Empirical Mode Decomposition for Modeling of Parallel Applications on Intel Xeon Phi Processors

Gary Lawson[†], Masha Sosonkina[†], Tal Ezer[*], Yuzhong Shen[†]
Department of Modeling, Simulation and Visualization Engineering[†]
Department of Ocean, Earth, and Atmospheric Sciences[*],
Old Dominion University
Norfolk VA 23529, USA
{ glaws003, msosonki, tezer, yshen } @odu.edu

***Abstract*—For modern parallel applications, modeling their general execution characteristics, such as power and time, is difficult due to a great many factors affecting software-hardware interactions, which is also exacerbated by the dearth of measuring and monitoring tools for novel architectures, such as Intel Xeon Phi processors. To address this modeling challenge, the present work proposes to employ the Empirical Mode Decomposition (EMD) method to describe an execution as a series of modes culminating in a single residual trend, for which, in its turn, a model equation is obtained as a non-linear fit. As outcome, an overall energy consumption may be predicted using this model. A real-world quantum-chemistry application GAMESS and a molecular-dynamics proxy application CoMD were considered in the experiments. The results demonstrate that the energy modeled ranges within 10–30% of the measured energy, depending on the length of execution.**

***Keywords*-EMD, Modeling, Time, Power, Energy, Power Limiting, Intel Xeon Phi, Knights Landing, GAMESS, CoMD**

I. Introduction

The imposed 20MW power cap [19] has placed great strain on achieving exascale performance in the near future. Although processor hardware is becoming more energy-efficient, over 60% of the power budget is dedicated to peripheral devices, such as network and cooling devices. This leaves only 40% of the power budget towards exascale performance (processor and memory components)—thus, high performance systems are focused on getting the most performance out of the hardware available.

When using hardware, especially large-scale, multi-node systems, it is important to run software with the best available execution configuration. However, testing all available configurations is prohibitively expensive and wasteful; this is where modeling is beneficial. Currently, a general model to predict energy, power, and time for a given application and hardware configuration does not exist. Therefore, this work paves the way for such a model. Here, power traces are processed using the Empirical Mode Decomposition (EMD) method to obtain the underlying basic trend (denoted typically as "residual") of power over time. For each application run, a non-linear function is fitted into its residual data points, and thereby providing a model of the execution for the tested software and hardware configuration.

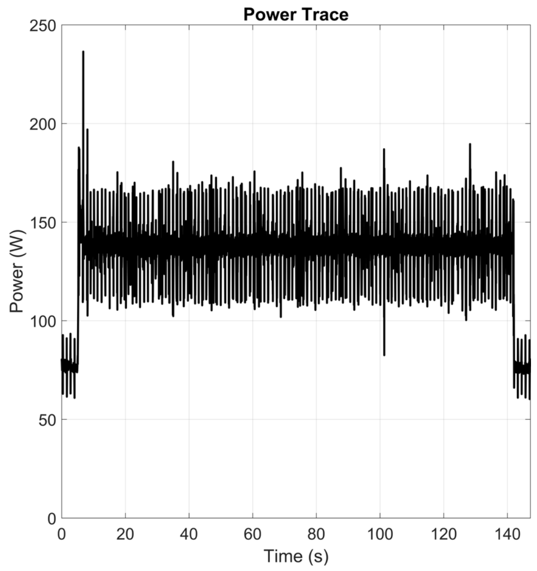
The new Intel Xeon Phi, code-named "Knights Landing" (KNL), is now available as a processor or co-processor. It is capable of hosting a full Linux OS [30], [18], and supports all legacy codes. The previous generation of Xeon Phi, code-named "Knights Corner" (KNC), was available only as a co-processor. For KNL, there are a number of hardware improvements over KNC, e.g., a higher performance-per-watt ratio. Also, KNL supports the Multi-Channel DRAM (MCDRAM) which is a 3D-stacked DDR memory, which is capable of higher than the traditional DDR memory bandwidth. KNL may deliver up to three TFLOPS double-precision performance, whereas KNC is capped at one TFLOPS [30]; and the KNL thermal design power (TDP) is lower: 215 W for KNL as compared to 245 W for KNC. Due to the many enhancements and advantages of KNL over KNC, this work focuses on KNL as processor.

Real-world applications typically consist of highly variable workloads, some are compute-intensive and others are memory-intensive. The parallel applications chosen in this work are GAMESS [15], [14], [28] and CoMD [8], such that their problem characteristics considered here make them both compute-intensive, although GAMESS requires significantly more main memory than CoMD. In a nutshell, the contributions of this work are as follows:

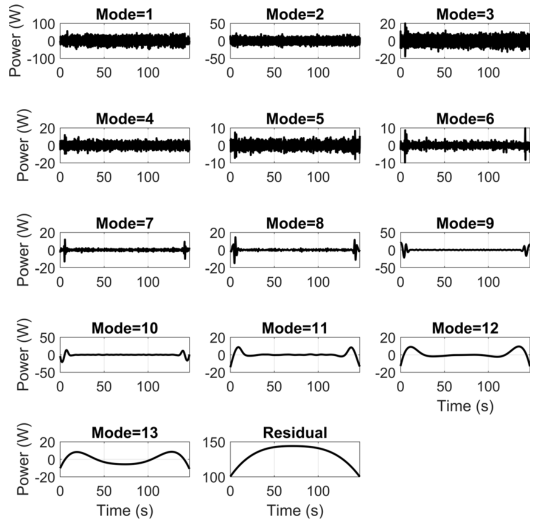
- Proposed a novel procedure for modeling of application power consumption on a given computing platform by using runtime traces and variants of the EMD method.
- Validated the accuracy of the obtained model on real-world applications, GAMESS and CoMD.
- Demonstrated benefits of using MCDRAM memory on KNL for parallel applications.

A. Related Work

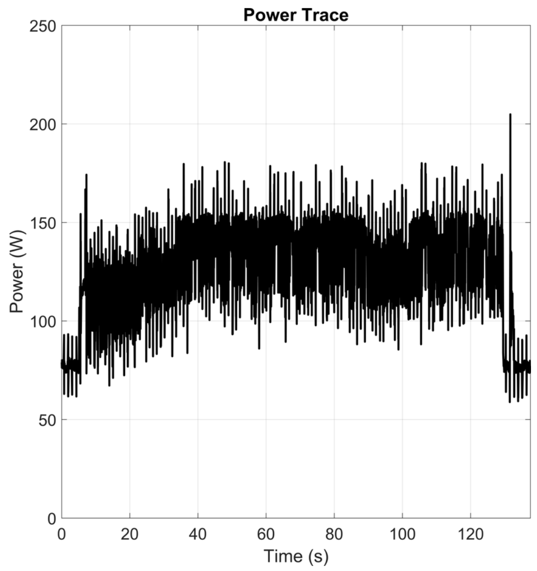
A general performance model—known as "LogP"—for parallel architectures and applications has been proposed already at the dawn of parallel computing [5]. LogP uses communication latency, memory transfer overhead, the reciprocal of per-processor communication bandwidth, and the number of available processor/memory modules to calculate the application performance. Following in the footsteps of the LogP model, the "roofline" model [32] has also been proposed as a general way to model parallel application runtime performance. It de-



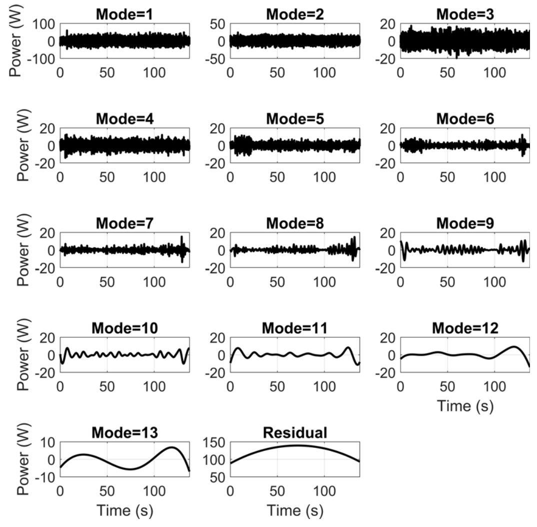
(a) Power Trace for CoMD



(b) IMF Amplitudes for CoMD



(c) Power Trace for GAMESS



(d) IMF Amplitudes for GAMESS

Figure 1: Illustration of EMD: Original power trace for CoMD (a) and GAMESS (c) decomposed into IMFs (b) and (d), respectively, where the final mode is the the residual trend.

scribes the relationship between the data movement and computational throughput, which helps to identify performance bottlenecks with respect to the theoretical performance of the hardware. These models, however, do not consider power and energy consumption of a software-hardware combination. Building upon the roofline model, [4] includes power and energy contributions of the parallel architecture, however, is not easily extendable to real-world applications, such as GAMESS or CoMD, which typically stress multiple platform components simultaneously. Instruction-level modeling [29] is another way for characterizing the hardware, which is also not easily extended to real-world applications. McPAT is a state-of-the-art power, area, and timing modeling framework [26] for manycore and multicore processors. It models the low-level interactions of the hardware at the transistor level as well as the critical path of an application. The framework is especially useful in finding underlying performance bottlenecks, experimenting with theoretical workloads in simulators and stress-testing new hardware designs. The methodology proposed here has a higher-level of applicability: It considers applications in their entirety, does not require low-level hardware component knowledge, which may not be readily available from vendors, and may be easily applied by application developers and end-users.

Over the past few years, a large body of knowledge has been cultivated by researchers interested in the Intel Xeon Phi. However, only few of them investigate power-performance tradeoffs [4], [25], [33], [2], [20]. The effect of thread affinity on the energy consumption has been evaluated in [22], and a model for energy on KNC in the offload mode has been proposed in [23].

The remainder of the paper is organized as follows: Section II presents the procedure for modeling execution traces using EMD; Section III describes the experimental results and validation for the parallel applications and platform used; Section IV concludes.

II. Modeling with EMD

Since modeling complex interactions between a computing platform and parallel application has proven difficult when singling out specific platform components or concentrating on specific workloads, this work models the execution from a high-level perspective given the residual trend (power as a function of time) obtained using the EMD method.

A. Empirical Mode Decomposition

The Empirical Mode Decomposition (EMD) method [16], [34] is considered here to analyze the entire calculation without breaking it into smaller segments or phases, which has proven to be difficult to do in conjunction with power readings [23]. EMD provides non-parametric non-stationary time-series analysis, which has been already successfully applied in a variety of fields, such as medicine, finance, engineering, and more recently in geosciences. The EMD code used here is based on the code adapted for analysis of sea level data [11] and climate change studies [10]. To the authors knowledge, EMD has not been used before to investigate parallel application performance and energy consumption. The main advantage of EMD over standard spectral methods is that it detects oscillating modes with time-dependent amplitudes and frequencies, so it is useful for analyzing irregular data with unknown frequencies. For application execution data, however, the interpretation of the EMD results is not straightforward, since individual modes do not necessarily represent particular execution characteristics.

The EMD implementation used here is based on the original one from [16], [34], as adapted in [11], and is available at [9]. Figure 1 presents two examples of the EMD procedure on a power trace; Fig. 1a shows the trace of the execution of the molecular dynamics application CoMD[1] and Fig. 1c shows the trace of the execution of the quantum-chemistry package GAMESS[2], both collected on the Intel Xeon Phi processor. Figure 1 provides a visual representation of each step in the EMD process. EMD decomposes the original power trace

[1]The EAM force kernel was used on MCDRAM with problem size 100, 63 cores, and a power limit of 120W (see Section III).

[2]The 1L2Y problem, MCDRAM, 32 cores, and a default power limit of 215W were used (see Section III).

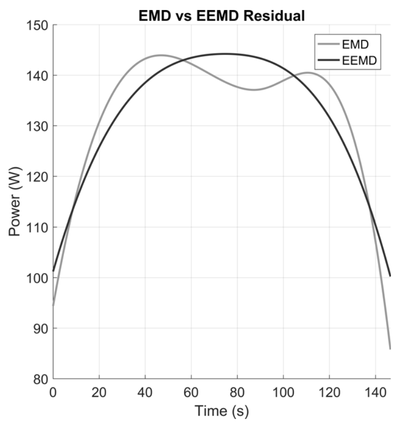


Figure 2: Illustration of the residual found using EMD vs EEMD applied to the same power trace.

(time-series) into *oscillating intrinsic mode functions* (IMFs) and into a *residual trend*, which is the final mode, with respect to time-variable amplitudes (in Figs. 1b and 1d). Note that the total number of IMFs, which output by EMD, depends on the trace characteristics. EMD extracts IMFs through a process called *sifting*. To sift, the minimum and maximum extrema of the time-series are used to calculate the average; the difference between the average and time-series is then treated as the time-series for the next sift. This process continuously refines the dataset until the standard deviation of the resulting time-series is less than 0.2 (see [16]). Once this standard deviation is obtained, the resulting time-series is accepted as an IMF, which is subsequently removed from the original time-series. This process is repeated until the residual is found from which no other IMFs may obtained. It may happen, however, that an intermittent mode cannot manifest under the standard deviation constraint and "contaminates" the residual trend with a spurious IMF. To alleviate this problem and obtain a more reliable shape of the residual, the Ensemble EMD (EEMD) method [34] may be applied. EEMD works by introducing white noise to the time-series to exhaust the sifting process. While in EMD, the sifting processes the original time-series once to extract each IMF, in EEMD, white noise and the sifting process are applied to the time-series multiple times, such that the white noise is averaged out and only the trace itself remains. This way, EEMD may avoid the residual contamination, as seen, e.g., in Fig. 2, which illustrates the difference between residuals found using EMD and EEMD. It is important to note that the number of modes produced by EEMD and EMD are the same for the traces explored in this work and possibly in general. The difference between the EEMD and EMD is in the shape of the resulting IMFs, where intermediate EEMD modes now include intermittent oscillations otherwise included in the residual for EMD.

B. Constructing the Model

The first step is to obtain a power trace. In this work, power measurements are sampled at a rate of 5ms, which is close to the maximum available sampling rate of 1ms. A sampling rate of 5ms ensures all samples return a reliable measurement as well as allows for a significant number of IMF modes to be extracted from the trace to get an accurate and reliable residual. Lower sampling rates, even on the order of hundreds of milliseconds, would suffice for producing a residual trend. However, the higher the sampling rate, the more IMF components that may be extracted since each IMF component resembles a particular time-scale (defined as the time between successive extrema). The residual is on the largest time-scale obtained by EMD.

The next step is to apply EEMD to the power trace. In this work, a white noise of 5 W was applied to the time-series and averaged over 50 EEMD passes. These are the smallest values, which were found to be sufficient to remove the final residual contamination, and fit the model to the residuals with $R^2 > 0.95$. In this work, a total of five traces are collected for each execution, which is specified by a particular system-application configuration, all of which are used for a fit. Note that several traces are used due to the variability in execution characteristics (e.g., memory stalls and conflicts).

In the final step, the model of the residual is defined as a quadratic fit into the obtained data points. In other words, three coefficients of a second-degree polynomial are determined. The reason for the quadratic polynomial fit is that, in this work, with the application of EEMD, the shape of the residual always looks like a quadratic. Furthermore, it is a concave-down quadratic because of the nature of high-performance applications in general and how the executions are organized here, in particular. Namely, to each application execution, "cool" periods are appended, equal to idle power draw before and after the execution. Therefore, the maximum power draw appears towards the center of the entire trace with minima at the ends (see Figs. 1a and 1c for examples).

Once the quadratic model is obtained, the execution parameters, such as the total time, average power, and total energy may be quite easily defined as follows:

- Total time is difference in the start and end times, the start time is always zero for the model, and the end time is taken as the time when the power draw equals to that at the start (zero) time.
- Average power is an average of the power draw as defined by the model.
- Energy is found by integrating the model between start and end times.

An example of a quadratic polynomial fit to several residuals obtained from multiple runs of a GAMESS problem[3] is illustrated in Fig. 3. Note that the EEMD residual captures the characteristics of the original power trace well: the energy approximated by the residual is within 10% of the energy measured for the trace [11]. For this and all the test cases presented in Section III, R^2 is above 95%.

[3]The 1L2Y problem, MCDRAM, 32 cores, and a default power limit of 215W were used (see Section III).

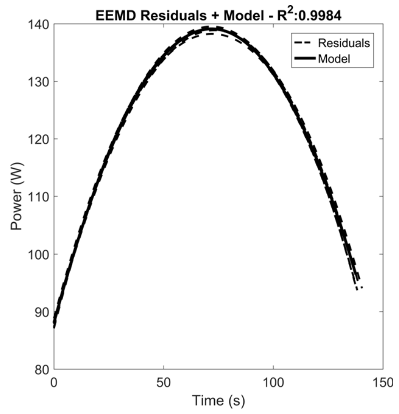


Figure 3: A quadratic fit to residual trends produced by EEMD from five sample executions of a GAMESS problem.

III. Experimental Validation

First, the computing platform and parallel applications are outlined. Then, the execution parameters obtained with the model are compared to those measured for the original trace, followed by the discussion.

A. Computing Platform

The experiment has been conducted on a single-node KNL system, nicknamed *Rulfo*, located at Old Dominion University and obtained through the Intel Developer Access Program (IDAP)[4]. The chosen system is a Colfax KNL Ninja Liquid Cooled Pedestal Developer Platform, as listed by IDAP. The Intel Xeon Phi 7210 processor has 64 cores @ 1.3 GHz (1.5 GHz turbo, and 1.0 GHz min). Each core has four hardware threads and two 512-bit vector processing units (VPU) for concurrent processing, and is capable of out-of-order execution. Cores are interconnected using a 2D mesh, and omni-path fabric is built into the system for scalability. The device has two levels of cache by default, L1 contains 32 KB instruction and 32 KB of data and L2 contains 1 MB shared between two cores (32 MB total).

KNL now contains the new 16GB multi-channel DRAM (MCDRAM) where various memory modes may now be adopted by the application: cache, flat, or hybrid mode. By default, MCDRAM is treated as regular (DDR) memory—flat mode—which is used in this work; cache mode configures the MCDRAM to serve as the L3 cache, and hybrid mode allows 8–12 GB to serve as a L3 cache and the remaining 4–8 GB as traditional DDR memory [30], [18]. In this work, the MCDRAM has 16 GB of data that expands the DRAM memory for the system; the system has an additional 98 GB of DRAM memory (6x DDR4 @ 16.384 GB and 2133 MHz) for a total of 112 GB DRAM. In this work, the MCDRAM and DDR are used separately; MCDRAM may be used in flat-mode without explicit code additions by using the `numactl` command as follows: `numactl --membind`

[4]https://software.intel.com/en-us/articles/developer-access-program-for-intel-xeon-phi-processor-codenamed-knights-landing

Table I: Best execution time, power, and energy across all the workloads in CoMD and GAMESS and DRAM memory types.

			DRAM Memory					
		Best Config	DDR			MCDRAM		
Application	Workload	# Cores, PLimit(W)	Time(s)	Power(W)	Energy(J)	Time(s)	Power(W)	Energy(J)
CoMD	LJ (60)	63, 120	19	141	2654	18	138	2528
	LJ (80)	63, 120	41	142	5830	40	138	5549
	LJ (100)	63, 120	82	142	11664	82	138	11279
	EAM (60)	63, 120	35	142	5007	35	138	4776
	EAM (80)	63, 120	75	143	10681	73	138	10153
	EAM (100)	63, 120	140	142	19906	137	138	18921
GAMESS	1L2Y	32, 215	130	130	16862	127	128	16190
	20w	32, 215	212	125	26332	201	123	24708
	S265	32, 215	35	126	4444	31	125	3865
	S301	32, 215	44	127	5534	39	126	4905

`1 ./<executable>` . By default, the DDR is used when executing the application.

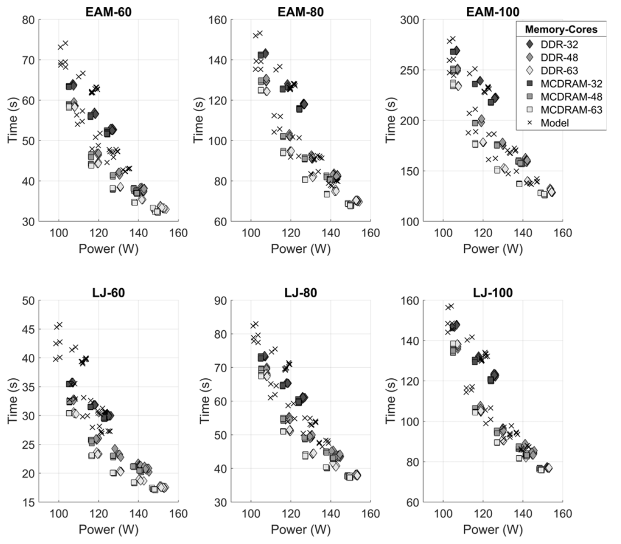
1) Power Limiting: For the Intel Xeon Phi, user-defined frequency scaling is not available; instead, the frequency may be changed indirectly by setting power limit thresholds for the device. The Xeon Phi System Management Controller (SMC) varies operating frequency as power surpasses the designated thresholds. Specifically, the Xeon Phi uses two power threshold values—*low* and *high*—each with a designated time window. By default, the low power threshold is set to the TDP with a time window of 100ms and the high threshold at 120% of the TDP and a time window of 10ms. When power exceeds the low threshold for the duration of the time window, frequency is decreased until power consumption is less than that of the threshold. When power exceeds the high threshold for the duration of the time window, the thermal throttling mechanism is engaged, which forces the device to the lowest operating frequency of around 500 MHz, as seen experimentally. More on Xeon Phi power limiting can be found in the datasheet [17]; however, it has not yet been updated for Knights Landing. In this work, the static power limiting is used, such that power limits are set prior to execution. (See [24] for setting power limits dynamically according to a power and performance loss model.)

2) Measurement Tools: The Sandia National Labs PowerAPI [21] is used to measure energy via the Linux Power Capping Framework (LPCF) [1] plugin which reads energy from the Running Average Power Limit (RAPL) [31], [6] counters. The PowerAPI uses the hardware locality (*hwloc*) API [27], [3] to detect the underlying hardware and is very portable. Hence, no modification to the API was required to measure energy for the KNL processor. LPCF is also used to update power thresholds. Power measurements are collected every 5ms, and measurements are collected for five seconds before and after the application is executed to establish the idle power draw, about 80 W. Although most of computing platforms support up to 1ms resolution for sampling power using RAPL, for KNL, it has been found empirically that 1ms sampling is unreliable on KNL. Thus, 2ms is the minimum suggested sampling rate. A sampling rate of 5ms was chosen for this work to accommodate both the accuracy and speed of the modeling with EEMD, provided that the execution times are often greater than 60 seconds.

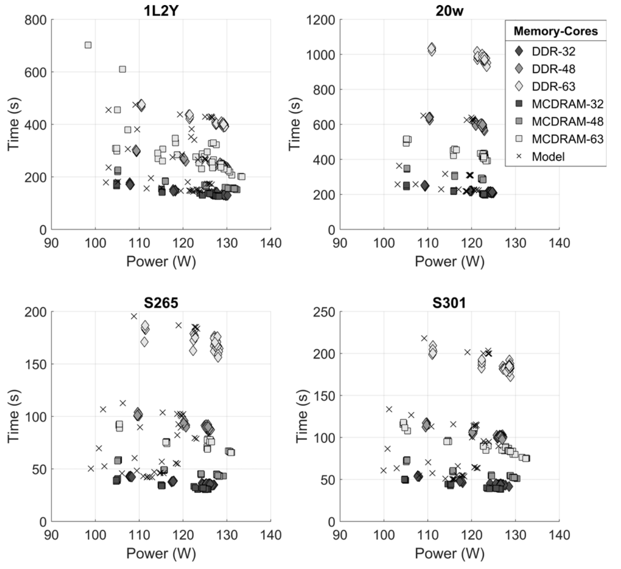
B. Parallel Applications

Applications tested here contain a wide range of workloads with respect to the real-word problems they are solving and computing-platform components they stress. Note that the applications have not been optimized or tuned specifically for KNL in order to investigate performance and energy consumption "as is" for hardware-software co-design that might be followed by joint hardware-software optimizations.

1) GAMESS: The General Atomic and Molecular Electronic Structure System (GAMESS) [14], [28] is a widely used quantum chemistry package capable of performing molecular structure and property calculations by a rich variety of *ab initio* methods finding an (approximate) solution of the Schrödinger equation for a given molecular system. An approximate (uncorrelated) solution is initially found using the Hartree-Fock (HF) method via an iterative self-consistent field (SCF) approach, and then is improved using various electron-correlated methods, such as second-order Møller-Plesset perturbation theory (MP2). To reduce the computational complexity for large molecular systems, a fragmentation approach, such as Fragment Molecular Orbital (FMO) method [13], is used, which divides the system into fragments and applies a quantum chemical method to each fragment, followed by the consideration of fragment interactions. The inputs used in this work are calculated using the MP2 method. Specifically, they are *20w*, a cluster of 20 water molecules; *1L2Y*, a synthetic protein tryptophan cage; *S256*, a 1-trichloromethylsilatrane (TCMS) molecule with 6-31G(d) basis set (265 basis functions), and *S301*, a TCMS molecule with 6-31G(d,p) basis set (301 basis functions). The inputs 20w and 1L2Y also use FMO approximations of short-range interactions up to trimers (when triples of fragments considered as a single fragment). OpenMP is not available in GAMESS, so half of the total MPI (Message Passing Interface) tasks are dedicated to computation and



(a) CoMD



(b) GAMESS

Figure 4: Measured and modeled time vs power for (a) CoMD and (b) GAMESS with two memory types (DDR and MCDRAM) and three core counts (32, 48, and 63); and one subplot per workload.

the remaining half to data movement via the generalized Distributed Data Interface (GDDI) [12].

2) CoMD: Co-design Molecular Dynamics (CoMD) is a proxy application developed as part of the Department of Energy co-design research effort [7] at the Extreme Materials at Extreme Scale (ExMatEx) center. CoMD is compute-intensive, where approximately 85–90% of the execution time is spent computing forces. In this work, both force kernels are used: the more accurate Embedded Atom Model (EAM) force kernel for short-range material response simulations, such as uncharged metallic materials [8], and the less accurate Lennard-Jones (LJ) force kernel. The LJ force kernel consists of one compute loop, whereas EAM consists of three compute loops and a small halo data exchange between the second and third loop.

Problem size is expressed as the number of atoms along an axis of the material; the default material is copper. In this work, each axis is equivalent (in atoms) which defines the material shape is a cube. A problem size of 40 equates to $4 \times 40^3 = 256,000$ atoms. The input sizes used here are 60, 80, and 100, which correspond to 864K, 2048K, and 4000K copper atoms, respectively. This work uses the hybrid MPI and OpenMP programming models to distribute the workload among processors. One MPI process is dedicted to each KNL core and OpenMP is used to spawn four hardware threads for each MPI process. In this work, 63, 48, and 32 cores are used for each workload. For CoMD, since the material shape is three-dimensional, the material workload is partitioned among cores as $9 \times 1 \times 7$, $8 \times 1 \times 6$, and $8 \times 1 \times 4$, for 63, 48, and 32 cores, respectively. Such partitionings were found experimentally to deliver the best performance for the corresponding numbers of cores.

C. Comparisons of Different Measured Configurations

Here, CoMD and GAMESS are tested for their various workloads on the Intel Xeon Phi processor using 32, 48, and 63 cores (with one core allotted solely to the measurement software, thereby mitigating performance overheads due to measurements). Either the *DDR* or *MCDRAM* memory type was used exclusively as DRAM (i.e., MCDRAM was in the "flat mode"). Power limits were set in the following way: (1) the default of 215 W and (2) from high of 140 W to low of 90 W in increments of 10 W, where the high power limit of 140 W was chosen because average power draw of the applications does not exceed 150 W. For CoMD, the EAM and LJ force computation kernels are tested for problem sizes of 60, 80, and 100. For GAMESS, four problems are tested, 1L2Y, 20w, S265, and S301. Among all the workloads, Table I states the execution parameters for the best configuration (column `Best Config`), which results in the minimum energy consumed and which is specified by the number of cores and power limit and shown as `# Cores` and `PLimit(W)`, respectively. It is remarkable that the test cases with MCDRAM always executed faster and required less power than those with DDR (cf. columns `MCDRAM` and `DDR`), and hence, always consumed less energy. The difference in power draw between DDR and MCDRAM for CoMD is larger than that for GAMESS, although the average power draw for GAMESS is lower than that CoMD. For CoMD, the obtained average power draw is 15–20 W higher than that specified by the power limit (cf. columns `Power(W)` and `PLimit(W)` in Table I). The energy savings obtained were larger for GAMESS than those for CoMD (see columns `Energy(J)` in Table I). Finally, the best configuration (column `Best Config`) was found to be the same for both DDR and MCDRAM either for CoMD or GAMESS across all the workloads. The best power limit for

CoMD was found to be 120 W for both the LJ and EAM force kernel, whereas, for GAMESS, the default of 215 W appeared to be the best power limit.

D. Comparisons of Modeled and Measured Results

Figure 4 presents the measured and modeled time-to-solution and average power draw for each workload, memory type, and number of cores. First, notice that the power limiting has a larger impact for CoMD than GAMESS. In particular, CoMD shows a linearly decreasing pattern as power draw increases while GAMESS shows a marginal decrease in the execution time as power draw increases and a large range of time-to-solution values, depending on the number of cores. These results may be explained by a general observation that, for CoMD, the maximum number of cores is always preferred while, for a GAMESS workload, smaller numbers of cores may lead to the best execution time, which is less affected by the power limiting and L2 cache saturation. Also, recall that the minimum power limit tested is 90 W, yet the resulting minimum measured power for CoMD or GAMESS is 105 W as indicated in Fig. 4, which is in line with authors' previous findings in [24]. When comparing modeled and measured values in Fig. 4 observe that the model calculates reasonably well both the time and average power usage for each configuration, even though power tends are underestimated and further model tuning may be warranted. In particular, for CoMD (Fig. 4a), the model closely matches the measured values with power underestimated by approximately 5–10 W. For GAMESS (Fig. 4b), on the other hand, the power is underestimated by almost 15 W in some cases.

Using the constructed model the same best configurations, specified by the (`# Cores`, `PLimit(W)`) pair, were found as those observed with measurements (see column `Best Config` in Table I). For these best configurations, Table II provides the modeled energy values and quadratic model coefficients a, b, and c. Observe that the model acceptably calculates the total energy consumption (cf. columns `Energy(J)` in Tables I and II). Specifically, the modeling error is within 10% for longer executions, i.e., for those taking greater than 100s, while the error increases up to 30% for shorter ones. For CoMD, which contains shorter traces, the overall average error has been found to be 15%. CoMD problem size of 100 shows the least error of 5–10%, whereas the problem size of 60 shows the largest error of 15–30%. For GAMESS, workloads 1L2Y and 20w result in the smallest error (less than 10%), but the errors in S265 and S301 are in the 15–25% range. Large model errors may be attributed, in part, to fitting the quadratic model into the outcome (residuals) of the EEMD procedure, which itself may incur errors of up to 10% [11], as verified empirically in the course of this work. Note that fitting into the raw traces is practically an impossible task, circumventing which is a principal objective of the current work. It may be possible to decrease the errors by increasing the power sampling rate, which is set to 5ms in this work (see Sections II-B and III-A2 for details on this rate choice).

It may be also observed in Table II that, for the test cases with MCDRAM, the model predicts always less energy consumption than that predicted for the cases with the DDR memory, which is in line with the measured results. From Table II, some tendencies of the model coefficients may be noticed. In particular, as problem size increases, the coefficients a and b decrease for both CoMD and GAMESS. Also, a is always negative, which is a trait of a concave-down shape of the quadratic residual. Finally, the coefficient c always increases with problem size. Further testing is necessary to observe how these tendencies hold across other platforms.

IV. Conclusions

A high-level modeling approach has been proposed in which EMD first captures the trend of an execution and then a quadratic model is used to reproduce this trend. The model may then be employed to approximate execution parameters, such as time, power, and energy consumption. The model accurately calculates the best configuration, comprising core counts and power limits, that minimizes energy consumption, which is promising in the quest for predicting the best application-platform configuration for a wide range of configuration parameters. Expanding this model to more general cases, which also do not involve the entire trace, constitutes the future work as well as fine tuning the present model to narrow the model error ranges.

Although only CoMD and GAMESS applications on the computing platform with KNL processor were used here to validate the model, the modeling approach has a general applicability since it does not rely on the application or platform specifics. Hence the proposed model may be applied to any hardware-software combination. The only requirement is that a power trace is collected such that EMD may be applied, e.g., that the sampling rate is of high enough resolution. The intrinsic mode functions (IMFs) produced by EMD may also be used (along with the residual) to understand execution performance, which is left as a future work. IMFs represent physical interactions and may be useful to identify execution phase in real-world applications.

This work also investigated the use of KNL MCDRAM memory against DDR in flat-mode. The results unanimously show that using MCDRAM promotes faster execution and a decreased power draw, especially for larger problem sizes. Hence, it is suggested that MCDRAM is used whenever possible to improve performance of any application.

Acknowledgment

This work was supported in part by the Air Force Office of Scientific Research under the AFOSR award FA9550-12-1-0476, by the U.S. Department of Energy, Office of Advanced Scientific Computing Research, through the Ames Laboratory, operated by Iowa State University under contract No. DE-AC02-07CH11358, and by the U.S. Department of Defense High Performance Computing Modernization Program, through a HASI grant. The authors would like to thank the reviewers for their constructive comments.

Table II: Model coefficients and calculated energy for all the configurations from Table I.

		DRAM Memory							
		DDR				MCDRAM			
Application	Workload	*a*	*b*	*c*	Energy (J)	*a*	*b*	*c*	Energy (J)
CoMD	LJ (60)	-0.432	12.161	63.461	3388	-0.401	11.202	64.754	3267
	LJ (80)	-0.105	5.340	84.493	6561	-0.102	5.145	83.051	6362
	LJ (100)	-0.026	2.414	96.495	12297	-0.025	2.298	95.328	11933
	EAM (60)	-0.134	6.138	80.571	5844	-0.133	5.940	80.123	5537
	EAM (80)	-0.030	2.548	97.848	11321	-0.029	2.481	95.512	10866
	EAM (100)	-0.008	1.134	108.488	20461	-0.008	1.130	104.974	19512
GAMESS	1L2Y	-0.009	1.381	90.230	18109	-0.010	1.406	88.682	17565
	20w	-0.003	0.599	99.350	27176	-0.003	0.640	95.837	25944
	S265	-0.107	4.912	76.968	5286	-0.128	5.428	73.563	4746
	S301	-0.075	4.128	80.108	6454	-0.088	4.400	79.125	5768

References

[1] Power capping framework, 2016. https://www.kernel.org/doc/Documentation/power/powercap/powercap.txt.

[2] D. Abdurachmanov, B. Bockelman, P. Elmer, G. Eulisse, R. Knight, and S. Muzaffar. Heterogeneous high throughput scientific computing with APM X-Gene and Intel Xeon Phi. *CoRR*, abs/1410.3441, 2014.

[3] F. Broquedis, J. C. Ortega, S. Moreaud, N. Furmento, B. Goglin, G. Mercier, S. Thibault, and R. Namyst. hwloc: A generic framework for managing hardware affinities in hpc applications. In *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*, pages 180 –186, Feb. 2010.

[4] J. Choi, M. Mukhan, X. Liu, and R. Vuduc. Algorithmic time, energy, and power on candidate HPC compute building blocks. In *2014 IEEE 28th International Symposium on Parallel Distributed Processing (IPDPS)*, Arizona, USA, May 2014.

[5] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a realistic model of parallel computation. In *Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPOPP '93, pages 1–12, New York, NY, USA, 1993. ACM.

[6] H. David, E. Gorbatov, U. R. Hanebutte, R. Khannal, and C. Le. RAPL: memory power estimation and capping. In *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, ISLPED'10, pages 189–194, New York, NY, USA, 2010. ACM.

[7] DOE. Co-design, 2013. http://science.energy.gov/ascr/research/scidac/co-design/.

[8] ExMatEx. CoMD proxy application, 2012. http://www.exmatex.org/comd.html.

[9] T. Ezer. EEMD/HHT, 2015. http://www.ccpo.odu.edu/~tezer/HHT/.

[10] T. Ezer, L. P. Atkinson, W. B. Corlett, and J. L. Blanco. Gulf stream's induced sea level rise and variability along the u.s. mid-atlantic coast. *Journal of Geophysical Research: Oceans*, 118(2):685–697, 2013.

[11] T. Ezer and W. Corlett. Is sea level rise accelerating in the Chesapeake Bay? A demonstration of a novel new approach for analyzing sea level data. *Geophysical Research Letters*, 39(19), 2012.

[12] D. G. Fedorov, R. M. Olson, K. Kitaura, M. S. Gordon, and S. Koseki. A new hierarchical parallelization scheme: Generalized distributed data interface (GDDI), and an application to the fragment molecular orbital method (FMO). *Journal of Computational Chemistry*, 25, Issue 6:872–880, 2004.

[13] M. S. Gordon, D. G. Fedorov, S. R. Pruitt, and L. V. Slipchenko. Fragmentation methods: A route to accurate calculations on large systems. *Chemical Reviews*, 112(1):632–672, 2012. PMID: 21866983.

[14] M. S. Gordon and M. W. Schmidt. Advances in electronic structure theory: GAMESS a decade later, 2005.

[15] Gordon Research Group. The general atomic and molecular electronic structure system (GAMESS), 2016. http://www.msg.ameslab.gov/gamess/index.html.

[16] N. Huang, Z. Shen, S. Long, M. Wu, H. Shih, Q. Zheng, N. Yen, Chi C. Tung, and H. Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454(1971):903–995, 1998.

[17] Intel. Intel Xeon Phi coprocessor: Datasheet, 2015. http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-coprocessor-datasheet.html.

[18] J. Jeffers, J. Reinders, and A. Sodani. *Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition*. MK Publishers, 2015. http://lotsofcores.com/.

[19] D. Kusnezov, S. Binkley, B. Harrod, and B. Meisner. DOE exascale initiative, 2013. http://www.industry-academia.org/download/20130913-SEAB-DOE-Exascale-Initiative.pdf.

[20] D. LaKomski, Z. Zong, T. Jin, and R. Ge. Optimal balance between energy and performance in hybrid computing applications. In *Green Computing Conference and Sustainable Computing Conference (IGSC), 2015 Sixth International*, pages 1–8, Dec 2015.

[21] J. Laros. Sandia national laboratories high performance computing power application programming interface (API) specification, 2016. http://powerapi.sandia.gov/.

[22] G. Lawson, M. Sosonkina, and Yuzhong S. Energy evaluation for applications with different thread affinities on the Intel Xeon Phi. In *Computer Architecture and High Performance Computing Workshop (SBAC-PADW), 2014 International Symposium on*, Oct 2014.

[23] G. Lawson, V. Sundriyal, M. Sosonkina, and Y. Shen. Modeling performance and energy for applications offloaded to Intel Xeon Phi. In *Proceedings of the 2Nd International Workshop on Hardware-Software Co-Design for High Performance Computing*, Co-HPC '15, pages 7:1–7:8, New York, NY, USA, 2015. ACM.

[24] G. Lawson, V. Sundriyal, M. Sosonkina, and Y. Shen. Runtime power limiting of parallel applications on Intel Xeon Phi processors. In *Proceedings of the 4th International Workshop on Energy Efficient Supercomputing*, E2SC '16, pages 39–45, Piscataway, NJ, USA, 2016. IEEE Press.

[25] B. Li, H. C. Chang, S. Song, C. Y. Su, T. Meyer, J. Mooring, and K. W. Cameron. The power-performance tradeoffs of the Intel Xeon Phi on HPC applications. In *Parallel Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*, pages 1448–1456, May 2014.

[26] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 469–480, Dec 2009.

[27] Open MPI Project. Portable hardware locality (hwloc), 2016. https://www.open-mpi.org/projects/hwloc/.

[28] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, T. L. Windus, M. Dupuis, and Jr. J. A. Montgomery. General atomic and molecular electronic structure system. *J. Comput. Chem.*, 14:1347–1363, Nov. 1993.

[29] Y. S. Shao and D. Brooks. Energy characterization and instruction-level energy model of Intel's Xeon Phi processor, 2013. http://www.eecs.harvard.edu/~shao/papers/shao2013-islped.pdf.

[30] A. Sodani, R. Gramunt, J. Corbal, H. S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. C. Liu. Knights landing: Second-generation Intel Xeon Phi product. *IEEE Micro*, 36(2):34–46, Mar 2016.

[31] V. Weaver. Reading RAPL energy measurements from linux, 2011. http://web.eece.maine.edu/~vweaver/projects/rapl/.

[32] S. Williams, A. Waterman, and D. Patterson. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, April 2009. http://doi.acm.org/10.1145/1498765.1498785.

[33] J. Wood, Z. Zong, Q. Gu, and R. Ge. Energy and power characterization of parallel programs running on Intel Xeon Phi. In *2014 43rd International Conference on Parallel Processing Workshops*, pages 265–272, Sept 2014.

[34] Z. Wu and N. Huang. Ensemble empirical mode decomposition: A noise-assisted data analysis method. *Advances in Adaptive Data Analysis*, 01(01):1–41, 2009.