Power-aware Computing: Measurement, Control, and Performance Analysis for Intel Xeon Phi

Azzam Haidar*, Heike Jagode*, Asim YarKhan*, Phil Vaccaro*, Stanimire Tomov*, Jack Dongarra*†‡
{haidar|jagode|yarkhan|tomov|dongarra}@icl.utk.edu,
*Innovative Computing Laboratory (ICL), University of Tennessee, Knoxville, USA
†Oak Ridge National Laboratory, USA
‡University of Manchester, UK

***Abstract*— The emergence of power efficiency as a primary constraint in processor and system designs poses new challenges concerning power and energy awareness for numerical libraries and scientific applications. Power consumption also plays a major role in the design of data centers in particular for peta- and exa-scale systems. Understanding and improving the energy efficiency of numerical simulation becomes very crucial.**

We present a detailed study and investigation toward controlling power usage and exploring how different power caps affect the performance of numerical algorithms with different computational intensities, and determine the impact and correlation with performance of scientific applications.

Our analyses is performed using a set of representatives kernels, as well as many highly used scientific benchmarks. We quantify a number of power and performance measurements, and draw observations and conclusions that can be viewed as a roadmap toward achieving energy efficiency computing algorithms.

I. Introduction

Power, energy, and temperature concerns have become major factors influencing the designs of today's processors. These concerns have led to the stagnation of CPU clock frequencies and to the reliance on parallelism—at both hardware and software level—for future performance and power efficiency increases. Hardware accelerated computing systems (e.g., GPUs, accelerators) have drawn the attention of researchers with designs based on many low-frequency cores that obtain tremendous computational power and high memory bandwidth. At the same time, it has become important to not only achieve high performance but also manage power usage, for example, using a metric measuring the number of floating point operations (flops) per Watt (W).

While hardware advancements are important, the software development of numerical libraries that are power and energy-aware is also becoming of high interest, and may be even more important. Indeed, although some hardware setups can be a few times more power efficient than others, the software design can bring more than an order of magnitude improvement in addition to the hardware. A necessary step toward energy efficiency is the ability to easily measure power and energy consumption, and determine the correlation with performance.

In this paper, we use the PAPI library that provides a generic, portable interface for the hardware performance counters available on all modern CPUs and other components of interest that are scattered across the compute system. All our experiments are set up and executed on Intel's Xeon Phi Knights Landing (KNL) architecture. The Knights Landing processor demonstrates many of the interesting architectural characteristics of the newer many-core architectures, with a deeper hierarchy of explicit memory modules and multiple modes of configuration. Our analysis is performed using Dense Linear Algebra (DLA) kernels, specifically BLAS (Basic Linear Algebra Subroutine) kernels.

In order to measure the relationship between energy consumption and attained performance for algorithms with different complexities, we run experiments with representative BLAS kernels that have different computational intensities. We use the newly developed power capping functionality of PAPI to examine the effects of managing power on algorithmic performance.

II. Contributions

The contributions of this work are in the areas of power management and in the analysis of the power efficiency of compute/memory bound computational kernels in the context of new high-bandwidth memory and many-core processor architectures. The contributions described in this work are:

- A novel component (*powercap*) in the PAPI performance library for power management. This new PAPI powercap component has an active interface that allows *writing* values in addition to *reading* them. This is a significant change from the prior PAPI RAPL component that had an entirely passive measurement interface to read information and events. Having this power capping functionality available through a common interface like PAPI provides portability and is beneficial to many scientific application developers, independent of whether they use PAPI directly, or via 3rd-party performance toolkits.
- A detailed study of the usage of the powercap component, extending the notion of *performance monitoring* to include power capping capabilities. This will enable the joint monitoring of hardware performance events and power information, together with the power capping functionality, in a uniform way through one consistent PAPI interface.
- A study of the correlation between *power usage* and *performance* for kernels that are representative for a wide range of real scientific applications. This provides

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

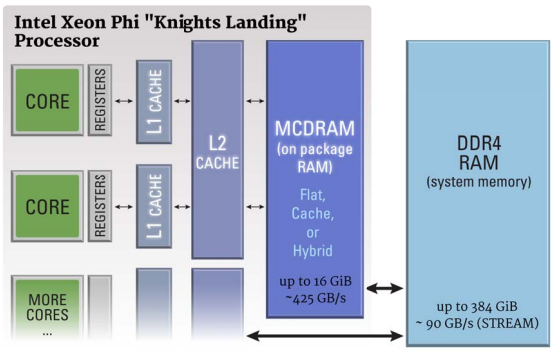


Fig. 1. Physical memory organization of Intel's Knights Landing processors.

a clear understanding about the factors that contribute to energy saving and performance. We are using the newly implemented PAPI power control mechanism because we are not only interested in reducing power usage but, more importantly, we are interested in exploring energy savings while keeping the execution time constant (ideally).

- This is the first paper that evaluates the power efficiency as related to the algorithmic intensity on the KNL architecture, allowing researchers to understand and predict what to expect from using high-bandwidth MCDRAM memory versus standard DDR4 memory, and from managing power on this architecture.

III. Hardware Description

Memory bandwidth in computing systems is one of the common bottlenecks for performance in computational applications. The memory hierarchy has become increasingly more complex, with the standard cache hierarchy being enhanced with connectivity between multiple memory devices. A specific example of current memory architectures is provided by the many-core Intel Xeon Phi Knights Landing (KNL) processor. It offers 16 GB of high-bandwidth memory (HBM) based on Multi-Channel Dynamic Random Access Memory (MCDRAM) with a bandwidth of around 425 GB/s, in addition to the more standard DDR4 memory with a bandwidth of about 90 GB/s. It is important to note that the high bandwidth MCDRAM is a 3-D stacked memory that is connected on top of the CPU chip die. Each core on the CPU die has a separate channel to its own region of the MCDRAM chip, hence the multi-channel designation. Figure 1 shows the physical memory organization of the Intel KNL processors. The MCDRAM can be configured in BIOS in one of three different usage modes. In *flat mode*, the entire MCDRAM is visible to the operating system as addressable memory that is available via a NUMA node. Memory allocations and initialization can be done specifically in the MCDRAM. These memory allocations are treated similarly to standard DDR4 memory allocations and will be cached by the L2 cache as needed. However, there are other ways of configuring this high bandwidth MCDRAM memory: for example, as a large cache between L2 cache and the DDR4 memory, which is referred to as *cache mode*. In this case, the MCDRAM is not visible to the OS, but is transparently used as a cache. A third way is referred to as *hybrid mode* where a portion of MCDRAM is used as addressable memory and the rest is used as cache. This allows for locating frequently written data objects to MCDRAM, and relegating read-mostly data objects to DDR4 memory. An application would need to profile its data usage with the memory access counters to choose the appropriate usage model. We conducted experiments and performance analysis study on the three different MCDRAM configurations on the KNL. We used the Intel C compiler from the Intel Composer XE 2016 suite, and the linear algebra routines from the Intel Math Kernel Library, MKL [7], optimized for the Intel Knights Landing architecture.

IV. Representative Kernels

In order to study and analyze the effect of real applications on power consumption and energy requirements, we chose kernels that can be found in high performance computing (HPC) applications and that can be distinctly classified through their computational intensity. We decided to perform an extensive study over linear algebra routines that are representative to many methods (such as, Jacobi iteration, Gauss-Seidel methods, Newton-Raphson, etc) used in real scientific applications, as for example in climate modeling, computational fluid dynamics simulations, material science simulations. The BLAS or Basic Linear Algebra Subprograms are a specification for a fundamental set of low-level linear algebra operations that are important for a wide variety of algorithms and scientific applications. BLAS operations are categorized into three levels by the type of operation. Level 1 addresses scalar and vector operations, Level 2 addresses matrix-vector operations, and Level 3 addressed matrix-matrix operations. The BLAS routines provide an excellent route to examine power and performance characteristics given that they are of high importance to scientific computations, well defined and well understood operations, their implementations are highly optimized by vendor libraries, and they provides different memory footprint, performance and power characteristics. The Level 1 and Level 2 BLAS routines belong to the "*memory-bound class*" of functions and thus provide similar behavior in term of performance and power requirement while the Level 3 BLAS routines belong to the "*compute intensive class*" of functions. We present the analysis and study for the compute intensive routine dgemm, and the memory bound routine dgemv as they demonstrate a wide range of computational intensities. Our goal is to find answers to the following questions: (1) What is the performance that can be attained, (2) Can power requirement be predicted for a real application, and (3) What energy requirement should we expect from applications?

V. PAPI: The Performance API

The PAPI performance monitoring library provides a coherent methodology and standardization layer to performance counter information for varied hardware and software components, including CPUs [16], GPUs [11], memory, net-

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

works [13], [4], I/O systems [16] and power interfaces [12], [8], as well as virtual cloud environments [9]. PAPI can be used independently as a performance monitoring library and tool for application analysis. However, PAPI finds its greatest utility as middleware component for a number of third-party profiling, tracing, and sampling toolkits (e.g., CrayPat [5], HPCToolkit [1], Scalasca [6], Score-P [14], TAU [15], Vampir [2], PerfExpert [3]), making it the de facto standard for performance counter analysis. As the middleware, PAPI handles the details for each hardware component in order to provide a consistent API platform to the higher-level toolkits, as well as operating system-independent access to performance counters within CPUs, GPUs, and the system as a whole.

A. Power Monitoring Capabilities

Energy efficiency has been identified as a major concern for extreme-scale platforms [10], and PAPI offers a number of components for different architectures that allows for transparent monitoring of power usage and energy consumption through different interfaces such as the Intel RAPL (Running Average Power Limit) interface, the libmsr [17] library, the MicAccess API, the high speed power measurement API on IBM Blue Gene/Q systems called EMON, and the NVML (NVIDIA Management Library) interface.

We build on this work and extend PAPI by incorporating power monitoring and power capping functionalities for recent Intel CPUs via the Linux `powercap` interface. The newly developed PAPI `powercap` component is publicly available since the PAPI 5.5.0 release. The addition of power controlling through PAPI has unlocked new opportunities for managing energy efficiency of applications. For instance, it can determine the choice between alternative algorithmic implementations of an operation, leading to the design of energy-efficient algorithms [8], [10]. Even though power optimization opportunities may not occur with every algorithm, PAPI can provide the information to detect the existence of such opportunities (i.e., power/time consumed by various tasks), and, furthermore, PAPI provides the API to allow the application to control the power on supported hardware.

The two power domains supported on Intel Xeon Phi Knights Landing processor (KNL) are (1) power for the entire package (`PACKAGE_ENERGY`), and (2) power for the memory subsystem (`DRAM_ENERGY`). As mentioned in section III, the MCDRAM is an on-package memory that resides on the CPU chip, while the DDR4 memory modules are not installed on the CPU chip. For that reason, energy consumption of the MCDRAM is accounted for in the `PACKAGE_ENERGY` domain only, while DDR4 is accounted for in `DRAM_ENERGY` domain.

VI. Discussion

We run an extensive set of experiments using the `powercap` component from PAPI. Figures 2 and 3 illustrate the performance results and the power consumption of both the package and the memory for the dgemm and dgemv kernels on the Intel Xeon Phi Knight Landing (KNL 7250). For each of the BLAS routines, we examined the behavior of the following four types of memory/data-allocation configurations in our experiments. The KNL is configured in FLAT or HYBRID mode, and for each of these memory modes the data can be allocated in the fast MCDRAM memory or in the standard DDR4 memory.

A. Study of the dgemm kernel behavior

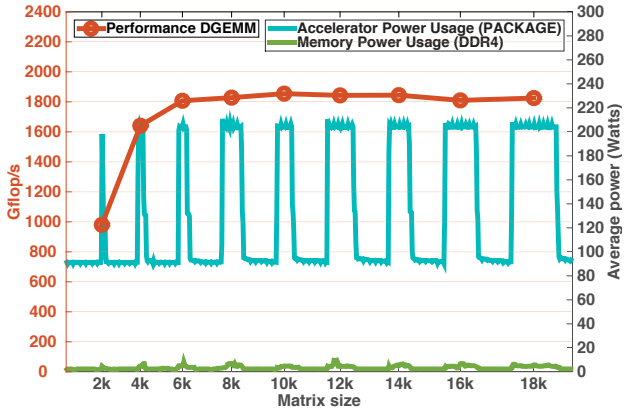
We begin this discussion with the compute intensive dgemm kernel. In order to condense a large amount of information, in Figure 2 we represent the performance achieved in GFlop/s (left vertical axis) and the average power in Watt (right vertical axis). The size of the matrices used in the dgemm computation varies from 2,000 to 18,000 (i.e., 2,000, 4,000, ..., 18,000).

In Figure 2a we show the measurements when the KNL is booted in HYBRID mode and the data is allocated in the DDR4 while Figure 2b illustrate the measurments when the data is allocated in MCDRAM. The performance of the dgemm routine in HYBRID mode is not affected by where the data is allocated. Since the dgemm is a compute intensive routine, once a block of data is brought from the DDR4 to the cache it is kept there until it is no longer needed. As a results, the data transfer from the DDR4 is minimal (probably once per block) and the performance is similar to the one obtained when the data is allocated in the high speed MCDRAM. In Figure 2a the DRAM memory power consumption is slightly higher than the one of Figure 2b because the data originates in DDR memory and additional power is used to transfer it to the cache. Once the data is in cache, there will be enough available data to satisfy all the cores. For that reason, both the DDR4 and MCDRAM data allocation experiments will keep all the cores busy and will have roughly the same package power consumption.

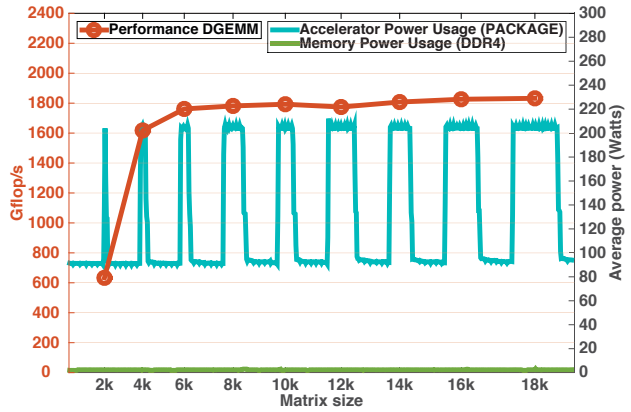
In Figures 2c and 2d, we did similar experiments and measurements for dgemm when the KNL was booted in FLAT mode. The FLAT mode does not use the MCDRAM as a cache, but as physical addressable memory space. In term of package power consumption, the behavior observed for the FLAT mode is similar to the one observed with the HYBRID mode for both memory options. This was expected since the dgemm is a compute intensive routine that relies on the efficiency of the cores rather than the speed of data transfer. The DDR4 power usage in Figure 2c is slightly higher than the one in Figure 2a because the data may be moved multiple times to/from the DDR4 while it will be held in the 8GB of MCDRAM-cache when the KNL is in HYBRID mode. The DDR4 power consumption for the MCDRAM option (Figure 2d) is minimal since no data is allocated there.

The lesson learned is, for compute intensive kernels, like dgemm, in HYBRID mode, it is better to allocate in the MCDRAM (if the data fits) in order to avoid the power requirement of the DDR4 data movement without any loss in term of performance. For the FLAT mode, it is clearly better to allocate in MCDRAM (16GB). If the data is too large to fit in the MCDRAM, it is advised to switch the device to

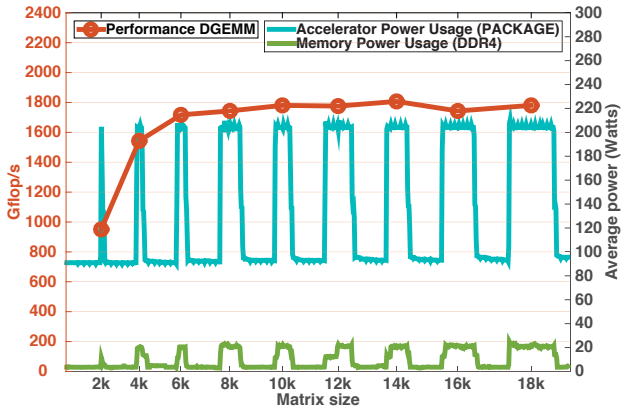
978-1-5386-3472-1/17/$31.00 ©2017 IEEE



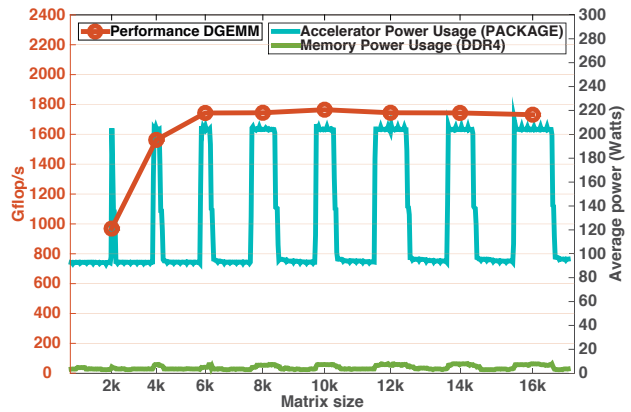
(a) HYBRID mode: dgemm: data allocated on DDR4.



(b) HYBRID mode: dgemm: data allocated on MCDRAM.



(c) FLAT mode: dgemm: data allocated on DDR4.



(d) FLAT mode: dgemm: data allocated on MCDRAM.

Fig. 2. Performance (GFlop/s on left axis) and average power (Watt on right axis) measurements of the Level 3 BLAS dgemm routine when the KNL is in either HYBRID or FLAT mode. The dgemm kernel is run successively for different matrix sizes.

HYBRID mode to avoid getting slower performance at higher power consumption.

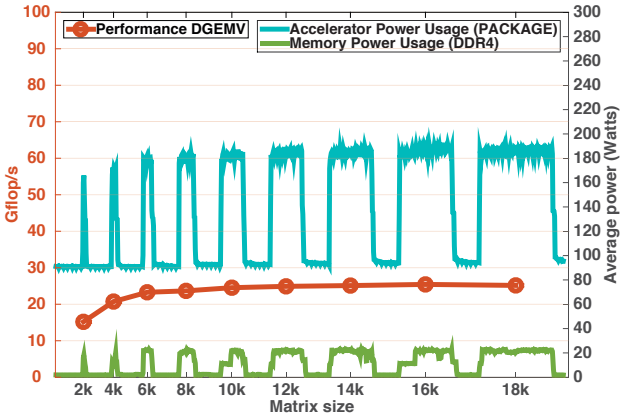
B. Study of the dgemv kernel behavior

Bandwidth-limited applications are characterized by memory-bound algorithms that perform relatively few FLOPs per memory access. For these type of routines (low arithmetic intensity routines), the floating-point capabilities of a processor are generally not important, however, the memory bandwidth limits the application's performance. We created a set of experiments that is similar to our previous dgemm investigation, to analyze the power and the performance behavior of the memory bound dgemv kernel. Here, we expect different behavior when allocating data using the two different memory areas, the MCDRAM or the DDR4, which have very different bandwidths. We expect these differences to be illustrated for both KNL booting models (HYBRID and FLAT).

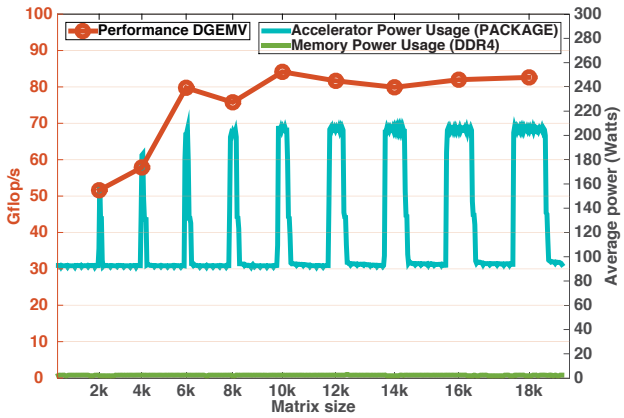
Figures 3a and 3b illustrate our measurements for the HYBRID mode. Because the kernel is memory-bound, there are limited benefits from cache reuse or from having blocks of data remain in fast memory. The performance drops dramatically between the two storage options. When allocating data in MCDRAM, the performance is about 3 times higher than when data is allocated in the DDR4. That is as predicted since the MCDRAM provides a bandwidth of about 400 GB/s versus around 90 GB/s for the DDR4. When the data is allocated in the DDR4, we can see a lower level of package power consumption than when data is in the MCDRAM. This can be explained by the fact that when data is in the MCDRAM, the elapsed time required to bring the data to the computational cores is less than when the data is in the DDR4, thus, within an interval of time the MCDRAM option brought more data allowing more work bringing up the power consumption. We can see that in the MCDRAM option the power can range between 200-210 Watt while it is about 180-190 Watt for the DDR4 option. In term of total energy, once we add the power of the DRAM to the power of the package, both MCDRAM and DDR4 consume about the same amount of power with 3X speedup in performance when using the MCDRAM option.

Figures 3c and 3d illustrate our measurements for the FLAT mode. The same observations as with the HYBRID mode can be reported here, except that for the DDR4 option the performance is about 4 times slower than for the MCDRAM option and slightly lower than its counterpart in HYBRID

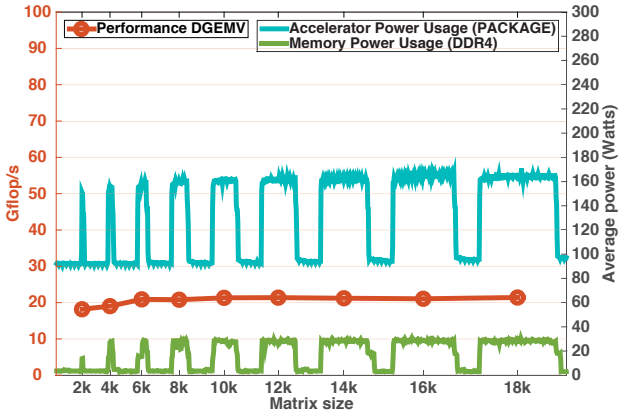
978-1-5386-3472-1/17/$31.00 ©2017 IEEE



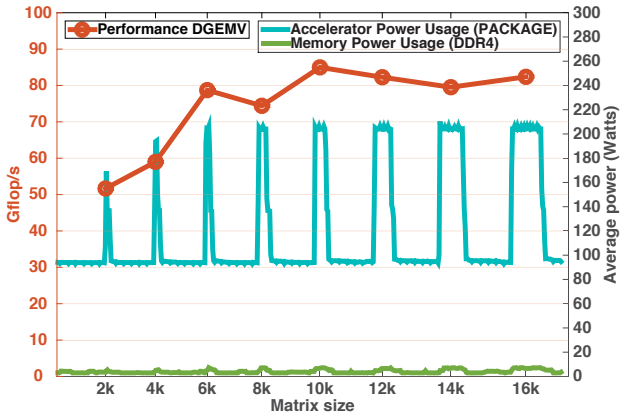
(a) HYBRID mode: dgemv: data allocated on DDR4.



(b) HYBRID mode: dgemv: data allocated on MCDRAM.



(c) FLAT mode: dgemv: data allocated on DDR4.



(d) FLAT mode: dgemv: data allocated on MCDRAM.

Fig. 3. Performance (GFlop/s on left axis) and average power (Watt on right axis) measurements of the BLAS-2 dgemv routine when the KNL is in either HYBRID or FLAT mode. The dgemv kernel is run successively for different problem sizes.

mode (figure 3a). We believe that is because in HYBRID mode both of the vectors x and y can be held in the cache portion of the MCDRAM.

The lesson learned is, for both power and performance optimization of memory-bound algorithms the MCDRAM option is always preferred, and a speedup of 3-4 times can be observed. If the data is relatively small (8GB<data<16GB) use FLAT mode and allocate the data in MCDRAM. Otherwise, if the data is large or needs to be stored in the DDR4, the HYBRID mode is preferable with a slight performance improvement of approx. 10%.

C. Power Capping as a Strategy for Improved Energy Efficiency

In this section we present the results of various power capping experiments. The objective is to utilize power control mechanisms in an effort to *save energy by keeping the execution time constant* (ideally).

Power usage and energy consumption is a very crucial topic in all scientific domains, in particular for high performance computing and big data centers. Energy efficiency becomes a major constraint for a computing center and it directly influences the operational budget. Data centers host hundreds of thousands of multi-core servers that provide many of our real life online services such as social networks, maps, big-data analytics, to name but a few. These centers host a

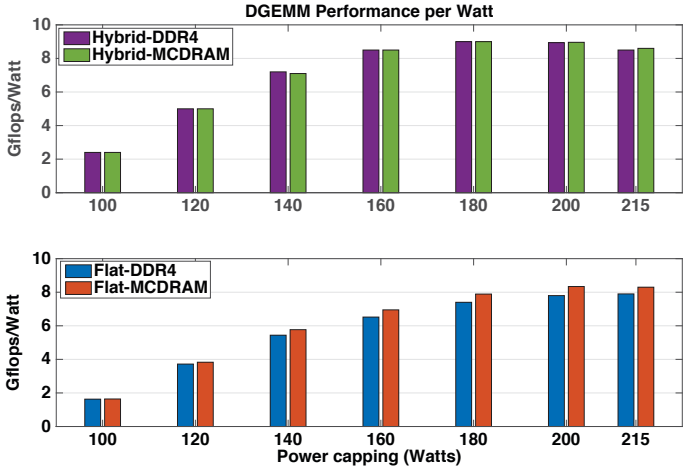


Fig. 4. Investigation of the energy efficiency of the BLAS-3 dgemm routine for various power caps. Performance is displayed in Gflops per Watt. Experiments are run on the KNL 7250 in HYBRID and FLAT mode.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

variety of applications that are constrained by memory-bound operations rather than by processor performance and thus it might be advantageous to throttle back the processor without affecting overall performance. Therefore, energy efficiency can be increased with the use of power capping techniques, which will result in reduced costs of operation. Moreover, large scale scientific applications running on HPC systems consist of different types of algorithms that can be compute or memory intensive and have different performance requirements. In this section, we focus our attention on studying the behavior of different compute or memory intensive benchmarks, and investigate how we can increase their productivity without decreasing their performance.

The new PAPI release provides the new component that enables transparent read and write access to power/energy information and controls through a simple and consistent interface. This PAPI `powercap` component uses the Linux powercap interface under the covers, which was designed to expose power constraint values for various Intel processors. Each processor has a different set of power constraints available via this Linux interface, and PAPI will only show the available constraints for the specific system that it is being run on. We performed extensive power capping studies with the representative linear algebra kernels to find good trade-off configurations.

Figures 4, 5 show the energy efficiency (performance per watt) for the BLAS kernels dgemm, dgemv respectively, computed on the Intel Xeon Phi Knights Landing processor. The energy efficiency was computed based on the asymptotic behavior of each routine, meaning when the performance of the routine reached around 90% of its practical peak. The energy efficiency is displayed in Gflops per Watt for various power capping values, ranging from 215 Watt (default power setting on the KNL) down to 100 Watt. For the power readings and writings, we are using the PAPI `powercap` component and the measurement utility that ships with the component, using a sampling rate of 100 milliseconds.

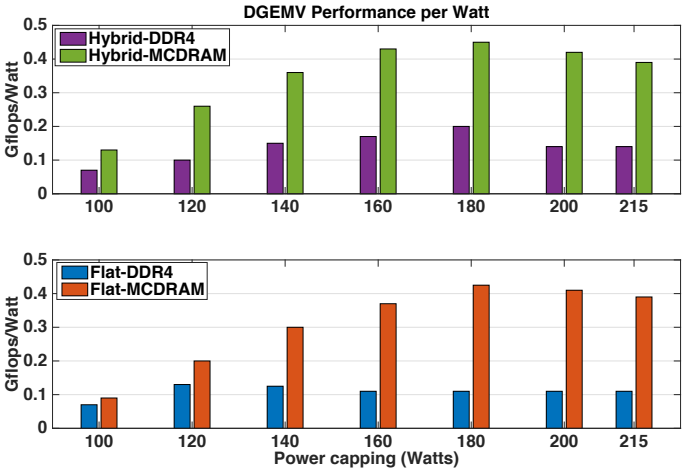


Fig. 5. Investigation of the energy efficiency of the BLAS-2 dgemv routine for various power caps. Performance is displayed in Gflops per Watt. Experiments are run on the KNL 7250 in HYBRID and FLAT mode.

The goal is to investigate which of the BLAS benchmarks *computes in the same amount of time with capped power, which ultimately results in energy saving*. The data is analyzed, evaluated, and the conclusions are summarized systematically for each benchmark. In general, power capping appears more beneficial when we use memory bound, low computational intensity kernels (like, e.g. dgemv) compared to compute bound kernels (like, e.g. dgemm).

In Figure 4, we observe that HYBRID mode enables energy savings by capping the power to either 180 Watt or 200 Watt for both DDR and MCDRAM option. Capping to 160 Watt will provide the same energy efficiency as a power of 215 Watt. For FLAT mode we can decrease the power requirements down to 180 Watt without any energy efficiency loss.

The more interesting energy behavior is observed for the memory bound routines. Figure 5 shows that we can save energy and gain performance/Watt by capping to 180 Watt for both DDR and MCDRAM option. Interestingly, the performance of the dgemv routine, in term of Gflop/s, is not affected when we cap to 180 Watt (as shown in Figure 6), meaning that an algorithm which is memory bound or that relies on dgemv operations (such as, finite element applications, iterative solvers like GMRES, conjugate gradient, preconditioning) will complete its computation within the same elapsed time but at a lower power. As a result, one can expect a significant energy saving, approx. 27%.

VII. Conclusion

We have used PAPI, with its newly developed `powercap` component, to explore the performance and power characteristics of various DLA kernels on the Intel Xeon Phi Knights Landing architecture. This exploration provides insights into how the available memory configurations on the KNL affect the performance of computation kernels with differing computational intensities. These insights allowed us to explore power management and capping strategies matching the computational intensities which can result in energy savings while keeping computational time constant.

Taking maximum advantage of the high bandwidth MCDRAM on the KNL is vital to achieving high performance and minimal power consumption, in particular for bandwidth-sensitive applications. We performed an extensive study of computational kernels from the highly used BLAS library, choosing kernels from each of the three BLAS levels in order to explore the performance and power characteristics of the system on compute-intensive and memory-bound computations. We explored the behavior of these kernels using different booting configurations for the MCDRAM on the KNL, using different data allocations to discover the best strategy in each booting scenario.

As a result from our analysis, we can make recommendations on the data allocation and on the memory mode that should be used for an application: If the application requires less than 8 GB, then the developer should allocate the data in the MCDRAM and the KNL can be used either in HYBRID mode or FLAT mode. Alternatively, if the data fits into the

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

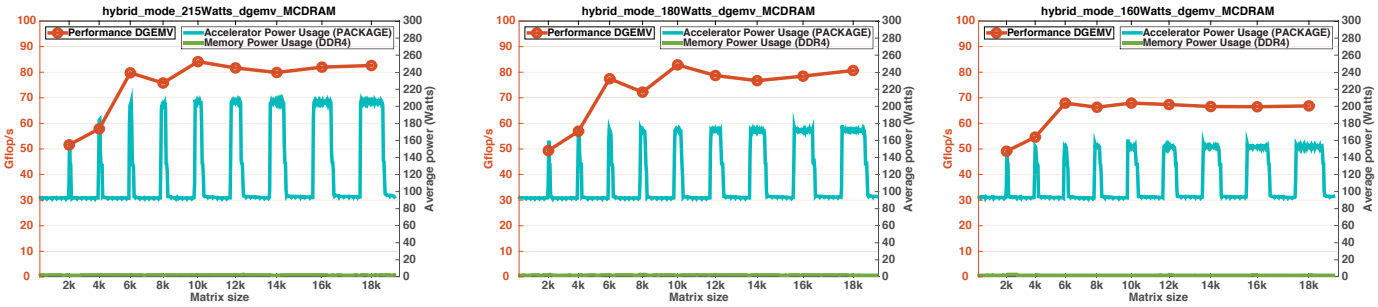


Fig. 6. Effect of different power caps (215, 180, 160 Watt) on the performance of the BLAS-2 dgemv routine for various problem sizes. Performance is displayed in Gflops per second. Experiments are run on the KNL 7250 in HYBRID mode.

16 GB of MCDRAM then the FLAT mode would be the best choice. If the data is larger than 16 GB then there are two possible paths: If the application is compute-intensive then the HYBRID mode is the best choice. Otherwise, the best path forward is to identify the memory bound portion of the code and allocate its data in the MCDRAM. One can use either HYBRID or FLAT mode based on the amount of memory bound data, that is, for memory bound data less than 8 GB choose HYBRID, otherwise choose FLAT mode.

ACKNOWLEDGMENTS

This material is based upon work supported in part by the National Science Foundation NSF under awards No. 1450429 “Performance Application Programming Interface for Extreme-scale Environments (PAPI-EX)” and No. 1514286.

REFERENCES

[1] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent. HPCTOOLKIT: tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience*, 22(6):685–701, 2010.

[2] H. Brunst and A. Knüpfer. Vampir. In D. Padua, editor, *Encyclopedia of Parallel Computing*, pages 2125–2129. Springer US, 2011.

[3] M. Burtscher, B.-D. Kim, J. Diamond, J. McCalpin, L. Koesterke, and J. Browne. PerfExpert: An Easy-to-Use Performance Diagnosis Tool for HPC Applications. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC ’10, pages 1–11, Washington, DC, USA, 2010. IEEE Computer Society.

[4] Using the PAPI Cray NPU Component. http://docs.cray.com/books/S-0046-10//S-0046-10.pdf.

[5] The CrayPat Performance Analysis Tool. http://docs.cray.com/books/S-2315-50/html-S-2315-50/z1055157958smg.html.

[6] M. Geimer, F. Wolf, B. J. N. Wylie, E. Ábrahám, D. Becker, and B. Mohr. The Scalasca performance toolset architecture. *Concurrency and Computation: Practice and Experience*, 22(6):702–719, Apr. 2010.

[7] Intel. Math kernel library. https://software.intel.com/en-us/en-us/intel-mkl/.

[8] H. Jagode, A. YarKhan, A. Danalis, and J. Dongarra. Power Management and Event Verification in PAPI. In *Tools for High Performance Computing 2015: Proceedings of the 9th International Workshop on Parallel Tools for High Performance Computing, September 2015, Dresden, Germany*, pages 41–51, Cham, 2016. Springer International Publishing.

[9] M. Johnson, H. Jagode, S. Moore, P. Mucci, J. Nelson, D. Terpstra, V. Weaver, and T. Mohan. PAPI-V: Performance monitoring for virtual machines. In *Proc of CloudTech-HPC Workshop*, Sept. 2012.

[10] R. Lucas, J. Ang, K. Bergman, S. Borkar, W. Carlson, L. Carrington, G. Chiu, R. Colwell, W. Dally, J. Dongarra, et al. Top ten exascale research challenges. *DOE ASCAC Subcommittee Report*, 2014.

[11] A. D. Malony, S. Biersdorff, S. Shende, H. Jagode, S. Tomov, G. Juckeland, R. Dietrich, D. Poole, and C. Lamb. Parallel performance measurement of heterogeneous parallel systems with gpus. In *Proceedings of the 2011 International Conference on Parallel Processing*, ICPP ’11, pages 176–185, Washington, DC, USA, 2011. IEEE Computer Society.

[12] H. McCraw, J. Ralph, A. Danalis, and J. Dongarra. Power Monitoring with PAPI for Extreme Scale Architectures and Dataflow-based Programming Models. In *Workshop on Monitoring and Analysis for High Performance Computing Systems Plus Applications (HPCMASPA 2014), IEEE Cluster 2014*, pages 385–391, Sep 2014.

[13] H. McCraw, D. Terpstra, J. Dongarra, K. Davis, and M. R. Beyond the CPU: Hardware Performance Counter Monitoring on Blue Gene/Q. In *Proceedings of the International Supercomputing Conference 2013*, ISC’13, pages 213–225. Springer, Heidelberg, June 2013.

[14] M. Schlütter, P. Philippen, L. Morin, M. Geimer, and B. Mohr. Profiling Hybrid HMPP Applications with Score-P on Heterogeneous Hardware. In *Parallel Computing: Accelerating Computational Science and Engineering (CSE)*, volume 25 of *Advances in Parallel Computing*, pages 773 – 782. IOS Press, 2014.

[15] S. S. Shende and A. D. Malony. The Tau Parallel Performance System. *Int. J. High Perform. Comput. Appl.*, 20(2):287–311, May 2006.

[16] D. Terpstra, H. Jagode, H. You, and J. Dongarra. Collecting Performance Data with PAPI-C. *Tools for High Performance Computing 2009*, pages pp. 157–173, 2009.

[17] S. Walker, K. Shoga, B. Rountree, and L. Morita. Libmsr, 2015. https://github.com/scalability-llnl/libmsr.