Performance and Energy Usage of Workloads on KNL and Haswell Architectures

Tyler Allen[1(✉)], Christopher S. Daley[2], Douglas Doerfler[2], Brian Austin[2], and Nicholas J. Wright[2]

[1] Clemson University, Clemson, SC, USA
tnallen@clemson.edu
[2] Lawrence Berkeley National Laboratory, Berkeley, CA, USA
{csdaley,dwdoerf,baustin,njwright}@lbl.gov

Abstract. Manycore architectures are an energy-efficient step towards exascale computing within a constrained power budget. The Intel Knights Landing (KNL) manycore chip is a specific example of this and has seen early adoption by a number of HPC facilities. It is therefore important to understand the performance and energy usage characteristics of KNL. In this paper, we evaluate the performance and energy efficiency of KNL in contrast to the Xeon (Haswell) architecture for applications representative of the workload of users at NERSC. We consider the optimal MPI/OpenMP configuration of each application and use the results to characterize KNL in contrast to Haswell. As well as traditional DDR memory, KNL contains MCDRAM and we also evaluate its efficacy. Our results show that, averaged over our benchmarks, KNL is 1.84× more energy efficient than Haswell and has 1.27× greater performance.

Keywords: Benchmarking · Power consumption · Energy
Hyperthreads · Manycore architecture · Intel Knights Landing
Haswell

1 Introduction

Manycore architectures promise significant gains in application performance and energy efficiency over past High Performance Computing (HPC) architectural designs. The first mainstream manycore architecture, Intel Knights Landing (KNL), already boasts early adoption in several clusters hosted by major HPC facilities, including Cori at the National Energy Research Scientific Computing (NERSC) Center [16,30], Trinity at Los Alamos National Laboratory (LANL) [15], and Theta at Argonne National Laboratory (ANL) [39]. These first pre-exascale manycore systems are intended to pave the way towards exascale-at-twenty-megawatt computing for the DoE [19]. In this paper, we use modern HPC workloads to evaluate how well the KNL satisfies the trajectory requirements for exascale.

S. Jarvis et al. (Eds.): PMBS 2017, LNCS 10724, pp. 236–249, 2018.
https://doi.org/10.1007/978-3-319-72971-8_12

We use applications representative of the NERSC workload to characterize and quantify the performance and efficiency benefits of KNL. This application-suite is a broad composite of prominent applications used on NERSC systems [10] with some traditional micro-benchmarks for fine-grain evaluation. We evaluate this workload on the current NERSC flagship supercomputer, Cori. We use the most popular KNL configuration: KNL with MCDRAM memory configured to operate as a cache, where MCDRAM is Intel's on-package high bandwidth memory. We contrast this by evaluating the same workload on Cori Haswell nodes at the same node count. To make this a fair comparison, we use optimal Message Passing Interface (MPI) and Open Multi-Processing (OpenMP) configurations for both KNL and Haswell, as well as compiling with the appropriate vector Instruction Set Architecture (ISA) (AVX512, AVX2 respectively). We also contrast against KNL runs that use only Double Data Rate (DDR) memory. Henceforth, we will refer to the KNL cache mode configuration as KNL-Cache and the KNL flat mode configurations as KNL-DDR and KNL-MCDRAM depending on whether data is explicitly allocated in DDR or MCDRAM.

Our key findings are:

- Optimized micro-benchmark configurations run 1.5 to 4.0 times faster on KNL nodes than Haswell nodes.
- The STREAM memory bandwidth is approximately 140 GiB/s less when using KNL-Cache compared to KNL-MCDRAM. In addition, cache conflicts can reduce KNL-Cache memory bandwidth by up to an additional 100 GiB/s.
- The average performance of the full benchmark suite is only better on KNL compared to Haswell when using KNL-Cache. KNL-Cache improves the performance of every single benchmark compared to KNL-DDR. This indicates the value of MCDRAM to overall performance.
- The average energy efficiency of the full benchmark suite is better on KNL than Haswell in both KNL-Cache and KNL-DDR modes. This indicates that there can still be an overall win in terms of work done within a given energy budget by using an architecture with small cores and no MCDRAM.
- The performance and energy gains of hyperthreads are dependent on the individual application, indicating the importance of understanding application characteristics.

2 Related Work

Barnes et al. provide an initial evaluation of strictly performance of some NERSC applications on KNL in comparison to Haswell [12]. A similar evaluation was performed on Trinity by Agelastos et al. [7]. Parker et al. perform a KNL performance study in their evaluation of Theta [31]. The Theta evaluation analyzes power for two micro-benchmarks, whereas we provide a more in depth analysis of the power efficiency improvement of KNL compared to Haswell. Several authors have analyzed the efficacy of hyperthreads, but this work predates KNL and do not give insights into the performance/power benefits on KNL [14,18,37,41].

Lawson et al. evaluated dynamic voltage and frequency scaling (DVFS) techniques on KNL for power limiting and provide a power model, whereas we evaluate efficiency gains and optimal usage without limiting power [28]. Peng et al. [33] evaluate the performance of micro-benchmarks and mini-apps on a KNL system with a focus on the performance impact of various KNL memory modes, and Ramos and Hoefler create a performance model for KNL memory modes [34]. Our evaluation uses a fixed memory mode (KNL-Cache), and contrasts it with the performance of KNL-DDR to show the benefits of MCDRAM. Work has been done on providing single metrics for evaluating trade-offs between power and performance [35]. However, in this paper we optimize applications based on performance and then report the energy consumption of this configuration.

3 NERSC's Cori Supercomputer

The Cori supercomputer is located at the U.S. Department of Energy's Office of Science National Energy Research Scientific Computing Center [16,30]. Cori is based on the Cray XC40 architecture [17] and was deployed in two phases. Phase 1 consists of 2,388 nodes based on dual-socket, 16-core Intel E5-2698 v3 Xeon® processors clocked at 2.3 GHz and a total 128 GB of DDR4 2133 memory [4]. Phase 2 is 9,688 nodes in size and utilizes a single Intel Xeon Phi™ 7250 Knights Landing processor with 68 cores at 1.4 GHz and 96 GB of DDR4 2400 memory [3]. The two node types share a Cray Aries dragonfly high-speed network and a common storage subsystem. The KNL processor can be configured to support a variety of non-uniform memory access (NUMA) and memory modes that are meant to allow the processor to be configured to the particular needs of a given application [38]. In particular, the on-chip mesh can be configured in 3 different clustering modes: all-to-all, quadrant (quad), and sub-NUMA (with the option of 2 or 4 NUMA regions). In addition, the KNL has a 16 GB on-chip high-speed memory (MCDRAM) that can be configured as a directly addressable memory region in its own NUMA region (flat) or it can be used as a cache for DDR. The majority of KNL nodes on Cori are configured to quad/cache mode as it provides an easy on-ramp for users coming from traditional Xeon® nodes. However, Cori does support the other available modes dynamically via a subset of the nodes allocatable in a dedicated *reboot queue* of the scheduler for those users that want to set a mode better suited to the needs of their application.

4 Method and Instrumentation

The key areas of evaluation and characterization for the NERSC workload on Cori are performance per watt and the efficacy of unique features of the KNL architecture including MCDRAM and four hardware-threads-per-core. We used the Integrated Performance Monitoring (IPM) profiling tool in our experiments

to map the performance over the parameter space of the selected NERSC workload applications on KNL. We also performed these experiments on KNL without utilizing MCDRAM and on Haswell nodes. In this section we describe our experimental design and methodology and discuss IPM and the enhancements we made to IPM for our experiments.

4.1 Integrated Performance Monitoring Tool

We introduce the power monitoring enhancement to the IPM library as part of this work[1] [20–22]. IPM is a NERSC profiling library for high-performance applications, first introduced by Fürlinger et al. [21]. IPM aggregates several low level interfaces to provide a large quantity of performance data. In order to adapt IPM to the new manycore architecture HPC paradigm and energy-constrained computing, we added energy measurement to the IPM feature set.

IPM now supports measurement of energy consumption over the course of application execution. The newly added IPM `PMON` module is included in IPM by using the `−enable-pmon` configure option. The PMON module follows the standard IPM module interface and requires only the additional `IPM_PMON`= 1 environment flag at runtime in order to activate energy collection. Energy measurements through IPM are currently only supported on Cray systems through the Cray power monitoring and management stack [29]. IPM measures energy over the full duration of the application, or programmer-specified sections. By default, the energy counter is initialized when the application calls `MPI_Init` and accumulated until the application calls `MPI_Finalize`. We are able to measure energy at three sources: the full node energy prior to distribution, CPU power, and DDR memory energy [36]. As a consequence of the architecture, the KNL MCDRAM energy consumption is included in the CPU power measurement and cannot be measured separately [36]. Using this energy measurement, we are also able to compute the average memory, CPU, and total power consumption of an application. We derive an average power value (W) by dividing this accumulated energy (J) by the application wall clock time (s). IPM energy information can be found in both the IPM standard output summary information and the standard IPM XML output files. Users should note that in the XML output, every rank provides an energy reading for the entire node. Therefore, user post-processing should appropriately handle the case where multiple ranks from the same node producing duplicate values.

4.2 Experiment Methodology

We designed our experiments to show the effect of the following parameter variations:

– Varying MPI ranks-per-node and OpenMP threads-per-rank with a fixed amount of total concurrency

[1] IPM is open-source and available on github: https://github.com/nerscadmin/IPM.

– Varying the total concurrency to evaluate the effects of using one, two, three, and four threads-per-physical-core
– Using KNL nodes in various modes, including KNL-Cache, KNL-MCDRAM and KNL-DDR, and Haswell nodes

We performed experiments using every combination of the parameter variations. The lightweight attribute of IPM makes it possible to collect all of the data required from each of these experiments in a single run per configuration. All applications are built with `icc` version 17.0.2.174 using the `-xMIC-AVX512` optimization flag to enable the 512-bit vector optimizations for KNL. (Haswell builds used the `-xAVX2` flag instead.)

With the rise of manycore architectures, MPI/OpenMP hybrid parallelism is seeing increased popularity. We designed an experiment to explore the performance relationship between MPI/OpenMP for our applications. We first fixed the amount of total concurrency such that there is only one MPI rank or OpenMP thread per core. One thread or process-per-core is exactly 68 threads-per-node on Cori. At times we needed to use MPI counts and/or OpenMP thread counts in powers of 2 because of the domain decomposition requirements of the applications. Experimentally, most applications did not receive a significant performance increase when using 68 cores instead of 64. To evaluate with two-or-more threads per core, we simply used appropriate values for ranks-per-node and threads-per-rank. We used the `OMP_PLACES=threads` and `OMP_PROC_BIND=spread` environment variable settings for our experiments to ensure OpenMP threads are not grouped on a single core. We also used the *Slurm* option `--cpu_bind=cores` to ensure MPI ranks are spread across different cores.

We conducted our experiments using KNL and Haswell nodes. We used the flat KNL modes to evaluate the benefit of MCDRAM for performance and power consumption. The `numactl` tool is used to explicitly allocate memory in MCDRAM or DDR. We also used Haswell nodes to compare the performance, but also the energy efficiency, of KNL nodes to Haswell nodes. This required us to modify our experiment somewhat, as Haswell nodes have fewer cores and threads-per-core than KNL. We only use one and two threads-per-core for Haswell experiments, and limit our MPI concurrencies accordingly. Haswell nodes also do not support the `AVX512` instruction set, and so `AVX2` optimizations are used instead.

4.3 Applications and Micro-benchmarks

Table 1 lists the applications used for this study. For application descriptions please refer to the respective references. The table lists the details of decomposition in addition to a brief description of the level of tuning performed for the KNL processor. Minimal refers to no source codes changes, but compiler optimizations may have been performed. Significant refers to code restructuring, thread and/or vectorization optimizations performed specifically for the KNL architecture.

Table 1. Application benchmark details

Application	Science area	Level of tuning	Nodes	Rnks-Thds/Rnk		GiB/
				HSW	KNL	node
STREAM [6]	Memory bandwidth	Minimal	1	32t	68t	6.7
RandN [11]	Random memory access	Minimal	1	64t	256t	6.5
DGEMM [1]	Dense linear algebra	Intel MKL	1	32t	136t	2.3
GTC-P [2,40]	Fusion	Moderate OpenMP	8	32r-1t	32r-8t	0.17
MILC [13]	Quantum chromodynamics	QPhiX dslash solver	8	32r-1t	32r-2t	8.3
Nyx-AMReX [9]	Cosmology	Minimal	2	16r-4t	16r-16t	58
Castro-AMReX [8]	Astrophysics	Minimal	4	32r-1t	32r-2t	6.5
Quantum Espresso [23]	Quantum chemistry	Significant	4	4r-8t	4r-16t	21
BD-CATS [32]	Data analytics for cosmology	Minimal	16	16r-4t	16r-16t	5.4

5 Results

5.1 Micro-benchmarks

In this section we evaluate the performance and energy efficiency of the DGEMM, STREAM and RandN micro-benchmarks on the KNL and Haswell architectures. The micro-benchmarks stress peak floating point, sequential memory access, and random memory access performance, respectively. The KNL results are obtained in the three modes discussed earlier: KNL-Cache, KNL-MCDRAM and KNL-DDR. We choose the problem size so that the memory footprint per compute node is less than the memory capacity of MCDRAM. This allows us to evaluate the benefit of MCDRAM under ideal circumstances.

Figure 1 shows the absolute performance and energy efficiency of the micro-benchmarks. The top row of the figure shows the performance of the benchmarks in the appropriate units: floating point rate for DGEMM in units of TFLOP/s and memory bandwidth for STREAM and RandN in units of GiB/s. The bottom row of the figure shows the energy cost of performing a single operation in the benchmark: 1 double precision FLOP in DGEMM and transferring a single 8-byte word to/from memory in STREAM and RandN. The energy metric is calculated by dividing the average power usage in Watts (J/s) by the micro-benchmark performance metric printed to standard output. In the case of DGEMM, we divide [J/s] by [FLOP/s] to obtain [J/FLOP].

The results in this figure show that the optimal micro-benchmark configurations run 1.5 to 4.0 times faster on the KNL architecture compared to the Haswell architecture. Our best KNL performance results are a peak floating point rate of 2 TFLOP/s, a sequential memory bandwidth of 466 GiB/s and a random memory access bandwidth of 6 GiB/s. The STREAM micro-benchmark performs better in KNL-MCDRAM mode than in KNL-Cache mode (466 GiB/s vs 327 GiB/s), indicating that KNL-Cache mode introduces some overhead for memory-bandwidth bound applications. This is because streaming stores incur an additional memory read in KNL-Cache mode to determine whether a line is already present in MCDRAM [27, p. 565]. We have found that switching off streaming stores with the compiler option `-qopt-streaming-stores=never`

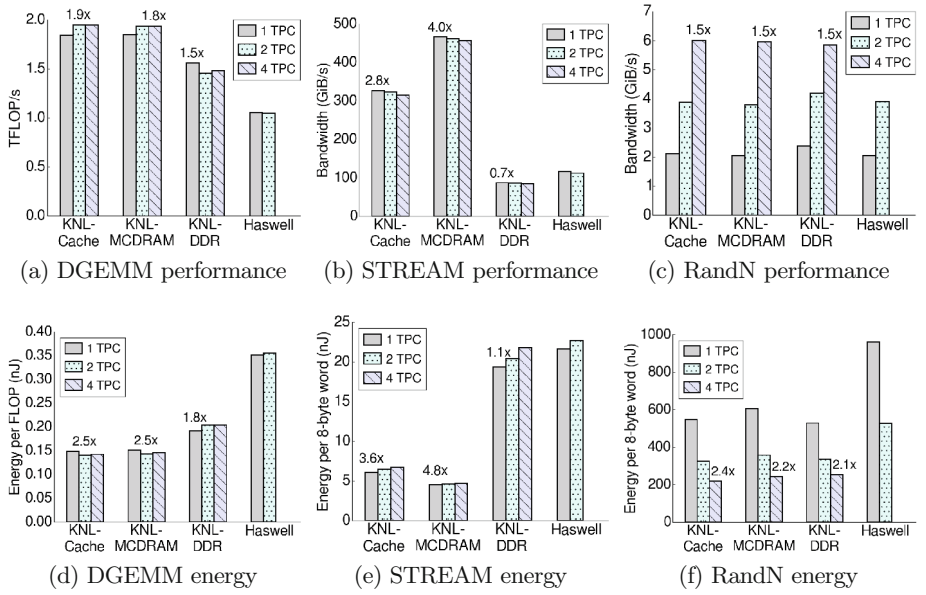


Fig. 1. Performance and energy efficiency of DGEMM, STREAM and RandN micro-benchmarks on KNL and Haswell architectures with various counts of threads per core (TPC). The optimal KNL configuration in each mode is marked with the relative improvement over the optimal Haswell configuration.

reduces STREAM performance in KNL-MCDRAM mode to 347 GiB/s, explaining most of the performance difference. All micro-benchmarks perform worse in KNL-DDR mode than the other modes, with the exception of RandN in 1 and 2 threads per core (TPC) configurations (Fig. 1c). These lower concurrency configurations are impacted by the higher memory latency of MCDRAM [33]. It is only at the highest TPC concurrency where MCDRAM can support the large number of memory requests needed to hide the memory latency disadvantage. The DGEMM and STREAM micro-benchmarks benefit more from MCDRAM than RandN (Figs. 1a and b). DGEMM and STREAM performance remains approximately the same when adding multiple threads per core because the floating point performance and memory bandwidth are already saturated.

The figure also shows that the optimal KNL configurations are 2.4 to 4.8 times more energy efficient than the Haswell architecture. Energy efficiency improves more than performance because of the difference in power usage of the nodes. For example, the optimally performing DGEMM configuration consumes 270 W on KNL and 360 W on Haswell[2]. The best KNL energy results are 0.15 nJ/FLOP in DGEMM, 4.5 nJ/word in STREAM and 200 nJ/word in

[2] The DGEMM power consumption is approximately 2 to 8 W higher on KNL over a range of concurrencies than the synthetic Firestarter benchmark designed to create near-peak power consumption [24].

RandN. These are significantly larger than the exascale target of 20 pJ/FLOP (i.e. 0.02 nJ/FLOP) to achieve an exaFLOP within a 20 MW power budget. Finally, the order of magnitude difference between STREAM and RandN indicates the high energy cost of random memory access workloads in traditional CPU architectures.

The evaluated Xeon Phi™ and Xeon® processors use a 14 nm and 22 nm technology size, respectively. This is a slightly unfair comparison because a smaller feature size is more energy efficient. Product sheets show that a 14 nm 16-core Xeon® Broadwell processor has a Thermal Design Power (TDP) of 115 W [5] compared to the Haswell Xeon® in Cori which has a TDP of 135 W [4]. Therefore, a rough estimate of the energy efficiency improvement over a 14 nm Xeon® can be obtained by multiplying the energy efficiency improvement in the figures by [115/135]. The optimal STREAM configuration on KNL would therefore be 4.1x more energy efficient than a 14 nm Xeon®.

5.2 STREAM Variability

The performance of the STREAM micro-benchmark varies considerably in KNL-Cache mode because of cache conflicts in the direct-mapped MCDRAM cache. This effect cannot be controlled and depends on the specific physical memory pages allocated to a job at runtime. It is a known issue that Intel has partially mitigated by creating a kernel module named Zonesort [25,26]. The kernel module reorders memory pages to reduce cache-conflicts and is run on Cori before every Slurm job step.

Figure 2 is a cumulative density plot showing STREAM performance over 48 trials. The results show that over 50% of trials achieve a bandwidth of 324–327 GiB/s and that there is a long tail of degradation towards 225 GiB/s. We monitored a performance counter measuring DDR traffic named `OFFCORE_RESPONSE_0:ANY_REQUEST:DDR` in each trial and found that high values correlate with poor

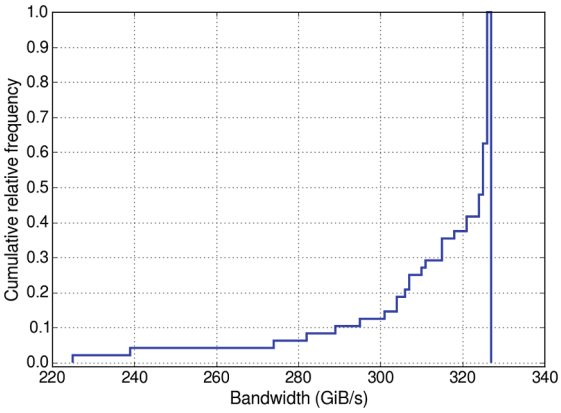


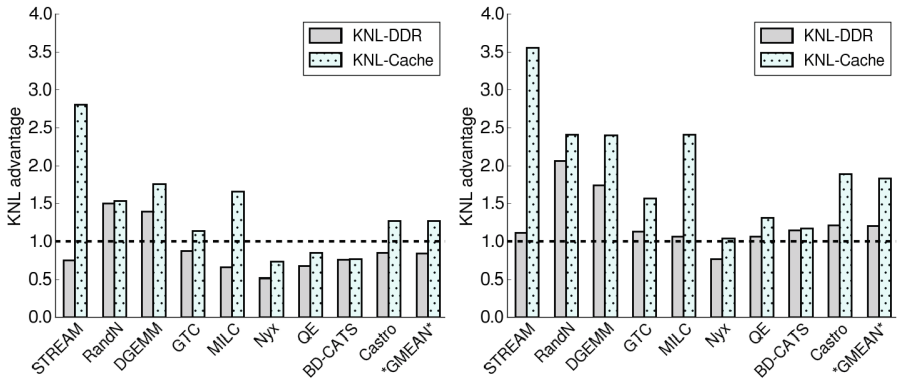
Fig. 2. STREAM bandwidth in KNL-Cache mode over 48 trials

STREAM memory bandwidth and high DDR memory power. This indicates that the direct-mapped cache on KNL can cause performance and energy inefficiencies and that Zonesort does not eliminate all cache-conflicts. It is significant because this effect can reduce STREAM memory bandwidth by a factor of 2 compared to the optimally performing KNL-MCDRAM configuration.

5.3 Performance and Energy Consumption Across Applications

In this section we compare application performance and energy consumption across architectural configurations for the application benchmark suite. We identify the fastest MPI/OpenMP configuration for each benchmark on KNL and then use this MPI count and a variable number of OpenMP threads for every experiment. Our tests are designed to show application sensitivity to memory bandwidth and hyperthreading on KNL. The later experiment studying hyperthreads is performed in KNL-cache mode.

Figure 3 summarizes the application performance and energy consumption on KNL and Haswell. The results are normalized so that values greater than 1.0 indicate that the application has a higher figure of merit on KNL than Haswell. The results show that 6 out of 9 applications perform better on the KNL node architecture. The KNL performance is best when using KNL-Cache mode in every experiment, and the KNL-DDR mode is worse than Haswell for all scientific applications, indicating the importance of MCDRAM to application performance. The greatest MCDRAM gains occur in STREAM and MILC which are applications bound by memory bandwidth. In some cases, the opposite is true: when applications like RandN and BD-CATS are dominated by random memory access, MCDRAM provides negligible gains in performance. Perhaps the most significant result is that all applications consume less energy on KNL compared



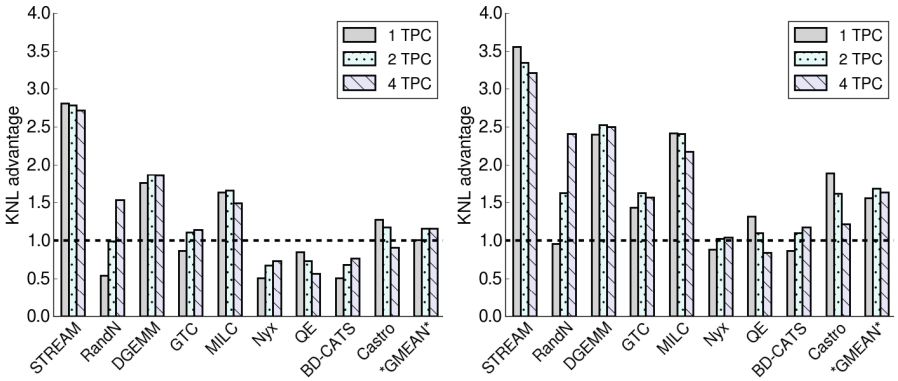
(a) Performance
Geometric mean: 0.84 and 1.27

(b) Energy consumption
Geometric mean: 1.21 and 1.84

Fig. 3. Figures of merit improvement of KNL relative to Haswell. The best KNL configuration is compared against the best Haswell configuration.

to Haswell. If we follow the approach from earlier then we estimate that the 1.84x energy improvement over Haswell would be a 1.56x energy improvement over a 14 nm Broadwell processor.

Figure 4 shows that application performance is more variable when changing the number of threads per core. Several applications, e.g. STREAM, Quantum Espresso and Castro, perform worse when using hyperthreads because of either resource saturation or increased overhead of using more threads. Other applications with random memory access, e.g. RandN and BD-CATS, have significant gains when using all 4 threads per core. On average, hyperthreads improve performance by approximately 16% over the optimal Haswell configuration. We find that 2 and 4 hyperthreads per core deliver similar average performance, however, we find that the 4 hyperthreads per core configuration consumes more energy than the 2 hyperthreads per core configuration. Therefore, based on energy consumption, 4 threads per core configurations are only helpful for a very specialized workload, e.g. a graph analytics workload dominated by random memory access.



(a) Performance
Geometric mean: 1.01, 1.16 and 1.16

(b) Energy consumption
Geometric mean: 1.56, 1.68 and 1.64

Fig. 4. Figures of merit improvement of KNL relative to Haswell. The KNL configuration at each thread count is compared against the best Haswell configuration. The KNL results are obtained in KNL-Cache mode.

6 Conclusions

We have shown that KNL is a solid step towards exascale efficiency, but that there is still significant progress left to be made. On the NERSC workload, we have shown that KNL improves performance for 6 out of 9 applications vs. Haswell, but manages to reduce energy consumption for every application. Also, for applications with memory locality, the MCDRAM present on KNL can provide enormous performance benefits in comparison to DDR4, and simultaneously reduces the energy-per-operation for every application. MCDRAM is a critical

feature of this architectural shift. Future architectures will need to make even greater strides towards efficiency.

7 Future Work

The work in this paper relied on our experience with the applications to explain the observed performance results. We plan a more thorough approach that will automatically characterize applications using hardware performance counters. We have already started to create this performance analysis framework by adding PAPI multiplexing support to IPM and developing scripts to create derived performance metrics based on this data to quantify the performance requirements of applications. This will allow us to understand at a deeper level the overall sensitivity of the larger NERSC workload to features on modern CPUs, such as MCDRAM and hyperthreads.

Acknowledgment. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

1. DGEMM. http://www.nersc.gov/research-and-development/apex/apex-benchmarks/dgemm/
2. GTC-P. http://www.nersc.gov/research-and-development/apex/apex-benchmarks/gtc-p/
3. Intel Xeon Phi Processor 7250 16GB, 1.40 GHz, 68 core. https://ark.intel.com/products/94035/Intel-Xeon-Phi-Processor-7250-16GB-1_40-GHz-68-core
4. Intel Xeon Processor E5–2698 v3 40M Cache, 2.30 GHz. https://ark.intel.com/products/81060/Intel-Xeon-Processor-E5-2698-v3-40M-Cache-2_30-GHz
5. Intel Xeon Processor E7–4850 v4 40M Cache, 2.10 GHz. https://ark.intel.com/products/93806/Intel-Xeon-Processor-E7-4850-v4-40M-Cache-2_10-GHz
6. STREAM: Sustainable Memory Bandwidth in High Performance Computers. https://www.cs.virginia.edu/stream/FTP/Code/
7. Agelastos, A.M., Rajan, M., Wichmann, N., Baker, R., Domino, S., Draeger, E.W., Anderson, S., Balma, J., Behling, S., Berry, M., Carrier, P., Davis, M., McMahon, K., Sandness, D., Thomas, K., Warren, S., Zhu, T.: Performance on Trinity phase 2 (a Cray XC40 utilizing Intel Xeon Phi processors) with acceptance applications and benchmarks. In: Cray User Group CUG, May 2017. https://cug.org/proceedings/cug2017_proceedings/includes/files/pap138s2-file1.pdf
8. Almgren, A.S., Beckner, V.E., Bell, J.B., Day, M.S., Howell, L.H., Joggerst, C.C., Lijewski, M.J., Nonaka, A., Singer, M., Zingale, M.: CASTRO: A new compressible astrophysical solver. I. hydrodynamics and self-gravity. Astrophys. J. **715**, 1221–1238 (2010)
9. Almgren, A.S., Bell, J.B., Lijewski, M.J., Lukić, Z., Andel, E.V.: Nyx: A massively parallel AMR code for computational cosmology. Astrophys. J. **765**(1), 39 (2013). http://stacks.iop.org/0004-637X/765/i=1/a=39

10. APEX Benchmark Distribution and Run Rules. http://www.nersc.gov/research-and-development/apex/apex-benchmarks/
11. Austin, B., Wright, N.J.: Measurement and interpretation of microbenchmark and application energy use on the Cray XC30. In: Proceedings of the 2nd International Workshop on Energy Efficient Supercomputing, pp. 51–59. IEEE Press (2014)
12. Barnes, T., Cook, B., Deslippe, J., Doerfler, D., Friesen, B., He, Y., Kurth, T., Koskela, T., Lobet, M., Malas, T., Oliker, L., Ovsyannikov, A., Sarje, A., Vay, J.L., Vincenti, H., Williams, S., Carrier, P., Wichmann, N., Wagner, M., Kent, P., Kerr, C., Dennis, J.: Evaluating and optimizing the NERSC workload on knights landing. In: 2016 7th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), pp. 43–53, November 2016
13. Bauer, B., Gottlieb, S., Hoefler, T.: Performance modeling and comparative analysis of the MILC Lattice QCD application su3_rmd. In: Proceedings CCGRID2012: IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (2012)
14. Coghlan, S., Kumaran, K., Loy, R.M., Messina, P., Morozov, V., Osborn, J.C., Parker, S., Riley, K.M., Romero, N.A., Williams, T.J.: Argonne applications for the IBM Blue Gene/Q, Mira. IBM J. Res. Dev. **57**(1/2), 12:1–12:11 (2013)
15. LANL Trinity Supercomputer. http://www.lanl.gov/projects/trinity/
16. NERSC Cori Supercomputer. https://www.nersc.gov/systems/cori/
17. Cray XC Series Supercomputers. http://www.cray.com/products/computing/xc-series
18. Evangelinos, C., Walkup, R.E., Sachdeva, V., Jordan, K.E., Gahvari, H., Chung, I.H., Perrone, M.P., Lu, L., Liu, L.K., Magerlein, K.: Determination of performance characteristics of scientific applications on IBM Blue Gene/Q. IBM J. Res. Dev. 57(1), 99–110 (2013). https://doi.org/10.1147/JRD.2012.2229901
19. The Opportunities and Challenges of Exascale Computing. https://science.energy.gov/~/media/ascr/ascac/pdf/reports/Exascale_subcommittee_report.pdf
20. Fuerlinger, K., Wright, N.J., Skinner, D.: Effective performance measurement at petascale using IPM. In: 2010 IEEE 16th International Conference on Parallel and Distributed Systems, pp. 373–380, December 2010
21. Fürlinger, K., Wright, N.J., Skinner, D.: Performance analysis and workload characterization with IPM. In: Müller, M., Resch, M., Schulz, A., Nagel, W. (eds.) Tools for High Performance Computing 2009, pp. 31–38. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11261-4_3
22. Fürlinger, K., Wright, N.J., Skinner, D., Klausecker, C., Kranzlmüller, D.: Effective holistic performance measurement at petascale using IPM. In: Bischof, C., Hegering, H.G., Nagel, W., Wittum, G. (eds.) Competence in High Performance Computing 2010, pp. 15–26. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24025-6_2
23. Giannozzi, P., Baroni, S., Bonini, N., Calandra, M., Car, R., Cavazzoni, C., Ceresoli, D., Chiarotti, G.L., Cococcioni, M., Dabo, I., Dal Corso, A., de Gironcoli, S., Fabris, S., Fratesi, G., Gebauer, R., Gerstmann, U., Gougoussis, C., Kokalj, A., Lazzeri, M., Martin-Samos, L., Marzari, N., Mauri, F., Mazzarello, R., Paolini, S., Pasquarello, A., Paulatto, L., Sbraccia, C., Scandolo, S., Sclauzero, G., Seitsonen, A.P., Smogunov, A., Umari, P., Wentzcovitch, R.M.: QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. J. Phys. Condens. Matter **21**(39), 395502 (19pp) (2009). http://www.quantum-espresso.org

24. Hackenberg, D., Oldenburg, R., Molka, D., Schöne, R.: Introducing FIRESTARTER: a processor stress test utility. In: 2013 International Green Computing Conference Proceedings, pp. 1–9, June 2013
25. He, Y., Cook, B., Deslippe, J., Friesen, B., Gerber, R., Hartman-Baker, R., Koniges, A., Kurth, T., Leak, S., Yang, W.S., Zhao, Z.: Preparing NERSC users for Cori, a Cray XC40 system with Intel many integrated cores. In: Cray User Group CUG, May 2017. https://cug.org/proceedings/cug2017_proceedings/includes/files/pap161s2-file1.pdf
26. Hill, P., Snyder, C., Sygulla, J.: KNL system software. In: Cray User Group CUG, May 2017. https://cug.org/proceedings/cug2017_proceedings/includes/files/pap169s2-file1.pdf
27. Jeffers, J., Reinders, J., Sodani, A.: Intel Xeon Phi Processor High Performance Programming: Knights, Landing edn. Morgan Kaufmann, Boston (2016)
28. Lawson, G., Sundriyal, V., Sosonkina, M., Shen, Y.: Runtime power limiting of parallel applications on Intel Xeon Phi Processors. In: 2016 4th International Workshop on Energy Efficient Supercomputing (E2SC), pp. 39–45, November 2016
29. Martin, S.J., Kappel, M.: Cray XC30 power monitoring and management. In: Cray User Group 2014 Proceedings (2014)
30. National Energy Research Scientific Computing Center. https://www.nersc.gov
31. Parker, S., Morozov, V., Chunduri, S., Harms, K., Knight, C., Kumaran, K.: Early evaluation of the Cray XC40 Xeon Phi System 'Theta' at Argonne. In: Cray User Group CUG, May 2017. https://cug.org/proceedings/cug2017_proceedings/includes/files/pap113s2-file1.pdf
32. Patwary, M.M.A., Dubey, P., Byna, S., Satish, N.R., Sundaram, N., Lukić, Z., Roytershteyn, V., Anderson, M.J., Yao, Y., Prabhat: BD-CATS: big data clustering at trillion particle scale. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC 2015, pp. 1–12. ACM Press, New York (2015). http://dl.acm.org/citation.cfm?doid=2807591.2807616
33. Peng, I.B., Gioiosa, R., Kestor, G., Laure, E., Markidis, S.: Exploring the Performance Benefit of Hybrid Memory System on HPC Environments. CoRR abs/1704.08273 (2017). http://arxiv.org/abs/1704.08273
34. Ramos, S., Hoefler, T.: Capability models for manycore memory systems: a case-study with Xeon Phi KNL. In: 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 297–306, May 2017
35. Roberts, S.I., Wright, S.A., Fahmy, S.A., Jarvis, S.A.: Metrics for energy-aware software optimisation. In: Kunkel, J.M., Yokota, R., Balaji, P., Keyes, D. (eds.) ISC 2017. LNCS, vol. 10266, pp. 413–430. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58667-0_22
36. Rush, D., Martin, S.J., Kappel, M., Sandstedt, M., Williams, J.: Cray XC40 power monitoring and control for knights landing. In: Cray User Group CUG, May 2017. https://cug.org/proceedings/cug2016_proceedings/includes/files/pap112s2-file1.pdf
37. Saini, S., Jin, H., Hood, R., Barker, D., Mehrotra, P., Biswas, R.: The impact of hyper-threading on processor resource utilization in production applications. In: Proceedings of the 2011 18th International Conference on High Performance Computing, pp. 1–10, HIPC 2011, IEEE Computer Society, Washington, DC, USA (2011). https://doi.org/10.1109/HiPC.2011.6152743

38. Sodani, A.: Knights landing (KNL): 2nd generation Intel Xeon Phi Processor. In: Hot Chips 27, Flint Center, Cupertino, CA, August 23–25 2015. http://www.hotchips.org/wp-content/uploads/hc_archives/hc27/HC27.25-Tuesday-Epub/HC27.25.70-Processors-Epub/HC27.25.710-Knights-Landing-Sodani-Intel.pdf
39. ANL Theta Supercomputer. https://www.alcf.anl.gov/theta
40. Wang, B., Ethier, S., Tang, W.M., Ibrahim, K.Z., Madduri, K., Williams, S., Oliker, L.: Modern Gyrokinetic Particle-In-Cell Simulation of Fusion Plasmas on Top Supercomputers. CoRR abs/1510.05546 (2015). http://arxiv.org/abs/1510.05546
41. Zhao, Z., Wright, N.J., Antypas, K.: Effects of hyper-threading on the NERSC workload on Edison. In: Cray User Group CUG, May 2013. https://www.nersc.gov/assets/CUG13HTpaper.pdf