8-2014

Energy Evaluation for Applications with Different Thread Affinities on the Intel Xeon Phi

Gary Lawson
Old Dominion University, glaws003@odu.edu

Masha Sosonkina
Iowa State University, masha@scl.ameslab.gov

Yuzhong Shen
Old Dominion University, yshen@odu.edu

Energy Evaluation for Applications with Different Thread Affinities on the Intel Xeon Phi

Abstract
The Intel Xeon Phi coprocessor offers high par- allelism on energy-efficient hardware to minimize energy consumption while maintaining performance. Dynamic frequency and voltage scaling is not accessible on the Intel Xeon Phi. Hence, saving energy relies mainly on tuning application performance. One general optimization technique is thread affinity, which is an important factor in multi-core architectures. This work investigates the effects of varying thread affinity modes and reducing core utilization on energy and execution time for the NASA Advanced Supercomputing Parallel Benchmarks (NPB). Energy measurements are captured using the micsmc utility tool available on Xeon Phi. The measurements are checked against total power captured using Wattsup power meters. The results are compared to the system-default thread affinity and granularity modes. Mostly positive impacts on performance and energy are observed: When executed at the maximum thread count on all unoccupied cores, all the benchmarks but one exhibited energy savings if a specific affinity mode is set.

Keywords
Intel Xeon Phi, energy, thread affinity, NAS benchmarks

Disciplines
Systems Architecture

Energy Evaluation for Applications with Different Thread Affinities on the Intel Xeon Phi

Gary Lawson*, Masha Sosonkina*†, Yuzhong Shen*
*Department of Modeling, Simulation, and Visualization Engineering
Old Dominion University
Norfolk, VA 23529
Email: {glaws003, msosonki, yshen}@odu.edu
†Department of Computer Science
Iowa State University, Ames, IA 50011
Email: mashas@iastate.edu

***Abstract*—The Intel Xeon Phi coprocessor offers high parallelism on energy-efficient hardware to minimize energy consumption while maintaining performance. Dynamic frequency and voltage scaling is not accessible on the Intel Xeon Phi. Hence, saving energy relies mainly on tuning application performance. One general optimization technique is thread affinity, which is an important factor in multi-core architectures. This work investigates the effects of varying thread affinity modes and reducing core utilization on energy and execution time for the NASA Advanced Supercomputing Parallel Benchmarks (NPB). Energy measurements are captured using the *micsmc* utility tool available on Xeon Phi. The measurements are checked against total power captured using Wattsup power meters. The results are compared to the system-default thread affinity and granularity modes. Mostly positive impacts on performance and energy are observed: When executed at the maximum thread count on all unoccupied cores, all the benchmarks but one exhibited energy savings if a specific affinity mode is set.**

I. Introduction

The Intel Xeon Phi coprocessor utilizes the Intel Many Integrated Core (MIC) architecture to accelerate parallel computing. The device is intended for highly parallel workloads and offers a high theoretical computational performance and a high theoretical memory bandwidth, one teraFLOP at double-precision and 320 GB/s, respectively [1]. These attributes as well as the energy-efficient hardware and software of Xeon Phi make the coprocessor an appealing option for exascale computing. In the Top500 list of November 2013, two clusters incorporating Intel Xeon Phi coprocessors, Tianhe-2 and Stampede, were ranked the first and seventh, respectively [2] while, in the Green500 list of November 2013, the first Intel Xeon Phi-based cluster ranks the 37th [3].

Reaching the theoretical performance is not easy, however, as the underlying architecture of the Xeon Phi differs from the traditional Intel Xeon processor. The Xeon Phi coprocessor incorporates many small power-efficient cores as well as offers high bandwidth to the local memory. Thread affinity offers applications a general optimization strategy for improving performance on multi-core architectures [4], such as Intel Xeon Phi. Although there have been several investigations into the performance impacts of thread affinity, often overlooked is energy, which heavily impacts the road to the exascale. This work investigates both the performance and energy effects of thread affinity on the NASA Advanced Supercomputing (NAS) Parallel Benchmarks, which are compiled to run natively on the Intel Xeon Phi. Specifically, the execution time and energy consumed by the Intel Xeon Phi are evaluated under different thread affinity modes. Going beyond measuring the execution time, other performance metrics relevant for Xeon Phi are explored, such as average cycles per instruction (CPI) per thread, memory bandwidth, and vectorization intensity. The energy and approximate execution time are computed based on measurements captured using the MIC System Management and Configuration *micsmc* utility tool [5].

This paper is organized as follows: Section II provides discussions related to the Intel Xeon Phi architecture, NPB suite, performance metrics, and the energy profiling method used here. Section III presents the experimental set-up and results. Section IV concludes and outlines the future research.

A. Related Work

Benchmarking applications on the Intel Xeon Phi has been well documented, focusing mainly on the performance. Krishnaiyer *et al.* [6] benchmark the performance of various applications on the Intel Xeon Phi with respect to data prefetching and non-temporal streaming store instructions. Fang *et al.* [7] benchmark the performance of various optimization techniques with a focus on guiding kernel design. Heinecke *et al.* [8] implemented the Linpack benchmark on single- and multi-node systems and achieved 80–90% peak efficiency. Pennycook *et al.* [9] explored performance of SIMD instructions for molecular dynamics applications. Saule *et al.* [10] investigate performance with respect to sparse matrix multiplication kernels. For the NAS parallel benchmarks, Ramachandran *et al.* [11] undertook a detailed analysis of the performance pitfalls and implemented several optimization techniques to increase performance.

The work presented here looks at the performance of the NAS benchmarks with their energy consumption as the primary focus with respect to thread assignment to cores. An instruction-level energy model has been used by Shao and Brooks [12] with the Linpack benchmark suite to observe increases in energy efficiency as high as 10%. Choi *et al.* [13] conducted a microbenchmarking study and found that the Intel Xeon Phi offers energy benefits to highly irregular data pro-

cessing workloads. The Xeon Phi requires one magnitude less energy per access during random memory access operations.

II. Background

A. Thread Placement in Intel Xeon Phi

The Xeon Phi coprocessor is typically composed of 50+ cores at approximately 1GHz. Each core is capable of concurrently processing four hardware threads [1]. It is not possible to execute instructions from a single thread in back-to-back cycles; therefore, a minimum of two hardware threads per core is suggested. Threads may be mapped to cores through the *affinity* environment variable that governs six possible modes: *balanced*, *compact*, *scatter*, *none*, *disabled*, and *explicit* [14], [15]. The "disabled-affinity" setting provides no thread affinity interface, while the "explicit-affinity" setting allows the user to manually assign each thread to every core. These two affinity settings are not considered in this work however. The *balanced* affinity mode evenly distributes threads among the cores. This mode attempts to use all the available cores while keeping the thread neighboring logical IDs physically close to one another. The *scatter* affinity also evenly distributes threads among the cores but it does so in a round-robin fashion. Hence, threads with the adjacent IDs are not guaranteed to be physically adjacent. The *compact* affinity distributes threads by assigning the maximum number of threads (which is 4) to a core before assigning threads to another core. This mode keeps the threads grouped tightly together and uses fewer cores than other affinity modes unless the maximum thread count is set, at which point the *balanced* and *compact* thread mappings are identical. When the thread affinity is not specified explicitly, the system-default setting is *none*.

Thread granularity specifies the way various affinity modes are applied. There are three levels of granularity: *fine*, *thread*, and *core*. The *fine* and *thread* levels are similar in that they bind threads to a single context when the thread is assigned to a core. The *core* granularity binds threads to a core, such that the threads may float within the context of the physical core [14], [16]. Using the former may improve reproducibility of the results and avoid the overhead from thread context switch. The system-default granularity setting is *core*.

B. NAS Parallel Benchmarks

The NAS Parallel Benchmark suite [17] provides a set of applications with varying communication and computation schemes. Each benchmark offers a range of class sizes. NPB categorizes the classes as A, B, C, D, or E in the ascending order of their problem sizes. To reflect major computational kernels in real-world scientific and engineering applications, this work focuses on the EP, CG, FT, IS, and MG benchmarks. Specifically, EP provides an estimate of the upper achievable limits for floating point performance, without communication overhead; CG implements unstructured matrix vector multiplications and dot products; FT provides a rigorous test for long-distance communication performance; IS tests integer computation speed; and MG tests both short and long distance data communication.

Table I presents execution time required to complete each benchmark for the various classes at 59 and 236 threads. Only the EP benchmark was able to execute class sizes larger than C on the Xeon Phi. This is due to the small memory footprint required for EP. Therefore, each benchmark is executed with class C problem sizes: 8,589,934,592 random numbers generated for EP, the matrix size of 150,000 for CG, and grid size of $512 \times 512 \times 512$ for FT, IS, and MG.

C. Energy

Energy consumption is calculated by the approximation of power timeslices provided by the Intel utility tool, *micsmc*. From the command line, *micsmc* can output a wealth of data including frequency, power, temperature, memory usage, and CPU utilization per core [18]. However, as additional types of data are requested, the delay between calls increase. Hence, to capture the energy readings in the smallest timeslice available, only frequency and power data are recorded (both are printed when the same input parameter is given). The *micsmc* tool measured and reported every 22–28ms, which is sufficient for a detailed evaluation. Data are collected from each unit on the device via the performance monitoring unit (PMU) [1], [19]. Each core contains an independently programmable PMU, which supports four hardware threads, two hardware counters per thread, and 40-bit precision per hardware counter.

Dynamic Voltage and Frequency Scaling (DVFS) involves changing the voltage and frequency levels of the processor to reduce or increase power, which may be performed in application software. This technique generally requires a careful implementation to reduce potentially severe performance penalties. When applied judiciously, however, it may yield as much energy savings as 14% with a modest performance loss of 2% on certain NAS parallel benchmarks as was shown by the authors in [20]. Unfortunately, Xeon Phi does not allow user-controlled DVFS. The hardware performance levels (P-states) are selected and set through the coprocessor OS kernel. P-states may change depending on the thermal or power readings. A new P-state is selected by the OS upon crossing a high thermal threshold or approaching one of the upper power limits [1]. Additionally, the Intel Xeon Phi may perform DVFS selectively for inactive cores. Hence, varying power consumption is explored here by varying the active core count under the *compact* affinity mode, in which it is possible to leave some cores idle when mapping threads numbered greater than the total core count. These idle cores may allow the device to save power with a certain performance loss.

To demonstrate the accuracy of the data obtained by *micsmc*, its output is compared to that collected by the Wattsup power meters[1], having a sampling rate of 1 Hz, which does not affect the measurements considerably since the benchmark used here has a rather large execution time. Figure 1 presents the power profiles, averaged over three runs, for *micsmc* alone (curve `Micsmc`), for the sum of the two Wattsup power (curve `Wattsup`) meters, and for the Wattsup power meters during `Micsmc` (curve `Micsmc-Wattsup`) when the CG benchmark is executed at 180 threads in the *compact* affinity mode. From Fig. 1, it is clear that *micsmc* and Wattsup are reporting power traces of similar patterns, with fluctuations appearing at the same time points. However, from the difference between the "pure" `Wattsup` and `Micsmc-Wattsup` power profiles,

[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 used has two power outlets.

TABLE I. EXECUTION TIME (SECONDS) OF 56 THREADS AND 236 THREADS FOR VARIOUS BENCHMARK PROBLEM-CLASS SIZES.

	59 Threads					236 Threads				
NPB	A	B	C	D	E	A	B	C	D	E
EP	1.97	7.83	28.94	455.84	—	1.05	4.18	16.44	268.69	4229.20
CG	0.49	49.61	206.82	†	†	0.21	12.98	53.31	†	†
FT	0.93	12.54	52.50	†	†	0.71	10.57	45.15	†	†
IS	0.17	0.52	3.26	†	NA	0.39	0.59	1.99	†	NA
MG	0.26	1.41	8.28	†	†	0.24	1.19	8.14	†	†

— - Did not test NA - Class size not available † - Out of memory

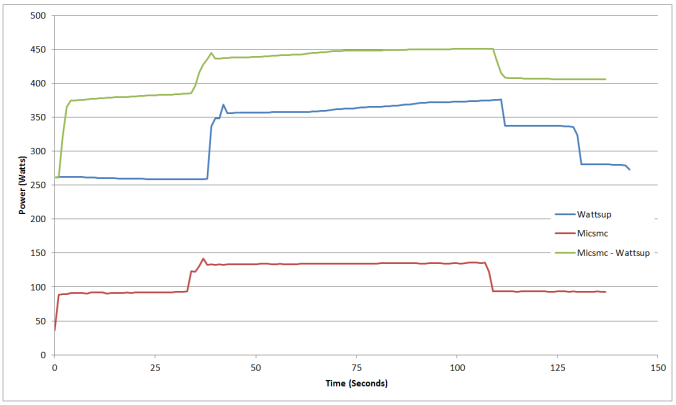

Fig. 1. The power profiles as obtained by Wattsup and *micsmc* for the CG benchmark with the *compact* affinity mode at 180 threads.

it may be observed that *micsmc* incurs a substantial overhead, which is caused by the fact that *micsmc* always puts the device cores into a rather high energy state (around 120 watts, as seen between the 5th and 30th second in Fig. 1), even if they are idle, when polling each core for the power data. In particular, as depicted in Fig. 1, the CG execution begins around 30 seconds into the measurement; the remainder is idle power draw. For `Wattsup`, this is the true idle power draw of the system since Wattsup is a non-intrusive monitor. However, for `Micsmc-Wattsup`, the idle power draw is observed to increase over 100 watts, which corresponds to the time when *micmsc* began polling the device.

III. EVALUATION

A. Experimental Setup

The experiments were conducted on the Borges computing platform at Old Dominion University. The system contains two Intel Xeon E5-2650 8-core processors with Hyper-Threading technology, each core at 2GHz (2.8GHz Turbo), 20MB of L3 cache, and 65GB total RAM. The system also contains two Intel Xeon Phi coprocessors 5110P. Each coprocessor connects to the host via a PCIe connection and has 30MB total of the L2 cache, 32KB of the L1 data and instruction caches, 8GB DRAM (GDDR5) 60 cores, each at 1.05GHz. Each coprocessor core contains a wide 512-bit single instruction multiple data (SIMD) vector processing unit (VPU), which implements fused multiply-add operations; 8 double-precision or 16 single-precision data elements may be operated upon in a single execution phase.

The execution time is obtained from the NPB output, and Intel VTune Amplifier XE software samples the hardware event counters. The Intel Xeon Phi power was measured with *micsmc* to compare the energy of different affinity modes used on the device only.

B. Results

1) Execution Time: Figure 2 presents the average normalized execution time observed for the EP, FT, IS, and MG benchmarks with different affinity modes and *thread* granularity. The execution time shown is an average over four runs, which is normalized against the default test. (Thus, values less than one are an improvement over the default case.) Recall that the default case uses the *core* granularity and *none* affinity. Figure 2 provides several examples of applications that perform better with a specified affinity and granularity. IS and MG under the *compact* affinity perform exceptionally well after 180 threads when compared to the default test. However, it also provides several examples of applications which underperform when fewer cores are utilized: All four benchmarks perform worse (value less than 1) while using the *compact* affinity for thread counts fewer than 180. In EP and FT, the *balanced*, *scatter*, and *none* affinities chaotically shift between being more or less efficient than the default. However, after 180 threads, each affinity observes a boost in performance, except for the *scatter* one. Thus, NPB perform best when each Intel Xeon Phi core is running least three threads per otherwise empty core, which is in agreement with the general optimization guideline of running four threads per core. After 236 threads, the performance of each affinity dramatically increases or decreases, depending on how well the threads are able to compete with the OS for resources, which occupy one core. Generally, the *scatter* and *none* affinities perform better at 240 threads.

2) Performance Metrics: To gain further insights into the performance of each benchmark, this work considers certain performance metrics as obtained from the hardware event counters provided by the Intel VTune Amplifier XE 2013 [21]. In particular, these metrics are average CPI per thread, vectorization intensity (VI) and memory bandwidth. The average CPI per thread (CPI) is the average number of CPU cycles required to retire an instruction, averaged per thread. It can be used to detect latency in the system which affects the applications execution. VI provides an indicator for how well the entire code maps to the VPU. For double-precision and single-precision operations, the VI ideally should be close to 8 and 16, respectively. The memory bandwidth metric describes the average streaming memory bandwidth during execution.

After running the Intel VTune Amplifier, it was observed that VI did not vary when varying affinity for any of the benchmarks. The EP, CG, FT, IS, and MG benchmarks exhibited a VI of 3.2, 3.0, 8.0, 1.33, and 6.4, respectively. This is attributed to the NPB benchmarks applications using 64-bit precision and the VPU length is 512-bit. FT and MG are

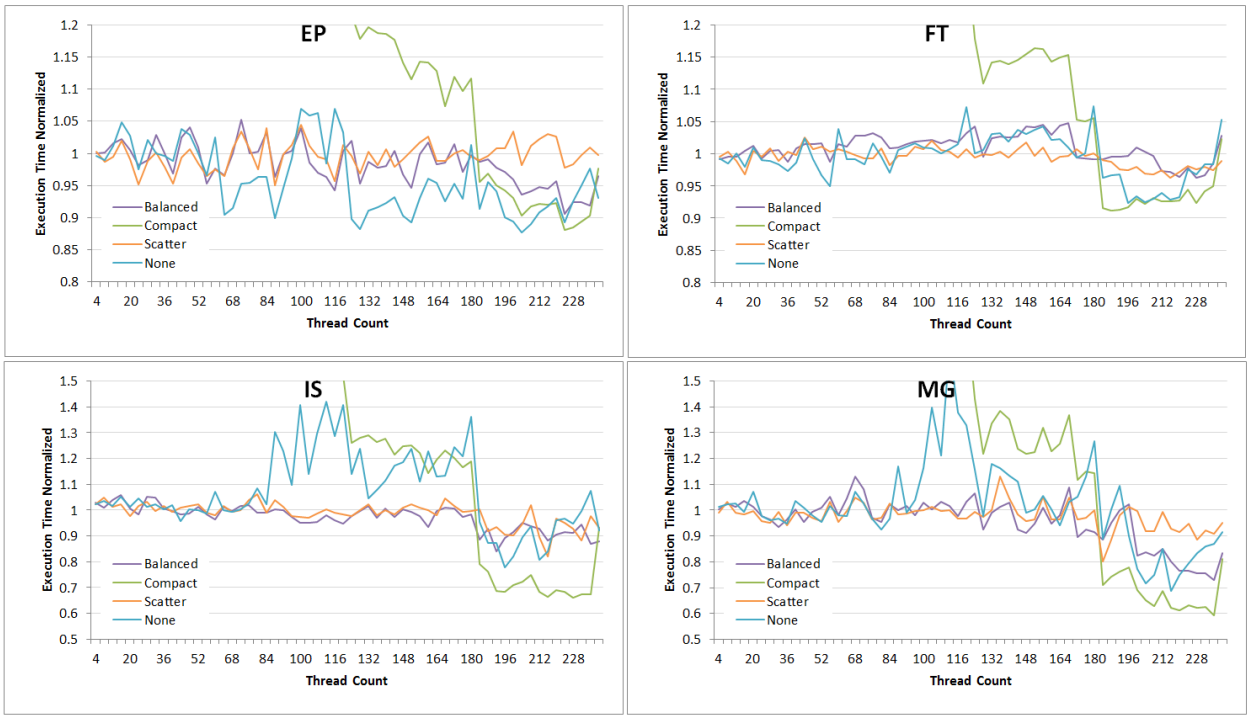

Fig. 2. Normalized execution time for the EP, FT, IS, and MG benchmarks with different affinity modes and the *thread* granularity. Each value is the average of four runs for each benchmark, and is normalized for each affinity against the default test, which has the affinity *none* and granularity: *core*. The data for *compact* affinity are shown starting at the number of threads greater than for the other affinities to permit good scaling of the plots.

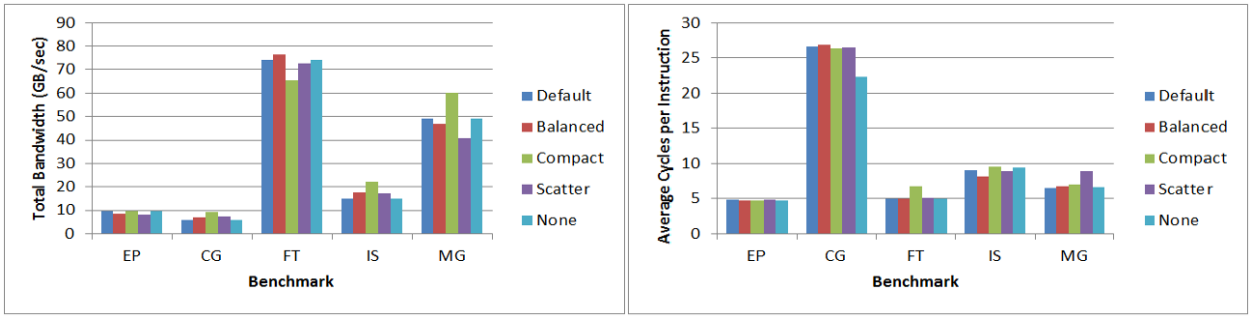

Fig. 3. Total bandwidth (left) and average CPI (right), for each benchmark and the affinity mode with the *thread* granularity and the thread count from Table II.

observed with high vectorization utilization, suggesting few or no data dependencies within the vectorized loops. The total bandwidth (Fig. 3, left) is much lower than the peak value of 320 GB/s or even than the observable 140 GB/s, as presented in [22]. The measured CPI (Fig. 3, right) is much closer to the expected value of four, except for CG which incurs around 26 cycles per instruction on average. In general, FT, with over 70 GB/s and a low CPI, utilizes the resources of the Xeon Phi well compared to the other benchmarks. MG features moderate bandwidth, acceptable CPI, and a high VI.

3) Energy: For the executions shown in Fig. 2, Fig. 4 presents the average normalized energy observed for the EP, FT, IS, and MG benchmarks, which is averaged and normalized as in Fig. 2. From Fig. 4, it is seen that, for the thread counts greater than 180, specifying affinity and granularity can decrease energy consumption. It is not until after 180 threads that the energy savings of using fewer cores outweigh the loss of the performance. Consecutively, EP results in almost 8% of energy savings with the *compact* affinity, FT results in 6%, IS results in 13%, and MG results in almost 23% energy savings.

For the CG benchmark (Fig. 5, left), energy savings may be observed already for fewer cores under the *compact* affinity mode, starting at 28 threads even though execution time is larger than the default test (normalized value greater than one in Fig. 5, right) and than other affinity modes until 180 threads. Since CG is the most memory-accessing application among the ones considered here as was shown in [23], this situation may be explained by the findings in [13] that Intel Xeon Phi uses less energy for memory accesses. Under the *compact* affinity mode, the neighboring threads, which are more likely to access memory simultaneously, are located in the same core. Hence, an entire core may be considered as memory-accessing to enjoy the lower energy usage, while the execution time is high because of thread contention.

After 180 threads, which was the same threshold observed

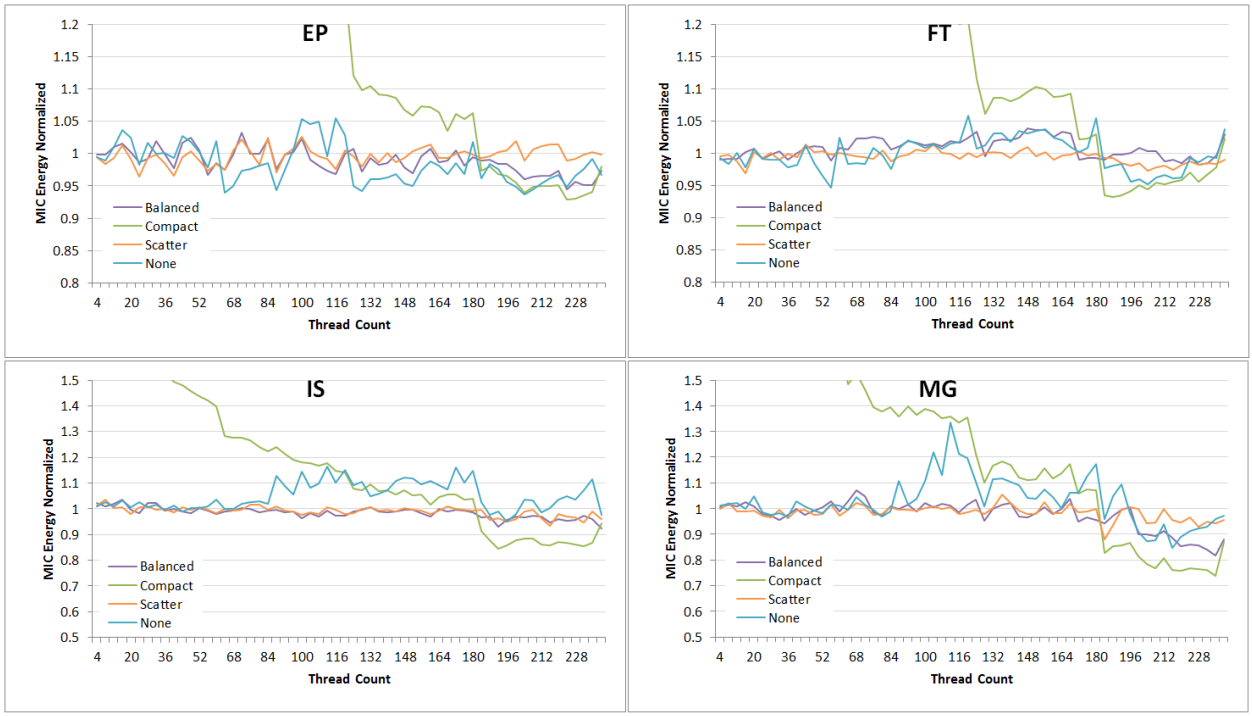

Fig. 4. Normalized energy for the EP, FT, IS, and MG benchmarks with different affinity modes and the *thread* granularity. Each value is the average of four runs for each benchmark, and is normalized for each affinity against the default test, which has the affinity *none* and granularity *core*. The data for *compact* affinity are shown starting at the number of threads greater than for the other affinities to permit good scaling of the plots.

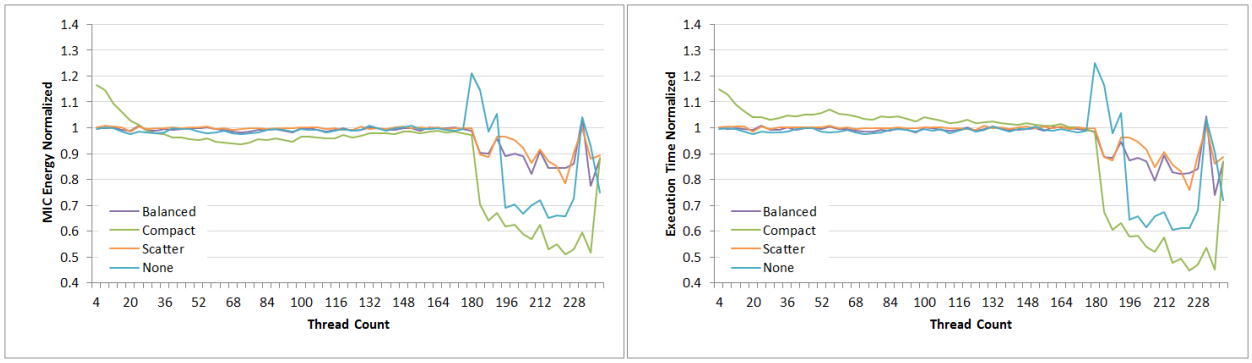

Fig. 5. Normalized execution time and energy for the CG benchmark with different affinity modes and the *thread* granularity. Each value is the average of four runs for each benchmark, and is normalized for each affinity against the default test, which has the affinity *none* and granularity *core*.

in every other benchmark, the CG performance increases drastically leading to substantial energy savings. At the 180-thread count, better thread load balancing takes place as threads are evenly distributed to cores, at three per core. Good load balancing occurs at other thread counts as well, such as 60, 120, and 240. In particular, after 120 threads (two per core), one of the two compute cycles will gain the option to switch between threads during execution and an increase in the performance is observed. An even greater increase is observed when the fourth thread is added and the second cycle has an additional thread (after 180 threads). Such performance increases, which lead to energy savings, are noticeable for all the user-specified affinity modes with the *thread* granularity considered here.

Figures 2, 4 and 5 indicate that the DVFS was not applied during the benchmark executions since the performance and energy correlate so closely in all the cores. Further, this is supported by the output from the *micsmc* that shows that frequency did not change during any of the benchmark executions.

TABLE II. LOWEST ENERGY CONSUMED WHEN SPECIFYING THE AFFINITY MODES AS COMPARED TO THE SYSTEM-DEFAULT SETTING.

	EP	CG	FT	IS	MG
`E_Default (J)`	3,173.16	17,005.39	8,274.34	1,042.97	1,813.89
`E_Test (J)`	2,986.20	8,787.69	8,386.27	892.59	1,731.50
`Aff_Mode`	compact	compact	scatter	compact	balanced
`Thread Count`	236	236	104	232	180
`E_Saved`	+5.89%	+48.32%	-1.35%	+14.42%	+4.54%

For each benchmark, Table II provides the lowest energy (row `E_Test`) along with the affinity mode (row `Aff_Mode`) and thread count (row `Thread Count`) at which it was observed. The corresponding energy value of the default case

is in row `E_Default` and the difference in the energy consumption is in row `E_Saved`. CG features the highest energy saving, a staggering 48% with respect to the default test. The FT benchmark was the only benchmark to consume more energy as a result of specifying affinity. It may be due to the granularity difference with the default case because the energy consumed by the *none*, *scatter*, and *balanced* affinities are almost the same (within 0.1% of one another).

IV. Conclusion

This work investigates the impacts of thread affinity and granularity for applications running natively on the Intel Xeon Phi architecture. In particular, NAS parallel benchmark suite is considered with the available affinity modes and *thread* granularity and compared to its execution at the system-default case, i.e., when the affinity is *none* (not pre-defined) and the granularity is *core*. In this paper, it was shown that varying thread affinity may improve both performance and energy, which is the most apparent under the *compact* affinity tests when the number of threads is larger than three per core. The energy savings reached as high as 48% for the CG NAS benchmark. The runs with the *compact* affinity have also shown that reducing the number of active cores(i.e., using few threads) may not save energy since the performance loss is too large for large-scale applications; This work used the *micsmc* tool to show the device-only power measurements, and demonstrated its usability by the Wattsup measurements. In particular, the fluctuations in power observed by *micmsc* correlated with those of Wattsup. However, further investigation of Intel Xeon Phi power monitoring techniques is necessary since *micsmc* incurs a large overhead in power. Future directions include projecting the current findings to "mini-applications" [24] and real-world applications and to the hybrid execution mode on Intel Xeon Phi consisting of offloading from the CPU.

Acknowledgments

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.

References

[1] Intel Corporation, "Intel Xeon Phi coprocessor system software developers guide," 2013. [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/xeon-phi-coprocessor-system-software-developers-guide.pdf

[2] Top500, "Top 500 supercomputer sites," 2013. [Online]. Available: \url{http://www.top500.org/}

[3] Green500, "The Green 500," 2013. [Online]. Available: http://www.green500.org/

[4] G. Hager and G. Wellein, *Introduction to High Performance Computing for Scientists and Engineers*. CRC Press, 2011.

[5] A. Birkland, "Cornell virtual workshop," 2013. [Online]. Available: https://www.cac.cornell.edu/vw/mic/default.aspx

[6] R. Krishnaiyer, E. Kultursay, P. Chawla, S. Preis, A. Zvezdin, and H. Saito, "Compiler-based data prefetching and streaming non-temporal store generation for the Intel Xeon Phi coprocessor," in *2013 IEEE 27th International Symposium on Parallel Distributed Processing (IPDPS)*, May 2013, pp. 1575–1586.

[7] J. Fang, A. L. Varbanescu, and H. Sips, "Benchmarking Intel Xeon Phi to guide kernel design," Delft University of Technology, Netherlands, Tech. Rep. PDS-2013-005, Apr. 2013.

[8] A. Heinecke, K. Vaidyanathan, M. Smelyanskiy, A. Kobotov, R. Dubtsov, G. Henry, A. Shet, G. Chrysos, and P. Dubey, "Design and implementation of the Linpack benchmark for single and multi-node systems based on Intel Xeon Phi coprocessor," in *2013 IEEE 27th International Symposium on Parallel Distributed Processing (IPDPS)*, May 2013, pp. 126–137.

[9] S. Pennycook, C. Hughes, M. Smelyanskiy, and S. Jarvis, "Exploring simd for molecular dynamics, using Intel Xeon processors and Intel Xeon Phi coprocessors," in *2013 IEEE 27th International Symposium on Parallel Distributed Processing (IPDPS)*, May 2013, pp. 1085–1097.

[10] E. Saule, K. Kaya, and Ü. V. Çatalyürek, "Performance evaluation of sparse matrix multiplication kernels on Intel Xeon Phi," *CoRR*, vol. abs/1302.1078, 2013.

[11] A. Ramachandran, J. Vienne, R. Van Der Wijngaart, L. Koesterke, and I. Sharapov, "Performance evaluation of NAS parallel benchmarks on Intel Xeon Phi," in *2013 42nd International Conference on Parallel Processing (ICPP)*, Oct 2013, pp. 736–743.

[12] Y. Shao and D. Brooks, "Energy characterization and instruction-level energy model of Intel's Xeon Phi processor," in *2013 IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, Sept 2013, pp. 389–394.

[13] J. Choi, M. Mukhan, X. Liu, and R. 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.

[14] Intel Corporation, "Intel C++ Compiler 12.1 user and reference guides," 2013. [Online]. Available: http://software.intel.com/sites/products/documentation/studio/composer/en-us/2011Update/compiler_c/hh_goto.htm\#main/main_cover_title.htm

[15] R. Green, "OpenMP thread affinity control," 2012. [Online]. Available: https://software.intel.com/en-us/articles/openmp-thread-affinity-control

[16] M. Barth *et al.*, "Best practice guide Intel Xeon Phi v1.1," 2014. [Online]. Available: http://www.prace-ri.eu/IMG/pdf/Best-Practice-Guide-Intel-Xeon-Phi.pdf

[17] NASA, "NAS parallel benchmarks," 2013. [Online]. Available: http://www.nas.nasa.gov/publications/npb.html

[18] T. Kidd, "Intel Xeon Phi coprocessor power management configuration: Using the micsmc command-line interface," 2014. [Online]. Available: https://software.intel.com/en-us/blogs/2014/01/31/intel-xeon-phi-coprocessor-power-management-configuration-using-the-micsmc-command

[19] Intel Corporation, "Intel Xeon Phi coprocessor performance monitoring units." 2013. [Online]. Available: http://software.intel.com/sites/default/files/forum/278102/intelr-xeon-phitm-pmu-rev1.01.pdf

[20] V. Sundriyal, M. Sosonkina, and Z. Zhang, "Automatic runtime frequency-scaling system for energy savings in parallel applications," *The Journal of Supercomputing*, vol. 68, no. 2, pp. 777–797, 2014. [Online]. Available: http://dx.doi.org/10.1007/s11227-013-1062-0

[21] P. Wang, "Ready to run applications from multicore platform onto Intel Xeon Phi coprocessor," 2013. [Online]. Available: https://software.intel.com/en-us/blogs/2013/01/08/ready-to-run-applications-from-multicore-platform-onto-intel-xeon-phi-coprocessor

[22] S. Cepeda, "Optimization and performance tuning for Intel Xeon Phi coprocessors, part 2: Understanding and using hardware events," 2012. [Online]. Available: https://software.intel.com/en-us/articles/optimization-and-performance-tuning-for-intel-xeon-phi-coprocessors-part-2-understanding

[23] V. Sundriyal and M. 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*, ser. E2SC '13. New York, NY, USA: ACM, 2013, pp. 1:1–1:6. [Online]. Available: http://doi.acm.org/10.1145/2536430.2536433

[24] R. F. Barrett, M. A. Heroux, P. T. Lin, C. T. Vaughan, and A. B. Williams, "Poster: Mini-applications: Vehicles for co-design," in *Proceedings of the 2011 Companion on High Performance Computing Networking, Storage and Analysis Companion*, ser. SC '11 Companion. New York, NY, USA: ACM, 2011, pp. 1–2. [Online]. Available: http://doi.acm.org/10.1145/2148600.2148602