

Power Modeling and Thermal Management Techniques for Manycores

Rajib Nath

Computer Science and Engineering
University of California, San Diego

Douglas Carmean

Extreme Technology Lab
Intel Lab, Oregon

Tajana Rosing

Computer Science and Engineering
University of California, San Diego

Abstract—The rising number of cores in manycore architectures, along with technology scaling, results in high power densities and thermal issues on the die. To explore innovative thermal management techniques in such processors, we need an accurate online estimate of the power consumption. In this paper, we present the first ever power model for Intel many integrated core processors, which we use to show the benefit of a novel manycore specific thermal management technique called workload intermixing. Our proposed model leverages performance monitoring events and accounts for operating voltage and clock frequency. We validate our model for Intel Knights Ferry (*KNF*) design and show that we have an average of 4.73% prediction error vs. measurements. We provide the breakdown of total power into three main components: compute, memory, and interconnect and use it as an input for thermal management. Our simulation results show that our proposed intermixing of workloads in *KNF* architecture can reduce the total number of thermal emergency situations by 58% with energy savings of 14% on average.

I. INTRODUCTION

Manycore chips are becoming a baseline for future high performance computing solutions. However, the additional performance gained from these high end computing resources comes with significant increase in power consumption. High power dissipation causes thermal hotspots that may have a significant effect on reliability, performance, and leakage power. As the technology node scales down, it becomes more challenging to dissipate the heat using existing cooling mechanisms without sacrificing performance. Since the target applications of these manycore chips exhibit a lot of parallelism, a highly compute intensive application occupying all the cores can increase the temperature very quickly, causing performance degradation due to the state of the art dynamic thermal management (*DTM*) [14] techniques like throttling or dynamic voltage frequency scaling (*DVFS*). In order to explore new thermal management techniques, we need to understand and develop models of the power consumption of these manycore processors.

A number of power models for general purpose CPUs have been proposed. At first, Tiwari, et al. [1] introduced a processor power model based on instruction level power analysis. A survey on power model demonstrated that models based upon OS utilization metrics and performance monitoring unit (PMU) counters are generally sufficiently accurate [2]. However, there are very few publications on power estimation of manycore or similar architectures. Vendors generally do

not release power models for their chips. The lack of accurate power model is the main hindrance towards pursuing academic research on innovative thermal management techniques of manycore architecture.

Sheaffer, et al. [3] extended existing well studied CPU power models and derived a GPU power model based upon hypothetical GPU architectural simulation. The work in [4] introduced an architectural power modeling framework for GPUs which is composed of analytical, empirical and area based components. Even though it is an useful architectural exploration tool, it requires circuit level knowledge about the chip and does not validate the power model at the chip level to guarantee the accuracy of power estimation. Ma, et al. [5] have statistically analyzed and modeled the power consumption of a mainstream GPU (NVIDIA *GeForce880GT*). They exploited the relation among power consumption characteristics, runtime performance, and dynamic workloads to model GPU power consumption. Their model shows significant amount of error, as large as 30%, in number of cases because their model ignores bus and memory activities. They also neglect the effect of temperature on the static power consumption. Hong, et al. [6] presented a power model for recent GPGPUs from NVIDIA. As GPGPUs do not have speculative execution, they used dynamic instruction rate as an activity factor in the power model. Such models are inaccurate for a manycore processor like Intel's Knights Ferry (*KNF*) [7], where the speculation is introduced through branch predictions.

Thermal management for general purpose processors has become an active research area in recent years. Core level thermal management techniques for *CPUs* have two classes: reactive and proactive. Reactive *DTM* techniques remove the excess heat by slowing down the computation aggressively through pipeline throttling, power gating, DVFS, etc [14]. Activity migration reschedules the computation across redundant cores to manage excess temperature [16]. In order to address the performance overhead and non uniform thermal distribution of reactive techniques, several proactive techniques have been proposed that leverages temperature predictors [17]. Sheaffer et al. [15] have explored various *CPU* specific thermal management techniques for *GPU*. However, all these techniques have performance overhead due to the underlying throttling. Despite all the work in *CPU* and *GPU* areas, thermal management techniques for modern manycore architectures have never been explored.

Our work has two key contributions. First, we propose a new *DTM* technique called workload intermixing, which exploits the thermal heterogeneity in the manycore workloads. Second, we develop an efficient and accurate power model for a many integrated core architecture to evaluate the proposed *DTM* technique. In our power model, the total chip dynamic power is expressed as a function of three main components: (a) compute, (b) memory, and (c) interconnect, which is necessary for accurate thermal simulation and management. The dynamic power of each component is based on the event counts from *PMU*. The contribution of each event to the total power consumption is calibrated using a set of micro benchmarks running on the instrumented KNF card. Our work also studies two key components of power, one of which has not been addressed by previous work. First, we provide a model for the temperature dependent part of the total static power. In addition, our model enables accurate estimates of how power consumption and performance change over various operating voltage and clock frequency settings that allows us to accurately simulate *DVFS* for our target architecture. Our experiments show that the peak power consumption of a given benchmark can vary significantly (e.g., 2X) for a small change in input data size. We verify our calibrated power model for KNF with a set of dynamic workloads for different input sizes and different data mapping policies. No previous power model has addressed these dimensions before. The average error of our power model is within 4.73% of the measured power. This then enables us to develop a thermal simulator for *KNF* chip to assess the efficacy of workload intermixing as an innovative thermal management technique in manycore chips. Our simulation results show that our proposed intermixing of workloads in manycore architecture like *KNF* can reduce the total number of thermal emergency situations by 58% on average comparing to existing *DTM* techniques with energy savings of 14% on average.

II. MIC CHARACTERIZATION

This section provides a brief overview of the KNF architecture followed by the measurement methodologies and the benchmarks that we have used in Section III to develop a power model for manycore architecture. We leverage the calibrated power model to evaluate our proposed *DTM* technique in Section IV.

A. Knights Ferry Architecture

KNF is an X86 based processor with 32 small cores, each capable of executing 4 simultaneous threads. Each core has a wide (512-bit) vector processor unit (VPU), 32KB L1 data cache, 32KB L1 instruction cache, and a private 256KB L2 data cache. As shown in Figure 1, cores, L2 cache and memory components are connected through a ring interconnect to enforce coherency. The KNF card has 2GB GDDR5 memory running at 3.6 GT/s. We fix the clock speed and operating voltage of the chip at 900 MHz and 1.15 volt respectively for all our experiments.

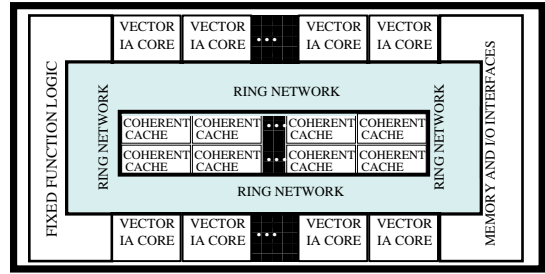


Fig. 1. KNF architecture

B. Measurements

We collect hardware event traces using Intel’s Sampling Enabling Product (SEP) tool [8] at default sampling frequency of SEP (1KHz). We instrument the KNF card and use National Instrument(NIS)’s Data Acquisition (DAQ) system for power measurements [9]. The power consumptions of two power supplies in the KNF card are added together to compute the total power consumption. We found that 1KHz sampling frequency is sufficient for all the benchmarks that we have used (relates to Nyquist frequency). We use system utility to read the on die register which stores the current temperature of the KNF chip. Temperature is measured every second since *KNF*’s thermal constant is on this order.

C. Workloads

Each of the benchmarks used in this study runs directly on the KNF card using 32 cores and 4 threads per core. A summary is given in Table I where *IC* indicates the iteration count, *L* is the execution time, and *F* indicates whether the benchmark has a flat or variable behavior in terms of power consumption. P_{Total} is the peak total power consumed by the KNF card. P_{Total} includes the chip’s dynamic power ($P_{Dynamic}$) and idle power of the KNF card (P_{Idle}), which is the total power consumption of the KNF card when no workload is running on the chip and when chip temperature (T_{Chip}) is at the steady state temperature (T_{CIdle}). Hence, board power, leakage power, and other sources of idle power are also included in P_{Total} . All the power numbers in this paper are normalized with respect to P_{Idle} at T_{CIdle} . We wait 9 minutes before taking any new measurements to allow the system to reach steady state temperature.

1) *Micro Benchmarks*: We have created a set of micro benchmarks (μ bench) to study the power consumption of different components of the KNF chip. All the μ benchs are compiled with Intel icc compiler (*icc -O0*) unless otherwise specified. The power consumed by μ benchs has a dynamic range of 28 watts. There are two classes of μ bench, those that focus on core and others which exercise memory and interconnect, *memint*.

Core μ benchmarks exercise different parts of the core, e.g., pipelines, register file, branch unit, integer unit, vector processor unit, etc. There are eight core benchmarks: (a) *Sleep*, (b) *SleepC*, (c) *Null*, (d) *Noop*, (e) *Count*, (f) *Vec*, (g) *Vec-Int*, and (h) *V-I-O3*.

Null is an empty while loop – $while(1)\{\}$. It is created to observe the minimum amount of power it takes to keep instruc-

tions running. *Null* has no data access from register or memory. On the other hand, μ bench *Count* is used to observe the change in power consumption due to data access from registers. *Count* is an infinite while loop which has an increment statement inside the loop body – `count=0; while(1){count++;}`. *Sleep* and *SleepC* highlight the idle power of the system, which is very much dependent on the chip’s temperature. *Sleep* goes through an infinite while loop with a sleep statement inside the loop body – `while(1){sleep(1);}`. *SleepC* is a slightly modified version of *Sleep*: `count=0; while(1){sleep(1); count++;}`. *Noop* is an infinite while loop where the loop body has a *noop* instruction– `while(1){noop;}`. We included *Noop* to observe the power consumption while the pipeline is full.

Type	Subtype	Name	$\frac{P_{Total}}{P_{Idle}}$ (ratio)	$T_{Chip}^{max} - T_{CIdle}$ (°C)	L (s)	F	IC	
μ bench	Core	Sleep	1.11	1	20	Y	1	
		SleepC	1.11	1	20	Y	1	
		Null	1.22	9	20	Y	1	
		Noop	1.20	8	20	Y	1	
		Count	1.19	8	20	Y	1	
		Vec	1.22	9	20	Y	1	
		Vec-Int	1.22	10	20	Y	1	
	MemInt	V-I-O3	1.27	13	20	Y	1	
		L1-Hit	1.15	6	20	Y	1	
		L2-Hit	1.17	8	20	Y	1	
bench	Application&Phase	L2-Miss	1.11	3	20	Y	1	
		mc	1.46	7	12	Y	100	
		nbody	1.41	4	5	N	1	
		libor	1.54	7	6	Y	1	
		stream	1.22	1	7	N	100	
		search	1.37	3	3	Y	1	
		bp	1.53	5	3	Y	1	
		fft	1.20	1	13	N	6000	
		scan	1.20	1	4	N	100	
		sgemm.f	1.54	7	14	N	10	
		Input	bs16384	1.35	5	15	N	4000
			bs16400	1.43	5	7	N	4000
	bs16512		1.63	6	4	N	4000	
	Data	bsa16384	1.31	6	40	N	4000	
		bsa16400	1.32	5	26	N	4000	
		bsa16512	1.54	6	7	N	4000	

TABLE I
INFOS ON μ BENCHS AND BENCHMARKS

In order to quantify the power contribution of VPUs, we developed three benchmarks, each with an infinite while loop. In case of *Vec*, the loop body is a sequence of vector instructions with no loop independent dependency among them. *Vec-Int*’s loop body contains a run of pairs containing one vector instruction and one integer instruction with similar dependency requirement as mentioned for *Vec*. *V-I-O3* is a modified version of *Vec-Int* and it is compiled with *icc -O3* option.

Memint μ benchs exercise memory hierarchy and interconnect. These μ benchs have an infinite while loop with stream of load instructions inside the body of the loop–`while(1){for(i=0;i<blocksize;i+=offset){load a vector register from location A+i;}}`. By changing the value of *blocksize* and *offset*, we get three different kinds of benchmarks: (i) *L1-Hit*, (ii) *L2-Hit*, and (iii) *L2-Miss*. In *L1-Hit*, all except the first load of a cache line are satisfied from L1 cache. In *L2-Hit*, all except the first load of a cache line are satisfied from L2 cache. All the loads in *L2-Miss* go to memory. A summary of all the

μ benchmarks is given in Table I

2) *Test Benchmarks*: To test the accuracy of our power model, we use ten benchmarks with a dynamic power consumption range of 100 watts. These benchmarks are selected from the parallel benchmark suits for multicore CPUs and GPGPUs. Lack of efficient implementation for Knight Ferry architecture has restricted us from expanding the test benchmark suit. We have selected Black-Scholes (*bs*) to verify our model across different input sizes: 16384, 16400, and 16512. In order to test the robustness of our power model across different data placement policies, we have used *bsa*, a different version of *bs*, with three different data mappings with adjustment sizes of 16384, 16400, and 16512 for the same input size, 16384. We use dense matrix multiplication *sgemm.f* as a very important kernel for high performance computing. Benchmark *nbody* is a simulation that predicts the motion of a group of celestial objects that interact with each other gravitationally. *Stream* is a simple synthetic benchmark program that measures sustainable memory bandwidth. The other selected benchmarks are: monte carlo simulation of option pricing (*mc*), discrete fourier transform (*2dff*), market model monte carlo (*libor*), image reconstruction process (*back-projection*), prefix sum (*scan*) and tree based search (*search*).

D. Thermal Simulator

We extend Hotspot [12] simulator with manycore thermal simulation based on the estimated floor plan and package characteristics of Intel *KNF*. The heatsink dimension and the case to ambient thermal resistance (K/W) are approximated as 0.07m and 0.25 respectively based on the data in [13]. We leverage our derived power model in Section III to generate the dynamic power traces for our benchmarks and to implement *DVFS*. We also account for thermally dependent leakage power based on our model. We include a baseline power of the *KNF* chip in the simulator. Each simulation starts from an initial temperature of 45°C with a warm-up period of 200s. We keep the *KNF* fan running at a fixed speed throughout the simulation to maintain a constant case temperature. There is one temperature sensor per core. The local ambient temperature within the computer and the critical temperature threshold on chip are set at 45°C and 90°C respectively. Since our goal is to design a new *DTM* technique that intermixes thermally heterogeneous workloads, we focus on cores only and do not do thermal management of memory or handle cooling.

III. POWER MODEL

This section provides our a power model for a many integrated core processor. To calibrate the model for *KNF*, we run μ benchs described in Section II on the instrumented *KNF*, measure the real power consumption using DAQ, collect events sample using SEP, and finally perform the fit of the power model.

A. Power Model Components

Total power consumption has two components: static power and dynamic power. Static power (P_{Static}) is a function of

chip layout, circuit technology, and operating temperature. On the other hand, dynamic power ($P_{Dynamic}$) is dependent on runtime switching activity of the circuit. As transistors become smaller and faster, the relative portion of static power in the total power consumption grows. P_{Static} is proportional to the power supply voltage (V_{CC}) and leakage current as follows:

$$P_{Static} = V_{CC} N K_{design} \hat{I}_{leakage} \quad (1)$$

Here, N is the number of transistors, K_{design} represents the characteristics of the device, and $\hat{I}_{leakage}$ is per device subthreshold leakage. K_{design} and $\hat{I}_{leakage}$ have a strong dependency on the temperature of the chip, T_{Chip} .

Hotleakage [10] provides a quadratic model between leakage power and T_{Chip} . Su, et al. [11] shows that this relation can be modeled as linear in normal operating temperature range. Thus we model static power at chip temperature T_{Chip} as shown in Equation 2. Here ΔT is the difference between current temperature (T_{Chip}) and the chip idle temperature (T_{CIdle}). ΔP is the rise in power consumption for 1°C rise in T_{Chip} .

$$P_{Static} = P_{Idle} + \Delta T \Delta P \quad (2)$$

$$\Delta T = T_{Chip} - T_{CIdle}$$

The dynamic power dissipated by a chip can be modeled as $\alpha CV^2 f$ where α is the activity factor in the chip, C is the capacitance being switched per clock cycle, V is the operating voltage of the chip, and f is the clock speed. It is a strong function of application's runtime activity. We divide the dynamic power of a manycore processor into three main components: compute, interconnect, and memory. We model the power of each component i as $\alpha_i C_i V^2 f$ where α_i is the activity factor of the component i , and C_i is the capacitance being switched per clock cycle in component i . Even though $\alpha_i C_i$ is unknown for each component, it can be modeled as a weighted sum of a few meaningful performance unit events as shown in equation 3.

$$\alpha_i C_i = \sum_{\substack{\text{for all event } e \\ \in MES_i}} w_{i,e} r_{i,e} \quad (3)$$

Here MES_i is the meaningful event set for component i , $r_{i,e}$ is the frequency of event e in event set MES_i , and $w_{i,e}$ is a weight for event e . We use this framework because PMU based power models have been shown to be very accurate. As defined here, the power model is general enough to be applied to any of the many core processor. In the following subsections, we define a small set of meaningful events specifically for KNF design and the respective weights, $w_{i,e}$, used by Equation 3.

B. Selecting MES for KNF

MES_{Compute}: Usually the dynamic power consumption of CPU is highly correlated to instructions executed per clock tick (IPC). The power is also influenced by the instruction mix and the presence of speculative execution (e.g. branch prediction). Moreover, in the presence of VPUs, the power consumption in applications can be greatly influenced by the percentage of vector instructions in the total number of instructions executed and the width of the VPU. Wider VPUs can lead toward larger power consumption. The power consumption also varies

significantly depending upon the source of data for each instruction. Accessing data from different parts of the chip has different power costs, e.g., register vs. L1 cache vs. off chip memory. It is important to study these factors while selecting PMU events $MES_{Compute}$.

From the analysis of μ benchs running on KNF, we have observed that the power consumed by a CPU bound application varies significantly across instruction mixes. For example, an execution of a single unconditional jump statement in an infinite while loop by 128 threads causes an increase in total power consumption by 22% of P_{Idle} . If the instruction stream in the application is accessing registers, the IPC may decrease, as seen in *count*. In some cases, it may also increase the power consumption if the computation and communication can be overlapped to some extent. We have also noticed that the fraction of vector instructions in the total instructions executed is a dominating factor in the total power consumption of KNF. Due to the wide (512 bit) VPUs in KNF, vector instructions consume more power than integer instructions. The amount of branch mispredicted is also an influencing factor of the power consumption in KNF as observed in general purpose cores.

To distinguish these different components of power consumption, we set $MES_{compute}$ to four events in the core: *total instructions executed*, *vpu instructions executed*, *branches mispredicted*, *data read or write*. Here *data read or write* is the number of access to registers or caches or memory. In KNF, a wide memory access is split into multiple accesses, and each split access is counted separately in *data read or write*. Moreover, since KNF does not have a counter for integer instructions, we derive the number of integer instructions from *instructions executed* and *vpu instructions executed*.

MES_{Interconnect}: The increased number of cores in manycore processors necessitates the need for a high performance interconnect. The more cores there are, the more traffic among cores and various levels of memory hierarchy. We have to explore these events while selecting $MES_{Interconnect}$.

All the cores, L2 caches and memory modules are connected through a ring network in KNF. Any load or store instruction that goes beyond the L1 cache involves this ring network and adds a power component to the total power consumption. After experiments and analysis with the μ benchs, we have selected three events to compute the power consumption in interconnect: (a) *L2 read miss mem fill*, (b) *data read miss or write miss*, and (c) *bus cycle duration*. Here, *data read miss or write miss* represents the total number of L1 misses, and *bus cycle duration* is the number of cycles when the bus is busy with traffic. *L2 read miss mem fill* is the actual number of memory accesses that are satisfied by memory after a miss from L2. The absences of uncore PMU events in SEP restricts us from exploring different variations of $MES_{Interconnect}$.

MES_{Memory}: The dynamic power consumption in memory can be computed as a function of memory activities. The events included in MES_{Memory} depend on memory technology. For example, counting the total number of operations (read and write) is sufficient for dynamic random access memory (DRAM), but a complex memory hierarchy may

require including more events in MES_{Memory} .

We model the dynamic power consumed in KNF main memory using a single core PMU event counter; namely *L2 read miss mem fill*. There was no *L2 write miss mem fill* event in D0 KNF card. Since SEP does not have access to uncore PMU events, we set MES_{memory} to one event: *L2 read miss mem fill*.

C. Calibrating the Dynamic Power Model

After selecting MES_i for each component i of KNF, the missing part of the power model in Equation 3 are the weights $w_{i,e}$ for all the events in MES_i . The frequency of event e , $r_{i,e}$, in Equation 3 is computed by dividing the total number of events in 1ms interval by the number of clock ticks in 1ms. We computed *iipc* (number of integer instructions executed per clock tick), *vipc* (number of vector instructions executed per clock tick), *data* (number of data accesses per clock tick), *bm* (number of branch mispredicted per clock tick), *bus* (number of bus usage per clock tick), *l2* (number of l2 accesses per clock tick), and *mem* (number of memory accesses per clock tick). The weight for each event is calibrated by correlating the real power value and $r_{i,e}$ of μ benchs. We sort the μ benchs in terms of number of sub components (e.g. integer pipeline, registers, vector pipeline, L1 cache, L2 cache, memory, interconnect) it is using in ascending order. As shown in Table II, the sorted sequence is the same as found in Table I for μ benchs. Benchmark *sleep* is in row 1 because it is not using any sub components. If a benchmark has no activity in a particular sub component then that event counter is not needed. The x sign in a particular cell in a row b and column e means that there is no activity of event e for benchmark b . Hence weight $w_{i,e}$ becomes meaningless for benchmark b . For example, we have x in column *vipc* for benchmark *count* because *count* has no vector instructions. Hence *vipc* is not necessary while fitting the model for row 5. We fit the model for row j by finding weights of all the meaningful events that minimize the squared error between measured and predicted power, and move to row $j + 1$. The weights computed in row j are unchanged while fitting the model for row $j + 1$. Finally, we find the weight for *bm* using a real benchmark that has a high branch misprediction rate. In Table II, $*$ indicates where certain weight was computed for the first time and \downarrow indicates the propagation of each weight to the final model.

Bench	iipc	vipc	data	bm	bus	l2	mem
Sleep	x	x	x	x	x	x	x
Sleep-C	x	x	x	x	x	x	x
Null	*	x	x	x	x	x	x
Noop	\downarrow	x	x	x	x	x	x
Count	\downarrow	x	x	x	x	x	x
Vec	\downarrow	*	*	x	*	x	x
Vec-Int	\downarrow	\downarrow	\downarrow	x	\downarrow	x	x
V-I-O3	\downarrow	\downarrow	\downarrow	x	\downarrow	x	x
L1-Hit	\downarrow	\downarrow	\downarrow	x	\downarrow	x	x
L2-Hit	\downarrow	\downarrow	\downarrow	x	\downarrow	*	x
L2-Miss	\downarrow	\downarrow	\downarrow	x	\downarrow	\downarrow	*
Final	\downarrow	\downarrow	\downarrow	*	\downarrow	\downarrow	\downarrow

TABLE II
CALIBRATING WEIGHTS FOR DIFFERENT EVENTS.

D. Calibrating the Static Power Model

In this section, we calibrate the static power model shown in equation 2 which has a temperature dependent leakage power component. For simplicity, we assume that the fan speed remains constant during the execution period of the benchmarks. Figure 2 shows the total power consumption while running *V-I-O3* for 20 seconds. It is observed that the total power consumption increases by 24% of P_{Idle} even for starting *V-I-O3*. This $0.24P_{Idle}$ watt is a fixed cost for *V-I-O3*. As we keep *V-I-O3* running, the temperature rises and the total power consumption increases by 5 watts in 20 seconds. The rise in temperature is from 75°C to 78°C in 20 seconds. In summary, there are 5 watts of power increment for a 3°C rise in temperature. A quick estimate for leakage power increment due to temperature is $\frac{5}{3} \frac{\text{watts}}{^\circ\text{C}}$. This is the value used for ΔP in equation 2.

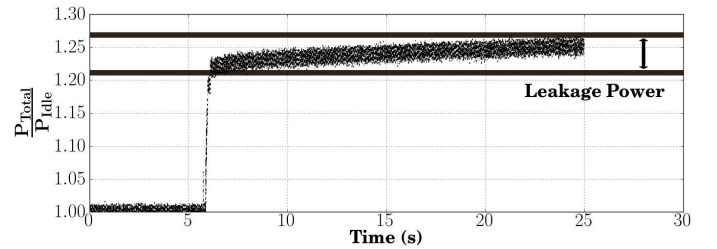


Fig. 2. Modeling Leakage Power.

E. Model Validation

Figure 3 and 4 depicts the accuracy of our proposed many integrated core power model, which is calibrated for KNF. The average error for the 12 μ benchs is 3.2% (mbAvg.). *V-I-O3* has maximum 12% error. For the test benchmark suite, the average error (bAvg.) is 4.73%. From these results, it is clear that our model provides high accuracy for predicting power consumption of 10 different benchmarks, which are highly optimized for the KNF card. The data also shows that our model is capable of predicting power consumption of dynamic workloads for different input sizes. For three different option sizes of *bs*, the average error is 3.40%. We also illustrate that our power model can estimate power consumption of a single application with diverse data mappings. In the case of Black-Scholes, we verify our model with three different data mappings with an average error of 4.54%.

The model can also accurately predict power consumption of workloads that go through different phases in a single execution. To illustrate this, we compare the measured and predicted power of *bs16384* in Figure 5. Our power model closely follows the measured power consumption trend of *bs16384*. We have analyzed the activity factors of three major components while running *bs16384*. During the first 10 seconds, activity in the cores is very low and in interconnect are very high. Due to the very long request queue in the memory module, memory requests are not serviced quickly, so as a result the *ipc* is very low. After the 10th second memory starts servicing requests quickly, interconnect activity slows down, and cores become much more active leading to high *ipc* and higher power consumption.

Figure 6 shows the normalized activity factors of different events for the test benchmark suite. The power consumption of bs is $1.35\times$, $1.43\times$, and $1.63\times$ of P_{Idle} for three different input size (16384, 16400, 16512) as shown in Table I. The activity factor shows that the dominating event behind this difference is $vipc$ and mem . Benchmark $bs16512$ has more memory activity than $bs16384$, which also implies that memory requests are served quickly in $bs16512$. $Vipc$ increases as a result. In the next section, we use our power model to generate power traces for different components of the chip so that we can perform thermal simulation.

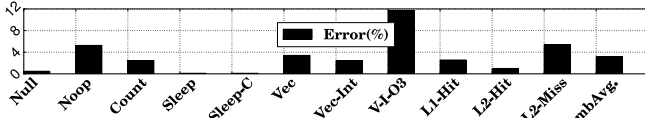


Fig. 3. Error of the model for micro benchmarks suite.

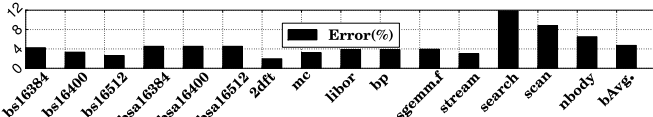


Fig. 4. Error of the model for test benchmark suite.

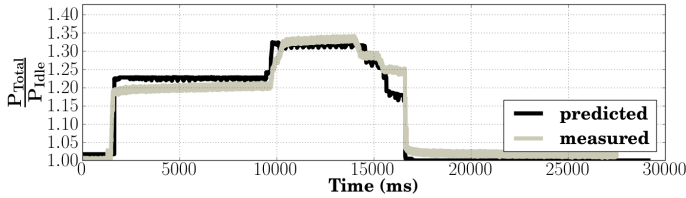


Fig. 5. Power prediction for $bs16384$.

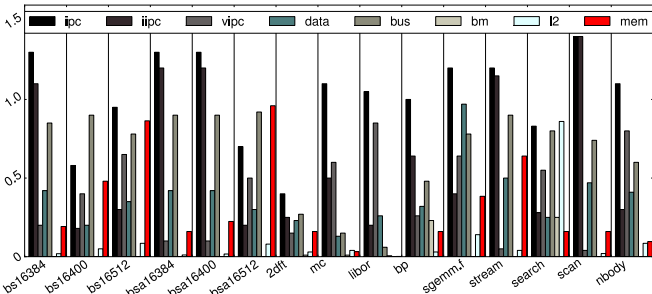


Fig. 6. Normalized activity factors of the events.

IV. THERMAL MANAGEMENT VIA INTERMIXING

In this section, we first show that workloads run on KNF have quite a bit of variability in power dissipation, which can be leveraged by our novel workload intermixing technique for efficient thermal management in such manycore processors. We use the power model we developed for more detailed analysis of the core level power consumption of the KNF applications. Figure 7 shows the breakdown of the total power consumption into three main components.¹ The compute unit is responsible for most of the chip’s dynamic power with a large range.

There is quite a bit of thermal variability between various KNF applications as summarized by Table I due to their

¹The idle power consumption of the KNF memory controller is very high due to a large analog component. For illustration, we have selected the idle power for memory as $0.2P_{Idle}$.

power heterogeneity. Compute intensive benchmarks such as mc , $libor$, bp make the cores hotter since they consume large amounts of core power, while more memory bound benchmarks, e.g., $stream$, $scan$, etc., use less core power, and thus keep the core cooler. To further understand the magnitude of these differences and the performance cost of state of the art thermal management used in KNF , we compare two different benchmarks: bp and $2dft$. We use the thermal simulator described in section II-D and run each benchmark for 800s. All the cores occasionally reach the critical temperature when running bp and thus experience 23% performance overhead as a result of default throttling mechanism. There is no throttling while executing $2dft$. While evenly distributing 32 cores among these two benchmarks, there is only 0.03% performance degradation while meeting thermal constraints. This observation has motivated us to intermix thermally heterogeneous workloads to reduce thermal hotspots in manycore architecture.

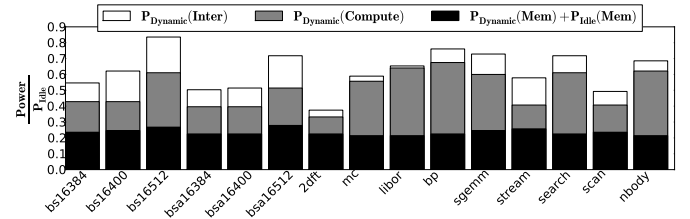


Fig. 7. Power breakdown into components.

The target applications of manycore architectures usually have the sufficient parallelism to utilize all the compute cores. However, assigning all the cores to a compute intensive application is inefficient from performance point of view due to the throttling that will result because of thermal hotspots. Instead we split the available compute resources between a compute intensive and a memory intensive workload. Since each of the cores in KNF has a private $L1$ cache and a dedicated portion of $L2$ cache, intermixing workloads should not introduce cache overhead. The available off chip memory bandwidth can be efficiently shared between two workloads. We call this technique workload intermixing (WI).

Workload	Mix	Workload	Mix
WL1	bp+2dft	WL6	libor+stream
WL2	libor+2dft	WL7	bp+libor
WL3	bp+scan	WL8	2dft+scan
WL4	libor+scan	WL9	nbody+bp
WL5	bp+stream	WL10	nbody+libor

TABLE III
WORKLOAD DESCRIPTION

In order to evaluate the efficacy of our workload intermixing (WI) strategy, we create 10 workloads using applications in the test benchmark suite to have a representative mix of hot and cold benchmarks as shown in Table III. Each benchmark in the workload runs 400s. The state of the art default DTM policy throttles the cores when the core temperature reaches the critical temperature threshold and clock gates the cores until the temperature falls below $85^{\circ}C$. We have also implemented $DVFS$ with five different settings: $(1.15V, 0.90GHz)$, $(1.12V, 0.85GHz)$, $(1.10V, 0.80GHz)$, $(1.07V,$

0.75GHz), and (1.04V, 0.70GHz) using scaling techniques in [14]. *DVFS* technique goes to the next lower step whenever a thermal emergency occurs and tries to maintain the maximum possible speed while avoiding thermal emergency. In the absence of thermal emergencies, *DVFS* increments the frequency to the next higher whenever possible.

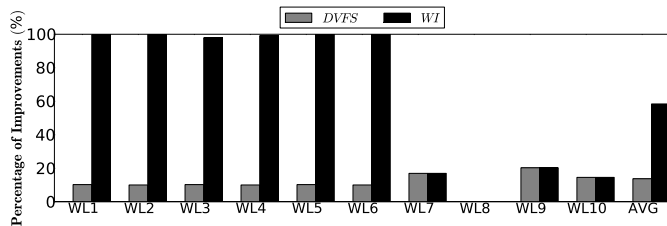


Fig. 8. Improvements over default policy.

The default policy experiences computation slowdown of up to 22.4% with an average of 11.46% for the workloads in Table III. As expected, the performance cost due to throttling is higher for workloads with *hot* benchmarks, e.g., *WL7*. Meanwhile, workloads with only *cold* benchmarks (e.g., *WL8*) do not experience any thermal hotspots. Figure 8 shows the reduction of computation slowdown when using our strategy of workload intermixing (*WI*) and *DVFS* over the default policy. *DVFS* performs better than the default policy by only 14% on average. Our proposed technique *WI* shares the cores among thermally heterogeneous workloads in interleaved fashion. It obtains on average 58% reduction in the computation slowdown as compared to the default policy. We also improve over *DVFS* by 51% on average. For a subset of workloads, *WL1*–*WL6*, *WI* improves over the default policy by 99%. *WI* has identical improvements (99%) over *DVFS* for the target workloads. If there is no thermal heterogeneity in the workload (e.g., *WL7*, *WL9*, *WL10*), our thermal management technique applies *DVFS* because: (a) there is no way to benefit from intermixing, (b) *DVFS* performs better than the default policy.

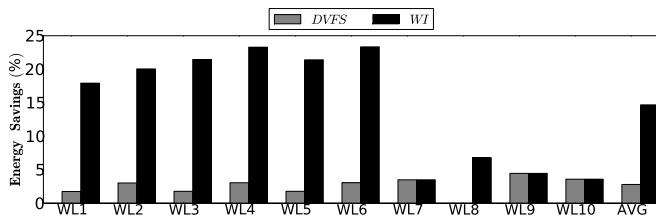


Fig. 9. Energy Savings over default policy.

Figure 9 shows the percentage of energy savings of *WI* and *DVFS* relative to the default thermal management policy. *WI* saves on average 14% of energy compared to 2.8% with *DVFS*. The savings of energy come from two different sources. The reduction in computation throttling helps the jobs finish faster. In addition, *WI* saves leakage power by reducing the average temperature of the cores. For the heterogeneous workloads representing our target cases (*WL1*, *WL2*, *WL3*, *WL4*, *WL5*, and *WL6*), *WI* saves 21% and 19% energy on average relative to the default policy and *DVFS* respectively. Energy savings are maximized when we mix the hottest with the coldest benchmarks.

To make such temperature aware intermixing possible in manycore architectures like *KNF*, the parallelism (number of threads) of manycore applications should be parametrized and included in the programming model of future manycore designs. Even though applications running on future manycore chips may request a specific number of cores, the actual level of parallelism and core allocation decisions will be dynamically made by the OS scheduler based on workload profiling and current thermal state of the system.

V. CONCLUSION

In this paper, we explore the prospect of workload intermixing as an efficient thermal management technique for manycore architecture. In order to perform an accurate thermal simulation for this purpose, we developed the first ever power model for a many integrated core architecture. We calibrate our model on Intel’s *KNF* with a set of benchmarks and real power measurement data. In contrast, when compared to measurement on Intel’s *KNF* card, our model has under 4.73% average error. The inclusion of operating voltage and clock frequency enables our model to be used as a part of the *DVFS* policy design. Our thermal simulation based on this power model revealed that intermixing thermally heterogeneous workloads in manycore chips can reduce the thermal hotspots by 58% on average compared to state of the art thermal management techniques with an energy savings of 14% on average.

VI. ACKNOWLEDGE

This work has been funded by Intel Lab and DARPA.

REFERENCES

- [1] V. Tiwari et al. Power analysis of embedded software: A first step towards software power minimization. *In IEEE Transactions on VLSI Systems*, 1994.
- [2] S. Rivoire et al. A comparison of high-level full-system power models. *In HotPower*, 2008.
- [3] J. Sheaffer et al. A flexible simulation framework for graphics architectures. *In HWWS*, 2004.
- [4] K. Ramani et al. Powerred: A flexible power modeling framework for power efficiency exploration in gpus. *In GPGPU*, 2007.
- [5] X. Ma et al. Statistical power consumption analysis and modeling for gpu-based computing. *In SOSP Workshop on Power Aware Computing and Systems*, HotPower, 2009.
- [6] S. Hong et al. An integrated gpu power and performance model. *In ISCA*, 2010.
- [7] Many Integrated Core (MIC) Architecture www.intel.com.
- [8] Intel Sampling Enabling Product (SEP) software.intel.com/file/35252.
- [9] National Instrument Data Acquisition www.ni.com/dataacquisition/.
- [10] Y. Zhang et al. Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects. *Technical report, University of Virginia*, 2003.
- [11] H. Su et al. Full chip leakage estimation considering power supply and temperature variations. *In ISLPED*, 2003.
- [12] K. Skadron et al. "Temperature-aware microarchitecture: Modeling and implementation. *TACO 2004*.
- [13] J. Wang et al. Vapor chamber in high-end vga card. *IMPACT*, 2010.
- [14] D. Brooks et al. Dynamic thermal management for high-performance microprocessors. *HPCA*, 2001.
- [15] J. Sheaffer et al. Studying thermal management for graphics-processor architectures. *ISPASS*, 2005.
- [16] J. Choi et al. Thermal-aware task scheduling at the system software level. *ISLPED*, 2007.
- [17] A. Coskun et al. Proactive temperature management in mpsoes. *ISLPED*, 2008.