Modeling Performance and Energy for Applications Offloaded to Intel Xeon Phi*

Gary Lawson[†] Vaibhav Sundriyal Masha Sosonkina Yuzhong Shen
Old Dominion University
1300 Engineering & Computational Sciences Bldg
Norfolk, Virginia, 23529
{glaws003, vsundriy, msosonki, yshen}@odu.edu

ABSTRACT

Accelerators are adopted to increase performance, reduce time-to-solution, and minimize energy-to-solution. However, employing them efficiently, given system and application characteristics, is often a daunting task. A goal of this work is to propose a general model that predicts performance and power requirements for an application, computational portions of which are offloaded to an accelerator. Intel Xeon Phi is the only accelerator type investigated here, and only in offload execution mode. This mode is also employed by other accelerator types, such as GPU; thus the proposed model is applicable directly. The predictive capabilities of the model are demonstrated by determining the best hardware-software configuration instances with respect to the minimum energy consumption for the CoMD proxy application executed on single or multiple nodes. For the CoMD problem sizes investigated here, the best modeled configuration was relatively close to the best measured configuration with relative error under 5% of the energy consumed for most configurations. Initial model validation also confirmed the model accuracy for a variety of model parameters, such as host computation time and power consumption on the host and accelerator. The model also provides estimates of the total data movement and computational throughput as well as of some key metrics, such as FLOPs-

per-joule and bytes-per-joule, which are commonly used to study the energy-performance trade-offs.

Keywords

heterogeneous architecture modeling, Xeon Phi, offload, energy, performance, HPC cluster

1. INTRODUCTION

Hardware-software co-design is a process currently accelerating as high performance computing nears the exascale era. Important problems, such as power capping and fault tolerance, may find solutions only by joining forces at the hardware and software/application levels while maximizing performance and minimizing energy-to-solution. Computational throughput is the most obvious performance metric to measure. However, as the exascale era nears, moving data among nodes and individual devices will become an ever increasing bottleneck, and therefore, it is another critical performance metric to measure.

Accelerators are often adopted to reduce time-to-solution with low energy costs. The Intel Xeon Phi is an accelerator that promotes high memory bandwidth (i.e. data movement) in addition to high computational throughput. From the work of Choi *et. al* [2], the Xeon Phi is capable of 11 GFLOPs/J and 880 MB/J for single-precision operations (measured throughput of 2 TFLOPs/s and 180 GB/s memory bandwidth). Double-precision has a measured maximum throughput of 1 TFLOPs/s, and therefore capable of 5.5 GFLOPs/J. The Xeon Phi co-processor also supports three execution modes: offload, symmetric, and native (i.e., device-only). Only offload execution is investigated in this work because it is the execution mode commonly used by other accelerator types (i.e., GPU, FPGA).

Together with the main processor and memory, accelerators constitute a *heterogeneous system* having a different type of hardware architecture. In a heterogeneous architecture, for each individual device¹, there are several configurable parameters affecting execution performance. In a multi-node environment, even more parameters may be tuned. At the same time, applications typically have a set of parameters pertaining to execution performance, which may be varied. Hence, exploring the configuration space for every possible combination of application-architecture parameters is not

*Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org. Co-HPC2015, November 15–20, 2015, Austin, TX, USA © 2015 ACM. ISBN 978-1-4503-3992-6/15/11\$15.00 DOI: <http://dx.doi.org/10.1145/2834899.2834903>

[†]Corresponding author.

¹The term “device” refers to either host or accelerator architecture.

feasible, especially if one wants to optimize the performance “on-demand”, i.e., for a specific application configuration on an available set of devices/nodes. In this work, a model is proposed to estimate the effect of a given system configuration on distributed application execution time and energy consumption in the offload mode, and to extrapolate this prediction for more system configurations. A major objective is to explore the configuration space and determine a “best” configuration, based on some criteria, such as energy consumed, execution time, and data movement.

1.1 Related Work

The model builds on the classic “roofline” model [20, 3], and the “time-frequency” model [4, 19, 14] to define execution time. Power is modeled in the same manner as in authors’ previous work [14]. An initial validation of the model is performed using either single- or multi-node computing platforms running the CoMD proxy application for molecular dynamics simulations [17, 7]. Other related work on modeling and performance profiling of the Xeon Phi has been conducted in [18] and [16]. However, those research efforts do not combine the accelerator execution modes with the host operation, as proposed here for heterogeneous architectures with accelerators used to offload computations from the host CPU.

In their earlier work [15], the authors have developed an experimentation procedure to obtain the desired performance metrics during execution. Specifically, host power is measured using RAPL (Running Average Power Limit) MSRs (Machine Specific Register), and Xeon Phi power is measured via the power output file `/sys/class/micras/power`. Power measurements for both the host and accelerator as well as host performance measurements are taken independently of the application and, therefore, may show the device power state before and after execution. On the other hand, the accelerator performance is captured within the application offload sections and requires code instrumentation. Additionally, [15] discusses how to synchronize measurements and application execution in order to extract key performance readings during execution phases selectively; thereby, accessing information on such measurements as device active and idle power draws, memory bandwidth during host communication, and PCI data transfers. A way to estimate the number FLOPs offloaded to the accelerator is also described in [15].

The paper is organized as follows. Section 2 presents the proposed heterogeneous-architecture model for applications decomposed into sub-domains. In Section 3, the experimental procedure is outlined. Section 4 discusses the model validation and analyzes the results. Section 5 concludes.

2. MODEL DESCRIPTION

A single-node heterogeneous architecture is composed of one multi-core host architecture and one or more multi-core accelerator architectures (A_i $i = 1, 2, \dots, n_{acc}$), where n_{acc} is the number of accelerators. Note that such an architecture may contain accelerators of different types (e.g., Xeon Phi and GPU). Each accelerator is connected to the host and one another by the PCI bus, contains a two-level memory hierarchy (with slow and fast memories), and is a many-core processing unit. It is also assumed that the slow and fast

memories are infinite and finite capacities, respectively, and that data must be moved between memories and processor (called resources) during application execution.

The parallel application is assumed to employ a domain-decomposition scheme [8], which is defined here as the division of a problem into sub-problems (called sub-domains) that are distributed among devices. Sub-domains may be computed in parallel, and may also require sharing data with neighboring sub-domains to solve the problem globally. It is assumed that data communication phase may not overlap computation phase. When executing an application, the total number of sub-domains is dependent both on the application and system configuration.

The distribution of sub-domains among resources is dependent on the execution mode: *device*, *offload*, or *symmetric*. For execution exclusively on the device, all work and data movement use only the resources of that particular device. For symmetric execution, sub-domains are distributed among the hosts and accelerators, serving as peers. For *offload*, on the other hand, the computations are performed either on the the host or accelerators, such that one host sub-domain is shared with one accelerator only. In other words, each sub-domain resides on the host while portions of its computational phase and data are copied to the accelerator for processing and the result returned to the host. It is assumed that host and accelerator computations do not overlap, i.e., one is idle while the other computes. The communications among sub-domain performed only by the host(s) while leaving the corresponding accelerators idle.

Certain accelerator types, such as Intel Xeon Phi, have all three execution modes whereas others, such as GPUs, do not have symmetric or native modes. Note that only the offload mode, which is the most common way to employ accelerators, is investigated in this work. Although, for the offload execution, the number of sub-domains off-loaded to the accelerator may be fewer or equal to the total number of sub-domains, this work considers only the latter case.

2.1 Execution Time

The execution of an application that employs domain-decomposition may be described as having the following four phases: *initialize*, *compute*, *communicate*, and *output*. The initialization phase sets up a problem to be solved, and the output phase relays important statistics and output upon completion. Solving each sub-problem requires an iterative pattern of computation and communication phases until a global solution is achieved. Note that the initialization and output phases are not modeled here because they are expected to affect little the overall performance for large-scale problems with multiple nodes. Modeling the influence on energy of peripheral sources, such as hard-drive and cooling, is also beyond the scope of this work.

The total execution time in the offload mode may modeled as

$$T = T_{comp} + T_{comm} \quad , \quad (1)$$

where T is the sum of the times required to perform all the computations and communications, respectively.

2.1.1 Computation Phase:

The total computation time is limited by the slowest time required to solve a sub-domain for a given execution mode. It is equivalent to being limited by the total time of a particular execution mode. Computation may be simply defined by the slowest execution mode because sub-domains of similar execution modes will require relatively the same time to solve; however sub-domains of differing execution modes may be vastly different, depending on load balance and the implementation. For the model, all sub-domains of similar execution modes are equivalent considering each would be modeled using the same parameters.

The total time to compute in offload mode T_{comp} is

$$T_{\text{comp}} = T_{\text{host}} + T_{\text{acc}} + T_{\text{pci}} . \quad (2)$$

It is defined by the execution time for the host T_{host} , accelerator T_{acc} , and communication time across the PCI bus T_{pci} . The time T_{host} to compute on the host is defined using the time-frequency model [14, 19] as:

$$T_{\text{host}} = t_{\text{on}} \cdot f_{\text{max}}/f + t_{\text{off}} , \quad (3)$$

where t_{on} is the time on-chip, t_{off} is the time off-chip, and f is the operational frequency during execution for the device such that $f \neq f_{\text{max}}$. This general equation is used simply to estimate host execution time and deduce the applications computational intensity on the host. The time T_{acc} to compute on the accelerator is defined using the roofline model [20, 3] as:

$$T_{\text{acc}} = \max(W_{\text{acc}} \cdot \tau_W, M_{\text{acc}} \cdot \tau_M) , \quad (4)$$

which is the maximum of the time to perform work W_{acc} and time to move data M_{acc} between memories with τ_W and τ_M being the times to perform a unit of work and to transfer a unit of data, respectively². The time T_{pci} to move data across the PCI bus is

$$T_{\text{pci}} = M_{\text{pci}} \cdot \tau_{\text{pci}} , \quad (5)$$

which is the product of the amount of data M_{pci} to be moved and the time per data movement τ_{pci} across the PCI bus.

2.1.2 Communication Phase:

Total communication time T_{comm} is limited by the slowest transfer between sub-domains, and it may be simply limited by the slowest communication type. For offload execution, there are two communication types to consider: transfers between sub-domains on the same node (called *intra-node*), and transfers between sub-domains on differing nodes (called *inter-node*). These two communication types may overlap. For configurations executed on one node, the intra-node communication model is used; and for multiple nodes the inter-node communication model is used.

Intra-node communication times T_{comm} may be defined as:

$$T_{\text{comm}} = M_{\text{comm}} \cdot \tau_{\text{comm}} , \quad (6)$$

where M_{comm} is the amount of data to be moved and τ_{comm} the time required to move a unit of data. Inter-node communication time T_{comm} is

$$T_{\text{comm}} = t_l + M_{\text{comm}} \cdot \tau_{\text{comm}} , \quad (7)$$

²Work and data are expressed as floating-point operations and bytes, respectively.

where t_l is the network latency. Note that, for a single-node configuration, network latency time is not present in Eq. (6).

2.1.3 Throughput:

The time τ_W to perform unit of work is computed by taking the inverse of throughput. The definition of throughput is generally the total number of cores performing work times the frequency per core. However, for devices, such as the Xeon Phi, throughput also depends on characteristics such as vectorization intensity [1] and operations per cycle:

$$\tau_W = (n_{\text{core}} \cdot n_{\text{ops}} \cdot VI \cdot f)^{-1} , \quad (8)$$

where number of cores n_{core} includes only those used in the computation, f is the device frequency, the number of operations per cycle n_{ops} is a value between one and two representing an average of single and fused multiply-add operations performed, and VI is the vectorization intensity, which is a measure of the number of SIMD instructions issued. For the Intel Xeon Phi, VI may be a value between one and eight for double-precision floating-point operations³. Note that Eq. (8) is applicable to all the Intel devices based on the Sandy-Bridge or newer microarchitectures.

2.2 Power and Energy

The total power draw P for the system is the sum of the power draw for each device; the total number of devices is n_{dev} , and power is defined as:

$$P = \sum^{n_{\text{dev}}} (P_{\text{dev}}) . \quad (9)$$

Device power is defined as static and dynamic power; however dynamic power may fluctuate during execution depending on whether the device is idle or active. A device is considered active if performing computation, and otherwise is idle (that is to include all communications). Device power may be defined using the weighted sum of the power draw for each execution state:

$$P_{\text{dev}} = P_{\text{active}} \cdot \frac{T_{\text{active}}}{T} + P_{\text{idle}} \cdot \frac{T_{\text{idle}}}{T} , \quad (10)$$

where the total execution time $T = T_{\text{active}} + T_{\text{idle}}$ and

$$P_{\text{state}} = P_{\text{static}} + \rho \cdot n_{\text{core}} \cdot f^3 \quad (11)$$

is the power draw for a given state and depends on the static power draw P_{static} , a power constant ρ , the number of cores for the device n_{core} , and the state frequency f (see, e.g., [19] and the references therein).

From Eqs. (1), (2) and (9), energy may be defined as:

$$E = E_{\text{host}} + E_{\text{acc}} + E_{\text{pci}} + E_{\text{comm}} , \quad (12)$$

where the energies E_{host} , E_{acc} , and E_{pci} correspond to the three terms of Eq. (2), respectively, and E_{comm} is obtained using either Eq. (6) or Eq. (7) for single- or multi-node executions, respectively.

3. EXPERIMENT SETUP

It is difficult to define all the necessary parameters analytically. Therefore certain model parameters are found by fitting to the data obtained during specially designed trial

³Single-precision operations are not investigated in this work.

Table 1: Hardware characteristics and software versions for host devices on Borges and Bolt.

		Borges	Bolt
		Sandy Bridge	Sandy Bridge
Hardware	Model	Xeon E5-2650	Xeon E5-1650
	Sockets	2	1
	Frequency (GHz)	2.001–1.2	3.201–1.2
	P-States	10	16
	Cores (p Socket)	8	6
	Threads (p Core)	2	2
	LL Cache (MB)	20	12
	DRAM (GB)	64	64
	TDP (Watts)	95	130
Software	Intel Compiler	2013 sp1.1.106	2013.3.163
	Intel MPI	4.1.2	5.0.2
	MPSS	3.4.1	3.4.4.1
	PAPI	5.3.2	5.3.0

Table 2: Accelerator hardware specifications, which are the same for Bolt and Borges.

Intel Xeon Phi 5110p	
Cores	60
Threads (per core)	4
Frequency (GHz)	1.053
LL Cache (MB)	30
DRAM (GB)	8
VPU (bits)	512
FMA (ops/cyc)	2
TDP (Watts)	245

executions of a given application. In other words these parameters are estimated. For example, to fit data using the time-frequency model in Eq. (3), the host frequency is varied to estimate the time on- and off-chip, and host power during application execution. To estimate the power on Intel Xeon Phi (in Eq. (9)), executions with a different number of threads are performed since frequency changing capability is not available on Intel Xeon Phi. An experimentation procedure, described in detail in [15], has been designed to estimate all the required parameters that cannot be obtained analytically. In particular, [15] depicts how host and Xeon Phi power, execution time, and performance are estimated experimentally using application code instrumentation. Measuring the offload performance requires instrumenting the offload sections with the code to capture the hardware counter data. In this work, PAPI [9] is used to access measured parameters from hardware counters on the host and Xeon Phi.

The experiment has been conducted on two computing systems: single-node (nicknamed *Borges*) located at Old Dominion University and multi-node cluster (nicknamed *Bolt*) located at Iowa State University. Tables 1 and 2 provide the detailed specifications for each system including host, accelerator hardware, and software versions on the host device. For each system, the same Intel Xeon Phi accelerator has been used, the 5110p. For the Bolt system, the compute nodes are QDR-connected with Infiniband. Thermal design power (TDP) is an estimate of the amount of power consumed by the device while running applications, provided by the vendor; it does not provide the peak power capable of each device.

3.1 Overview of CoMD

CoMD is a proxy application developed as part of the Department of Energy co-design research effort [6] Extreme Materials at Extreme Scale [7] (ExMatEx) center. CoMD is a compute-intensive application where approximately 85–90% of the execution time is spent computing forces. Although two methods are available for the force computation, this work focuses only on the more complex and accurate EAM force kernel for short-range material response simulations, such as uncharged metallic materials [17]. EAM was chosen here because its parallel performance generally receives less attention than the more commonly used Lennard-Jones potential, which easily yields itself to parallelism.

As in the authors previous work [12], the three compute loops in the EAM force kernel have been targeted for offload. Offloading the position and velocity functions slowed execution due to additional data transfer requirements. The time to move data and process it on the Xeon Phi was larger than the time to process data on the host. In addition to offload statements, 64-byte alignment, conversion from multi-dimension to one-dimensional arrays and from array of structures to structure of arrays, SSE3 instructions have been enforced during compilation [5], and utilization of the 2 MB buffers available through the environment variable [10] has been implemented.

The maximum number of atoms per cell has been changed to 16 from the default 64 for all the experiments per authors’ earlier finding [12] that the smaller atom count leads to superior performance. CoMD problem size is expressed as the number of atoms along an axis of the material cube, where each axis is equivalent. For example, a problem size of 50 equates to $4 \times 50^3 = 500,000$ atoms.

At this initial stage, the proposed models have been validated only on this large-scale application. Future work will include a diverse set of applications including memory-intensive ones, such as large sparse linear-system and eigenvalue-problem solutions.

3.2 Configurations Tested

The measured energy is averaged over five runs for each experiment. For the Borges system, only two configurations are investigated, termed *MIC 1* and *MIC 2*, corresponding to employing only one or both Xeon Phi devices⁴, respectively. On Bolt, six configurations are investigated, termed *N1 MIC 1*, *N1 MIC 2*, *N2 MIC 1*, *N2 MIC 2*, *N3 MIC 1*, and *N3 MIC 2*, where *N1*, *N2*, and *N3* correspond to one, two, and three nodes used to run CoMD⁵. For each configuration, the host frequency, number of Xeon Phi (MIC) threads, and model problem size were varied as follows:

- All ten power states were considered on Borges (from 1.2 to 2.001 GHz with the 100-MHz stepping). On Bolt, only were seven (3.201, 3.2, 2.8, 2.3, 1.9, 1.5, and 1.2 GHz) out of sixteen possible states (from 1.2 to 3.201 GHz with variable stepping) chosen to make the number of

⁴*MIC* stands for “many-integrate cores” technology used in Intel Xeon Phi.

⁵Only three nodes of Bolt are outfitted with two Xeon Phi devices.

Figure 1: Relative energy-model error per Eq. (13) on Borges.

measured configuration instances manageable while still having enough data to fit the models.

- Seven MIC-thread values ranging from 120 to 236 (four threads per core) were taken to execute CoMD. Note, that, since one core is always occupied by four threads dedicated to operating system tasks, such as servicing the offload daemon and to avoid thread oversubscription, the maximum of 236 application threads is reasonable to use on the 60-core Xeon Phi considered here.
- Although problem sizes of 50, 60, 70, and 80 are explored to observe the computational intensity of CoMD for given platforms in this work, all the results presented here are for the problem size of 50 only. Executions with the other problem sizes exhibited similar behavior but took significantly longer to complete.

The *compact* thread affinity and thread-level granularity were used on accelerator devices since they were found to perform better in the authors’ previous work [11, 12].

4. VALIDATION

The relative error between the measured energy E_{measured} and E_{modeled} , modeled by Eq. (12), is calculated as:

$$100(E_{\text{modeled}} - E_{\text{measured}})/E_{\text{measured}}. \quad (13)$$

Examples of error quantification are featured in Figs. 1 and 2 for Borges and Bolt systems, respectively, where each configuration is considered for all the chosen MIC-thread and host-frequency values. For Borges—as seen in Fig. 1—the majority of configuration instances are modeled with no more than 5% of error. Note that only does *MIC 1* at the frequency of 2.001 and lower MIC-thread values appear outside the 5% error range but is still confined within the 10% threshold. In Fig. 2, the relative error is also confined in the 10% interval with the *MIC 1* configurations showing a slightly better prediction accuracy, in general.

4.1 Execution Time

Table 3 presents the execution-time model parameters for each configuration (column **Config**) of the CoMD and the two systems (column **System**) as calculated from Eqs. (3) to (6) for the host compute time T_{host} , host communication time T_{comm} , Xeon Phi compute time T_{acc} , and PCI transfer time T_{pci} . The model parameters have been estimated using linear regression over a sample set of configuration instances with varying host frequencies and MIC thread counts. For detailed procedure to estimate parameters in Table 3 see [15]. The column n_{sub} lists the number of sub-domains for each configuration.

Figure 2: Relative energy-model error per Eq. (13) on Bolt.

The host computation time (column-set **Host Comp**) is modeled as a sum of on- and off-chip times (columns t_{on} and t_{off}) using linear regression while varying host frequencies in Eq. (3). The coefficient R^2 was 0.99 throughout this estimation for all the configurations; thereby, indicating a high accuracy of the model. Note that for both systems, the ratio of t_{off} to t_{on} (column $t_{\text{off}}/t_{\text{on}}$) is less than one for all cases, which shows that the host computation is compute, rather than memory, intensive. Recall that in CoMD, the host updates atom position, velocity, and (when needed) redistribution of atoms. The host communication time (column-set **Host Comm**) is measured and used in Eq. (6) to obtain the (inter-host) communication latency t_l by reading the hardware counters for the amount of data transferred M_{comm} and the transfer bandwidth $1/\tau_{\text{comm}}$.

The workload on the Xeon Phi (column W_{acc} in the column-set **MIC Comp**) is estimated by varying the MIC thread count and observing its effect on the execution time. The parameters M_{acc} and τ_M were estimated similarly to those for T_{comm} . Note that, although both systems use the same model (5110p) of the Intel Xeon Phi, the estimated workloads are somewhat different for the same configurations (e.g., cf. 4.12 and 4.46 TFLOPs on Borges and Bolt, respectively, for *MIC 1*), which is expected because the W_{acc} values were modeled rather than explicitly calculated, say, by counting all the algorithmic operations. The differences (amounting to about 8%) may be due, for example, to different numbers of co-processor stalls incurred during memory operations in each system, which are not accounted for

Table 3: Execution-time model parameters for Borges and Bolt.

System	Config	n_{sub}	Host Comp T_{host}			Host Comm T_{comm}			MIC Comp T_{acc}			PCI T_{pci}	
			t_{on} (s)	t_{off} (s)	t_{off} t_{on}	t_l	M_{comm} (MB)	$1/\tau_{\text{comm}}$ (MB/s)	W_{acc} (TFLOPs)	M_{acc} (MB)	$1/\tau_M$ (MB/s)	M_{pci} (GB)	$1/\tau_{\text{pci}}$ (GB/s)
Borges	MIC 1	1	1.62	0.23	0.14	0	59.17	450.51	4.12	83.5	2.48	5.33	4.61
	MIC 2	2	0.99	0.06	0.06	0	67.1	514.97	2.16	36.1	2.10	2.81	3.78
Bolt	N1 MIC 1	1	1.21	0.30	0.25	0	63.26	564.18	4.46	84.18	2.44	5.64	2.83
	N1 MIC 2	2	0.93	0.19	0.20	0	56.23	549.98	2.32	36.13	2.04	2.97	2.51
	N2 MIC 1	2	0.63	0.12	0.19	0.31	52.39	569.27	2.33	36.06	2.03	2.97	2.23
	N2 MIC 2	4	0.46	0.13	0.28	0.63	51.66	553.03	1.14	10.33	1.14	1.56	0.62
	N3 MIC 1	3	0.43	0.06	0.14	0.23	49.14	580.15	1.60	19.52	1.61	2.08	1.93
	N3 MIC 2	6	0.32	0.05	0.16	0.80	51.02	572.12	0.79	5.80	0.94	1.09	0.44

explicitly in the proposed model. It is also interesting to observe that the speed-up with respect to the number of sub-domains n_{sub} is very good in all the configurations with its lowest gains of 2.78 on three sub-domains and highest gains of 1.92 on two sub-domains of Bolt. Hence, one may infer that, even this non-optimized version of CoMD with VI of only 2.6 and with a moderate problem size of 50 scales well to multiple accelerators, either attached to a single node or to multiple nodes (cf. W_{acc} for *N1 MIC 2* and *N2 MIC 1* configurations, which have the same n_{sub} on Bolt).

The amount of data transferred over PCI (column M_{pci} from the column-set **PCI**) was read directly from the offload output reports, which most likely estimate this value since it appears to differ somewhat across the systems. The peak PCI bandwidth is 8 GB/s (PCI Express 2.0 x16) for Xeon Phi. As seen in Table 3, the configurations with fewer sub-domains (thus, more data to transfer per sub-domain), the bandwidth $1/\tau_{\text{pci}}$ reaches almost 3 and 5 GB/s on Bolt and Borges, respectively, which indicates that PCI transfers are well-optimized in offload executions.

4.2 Power

Table 4 presents the obtained power-model parameters for a Xeon Phi accelerator and the host CPU (column **Device**) when a given configuration (column **Config**) is considered on a single node of Borges or Bolt (column **System**). The parameters were calculated using linear regression over the configurations instances, in which different execution states, active or idle, were distinguished (column-sets **Active Power** and **Idle Power**, respectively). The samples are obtained from varying host frequency and MIC threads, for the host and Xeon-Phi device power models, respectively, obtained by Eq. (11). If the maximum frequency is used, then the maximum power draw P_{max} may be predicted for each state, as presented in Table 4. The total power P (column-set **Total Power**) is then calculated following Eq. (10). The model accuracy is assessed by the R^2 term. All the models appear well-correlated because the obtained R^2 coefficients, also provided in Table 4, are all close to one. For each device and configuration, a ratio of its active (idle) time to the total time is provided in column T_{active}/T (T_{idle}/T). Note that the sum of these two ratios is one. For a give device, the times T_{active} and T_{idle} mutually exclusive with an active-state time being equal to the device computation time, T_{acc} for the Xeon Phi and T_{host} for the host CPU. During any type of communications (host-only or PCI), both host and Xeon Phi are assumed to be idle since communication/computation overlap is not considered in this work.

Different amounts of idle time may indicate varying benefits of CPU-based dynamic voltage and frequency scaling (DVFS) for each device. For example, as seen in Table 4 for Borges and the *MIC 1* configuration, the Xeon Phi idles only for 12% of the execution time, so its DVFS potential may be rather small compared with that on the host, which is 93% for the same system and configuration. Even though the power draw is significantly lower on the host, the offloaded portions of the execution are ideal places to save energy by DVFS, as the authors have previously concluded [13], along with communication phases (see, e.g., [19]).

It may be observed from Table 4, that the maximum host power draw for either system is higher when only one accelerator is used (cf. lines 5 and 6; lines 7 and 8) because with two accelerators the host works on two smaller sub-domains in parallel. Using two sub-domains requires less computational intensity than one larger sub-domain does with only one accelerator in use. Observe also, that the *MIC 1* configurations required more total power than the *MIC 2* ones. Hence, using more than one accelerator may be beneficial not only to reduce execution time but also to expect less of a power surge in compute-intensive applications.

4.3 Energy

The proposed model aims to predict the *best* configuration instance defined as the one with the lowest energy consumption. For each system and the corresponding node count (column **N**), Table 5 presents configuration instances consuming (column-set **Config Instance**) the minimum amount of energy (column **Min Energy**), which is either measured or modeled (column **Type**). Since the problem size is fixed, each configuration instance shown in Table 5 is determined by the number of Xeon Phi's used (column **# MIC**), host frequency (column **Freq**) in GHz, and the number of threads per Xeon Phi (column **Thread**). The measured energy values are averaged over five runs for a given configuration instance.

As seen in Table 5, the models are able to predict a configuration close to the one obtained experimentally. Specifically, in each case, all the configuration-instance parameters matched except that the host frequency was consistently over-predicted by 0.3 GHz throughout but on Borges, where the predicted frequency was 0.6 GHz higher. Overall, for either Borges or Bolt, the best configuration instances were those using a single node with two Xeon Phi since the entire problem of size 50 fits on a single node. For larger problems, when only are multi-node systems employed efficiently, the

Table 4: Device power-model parameters found for the Borges and Bolt systems.

Device	System	Config	Active Power P_{active} , W					Idle Power P_{idle} , W					Total Power P , W	
			P_{static}	ρ	P_{max}	T_{active}/T	R^2	P_{static}	ρ	P_{max}	T_{idle}/T	R^2	P_{static}	P_{max}
Xeon Phi	Borges	MIC 1	168.39	1.05	241.95	0.88	0.99	170.07	0.95	236.62	0.12	0.95	168.59	241.31
		MIC 2	169.91	0.93	235.06	0.84	0.97	171.24	0.79	226.58	0.16	0.80	170.12	233.70
	Bolt	MIC 1	161.69	1.26	249.96	0.87	0.99	160.51	1.28	250.18	0.13	0.99	161.54	249.99
		MIC 2	162.51	1.17	244.47	0.84	0.99	168.37	1.07	243.33	0.16	0.84	163.45	244.29
Host CPU	Borges	MIC 1	24.30	0.28	60.19	0.07	0.85	24.22	0.28	60.11	0.93	0.85	24.23	60.12
		MIC 2	13.11	0.11	27.21	0.08	0.99	13.11	0.11	27.20	0.92	0.99	13.11	27.21
	Bolt	MIC 1	30.60	0.22	73.89	0.04	0.99	29.03	0.18	64.45	0.96	0.91	29.09	64.83
		MIC 2	25.21	0.12	48.83	0.08	0.96	25.82	0.12	49.44	0.92	0.98	25.77	49.39

Table 5: Minimum measured and modeled energy for Borges and Bolt.

System	Type	N	n_{sub}	Config Instance			Min Energy (J)	MIC Comp (GFLOPs/J)	MIC Mem (MB/J)	Host Comm (MB/J)	PCI (MB/J)
				# MIC	Freq	Thread					
Borges	Measured	1	2	2	1.3	236	6436.6	0.81	0.013	0.69	16.77
	Modeled	1	2	2	1.9	236	6583.2	0.77	0.013	1.03	15.62
Bolt	Measured	1	2	2	2.3	236	8112.1	0.70	0.011	0.51	8.27
	Modeled	1	2	2	2.6	236	7986.8	0.70	0.011	1.04	8.04
	Measured	2	2	1	1.5	236	8786.1	0.34	0.005	0.22	3.59
	Modeled	2	2	1	1.8	236	8725.8	0.33	0.005	0.23	3.96
	Measured	3	3	1	1.5	236	8985.6	0.23	0.003	0.20	2.10
	Modeled	3	3	1	1.8	236	8980.4	0.22	0.003	0.19	2.32

best configuration instances will include multiple node.

To further verify the minimum energy estimations obtained for a single node N of Bolt, all the 16 available frequencies were examined for 220 and 236 thread counts and the corresponding energies measured. Their values, however, were always higher than that shown as italicized in column **Min Energy** of Table 5. It may be observed also that, for multi-node executions, one-MIC configurations are better at large MIC-thread counts because the host frequency may be significantly reduced (cf. measured 2.3 and 1.5 GHz on Bolt for two sub-domains in column n_{sub}) to compute sub-domains distributed to multiple nodes rather than shared by a single node. Considering that the total execution time is similar in these cases (cf. $T_{\text{host}} + T_{\text{comm}}$ equal to 1.12 and 1.06 seconds for *N1 MIC 2* and *N2 MIC 1*, respectively, in Table 3), it may be inferred that an instance of *N2 MIC 1* is more energy efficient, as confirmed in column **Min Energy**.

The number of floating-point operations per joule as well as bytes per joule are typically used to assess architecture performance with respect to energy consumption (see [2, 3]). These metrics are modeled using the parameters from Tables 3 and 4 using Eq. (12) and are provided in columns 9–12 of Table 5, such that columns **MIC Comp**, **MIC Mem**, **Host Comm**, and **PCI** correspond to MIC-only computations and memory accesses, data movement in host communications and PCI transfers, respectively. The modeled values in each column were calculated from Eq. (8) as follows:

MIC Comp	$n_{\text{acc}} \cdot W_{\text{acc}}/E_{\text{acc}}$,
MIC Mem	$n_{\text{acc}} \cdot M_{\text{acc}}/E_{\text{acc}}$,
Host Comm	$N \cdot M_{\text{comm}}/E_{\text{comm}}$,
PCI	$n_{\text{acc}} \cdot M_{\text{pci}}/E_{\text{pci}}$.

5. CONCLUSION

To promote software-hardware co-design and co-development, this work proposes a model for predicting energy and power consumption in applications offloaded to Intel Xeon Phi accelerators in single- and multi-node architectures. The model uses widely-used performance metrics, such as data movement and computational throughput, to estimate execution time and relies on static and dynamic power estimates to calculate the total power draw on each participating device. In particular, the proposed model may be used to access the power-capping affects for a given application.

The model has been preliminary validated for the CoMD proxy application, and the relative error between the modeled and measured energy was generally within 5%, and always no more than 10%. By analyzing the maximum power draw and execution time predicted by the model, it has been determined that using hardware configuration featuring more than one accelerator in compute-intensive applications, such as CoMD, is beneficial both in terms of time-to-solution and reduction of the maximum power consumption for single-node configurations at this problem size. For multi-node configurations, one Xeon Phi per node is suggested as using more than two wastes energy.

The proposed model also enables users to predict the *best* configuration instance as to its lowest energy consumption. It has been shown here that this prediction matches well the *best* configuration as obtained from actual computation. The model is also capable of estimating the potential gains for DVFS by providing a ratio of execution time spent idle or active by a particular device, and by tweaking frequency, estimates of power draw under DVFS may be predicted.

In addition, the proposed model may be of value because it provides a means of considering an application, such as mini-application with well-instrumented source code, itself for algorithm- and architecture-based optimizations; thereby, expanding the rôle of high-performance benchmarks, which

as a rule, stress-test only a particular hardware subsystem and do not take into account energy consumption. For example, given an accurate application-workload estimate and the proposed model, mini-application-developers will be able to predict the maximum energy benefits when some optimization is made, such as increasing vectorization intensity or reducing power draw using DVFS.

The authors continued efforts include applying the model to more application types, such as memory-intensive ones, to expand the model validation. Additionally, the authors plan to augment the model with native and symmetric accelerator execution modes, to calibrate the model parameters by considering a variety of heterogeneous architectures, and to extend the proposed model for predicting performance and power for larger problem sizes.

Acknowledgments

This work was supported in part by the Air Force Office of Scientific Research under the AFOSR award FA9550-12-1-0476, by the National Science Foundation grants 0904782, 1047772, 1516096, 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.

6. REFERENCES

- [1] S. Cepeda. Optimization and performance tuning for Intel Xeon Phi coprocessors, part 2: Understanding and using hardware events, 2012. <https://software.intel.com/en-us/articles/>.
- [2] 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.
- [3] J. W. Choi, D. Bedard, R. Fowler, and R. Vuduc. A roofline model of energy. In *Parallel Distributed Processing (IPDPS)*, 2013 IEEE 27th International Symposium on, pages 661–672, May 2013.
- [4] K. Choi, R. Soma, and M. Pedram. Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, Jan 2005.
- [5] M. Corden. How to compile for Intel AVX, 2012. <https://software.intel.com/en-us/articles/how-to-compile-for-intel-avx>.
- [6] DOE. Co-design, 2013. <http://science.energy.gov/ascr/research/scidac/co-design/>.
- [7] ExMatEx. CoMD proxy application, 2012. <http://www.exmatex.org/comd.html>.
- [8] R. Hayashi and S. Horiguchi. Domain decomposition scheme for parallel molecular dynamics simulation. In *High Performance Computing on the Information Superhighway, 1997. HPC Asia '97*, pages 595–600, Apr 1997.
- [9] ICL:UT. Performance application programming interface PAPI, 2015. <http://icl.cs.utk.edu/papi/>.
- [10] Intel. How to use huge pages to improve application performance on pIntel Xeon Phi coprocessor, 2012. https://software.intel.com/sites/default/files/Large_pages_mic_0.pdf.
- [11] 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.
- [12] G. Lawson, M. Sosonkina, and Y. Shen. Performance and energy evaluation of CoMD on Intel Xeon Phi co-processors. In *Proceedings of the 1st International Workshop on Hardware-Software Co-Design for High Performance Computing*, Co-HPC '14, Piscataway, NJ, USA, 2014. IEEE Press. <http://dx.doi.org/10.1109/Co-HPC.2014.12>.
- [13] G. Lawson, M. Sosonkina, and Y. Shen. Changing CPU frequency in CoMD proxy application offloaded to Intel Xeon Phi co-processors. *Procedia Computer Science*, 51(0):100 – 109, 2015. International Conference On Computational Science, ICCS 2015.
- [14] G. Lawson, M. Sosonkina, and Y. Shen. Towards modeling energy consumption of Xeon Phi. *CoRR*, abs/1505.06539, 2015. <http://arxiv.org/abs/1505.06539>.
- [15] G. Lawson, V. Sundriyal, M. Sosonkina, and Y. Shen. Experimentation procedure for offloaded mini-apps executed on cluster architectures with Xeon Phi accelerators, 2015. <http://arxiv.org/abs/1509.02135>.
- [16] B. Li, H. Chang, S. L. Song, C. Su, T. Meyer, J. Mooring, and K. Cameron. The power-performance tradeoffs of the Intel Xeon Phi on HPC applications, 2014. <http://scape.cs.vt.edu/wp-content/uploads/2014/06/lsp14-Li.pdf>.
- [17] J. Mohd-Yusof, S. Swaminarayan, and T. C. Germann. Co-design for molecular dynamics: An exascale proxy application, 2013. http://www.lanl.gov/orgs/adts/publications/science_highlights_2013/docs/Pg88_89.pdf.
- [18] 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>.
- [19] V. Sundriyal and M. Sosonkina. Analytical modeling of the CPU frequency to minimize energy consumption in parallel applications. *Submitted for publication to: Elsevier*, 2015.
- [20] 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>.