2015

Changing CPU Frequency in CoMD Proxy Application Offloaded to Intel Xeon Phi Co-Processors

Gary Lawson
Old Dominion University

Masha Sosonkina
Old Dominion University, msosonki@odu.edu

Yuzhong Shen
Old Dominion University, yshen@odu.edu

Procedia Computer Science

Volume 51, 2015, Pages 100–109

ICCS 2015 International Conference On Computational Science

Changing CPU Frequency in CoMD Proxy Application Offloaded to Intel Xeon Phi Co-processors.

Gary Lawson, Masha Sosonkina, and Yuzhong Shen

Old Dominion University, Norfolk, Virginia, U.S.A.
{ glaws003, msosonki, yshen } @odu.edu

Abstract

Obtaining exascale performance is a challenge. Although the technology of today features hardware with very high levels of concurrency, exascale performance is primarily limited by energy consumption. This limitation has lead to the use of GPUs and specialized hardware such as many integrated core (MIC) co-processors and FPGAs for computation acceleration. The Intel Xeon Phi co-processor, built upon the MIC architecture, features many low frequency, energy efficient cores. Applications, even those which do not saturate the large vector processing unit in each core, may benefit from the energy-efficient hardware and software of the Xeon Phi. This work explores the energy savings of applications which have not been optimized for the co-processor. Dynamic voltage and frequency scaling (DVFS) is often used to reduce energy consumption during portions of the execution where performance is least likely to be affected. This work investigates the impact on energy and performance when DVFS is applied to the CPU during MIC-offloaded sections (i.e., code segments to be processed on the co-processor). Experiments, conducted on the molecular dynamics proxy application CoMD, show that as much as 14% energy may be saved if two Xeon Phi's are used. When DVFS is applied to the host CPU frequency, energy savings of as high as 9% are obtained in addition to the 8% saved from reducing link-cell count.

Keywords: Molecular dynamics, CoMD, Intel Xeon Phi, Energy savings, DVFS

1 Introduction

To reach exascale performance, computing nodes are supplied with accelerators, such as GPUs and, recently, many-integrated cores (MIC) co-processors that feature very high levels of concurrency. At the same time, energy consumption has gained much attention since it is a limiting factor in attaining exascale performance. For example, the U.S. Department of Energy has set the cap of 20MW power draw for supercomputer operation [12]. Applications that exploit high degrees of parallelism and take advantage of the different hardware components available execute with improved efficiency and may result in reduced execution times and energy savings.

doi:10.1016/j.procs.2015.05.207

The Intel MIC architecture is at the heart of the Intel Xeon Phi co-processor. The co-processor is typically composed of 50+ cores at approximately 1GHz. Each core is capable of concurrently processing four hardware threads [7]. Threads may be mapped to cores through the *affinity* environment variable [6]. Each core is also equipped with a 512-bit vector processing unit (VPU) capable of processing up to 8 or 16 operations per second depending on data precision, double- and single-precision respectively.

To reduce energy consumption by the CPU, a dynamic voltage and frequency scaling (DVFS) technique is commonly used at the application runtime (see, e.g., [20]). The current generation of Intel processors provides various P-states for frequency scaling. Frequency may be specified by manipulating the model-specific registers (MSR). P-states are defined as $(f_i, \dots f_n$, where $f_i > f_j$ for $i < j)$. For the older generations of the Intel Xeon Phi, DVFS is not software-accessible. DVFS may be used on the host CPU only to reduce host-side energy consumption while computation is being performed on Xeon Phi.

Molecular dynamics applications has such a potential on the road towards exascale. They are used extensively in many scientific domains. Co-design Molecular Dynamics (CoMD) is a *proxy application* provided by the Exascale Co-Design Center for Materials in Extreme Environments (ExMatEx) [5, 16]. A proxy application retains the essential workload characteristics of the full production version and is used to explore new programming models and algorithms for emerging hardware and software capabilities [5]. ExMatEx maintains the original source code on Github [2] for CoMD and other applications, such as LULESH [1, 10].

This work investigates the impact on performance and energy when CoMD is adapted to use the Xeon Phi co-processor. The impacts of varying host-side frequency during offload executions and the maximum atom count of link cells are explored with focus on minimizing the energy-to-solution. Impacts to time-to-solution, performance, and offload timings are also discussed. The tests are performed on a single-node computing platform that offers two Xeon Phi's; hence one- and two- Xeon Phi configurations are explored. In this paper, Xeon Phi and MIC may be used interchangeably to reference the Intel Xeon Phi co-processor. The implementation used in this work has been defined in [14].

The remainder of this paper is organized as follows: Section 2 discusses the related work. Section 3 provides an overview for the Xeon Phi and Section 4 for CoMD. It also defines the execution time model relating CPU execution time, frequency, and the Xeon Phi execution time. Section 5 presents the configurations explored, computing platform used, and a discussion of the experiment conducted. The section also presents the execution time model validation discussion and additional results. Finally, Section 6 concludes this paper.

2 Related Work

Power and/or performance evaluations of the Intel Xeon Phi include [3], [4], [17], [19], [21], and [22]. In [21], the authors investigate the power and performance of auto-tuned applications, including a modified version of LULESH (v1.0), to determine the optimal energy on computing platforms which include the Intel Xeon Phi. However, LULESH was executed for the host CPU, and not for the Intel Xeon Phi. Modeling of power and performance of architectures including the Xeon Phi was conducted in [4] where extension of the roofline model was applied. The work presented in [3] measures power and performance for the Rodina and SHOC benchmarks. The authors compare executions between the multi-core, Xeon Phi, and GPU architectures and found that the best architecture was highly dependent on application workload and therefore limited by the applications tested. A performance evaluation of An instruction-level energy model for the Intel Xeon Phi was proposed in [22]. A high performance MPI library for clusters featuring the Intel Xeon Phi and Infiniband was proposed in [19] which is based on the SCIF API. An investigation of performance on molecular dynamics applications for the Intel Xeon Phi has been conducted in [17]. The authors optimized the Lennard-Jones (LJ) force kernel by improving the method for identifying atom pairs by using nearest neighbor lists. Vectorization was achieved by using Intel intrinsic functions as well as pragma directives. Research into improving vectorization by loop-blocking techniques [15], vectorization pragmas, and the general optimizations also being conducted by the ExMatEx team [5].

3 Overview of MIC Architecture

The Intel Xeon Phi supports three flexible usage models for interfacing the co-processor: it may run applications exclusively on the device (native), applications may transfer work to the device for processing via compiler directives (offload), and finally applications may treat the Xeon Phi as a separate MPI node (symmetric) [7].

Data transfer between the Xeon Phi and host is a significant bottleneck in the offload execution model for any iterative application that requires data to be returned to the host for communication. Improving the transfer rate may be accomplished by using the SCIF (Symmetric Communications Interface) which allows for remote memory access mechanisms for efficiently transferring data between node host (CPU) and devices [8]. Some applications provide a value which may control how many items each data container may hold. This limiting value effectively controls the memory footprint of the container, an important component during communications. It may be beneficial to offload memory transfers which implement the SCIF simple message passing interface for communicating across the PCIe bus to the Xeon Phi device. Data transfers over the PCIe bus which use a simple message passing interface are most efficient when the message size is small, about 4KB [8].

To measure the offload memory transfer time for each offload event, the `OFFLOAD_REPORT` environment variable was set to level 1 to collect offload event messages which including high precision timings. Two timings are provided per offload event message: *CPU time* and *MIC time*. CPU time represents the time required to complete the offload event measured by the CPU; it includes the time required by any memory transfers and to complete the offload event itself (i.e. data initialization, computation, etc.). MIC time represents the time required to complete the offload event measured by the Xeon Phi; it only measures the time required to complete the event. MIC time does not measure the time spent performing memory transfers. Therefore, offload memory transfer time is simply the difference in CPU and MIC time.

4 Overview of Co-Design Molecular Dynamics

CoMD is a molecular dynamics application which simulates materials where the inter-atomic potentials are short range (e.g., uncharged metallic materials) [5]. Performance is dependent upon two tasks: identification of all atom pairs within the cut-off distance and computing the force. The force computation accounts for approximately 90% of the execution time, making it an ideal candidate for execution on the Xeon Phi. Problem size is expressed as the number of atoms in the material: $4(nx \times ny \times nz)$, where nx, ny, and nz represent the number of atoms along each axis. To expand to multiple tasks, the total problem size, or domain, must be sub-divided into sub-problems (sub-domains) equal to the number of tasks. Each sub-domain is then processed by an individual MPI task.

Within each sub-domain, individual atoms are collected into containers called link cells.

Atoms are grouped by spatial locality within the material domain and provide a basic data structure used to transmit atom data between domains. To ensure that link cells do not get too large, a maximum number of atoms may be set per link cell. In this work, the maximum number of atoms per link cell has been varied to measure the difference in performance and energy. A metric for measuring execution performance is *atom rate*, which is a ratio of the average number of atoms updated per microsecond. This CoMD-specific metric allows to compare among various hardware and software configurations. Since, an atom update requires the force calculation, computation of velocity and position as well as communication with all the relevant sub-domains, the atom rate is indicative of the performance on the host and on the Xeon Phi.

A major advertised appeal of the the Xeon Phi is its ease of programming and compiling with the same native x86 Intel compilers used to compile on the host [18, 7]. No extensive modifications of the original MPI/OpenMP version of CoMD has been performed for this work to explore the possible energy savings potential "out-of-the-box". In this work, the offload model is used to target the computational portions of the force kernel in CoMD [14]. The work offloaded from the EAM force kernel to the Xeon Phi is distributed using OpenMP.

4.1 Execution Time Model

This section presents the execution time model for applications which feature offloaded code acceleration on the Intel Xeon Phi. The model may be used to determine accurately how compute- or memory-intensive an application is. This information may then be used to determine the effect MIC acceleration had with respect to the host and the effect of DVFS during MIC offloads with respect to MIC executions without DVFS. Execution time T for an application that offloads computation to the Xeon Phi may be defined as $T = t_{\text{host}} + t_{\text{mic}}$, where the execution time of the host and offloaded code sections are non-overlapping. t_{host} represents the execution time spent on the host multi-CPU and t_{mic} represents the execution time spent on the Xeon Phi. DVFS may save power during those CPU cycles that are not computationally intensive, e.g., when the CPU is stalled waiting for memory, I/O, branch mis-prediction, or reservation station stalls [20]. The host execution time t_{host} consists of the time on-chip t_{on}, when the CPU is engaged in computations, and time off-chip t_{off}, for the remainder of CPU cycles. DVFS affects only the time on-chip, which scales linearly with the change in frequency [20]. This relationship may be described as

$$t_{\text{host}} = t_{\text{on}} \frac{f_{\max}}{f_i} + t_{\text{off}} \ , \tag{1}$$

where $f_{\max}$ is the maximum allowable frequency and $f_i, \ \ i > 0$ is any lower available frequency level. Substituting t_{host} to solve for T yields

$$T = t_{\text{on}} \frac{f_{\max}}{f_i} + t_{\text{off}} + t_{\text{mic}} \ . \tag{2}$$

Equation (2) provides the most basic relationship between total execution time, CPU (host) frequency, and times spent on the host and MIC. It may be used to determine the energy-saving potential for a hybrid CPU-MIC application, indicated by the ratio of t_{off} to t_{on}.

5 Experiments

The experiments conducted in this work aim to compare the multicore execution of CoMD, referred to as *Host*, and the CPU-MIC execution of CoMD, referred to as *MIC*. Each execution is run using one or two MPI tasks, hence: *Host 1*, *Host 2*, *MIC 1*, *MIC 2*. When DVFS was applied to the host during MIC offloads, the corresponding tests are referred to as *MIC 1 DVFS* and *MIC 2 DVFS*. For the MIC executions, the number of MPI tasks is equivalent to the number of MIC devices employed, such that each MPI task is assigned a Xeon Phi for its portion of the computations. In each iteration of the EAM force kernel, data must be transmitted between the Xeon Phi and host four times for three offload events. In the initialization phase of CoMD, one offload to Xeon Phi is required to instantiate and allocate its memory. All static data, such as interpolation tables, are transmitted to the device at this time. The memory transfer times have been measured as discussed in Section 4.

Experiments were performed on the "Borges" computing platform at Old Dominion University. The Borges system contains two Intel Xeon E5-2650 8-core processors with HyperThreading technology, each core runs at 2GHz (2.8GHz Turbo), has 20MB of L3 cache, and 64GB total of RAM. The Intel Xeon E5-2650 processor supports 10 P-states and the frequencies range from 1.2 GHz to 2.001 GHz. A 100MHz stepping is used for all the states except the last one, where only a 1MHz stepping is used. "Borges" contains two Intel Xeon Phi co-processors 5110P, each connecting to the host via a PCIe and having 30MB of the L2 cache, 32KB of the L1 data and instruction caches, 8GB DRAM (GDDR5), and 60 cores running at 1.05GHz. Each core contains a wide 512-bit single instruction multiple data (SIMD) vector processing unit, which implements the fused multiply-add operation; 8 double-precision or 16 single-precision data elements may be operated upon in a single execution phase.

Power measurements are collected externally via two Wattsup meters which power and monitor the Borges computing platform[1]. Data are sampled at a rate of 1Hz, which does not affect the measurements considerably since the problem size used provides sufficiently longer execution times. Although several tools exist for measuring power on the Xeon Phi itself [11, 9], they often incur a substantial overhead from measuring the device power. Wattsup offers a coarse-grained sampling solution, which, however, does not impact power measurements. The power is measured for 15 seconds before and after execution of CoMD in all the experiments. Additionally, 45 seconds of idle time is allocated in-between executions to allow the idle power draw to reach steady state. The results have been averaged over five executions of each test.

In this work, two parameters are varied: link-cell count and host CPU frequency. Conversely, the parameters that are not varied during the experiment are problem size, thread affinity, and thread count. Problem size has been set to 70 (1,372,000 atoms) because this workload provides a sufficiently long execution time (more than 30 seconds) to collect power samples. Thread affinity has been set to *compact* and granularity is set to *thread* because this affinity combination has been shown to provide the most efficient execution [13, 14]. Thread count has been set to 236 threads (out of possible 240), which are mapped to 59 cores. One core is left free from CoMD because it is occupied by the MIC operating system.

5.1 Execution-Time Model Validation

Compute- or memory-intensity of an application may measured by the ratio t_{off}-to-t_{on} from Eq. (1). A ratio which is very close to 0 represents a computationally intensive application; similar, a value greater than 1 represents a memory intensive application. Here CoMD is executed on "Borges" for each configuration at different operating frequencies such that linear regression may be used to solve Eq. (1) for t_{on} and t_{off}. In all of the "one-MPI" configurations (Table 1), t_{off} is about the same while it is much smaller for those with two MPI tasks because,

[1]Wattsup meter (`https://www.wattsupmeters.com`) records the *total* power for the computing system to which it is connected. Two meters are used here because the system has two power outlets.

Configuration	t_{on}	t_{off}	$t_{\text{off}}/t_{\text{on}}$	R^2
Host 1	97.139	3.769	0.039	0.992
Host 2	94.965	1.019	0.011	0.973
MIC 1	6.990	2.630	0.376	0.995
MIC 2	4.820	1.293	0.268	0.965
MIC 1 DVFS	7.051	2.635	0.374	0.996
MIC 2 DVFS	4.460	1.680	0.377	0.985

Table 1: Execution times on host (as defined in Eq. (1)) in seconds and the associated goodness-of-fit metric R^2 for all explored configurations. The problem size is 1,372,000 atoms, affinity is *compact*, thread count is 236, and link cell count is 16.

in the latter, certain data sets are treated in a two-way parallel fashion. For the *MIC* executions, t_{on} is much smaller than *Host* because the CPU (host) counts the Xeon Phi computations as its own time off-chip. It is for this reason that the DVFS applied during Xeon Phi offloads may benefit energy without affecting the performance.

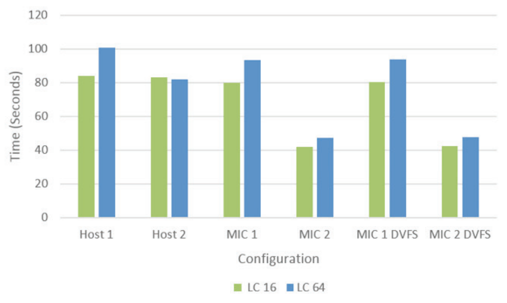
Notice that CoMD remains rather compute-intensive (t_{off}-to-t_{on} ratio is less than one) even when the force kernel is computed on the accelerator (rows *MIC* in Table 1). In particular, for all the *MIC* configurations, this ratio is approximately 0.3, which indicates that the computations remaining on the host, such as position and velocity updates, are also compute-bound. Offloading the update sections does not benefit the MIC implementation because additional atom data would need to be transmitted to the Xeon Phi each iteration. The extra memory transfer per iteration is comparable to the computation time to perform the update. Therefore, the velocity and position updates are computed by the host.

The *MIC 2* tests with and without DVFS have rather high variation in the t_{off}-to-t_{on} ratios, which is 0.1 higher for *MIC 2 DVFS* whereas these ratios are almost the same in the *MIC 1* tests. This observation may be explained by the fact that, in the *MIC 2 DVFS* case, DVFS is applied to the entire node by only one (first) MPI task. However, no synchronization is algorithmicly necessary between the MPI tasks. Hence, it is possible for the DVFS to be applied while one of the MPI tasks is still executing a host CPU portion, outside of the offloading segment. It has been verified experimentally that the host execution time t_{host} in the *MIC 2 DVFS* case was greater than that in the *MIC 2* case for several frequency levels, while t_{mic} remained constant independently of host DVFS.

5.2 Experimental Results

The impact on performance from varying link-cell count is investigated. Link-cell size has been set to either the default value of 64 atoms or to 16 atoms, which is the smallest size available for the problem size selected and, thus, may lead to more prominent differences in the performance. The highest CPU frequency $f_{\max}$ has been applied and remains constant throughout each execution, except in the *MIC DVFS* configurations. For *MIC DVFS*, CPU frequency is set to its lowest value of 1.2 GHz during offloads to the MIC(s), and restored to its highest value at the end of the offload section.

Link-cell. The results presented in Fig. 1 compare the performance of the two link-cell counts (LC of 16 and 64) in each execution configuration. In particular, Fig. 1(left) presents the total execution time while Fig. 1(right) shows the offload memory transfer time for the hybrid configurations with four main offload events: `Initialization` and `Loop 1, 2, 3` as noted in



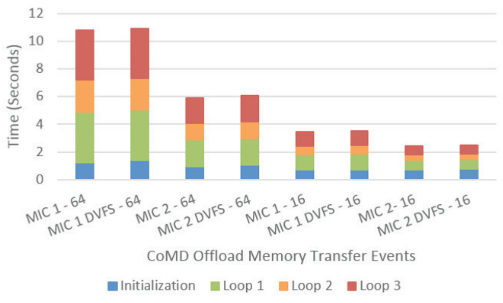
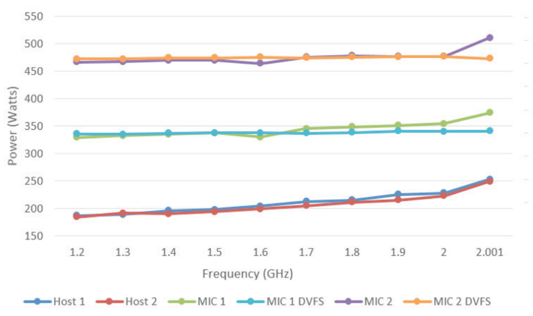


Figure 1: Total execution time (left) and total memory transfer time per offload event (right) for two link-cell counts, 16 and 64.



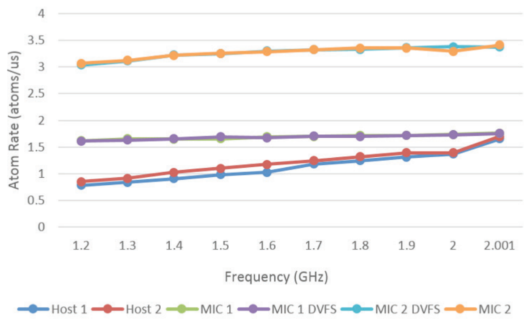
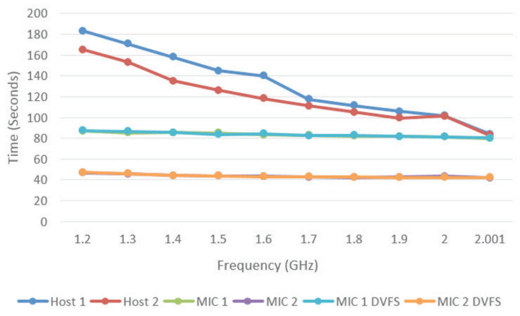


Figure 2: Average power (left) and atom rate (right) for different frequency levels with and without DVFS for link-cell count of 16. (The curves *MIC DVFS* and *MIC* partially overlap for one and two MICs, respectively.)

Section 5. The smaller LC of 16 reduces execution time for each configuration except *Host 2*, as seen in Fig. 1(left). *Host 2* does not benefit from the reduced link cell count because the link cell was able to fit into cache in the two MPI task version. *MIC 2* benefits from the reduced link cell count because its cache is much smaller than the hosts. In general, the cache utilization increases as a result of the reduced memory footprint of the link-cell container for LC of 16. For the *MIC* configurations, this observation may be particularly noticeable: Figure 1(right) shows that the memory transfer time decreases significantly with the reduced maximum atom count per link cell. Specifically, the total transfer time decreases 2.75 times (from eleven to four seconds) for *MIC 1* and two times (from six to three seconds) for *MIC 2*.

Frequency Scaling. Figures 2 and 3 provide the average power, atom rate, execution time, and energy plots for the experiments when frequency on the host is scaled from the lowest frequency to the highest frequency. The link-cell count is fixed as 16 in the plots.

Figure 2(left) presents the average power during the execution of CoMD derived from Wattsup measurements for each test configuration. For the *MIC 2* configurations, the curves show the actual power measurements. For the *MIC 1* and *Host* configurations, the curves represent the measured power values adjusted to exclude the power draw of unused Xeon Phi devices, which could not be removed from the compute node in the experiments, as follows. The



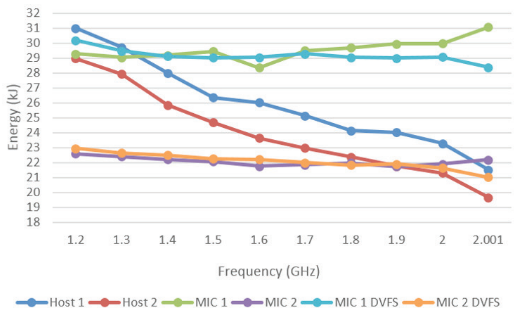


Figure 3: Total execution time (left) and energy consumed (right) for different frequencies with and without DVFS for link-cell count of 16. (The *MIC* curves partially overlap *MIC DVFS* for their respective tests.)

execution of CoMD on a single MIC causes both devices, if present, to enter an active power state. From the Xeon Phi datasheet, the active power state draws about 115W [7], therefore this number was subtracted from the actual power measurements and the resulting value shown on the *MIC 1* curves in Fig. 2(left). For the *Host* executions, both Xeon Phi's are in an idle state, which draws about 45W [7], therefore this number was subtracted from the actual power measurements and the resulting value shown in the *Host* curves in Fig. 2(left). Note that, power draw has not been measured on the Xeon Phi devices directly because existing software tools, such as *micsmc*, incur excessive overheads as was shown in authors' earlier work [13].

Figure 2(right) presents the *atom rate* metric of the performance, defined as number of atoms processed per microsecond. For lower frequencies, the atom rate is smaller because execution on the host longer. The atom rate is also indicative of the MIC performance. Hence, the rate is the largest for *MIC 2* cases, which also exhibit the best execution time as shown in Fig. 3(left). For the *MIC 2* tests, the atom rate decreases faster than that for the *MIC 1* ones because the executions of the *MIC 2* configuration are much shorter (cf. Fig. 2(right) and Fig. 3(left)). Hence, host operates at a lower frequency longer, which negatively affects the performance.

In Fig. 3(left), the total execution time is plotted against the available frequency levels. The *Host* executions are significantly impacted by frequency, as expected, due to the computationally intensive requirements of the CoMD application. The *MIC* executions are not impacted by DVFS as significantly as the *Host* because the host frequency affects only the host-side computations and not the ones offloaded to MIC.

Figure 3(right) presents the total energy consumed for the various configurations under different operating frequencies. The *MIC 1* and *Host* values have been adjusted to remove the unused Xeon Phi power draw from the measurements as explained for Fig. 2(left). The *MIC 2* tests consumed the least energy (with and without DVFS) for almost all the frequencies, except for $f_{\max}$, when *Host 2*, adjusted for unused MICs, shows the best result. *MIC 1* does not compare to *MIC 2* or to the *Host* executions, especially at the higher frequencies. Hence, further optimizations are required to make the *MIC 1* implementation energy efficient.

Energy savings for the *MIC 2* executions were measured when varying link-cell count and when applying DVFS. When link-cell count was reduced from 64 to 16, *MIC 2* saved 8.3% and *MIC 2 DVFS* saved 14.3% energy. The savings are the result of reduced host–MIC memory transfer in the cased of 16 link-cell count and of reduced cache misses due to the decreased memory footprint. When DVFS is applied and link-cell count is 16, *MIC 2 DVFS* saved 9% in energy compared with the *MIC 2* execution without DVFS while still consuming more energy than the *Host 2* execution at the highest frequency, when the *MIC 2* executions were not fast enough to compensate for the additional power consumption of the MICs (cf. Fig. 3(left) and Fig. 2(right)).

6 Conclusion

In this work, the molecular dynamics application CoMD was used experimentally to determine the validity of a simple execution time model for applications executed using the Intel Xeon Phi co-processor in offload mode. The experiments show that execution time on the Xeon Phi may be decomposed into two partitions: time to execute on the host and Xeon Phi. The developed model showed that, even under *MIC* execution, CoMD remains mostly compute intensive on the host. The *MIC 2* executions resulted in the lowest energy consumed among all the configurations for the frequencies below the maximum one, at which *Host 2* consumed the lowest energy. The *MIC 1* executions consumed the most energy of all configurations tested. More effort is required in the *MIC 1* implementation to achieve better performance. Furthermore, it was determined that, when a single MIC is specified for execution in a multi-MIC node, the other MICs are elevated to an active state, consuming extra power.

When the link-cell count was reduced from 64 to 16, energy savings as high as 14% and 8% were observed for the *MIC 2 DVFS* and *MIC 2* configurations, respectively. When DVFS was applied, 9% of energy was saved, in addition to the 8% saved from reducing link-cell count for the *MIC 2* configuration. Applying DVFS during offload sections did not affect significantly the performance: The execution time only did increase by 1%.

Future work includes fine-tuning of the execution time model to differentiate single- and multiple-MIC offloading, developing the power model, and combining the two to predict energy consumption for memory- and compute-intensive applications executed on different hardware configurations. Power measurement techniques on MIC are to be investigated further to provide the instantaneous power consumption on the MIC without incurring much overhead. Analysis of the time model will be performed with the definition of the energy model.

7 Acknowledgments

This work was supported in part by Ames Laboratory and Iowa State University under the contract DE-AC02-07CH11358 with the U.S. Department of Energy, This work was supported in part by the Air Force Office of Scientific Research under the AFOSR award FA9550-12-1-0476, and by the National Science Foundation grants NSF/OCI—0941434, 0904782, 1047772. This work has supported in part by the comments made by the reviewers.

References

[1] Hydrodynamics Challenge Problem, Lawrence Livermore National Laboratory. Technical Report LLNL-TR-490254.

[2] Exmatex, 2015. https://github.com/exmatex.

[3] Bo Li and Hung-Ching Chang and Shuaiwen Leon Song and Chun-Yi Su and Timmy Meyer and John Mooring and Kirk Cameron. The power-performance tradeoffs of the Intel Xeon Phi on HPC applications, 2014. http://scape.cs.vt.edu/wp-content/uploads/2014/06/lspp14-Li.pdf.

[4] Jee Choi, Marat Mukhan, Xing Liu, and Richard Vudue. 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] ExMatEx. CoMD proxy application, 2012. http://www.exmatex.org/comd.html.

[6] Ronald Green. OpenMP thread affinity control, 2012. https://software.intel.com/.

[7] Intel Corporation. Developer zone, 2013. https://www.software.intel.com.

[8] Intel Corporation. Intel manycore platform software stack (MPSS), 2014. https://software.intel.com/en-us/articles/intel-manycore-platform-software-stack-mpss.

[9] Intel Forums. how to measure energy consumption on the coprocessor?, 2013. https://software.intel.com/en-us/forums/topic/488782?language=en.

[10] Ian Karlin, Jeff Keasler, and Rob Neely. Lulesh 2.0 updates and changes. Technical Report LLNL-TR-641973, August 2013.

[11] Taylor Kidd. Intel Xeon Phi coprocessor power management configuration: Using the micsmc command-line interface, 2014. https://software.intel.com/.

[12] Dimitri Kusnezov, Steve Binkley, Bill Harrod, and Bob Meisner. Doe exascale initiative, 2013. http://www.industry-academia.org/download/20130913-SEAB-DOE-Exascale-Initiative.pdf.

[13] Gary Lawson, Masha Sosonkina, and Yuzhong Shen. 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.

[14] Gary Lawson, Masha Sosonkina, and Yuzhong 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.

[15] David Mackay. Optimization and performance tuning for Intel Xeon Phi coprocessors, part 1: Optimization essentials, 2012. https://software.intel.com/.

[16] Jamaludin Mohd-Yusof, Sriram Swaminarayan, and Timothy C. Germann. Co-design for molecular dynamics: An exascale proxy application, 2013. http://www.lanl.gov/orgs/adtsc/publications/science_highlights_2013/docs/Pg88_89.pdf.

[17] S J Pennycook, C J Hughes, M Smelyanskiy, and S A Jarvis. Exploring SIMD for molecular dynamics, using intel xeon processors and intel xeon phi coprocessors. *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, pages 1085 – 1097, 2013.

[18] James Reinders. An overview of programming for Intel Xeon processors and Intel Xeon Phi coprocessors, 2013. https://software.intel.com/.

[19] Sreeram Potluri and Khaled Hamidouche and Devendar Bureddy and Dhabaleswar K (DK) Panda. Mvapich2-mic: A high performance MPI library for xeon phi clusters with infiniband, 2013.

[20] Vaibhav Sundriyal and Masha Sosonkina. Initial investigation of a scheme to use instantaneous CPU power consumption for energy savings format. In *Proceedings of the 1st International Workshop on Energy Efficient Supercomputing*, New York, NY, USA, 2013. ACM.

[21] Wei Wang, John Cavazos, and Allan Porterfield. Energy autotuning using the polyhedral approach. *4th International Workshop on Polyhedral Compilation Techniques (IMPACT 2014), in conjunction with HiPEAC'14*, 2014.

[22] Yakun Sophia Shao and David 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.