56%

and without the default ckp(3, 5, 7, 11) and a bit-flip failure injection. "CPU" stands for the CPU power per node for the original application without checkpoint and bit-flip failure injection, and "CPU with ckp(3,5,7,11)/e1" stands for the CPU power per node for the application with the ckp(3, 5, 7, 11) and one bit-flip failure injection. The decrease in node power comes mainly from CPU because CPU power decreases significantly for activities related to checkpointing as shown in Figure 1 due to the dynamic power management support from KNL on Cray XC40. This explains the power decreases shown in Tables 5 and 6.

Bit-flip error injection uses different bit positions for possible error injections. For 32-bit data, there are 32 different bit positions. The different bit positions result in very small differences in runtime. Table 8 shows the percentage of the runtime, average power, and energy at five different bit positions for the ckp(3,5,7,11) with one bit-flip error injection on Cray XC40, where ckp(3,5,7,11)/e8 stands for the ckp(3,5,7,11) with one bit-flip failure injection at the 8th bit position. The frequency for the failure injection is 10 seconds for the five experiments. We observe that the percentage range is very small, from -0.82% to -0.09% in runtime, node power, and energy. Overall, the different bit positions for a bit-flip error injection have a little impact on power and energy on Cray XC40.

4.1.2. IBM BG/Q.

Table 9 provides the percentage for runtime, average node power, and energy per node for 10 checkpointing configurations using the original as the baseline. Compared with the original application, the multilevel checkpointing FTI causes an increase in runtime by up to 9.29%, in node power by up to 7.52%, and in energy by up to 17.51%. Because of the synchronization at the checkpointing levels, simultaneously writing the checkpointing file of 32 MB per MPI process (a total of 16 processes) on a node (16 cores) during the checkpointing ckp(1,2,3,4) results in the most FTI overhead. Among the checkpointing configurations, the percentage range is moderate, from 3.32% to 9.29% in runtime, from 2.21% to 7.52% in node power, and from 6.82% to 17.51% in energy. The moderate increase in energy is due to the relatively small checkpointing file size of 32 MB per MPI process. As expected, the checkpointing ckp(8,9,10,11) results in the lowest energy consumption because of the fewest number of checkpoints.

TABLE 9. Percentage for runtime (s), average node power (W) and energy (J) for HDC on BG/Q

Configuration	Runtime	Node Power	Energy
Original (baseline)	1626	50.77	82552.02
ckp(1,2,3,4)	9.29%	7.52%	17.51%
ckp(1,3,5,7)	8.74%	2.56%	11.53%
ckp(2,3,4,5)	7.44%	3.13%	10.80%
ckp(2,4,6,8)	4.31%	4.41%	8.91%
ckp(3,4,5,6)	6.39%	3.39%	10.00%
ckp(3,5,7,11)	5.29%	4.27%	9.79%
ckp(3,5,7.9)	5.03%	4.29%	9.54%
ckp(4,5,6,7)	5.65%	3.22%	9.17%
ckp(6,7,8,9)	8.99%	2.21%	11.39%
ckp(8,9,10,11)	3.32%	3.39%	6.82%

TABLE 10. Comparison of runtime (s), average power (W), and energy (J) on BG/Q

Configuration	Runtime	Node Power	CPU Power	Memory Power	Network Power	Energy
Original (baseline)	1626	50.77	30.99	6.89	1.72	82552.02
ckp(1,2,3,4)	9.29%	7.52%	8.55%	9.14%	8.14%	17.51%
ckp(8,9,10,11)	3.32%	3.39%	1.58%	13.79%	8.14%	6.82%

As shown in Table 9, the ckp(8,9,10,11) results in the minimum energy and the ckp(1,2,3,4) results in the maximum energy. Further, Table 10 summarizes the runtime, average power for different components (node, CPU, memory, and network), and energy with and without resilience. The average power consumption for different system components (node, CPU, memory, and network) increases. Because FTI launches an FTI-dedicated process per node to guarantee that the library does not cause any damage to the application communication channels, it results in more network power increase by 8.14%. The CPU power for ckp(1,2,3,4) is larger than that for ckp(8,9,10,11), however, the memory power for ckp(1,2,3,4) is smaller than that for ckp(8,9,10,11). This results in larger node power for ckp(1,2,3,4). Hence, the increase in runtime and power leads to the large increase in energy by 17.51%.

In Table 11, the ckp(8,9,10,11)/e1 resulted in the smallest runtime, but the ckp(6,7,8,9)/e1 resulted in the lowest energy because of its relatively small runtime and node power. Overall, the bit-flip failure injection for FTI has little impact on the runtime and power on BG/Q. Notice that among these checkpointing configurations, the percentage range is small, from -1.94% to 3.33% in runtime, from -0.79% to 2.90% in node power, and from -0.74% to 4.75% in energy.

In Figure 2, we compare the power consumptions (for the node, CPU, memory, and network) over time for the

TABLE 11. PERCENTAGE FOR RUNTIME (S), AVERAGE NODE POWER (W) AND ENERGY (J) FOR VARIOUS CHECKPOINTING FREQUENCIES WITH 1 ERROR INJECTION FOR HDC ON BG/Q

Configuration	Runtime	Node Power	Energy
ckp(3,5,7,11)/e1 (baseline)	1718	53.4	91741.20
ckp(1,2,3,4)/e1	3.33%	-0.79%	2.52%
ckp(2,3,4,5)/e1	1.79%	2.90%	4.75%
ckp(2,4,6,8)/e1	-1.24%	0.71%	-0.54%
ckp(4,5,6,7)/e1	-0.05%	0.66%	0.60%
ckp(6,7,8,9)/e1	-1.18%	0.45%	-0.74%
ckp(8,9,10,11)/e1	-1.94%	2.30%	0.31%

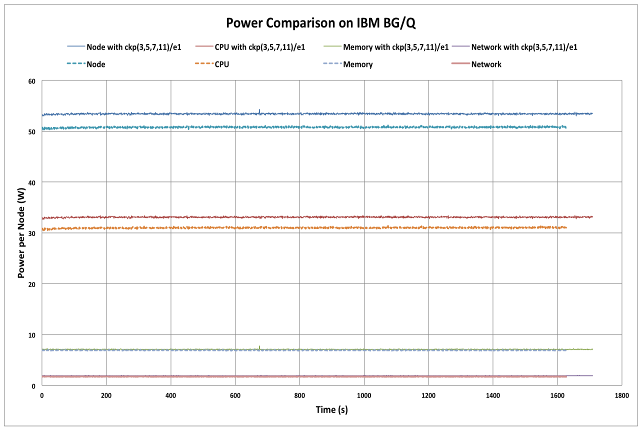

Figure 2. Power consumption over time for HDC on BG/Q

application with and without the default ckp(3, 5, 7, 11) and a bit-flip failure injection. The increase in node power comes mainly from CPU because CPU has to deal with additional activities related to checkpointing. This also explains the power increase shown in Tables 9 and 10.

Bit-flip error injection uses different bit positions for possible error injections. For 32-bit data, there are 32 different bit positions. The different bit positions result in very small differences in runtime. Table 12 shows the percentage

TABLE 12. COMPARISON OF RUNTIME (S), AVERAGE NODE POWER (W), AND ENERGY (J) FOR BIT FLIP FAULT INJECTION WITH DIFFERENT BIT POSITIONS FOR HDC ON BG/Q

Configuration	Runtime	Node Power	Energy
ckp(3,5,7,11)/e1 (baseline)	1718	53.40	91741.20
ckp(3,5,7,11)/e8	-0.14%	-1.76%	-1.89%
ckp(3,5,7,11)/e16	-0.03%	-2.85%	-2.88%
ckp(3,5,7,11)/e24	-0.17%	0.07%	-0.10%
ckp(3,5,7,11)/e31	0.04%	0.11%	0.15%

TABLE 13. PERCENTAGE FOR RUNTIME (S), AVERAGE NODE POWER (W), AND ENERGY (J) FOR HDC ON HASWELL

Configuration	Runtime	Node Power	Energy
Original (baseline)	1414.14	338.39	478530.83
ckp(1,2,3,4)	26.12%	1.74%	28.31%
ckp(1,3,5,7)	25.94%	1.66%	28.03%
ckp(2,3,4,5)	20.14%	2.93%	23.65%
ckp(2,4,6,8)	13.63%	5.93%	20.37%
ckp(3,4,5,6)	15.42%	5.40%	21.65%
ckp(3,5,7,11)	15.76%	5.12%	21.69%
ckp(3,5,7.9)	14.49%	5.85%	21.19%
ckp(4,5,6,7)	14.90%	4.80%	20.87%
ckp(6,7,8,9)	10.03%	7.09%	17.83%
ckp(8,9,10,11)	9.43%	8.40%	18.63%

of the runtime, average power, and energy at five different bit positions for the checkpointing ckp(3,5,7,11) with one bit-flip error injection on BG/Q. The frequency for the failure injection is 10 seconds for the five experiments. We observe that the percentage range is very small, from -0.17% to 0.04% in runtime, from -2.85% to 0.11% in node power, and from -2.88% to 0.15% in energy. Overall, the different bit positions for a bit-flip error injection have a small impact on power and energy on BG/Q.

4.1.3. Intel Haswell. Table 13 shows the percentage for runtime, average power and energy per node for 10 checkpointing configurations. Compared with the original application, the FTI causes an increase in time by up to 26.12%, in the node power by up to 8.40%, and in energy by up to 28.31%. Because of the regular NFS file system used on the Haswell and synchronization at the checkpointing levels, we find that simultaneously writing the checkpointing file of 32 MB per MPI process (a total of 32 processes) on a node (32 cores) during the checkpoints results in the most FTI overhead. Among these checkpointing configurations, the percentage ranges from 9.43% to 26.12% in runtime, from 1.66% to 8.40% in node power, and from 17.83% to 28.31% in energy. From the table, we observe that the checkpointing ckp(6,7,8,9) results in the lowest energy consumption among the checkpointing configurations, although ckp(8,9,10,11) results in the smallest runtime.

As shown in Table 13, the ckp(6,7,8,9) results in the minimum energy consumption and the ckp(1,2,3,4) results in the maximum energy consumption. Further, Table 14 summarizes the runtime, average power, and energy with and without resilience. For both checkpointing configurations, the power consumptions in node and CPU increase, and the disk power increases by more than 43%. We note that the most frequent checkpointing ckp(1,2,3,4) also results in 11.67% less memory power because of large, shared unified L3 cache (40 MB), large memory (128 GB), and frequent

TABLE 14. Comparison of runtime (s), average power (W), and energy (J) for HDC on Haswell

Configuration	Runtime	Node Power	CPU Power	Memory Power	Disk Power	Energy
Original (baseline)	1414.14	338.39	259.06	63.99	1.64	478530.83
ckp(1,2,3,4)	26.12%	1.74%	4.85%	-11.67%	43.29%	28.31%
ckp(6,7,8,9)	10.03%	7.09%	9.93%	-4.00%	44.51%	17.83%

TABLE 15. Percentage for runtime (s), average node power (W), and energy (J) for various checkpointing frequencies with 1 error injection for HDC on Haswell

Configuration	Runtime	Node Power	Energy
ckp(3,5,7,11)/e1 (baseline)	1640.15	357.23	585910.78
ckp(1,2,3,4)/e1	8.45%	-3.51%	4.64%
ckp(2,3,4,5)/e1	3.17%	-1.71%	1.40%
ckp(2,4,6,8)/e1	-2.75%	-0.10%	-2.84%
ckp(4,5,6,7)/e1	-0.40%	0.29%	-0.12%
ckp(6,7,8,9)/e1	-4.40%	1.53%	-2.93%
ckp(8,9,10,11)/e1	-5.83%	2.59%	-3.38%

checkpointing interrupts. This may results in the large runtime increase by 26.12%. Therefore, the large increase in runtime for the ckp(1,2,3,4) leads to the large increase in energy by 28.31%.

In Table 15, the checkpointing ckp(8,9,10,11)/e1 results in the lowest energy consumption because of the fewest numbers of checkpoints. Among these checkpointing configurations, the percentage ranges from -5.83% to 8.45% in runtime, from -3.51% to 2.59% in node power, and from -3.38% to 4.64% in energy. Overall, the bit-flip failure injection for FTI has a small impact on the runtime and power on Haswell.

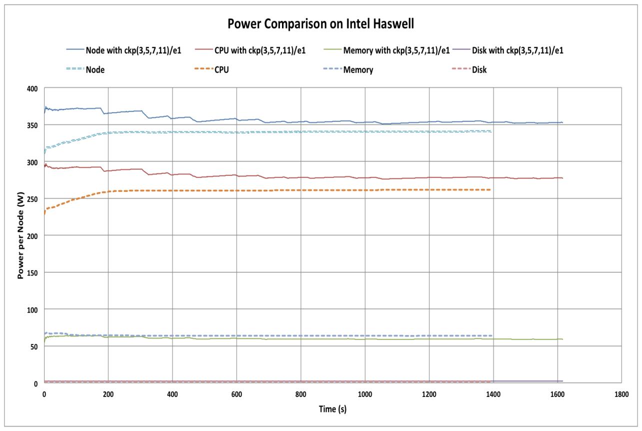

Figure 3. Power over time for HDC on Haswell

TABLE 16. Comparison of runtime (s), average node power (W), and energy (J) for bit-flip fault injection with different bit positions for HDC on Haswell

Configuration	Runtime	Node Power	Energy
ckp(3,5,7,11)/e1 (baseline)	1640.15	357.23	585910.78
ckp(3,5,7,11)/e8	0.18%	-0.27%	-0.09%
ckp(3,5,7,11)/e16	1.14%	-0.49%	0.65%
ckp(3,5,7,11)/e24	0.32%	-0.26%	0.06%
ckp(3,5,7,11)/e31	0.42%	-0.32%	0.09%

TABLE 17. Percentage for runtime (s), average node power (W), and energy (J) for HDC on Kaveri

Configuration	Runtime	Node Power	Energy
Original (baseline)	1380.92	75.14	103762.33
ckp(1,2,3,4)	7.15%	-0.37%	6.75%
ckp(1,3,5,7)	6.30%	-0.07%	6.23%
ckp(2,3,4,5)	5.64%	0.15%	5.79%
ckp(2,4,6,8)	4.31%	-0.17%	4.13%
ckp(3,4,5,6)	5.08%	-0.20%	4.87%
ckp(3,5,7,11)	5.10%	0.12%	5.22%
ckp(3,5,7.9)	5.16%	-0.20%	4.95%
ckp(4,5,6,7)	5.43%	0.32%	5.77%
ckp(6,7,8,9)	3.93%	-0.27%	3.65%
ckp(8,9,10,11)	3.15%	0.45%	3.62%

In Figure 3, we compare the power consumption (for node, CPU, memory, and disk) over time for the application with and without ckp(3, 5, 7, 11) and bit-flip failure injection, The largest power increase results from the CPU, and checkpointing causes less memory power over time as well.

Let us look at how different bit positions affect the power and energy on Haswell. Table 16 shows the percentage for the runtime, average power, and energy at five different bit positions for the checkpointing ckp(3,5,7,11) with one bit-flip error injection. The percentage range is very small, from 0.18% to 1.14% in runtime, from -0.49% to -0.26% in node power, and from -0.09% to 0.65% in energy. Overall, the different bit positions for a bit-flip error injection have little impact on power and energy on Haswell.

4.1.4. AMD Kaveri. Table 17 shows the percentage for runtime, average node power, and energy per node for 10 checkpointing configurations. Compared with the original, FTI causes an increase in time by up to 7.15%, but the power remains almost flat. The increase in runtime mainly results in an energy increase by up to 6.75%. From the table, we see that the checkpointing ckp(8,9,10,11) results in the lowest energy consumption among the checkpointing

TABLE 18. Comparison of runtime (s), average power (W), and energy (J) for HDC on Kaveri

Configuration	Runtime	Node Power	CPU Power	Memory Power	Disk Power	Energy
Original (baseline)	1380.92	75.14	44.54	13.01	0.92	103762.33
ckp(1,2,3,4)	7.15%	-0.37%	-0.47%	-0.38%	1.09%	6.75%
ckp(8,9,10,11)	3.15%	0.45%	0.36%	0.0%	0.0%	3.62%

TABLE 19. Percentage for runtime (s), average node power (W), and energy (J) for various checkpointing frequencies with 1 error injection for HDC on Kaveri

Configuration	Runtime	Node Power	Energy
ckp(3,5,7,11)/e1 (baseline)	1458.23	74.84	109133.93
ckp(1,2,3,4)/e1	2.08%	0.11%	2.19%
ckp(2,3,4,5)/e1	-0.004%	-0.11%	-0.11%
ckp(2,4,6,8)/e1	-1.66%	0.32%	-1.34%
ckp(4,5,6,7)/e1	-0.69%	-0.20%	-0.89%
ckp(6,7,8,9)/e1	-0.79%	0.29%	-0.50%
ckp(8,9,10,11)/e1	-1.52%	0.21%	-1.31%

configurations because of the fewest number of checkpoints. Among these checkpointing configurations, the percentage range is small, from 3.15% to 7.15% in runtime, from -0.37% to 0.45% in node power and from 3.62% to 6.75% in energy. So FTI has little impact on the power consumption because of the small overhead for simultaneously writing the checkpointing file of 32 MB per MPI process (only a total of 4 processes) on a node (4 cores) on Kaveri.

As shown in Table 17, the ckp(8,9,10,11) results in the minimum energy consumption and the ckp(1,2,3,4) results in the maximum energy consumption. Further, Table 18 shows the runtime, average power, and energy with and without resilience. As expected, the runtime increases with the complexity of the resilience strategy, however, the average power consumptions for different system components: node, CPU, memory, and disk remain almost flat with respect to the various resilience strategies. Therefore, the increase in runtime results mainly in an increase in energy.

In Table 19, the ckp(2,4,6,8)/e1 results in the smallest runtime but not smallest node power; this configuration also results in the lowest energy consumption. Among these checkpointing configurations, the percentage range is also very small, from -1.66% to 2.08% in runtime, from -0.20% to 0.32% in node power, and from -1.34% to 2.19% in energy. Overall, the bit-flip failure injection for FTI has little impact on the runtime and power on Kaveri.

Further, when we compare the power consumption over time for the application with and without ckp(3, 5, 7, 11) and a bit-flip failure injection in Figure 4, we observe that all power consumptions remain almost flat over time.

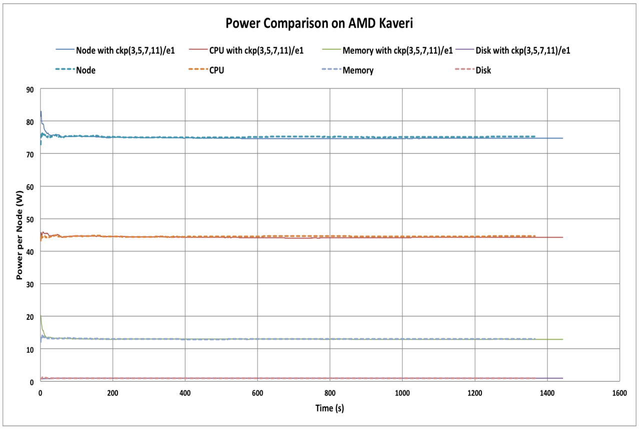

Figure 4. Power over time for HDC on Kaveri

TABLE 20. Comparison of runtime (s), average node power (W), and energy (J) for bit flip fault injection with different bit positions for HDC on Kaveri

Configuration	Runtime	Node Power	Energy
ckp(3,5,7,11)/e1 (baseline)	1458.23	74.84	109133.93
ckp(3,5,7,11)/e8	-0.46%	0.52%	0.06%
ckp(3,5,7,11)/e16	-0.18%	-0.25%	-0.43%
ckp(3,5,7,11)/e24	-1.28%	-0.20%	-1.48%
ckp(3,5,7,11)/e31	-0.52%	-0.01%	-0.53%

Let us look at how different bit positions affect the power and energy on Kaveri. Table 20 shows the percentage for the runtime, average power and energy at five different bit positions for the checkpointing ckp(3,5,7,11) with one bit-flip error injection. We observe that the percentage range is very small, from -1.28% to -0.18% in runtime, from -0.25% to 0.52% in node power, and from -1.48% to 0.06% in energy. Overall, the different bit positions for a bit-flip error injection have little impact on power and energy.

4.2. Case II: MPI Memory Benchmark STREAM

In this section, we use the memory benchmark STREAM (MPI version) [21] to develop the FTI version of STREAM to further address the power and energy effects of FTI with larger checkpointing file sizes. For the benchmark, we set the number of runs for each kernel at 5000, and we adjust the STREAM_ARRAY_SIZE to four times the size of the last level cache (256M for Cray XC40, 128M for BG/Q, 160M for Haswell, and 8M for Kaveri).

4.2.1. Cray XC40. Table 21 shows the percentage for runtime, average node power, and energy per node for 8 checkpointing configurations using the original as the baseline on Cray XC40. Although we set the STREAM_ARRAY_SIZE to 256M on Cray XC40, we observe that, compared with the original application, FTI causes an increase in runtime

TABLE 21. PERCENTAGE FOR RUNTIME (S), AVERAGE NODE POWER (W), AND ENERGY (J) FOR STREAM ON CRAY XC40

Configuration	Runtime	Node Power	Energy
Original (baseline)	323	295.83	95553.09
ckp(1,2,3,4)	5.11%	-1.50%	3.53%
ckp(1,3,5,7)	2.17%	-2.13%	-0.01%
ckp(2,3,4,5)	4.18%	-1.85%	2.26%
ckp(2,4,6,8)	0.46%	0.04%	0.51%
ckp(3,4,5,6)	1.39%	-0.86%	0.52%
ckp(3,5,7,11)	0.62%	-0.66%	-0.05%
ckp(3,5,7.9)	0.77%	-0.49%	0.28%
ckp(4,5,6,7)	0.31%	-0.74%	-0.44%

TABLE 22. COMPARISON OF RUNTIME (S), AVERAGE POWER (W), AND ENERGY (J) FOR STREAM ON CRAY XC40

Configuration	Runtime	Node Power	CPU Power	Memory Power	Energy
Original (baseline)	323	295.83	205.81	13.33	95553.09
ckp(1,2,3,4)	5.11%	-1.50%	-2.33%	4.43%	3.53%
ckp(4,5,6,7)	0.31%	-0.74%	0.26%	-4.13%	-0.44%

by up to 5.11%, a decrease in the node power by up to 2.13%, and an increase in energy by up to 3.53%. Because of user access to a Lustre PFS with 210 GB/s bandwidth on the Cray XC40, we find that, during the checkpoints, writing the 64 checkpointing files with a total size of 6 GB (3x8x256M) simultaneously on a node (64 cores) results in the small FTI overhead.

From the table, we observe that the checkpointing ckp(4,5,6,7) results in the lowest energy consumption among the checkpointing configurations because of the fewest number of checkpoints, the smallest runtime and relatively small node power. Among these checkpointing configurations, the percentage range is small, from 0.31% to 5.11% in runtime, from -2.13% to 0.04% in node power, and from -0.44% to 3.53% in energy. Compared with Table 5, the percentage ranges in runtime and energy slightly increase among the checkpointing configurations although the checkpointing file size is increased from 32 MB for HDC to 96 MB for STREAM.

As shown in Table 21, the ckp(4,5,6,7) results in the minimum energy consumption and the ckp(1,2,3,4) results in the maximum energy consumption. Further, Table 22 shows the runtime, average power, and energy with and without resilience. For both configurations, the node power decreases. For the ckp(1,2,3,4), the CPU power decreases by 2.33% and the memory power increases by 4.43%. However, for the ckp(4,5,6,7), the CPU power increases by 0.26%, and the memory power decreases by 4.13%. This results in the energy decrease by 0.44% for the ckp(4,5,6,7). Overall, the energy difference for both configurations is 3.97%, and it is still very small.

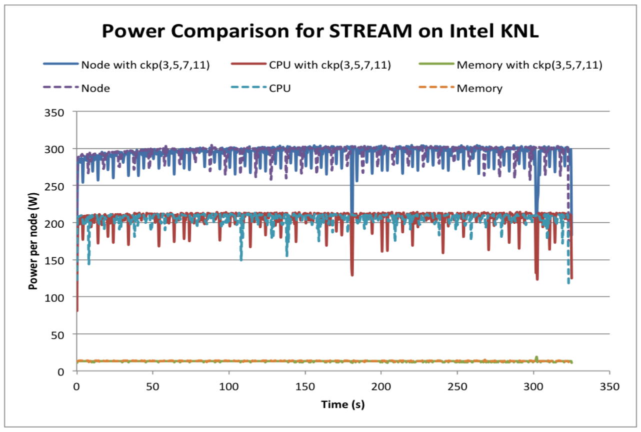

Figure 5. Power over time for STREAM on Cray XC40

TABLE 23. PERCENTAGE FOR RUNTIME (S), AVERAGE NODE POWER (W), AND ENERGY (J) FOR STREAM ON BG/Q

Configuration	Runtime	Node Power	Energy
Original (baseline)	216	53.56	11568.96
ckp(1,2,3,4)	24.07%	-0.39%	23.59%
ckp(1,3,5,7)	17.59%	-0.19%	17.37%
ckp(2,3,4,5)	18.52%	5.28%	24.78%
ckp(2,4,6,8)	5.09%	0.84%	5.98%
ckp(3,4,5,6)	6.02%	7.00%	13.44%
ckp(3,5,7,11)	6.02%	5.56%	11.92%
ckp(3,5,7.9)	6.94%	6.89%	14.31%
ckp(4,5,6,7)	0.46%	7.04%	7.53%

Figure 5 shows that FTI causes the power decrease for node and CPU on Cray XC40. The runtime is 323 s for the original application. For the application with checkpointing ckp(3,5,7,11), the first checkpoint happened at 3 minutes. During the checkpointing, the node power drop is caused mainly by the CPU power drop because of few CPU activities during writing 64 checkpointing files simultaneously. Both Figure 5 and Figure 1 have the similar power consumption trend: the checkpointing results in a power decrease in node and CPU on Cray XC40.

4.2.2. IBM BG/Q. Table 23 shows the percentage for runtime, average node power, and energy per node for eight checkpointing configurations using the original as the baseline on BG/Q. Although we set the STREAM_ARRAY_SIZE to 128M on BG/Q, we observe that, compared with the original application, FTI causes an increase in runtime by up to 24.07%, in the node power by up to 7.04%, and in energy by up to 24.78%. Because of user

TABLE 24. Comparison of runtime (s), average power (W), and energy (J) for STREAM on BG/Q

Configuration	Runtime	Node Power	CPU Power	Memory Power	Network Power	Energy
Original (baseline)	216	53.56	32.51	7.76	1.86	11568.96
ckp(2,3,4,5)	18.52%	5.28%	6.43%	9.79%	-0.54%	24.78%
ckp(2,4,6,8)	5.09%	0.84%	-0.95%	12.11%	1.61%	5.98%

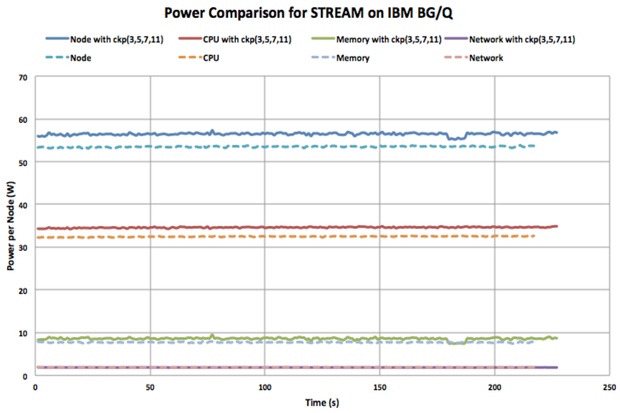

Figure 6. Power over time for STREAM on BG/Q

access to a GPFS file system with 240 GB/s bandwidth on the BG/Q, we find that, during the checkpoints, writing the 16 checkpointing files with a total size of 3 GB (3x8x128M) simultaneously on a node (16 cores) still results in the most FTI overhead, which is the main cause of the large runtime increase. This indicates that the high bandwidth GPFS file system was not fully utilized: the FTI overhead was still high. We believe that MPI-IO can be applied to FTI to fully utilize the GPSF system to reduce the overhead.

From the table, we observe that the checkpointing ckp(2,4,6,8) results in the lowest energy consumption among the checkpointing configurations although it does not have the smallest runtime and node power. Among these checkpointing configurations, the percentage range is large, from 0.46% to 24.07% in runtime, from -0.39% to 7.04% in node power, and from 5.98% to 24.78% in energy. Compared with Table 9, the percentage ranges in runtime and energy significantly increase among the checkpointing configurations because of the increased checkpointing file size, from 32 MB for HDC to 192 MB for STREAM.

As shown in Table 23, the ckp(2,4,6,8) results in the minimum energy consumption and the ckp(2,3,4,5) results in the maximum energy consumption. Further, Table 24 shows the runtime, average power, and energy with and without resilience. The memory and network power consumptions for the ckp(2,4,6,8) is larger than that for the ckp(2,3,4,5), however, its node and CPU power consumptions is smaller than that for the ckp(2,3,4,5). The large runtime increase mainly leads to the large energy increase.

Figure 6 shows that FTI causes the power increase for CPU, memory, and disk. The runtime is 229 s for the original application. For the checkpointing ckp(3,5,7,11), the first checkpoint happened at 3 minutes. During the checkpointing, the node power drop is caused mainly by the memory power drop because of few memory activities during writing 16 large checkpointing files simultaneously. Both Figure 6 and Figure 2 have the similar power consumption trend: the checkpointing results in a power increase in node, CPU, memory, and network on BG/Q, although the heat distribution application is compute intensive and the STREAM is memory intensive.

TABLE 25. Percentage for runtime (s), average node power (W), and energy (J) for STREAM on Haswell

Configuration	Runtime	Node Power	Energy
Original (baseline)	800	311.26	249008
ckp(1,2,3,4)	129.00%	-29.56%	61.30%
ckp(1,3,5,7)	131.00%	-26.01%	70.91%
ckp(2,3,4,5)	102.25%	-21.90%	57.95%
ckp(2,4,6,8)	63.88%	-16.06%	37.56%
ckp(3,4,5,6)	82.50%	-18.58%	48.59%
ckp(3,5,7,11)	74.50%	-15.87%	46.81%
ckp(3,5,7.9)	69.75%	-14.69%	44.82%
ckp(4,5,6,7)	81.25%	-16.77%	50.85%

4.2.3. Intel Haswell. Table 25 shows the percentage for runtime, average node power, and energy per node for eight checkpointing configurations using the original as the baseline on Haswell. Compared with the original, FTI causes not only an increase in runtime by up to 131% but also a node power decrease by up to 29.56% and an energy increase by up to 70.91%.The reason is that setting the STREAM_ARRAY_SIZE to four times the size of the last level cache (160M for Haswell) resulted in fully utilizing the whole node so that the power consumptions in node, CPU, and memory for the original are the largest (as we also observed in Figure 7). The checkpointing results in less power consumption in node, CPU, and memory because of the frequent checkpointing interrupts and large FTI overhead. The more frequent the checkpointing is, the less the average power consumption is. Because of the general NFS file system used on Haswell, during the checkpoints we find that writing 32 checkpointing files with a total size of 3.75 GB (3x8x160M) simultaneously on a node (32 cores) results in the most FTI overhead. The large FTI overhead significantly impacts the average node power because of the few CPU and memory activities during the I/O writing for checkpointing files. This indicates that FTI needs high bandwidth parallel I/O writing to significantly reduce the overhead. MPI-IO can be applied to FTI to accomplish this reduction.

From the table, we observe that the checkpointing

TABLE 26. Comparison of runtime (s), average power (W), and energy (J) for STREAM on Haswell

Configuration	Runtime	Node Power	CPU Power	Memory Power	Disk Power	Energy
Original (baseline)	800	311.26	230.34	64.83	2.32	249008.00
ckp(1,3,5,7)	131.00%	-26.01%	-25.84%	-33.18%	0.43%	70.91%
ckp(2,4,6,8)	63.88%	-16.06%	-15.80%	-21.07%	0.43%	37.56%

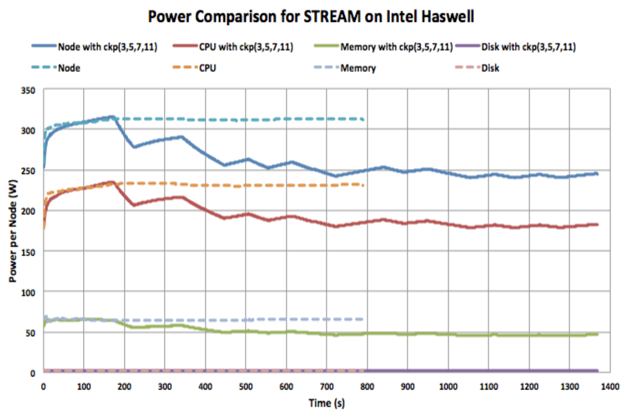

Figure 7. Power over time for STREAM on Haswell

ckp(2,4,6,8) results in the lowest energy consumption among the checkpointing configurations. Notice that among these checkpointing configurations, the percentage range is very large, from 63.88% to 131% in runtime, from -29.56% to -16.77% in node power, and from 37.56% to 70.91% in energy. Compared with Table 13, the percentage ranges in runtime and energy significantly increase among the checkpointing configurations because of the increased checkpointing file size, from 32 MB for HDC to 120 MB for STREAM on Haswell.

As shown in Table 25, the ckp(2,4,6,8) results in the minimum energy consumption and the ckp(1,3,5,7) results in the maximum energy consumption. Further, Table 26 shows the runtime, average power, and energy with and without resilience. The ckp(2,4,6,8) has smaller power decrease in node, CPU and memory than the ckp(1,3,5,7). This may result in the smallest runtime increase for the ckp(2,4,6,8).

Compared Figure 7 with Figure 3, we see that, for STREAM, because of setting the array size large enough to fully utilize the whole node, the checkpointing causes a decrease in node power, CPU power, and memory power, which are lower than for the original one. For HDC, the checkpointing causes an increase in node power and CPU power, which are higher than for the original, while the memory power is lower than for the original.

4.2.4. AMD Kaveri. Table 27 shows the percentage for runtime, average power, and energy per node for eight checkpointing configurations using the original as the baseline on Kaveri. Compared with the original, FTI causes an increase in time by up to 10.08%, but the node power consumption remains almost flat, and the energy increases by up to 9.6%. Because of the general NFS file system used on Kaveri, we find that, during the checkpoints, writing out our checkpointing files with a total size of 192 MB (3x8x8M) simultaneously on a node (4 cores) results in the most FTI overhead.

TABLE 27. Percentage for runtime (s), average node power (W), and energy (J) for STREAM on Kaveri

Configuration	Runtime	Node Power	Energy
Original (baseline)	486	78.17	37990.62
ckp(1,2,3,4)	10.08%	-0.43%	9.60%
ckp(1,3,5,7)	9.05%	-2.12%	6.74%
ckp(2,3,4,5)	8.44%	-1.62%	6.67%
ckp(2,4,6,8)	3.29%	-0.56%	2.71%
ckp(3,4,5,6)	6.79%	-1.19%	5.52%
ckp(3,5,7,11)	4.73%	-0.81%	3.89%
ckp(3,5,7.9)	4.73%	-0.96%	3.73%
ckp(4,5,6,7)	6.58%	-1.85%	4.61%

TABLE 28. Comparison of runtime (s), average power (W), and energy (J) for STREAM on Kaveri

Configuration	Runtime	Node Power	CPU Power	Memory Power	Disk Power	Energy
Original (baseline)	486	78.17	44.03	16.45	0.85	37990.62
ckp(1,2,3,4)	10.08%	-0.43%	0.27%	-2.67%	-3.53%	9.60%
ckp(2,4,6,8)	3.29%	-0.56%	-0.82%	-0.36%	-1.18%	2.71%

From the table, we observe that the checkpointing ckp(2,4,6,8) results in the lowest energy consumption among the checkpointing configurations because of the smallest runtime and relatively small node power. Among these checkpointing configurations, the percentage range is from 3.29% to 10.08% in runtime, from -2.12% to -0.43% in node power and from 2.71% to 9.60% in energy. Compared with Table 17, the percentage ranges in runtime and energy increase among the checkpointing configurations because of the increased checkpointing file size, from 32 MB for HDC to 48 MB for STREAM on Kaveri.

As shown in Table 27, the ckp(2,4,6,8) results in the minimum energy consumption and the ckp(1,2,3,4) results in the maximum energy consumption. Further, Table 28 shows the runtime, average power, and energy with and without resilience. The ckp(2,4,6,8) has smaller power decrease in memory and disk than the ckp(1,2,3,4), but has slightly larger power decrease in node and CPU. The energy difference between both configurations is 6.89%.

Figure 8 shows that FTI has little impact on power; and the power consumption remains flat for the node, CPU,

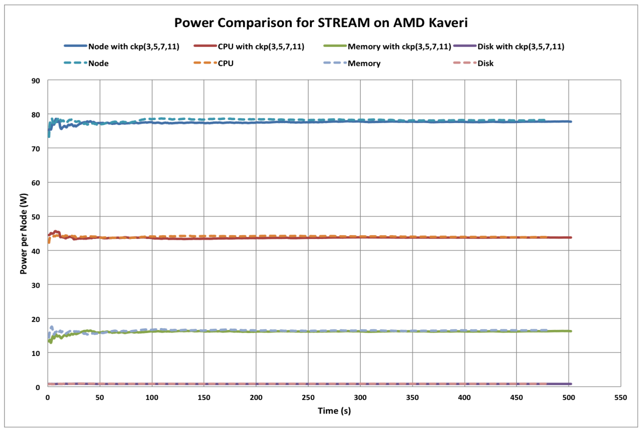

Figure 8. Power over time for STREAM on Kaveri

TABLE 29. SUMMARY OF THE MAXIMUM ENERGY PERCENTAGES

Application	Architecture	Configuration	Runtime	Node Power	Energy
HDC	Cray XC40	ckp(1,2,3,4)	1.67%	-0.59%	1.08%
	IBM BG/Q	ckp(1,2,3,4)	9.29%	7.52%	17.51%
	Intel Haswell	ckp(1,2,3,4)	26.12%	1.74%	28.31%
	AMD Kaveri	ckp(1,2,3,4)	7.15%	-0.37%	6.75%
STREAM	Cray XC40	ckp(1,2,3,4)	5.11%	-1.50%	3.53%
	IBM BG/Q	ckp(2,3,4,5)	18.52%	5.28%	24.78%
	Intel Haswell	ckp(1,3,5,7)	131.00%	-26.01%	70.91%
	AMD Kaveri	ckp(1,2,3,4)	10.08%	-0.43%	9.60%

memory, and disk. Both Figure 8 and Figure 4 have a similar power consumption trend, namely, the power consumption remains flat on Kaveri although the heat distribution application is compute intensive and the STREAM is memory intensive.

TABLE 30. SUMMARY OF THE MINIMUM ENERGY PERCENTAGES

Application	Architecture	Configuration	Runtime	Node Power	Energy
HDC	Cray XC40	ckp(4,5,6,7)	1.12%	-1.17%	-0.06%
	IBM BG/Q	ckp(8,9,10,11)	3.32%	2.21%	6.82%
	Intel Haswell	ckp(6,7,6,9)	10.03%	7.09%	17.83 %
	AMD Kaveri	ckp(8,9,10,11)	3.15%	0.45%	3.62%
STREAM	Cray XC40	ckp(4,5,6,7)	0.31%	-0.74%	-0.44%
	IBM BG/Q	ckp(2,4,6,8)	5.09%	0.84%	5.98%
	Intel Haswell	ckp(2,4,6,8)	63.88%	-16.06%	37.56%
	AMD Kaveri	ckp(2,4,6,8)	3.29%	-0.56%	2.71%

5. Summary

In this paper we presented an empirical study to evaluate runtime and power effects of multilevel checkpointing MPI applications using FTI. Tables 29 and 30 summarize the maximum and minimum energy percentages for various checkpointing configurations from our experiments for two FTI applications HDC and STREAM. Overall, the difference between maximum and minimum energy percentages is less than 4% on Cray XC40, less than 19% on IBM BG/Q, less than 34% on Intel Haswell, and less than 7% on AMD Kaveri. In most cases, the most frequent checkpointing ckp(1,2,3,4) results in the maximum energy increase, and for the cases other than ckp(1,2,3,4) the energy difference is small.

Our experimental results for the applications HDC and STREAM on Cray XC40, IBM BG/Q, Intel Haswell and AMD Kaveri indicate the following findings. First, the runtime and power consumption for both applications varied across the different architectures. In general, Both Cray XC40 and AMD Kaveri with dynamic power management exhibited the smallest impact, whereas Intel Haswell without dynamic power management manifested the largest impact. In most cases, we observe a proportional increase in application runtime and energy across these systems. Further, bit-flip fault injection had little impact on application runtime and power consumption, resulting in little impact on energy consumption across the systems. Multilevel checkpointing could benefit from high speed I/O writing to reduce the overhead caused by writing temporary checkpoint files. Shared file systems may affect the performance of multiple multilevel checkpointing applications executed simultaneously. The experimental work involved four different parallel systems for which the underlying architectures have different power requirements. Therefore, we do not compare the actual values of energy consumptions for the different systems but instead focus on the trends in term of percentage difference with respect to the baseline without fault tolerance.

Overall, this study provides us a good start to understand the tradeoffs among runtime, power, and resilience on different architectures. However, we will need to study the four architectures in depth to further analyze our observations. Because the four checkpointing levels for FTI correspond to coping with the four types of failures: software failure, single node failure, multiple node failure, and all other failures the lower levels cannot take care of, respectively, when setting the checkpointing frequency for each level to estimate the application runtime, it is important to consider how often the failure at the level occurs, how long it takes to execute the original application, the checkpointing file sizes and the file system I/O bandwidth. We plan to use performance counters to model tradeoffs among runtime, power, energy and resilience for various application-system configurations.

Acknowledgments

This work was supported in part by NSF grants CCF-1619236, CNS-0911023 and DMS-1317131, and in part supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under contract DE-AC02-06CH11357. We acknowledge L. Bautista-Gomez from the Barcelona Supercomputing Center for support of the FTI library, D. DeBonis from Sandia National Laboratories for support of PowerInsight, S. Wallace from Illinois Institute of Technology for providing the latest MonEQ, and Argonne Leadership Computing Facility for use of Cray XC40 Theta and BlueGene/Q Mira under the DOE INCITE project PEACES.

References

[1] G. Aupy, A. Benoit, T. Herault, Y. Robert, and J. Dongarra, Optimal checkpointing period: Time vs Energy, the 4th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems, 2013.

[2] P. Balaprakash, L. Bautista-Gomez, M. Bouguerra, S. M. Wild, F. Cappello and P. D. Hovland, Analysis of the tradeoffs between energy and run Time for multilevel checkpointing, the 5th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems, 2014.

[3] L. Bautista-Gomez, D. Komatitsch, N. Maruyama, S. Tsuboi, F. Cappello, and S. Matsuoka, FTI: High performance fault tolerance interface for hybrid systems, IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC2011), 2011.

[4] R. Bertran, Y. Sugawara, H. Jacobson, A. Buyuktosunoglu, and P. Bose, Application-level power and performance characterization and optimization on IBM Blue Gene/Q systems, IBM Journal of Research and Development, 57(1), 2013.

[5] A. Bouteiller, T. Herault, G. Bosilca, P. Du, and J. Dongarra, Algorithm-based fault tolerance for dense matrix factorizations, multiple failures and accuracy. ACM Trans. Parallel Computing Vol. 1, No. 2, January 2015.

[6] A. Branover, D. Foley and M. Steinman, AMD Fusion APU: Llano, IEEE Micro, vol. 32, no. 2, pp. 28-37, March-April 2012.

[7] Z. Chen and J. Dongarra. Algorithm-based checkpoint-free fault tolerance for parallel matrix computations on volatile resources, IEEE International Parallel and Distributed Processing Symposium (IPDPS'06), 2006.

[8] S. Chunduri, K. Harms, S. Parker, V. Morozov, S. Oshin, N. Cherukuri, and K. Kumaran, Run-to-Run Variability on Xeon Phi Based Cray XC Systems, IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC17), 2017.

[9] Cray, Monitoring and Managing Power Consumption on the Cray XC System, S-0043-7204.

[10] S. Di, M.S. Bouguerra, L. Bautista-Gomez, and F. Cappello. Optimization of multi-level checkpoint model for large-scale HPC applications, IEEE International Parallel and Distributed Processing Symposium (IPDPS'14), 2014.

[11] Cooper, Advanced Systems Technology Test Beds, Sandia National Laboratories, http://www.sandia.gov/asc/computational_systems/HAAPS.html.

[12] M. el Mehdi Diouri, O. Gluck, L. Lefevre, and F. Cappello, Energy considerations in checkpointing and fault tolerance protocols. 2nd International Workshop on Fault Tolerance for HPC at eXtreme Scale (FTXS), 2012.

[13] M. el Mehdi Diouri, O. Gluck, L. Lefevre, and F. Cappello. ECOFIT: A framework to estimate energy consumption of fault tolerance protocols for HPC applications. In 13th IEEE/ACM International Symp. Cluster, Cloud and Grid Computing, 2013.

[14] J. Elliott, K. Kharbas, D. Fiala, F. Mueller, K. Ferreira, and C. Engelmann. Combining partial redundancy and checkpointing for HPC, IEEE International Conference on Distributed Computing Systems (ICDCS'12), 2012.

[15] K. Ferreira, R. Riesen, P. Bridges, D. Arnold, J. Stearley, J. Laros, R. Oldfield, K. Pedretti, and R. Brightwell. Evaluating the viability of process replication reliability for exascale systems, IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC2011), 2011.

[16] D. Fiala, F. Mueller, C. Engelmann, R. Riesen, K. Ferreira, and R. Brightwell, Detection and correction of silent data corruption for large-scale high-performance computing, IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC2012), 2012.

[17] FTI: Fault tolerance interface, leobago.github.io/fti/, 2016.

[18] R. Ge, X. Feng, S. Song, et al., PowerPack: Energy profiling and analysis of high-performance systems and applications, IEEE Trans. on Parallel & Distributed Systems 21(5), 2010, pp. 658-671.

[19] A. Hoisie, L. Carrington, J. Hiller, D. Kerbyson, M. Schulz, D. Shaffer, A. Srivastava, J. Vetter, B. Ward, N. Wheeler, and S. Yalamachili, Report of Workshop on Modeling and Simulation of Systems and Applications, University of Washington, Seattle, 2014.

[20] K. Huang and J. Abraham. Algorithm-based fault tolerance for matrix operations. IEEE Transactions on Computers, 33(6):518-528, 1984.

[21] Intel MPI benchmarks, https://software.intel.com/en-us/imb-user-guide, Feb. 22, 2017.

[22] J. H. Laros III, P. Pokorny and D. DeBonis, PowerInsight — A commodity power measurement capability, 2013 International Green Computing Conference, 2013.

[23] M. Lim, A. Porterfield, and R. Fowler, SoftPower: Fine-grain power estimations using performance counters, the 19th Int'l Symp on High Performance Distributed Computing (HPDC'10), 2010.

[24] I. Marincic, V. Vishwanath, and H. Hoffmann, PoLiMEr: An Energy Monitoring and Power Limiting Interface for HPC Applications, SC2017 Workshop on Energy Efficient Supercomputing (E2SC'17). Nov. 13, 2017.

[25] S Martin, D Rush, M Kappel, M Sandstedt, and J Williams, Cray XC40 Power Monitoring and Control for Knights Landing, Proceedings of the 2016 Cray User Group (CUG) Conference, 2016.

[26] E. Meneses, O. Sarood and L.V. Kale, Assessing energy efficiency of fault tolerance protocols for HPC systems, the 24th IEEE International Symposium on Computer Architecture and High Performance Computing, 2012.

[27] MIRA, IBM BlueGene/Q system, https://www.alcf.anl.gov/mira.

[28] A. Moody, G. Bronevetsky, K. Mohror, and B. R. de Supinski. Detailed modeling, design, and evaluation of a scalable multi-level checkpointing system, IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC2010), 2010.

[29] A. Nagarajan, F. Mueller, C. Engelmann, and S. Scott, Proactive fault tolerance for HPC with Xen virtualization, ACM International Conference on Supercomputing (ICS2006), June 2006.

[30] NVIDIA, NVML API Reference Manual, 2012.

[31] J. Plank, K. Li, and M. Puening. Diskless checkpointing. IEEE Trans. on Parallel and Distributed Systems, 9(10), 1998.

[32] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, Power-management architecture of the Intel microarchitecture code-named Sandy Bridge, IEEE Micro, 32(2), 2012.

[33] Shepard, Advanced Systems Technology Test Beds, Sandia National Laboratories, http://www.sandia.gov/asc/computational _systems/HAAPS.html.

[34] Scalable checkpoint/restart project, https://computation.llnl.gov/ project/scr/.

[35] STREAM benchmarks, http://www.cs.virginia.edu/stream/.

[36] Theta, Cray XC40 system, https://www.alcf.anl.gov/theta.

[37] S. Wallace, V. Vishwanath, S. Coghlan, J. Tramm, Z. Lan, and M. E. Papka, Application power profiling on Blue Gene/Q, 2013 IEEE Conference on Cluster Computing, 2013.

[38] X. Wu, V. Taylor, C. Lively, H. Chang, B. Li, K. Cameron, D. Terpstra and S. Moore, MuMMI: Multiple Metrics Modeling Infrastructure (Book chapter), Tools for High Performance Computing, Springer, 2014.

[39] X. Wu, V. Taylor, J. Cook, and P. Mucci, Using performance-power modeling to improve energy efficiency of HPC applications, IEEE Computer, Vol. 49, No. 10, pp. 20-29, Oct. 2016.