Application Power Profiling on IBM Blue Gene/Q

Sean Wallace,*† Venkatram Vishwanath,† Susan Coghlan,† John Tramm,† Zhiling Lan,* and Michael E. Papka†‡

*Illinois Institute of Technology, Chicago, IL, USA
†Argonne National Laboratory, Argonne, IL, USA
‡Northern Illinois University, DeKalb, IL, USA
swallac6@iit.edu, venkat@anl.gov, smc@anl.gov, jtramm@mcs.anl.gov, lan@iit.edu, and papka@anl.gov

***Abstract*—The power consumption of state of the art supercomputers, because of their complexity and unpredictable workloads, is extremely difficult to estimate. Accurate and precise results, as are now possible with the latest generation of supercomputers, are therefore a welcome addition to the landscape. Only recently have end users been afforded the ability to access the power consumption of their applications. However, just because it's possible for end users to obtain this data does not mean it's a trivial task. This emergence of new data is therefore not only understudied, but also not fully understood.**

In this paper, we provide detailed power consumption analysis of microbenchmarks running on Argonne's latest generation of IBM Blue Gene supercomputers, Mira, a Blue Gene/Q system. The analysis is done utilizing our power monitoring library, MonEQ, built on the IBM provided Environmental Monitoring (EMON) API. We describe the importance of sub-second polling of various power domains and the implications they present. To this end, previously well understood applications will now have new facets of potential analysis.

***Index Terms*—Power Profiling, Energy Efficiency, Blue Gene/Q**

I. Introduction

As the field of computational science continues to push towards the exascale era of supercomputing, power consumption has emerged as an increasingly vital area of research. The leading system on the November 2012 Top500 list [1], Titan, achieves 17.5 PFlops Rmax while consuming 8.2 MW of power. It is projected that exascale systems will be capped at 20 MW [2]. This implies that to achieve exascale, current supercomputers will need to scale their performance by ∼60X while increasing their power consumption by just ∼2X – a challenging task.

Hardware manufacturers already recognize this problem, and, by leveraging better design choices and tradeoffs, have made significant strides toward mitigating it. Software too will play an important role in ensuring power is not wasted, by dynamically managing power consumption across the system.

The majority of performance studies conducted on large-scale high performance computing (HPC) systems (including those published by the Green500 [3]) focus on the Flops/Watt metric. While this metric is useful for measuring the energy consumption of a particular architecture, it fails to convey much information about the individual components and how they affect the system on the whole. Moreover, it is also recognized that a different metric to measure energy consumption, namely *time to solution*, would likely result in different rankings[4]. Therefore, the analysis of power consumption on state-of-the-art supercomputers is imperative to better understand how these new systems differ from their predecessors and in which direction key characteristics are heading.

Fortunately, hardware manufacturers have begun deploying various sensors on HPC systems to collect power-related data and provide relatively easy access to the data store. In this work, we will describe two power monitoring capabilities deployed on the Blue Gene/Q (BG/Q): one is an environmental database and the other is vendor-supplied application programming interfaces (APIs) to profile power usage through the code of jobs actually running on the system. They provide information about the power consumption at two different scales.

The environmental database is maintained primarily to help identify and eliminate insufficient cooling as well as inadequate distribution of power. The power profiling APIs, on the other hand, provide power consumption data directly to the running process across more power domains with respect to components and at much finer granularity with respect to time. Unfortunately, these APIs only provide the total power consumption of all domains and are quite complicated for application developers to use.

In this work, we present the use of these vendor-deployed power-monitoring mechanisms on IBM Blue Gene/Q systems. In particular, we first provide analysis of six months of environmental data collected from the production 48-rack Blue Gene/Q at Argonne National Laboratory. This detailed power data analysis enables us to get a better idea of what trends this latest generation of supercomputer exhibits. Next, we describe our development of our power profiling library called MonEQ, currently in development. MonEQ is designed to address the aforementioned issues of the vendor-supplied APIs. Using MonEQ, application developers can easily obtain the individual voltage and current data points of all domains by inserting as few as two lines of code. MonEQ is available to developers and distributed as open source to the community. Finally, we demonstrate the use of this library for power profiling by integrating it into several proxy applications on Mira and provide detailed analysis of the power data statistics collected by the library.

The remainder of this paper is as follows: A review of the IBM Blue Gene architecture as well as the environmental

978-1-4799-0898-1/13/$31.00 ©2013 IEEE

data collection of the system is presented in Section II. A detailed environmental analysis from our experiences with BG/Q is presented in Section III. Section IV provides a detailed description of our profiling library, MonEQ. Section V presents case studies of benchmarks obtained from our profiling code. The related work is provided in Section VI. Finally, we will provide our conclusions and future work in Section VII.

II. Blue Gene/Q Architecture and Environmental Data Collection

The Blue Gene/Q architecture is described in detail in [5]. Our analysis of power usage on BG/Q is based on Argonne National Laboratory's 48-rack BG/Q system, Mira. A rack of a BG/Q system consists of two midplanes, eight link cards, and two service cards. A midplane contains 16 node boards. Each node board holds 32 compute cards, for a total of 1,024 nodes per rack. Each compute card has a single 18-core PowerPC A2 processor [6] (16 cores for applications, one core for system software, and one core inactive) with four hardware threads per core, with DDR3 memory. BG/Q thus has 16,384 cores per rack.

In each BG/Q rack, bulk power modules (BPMs) convert AC power to 48 V DC power, which is then distributed to the two midplanes. Blue Gene systems have environmental monitoring capabilities that periodically sample and gather environmental data from various sensors and store this collected information together with the timestamp and location information in an IBM DB2 relational database – commonly referred to as the environmental database [7]. These sensors are found in locations such as service cards, node boards, compute nodes, link chips, BPMs, and the coolant environment. Depending on the sensor, the information collected ranges from various physical attributes such as temperature, coolant flow and pressure, fan speed, voltage, and current. This sensor data is collected at relatively long polling intervals (about 4 minutes on average but can be configured anywhere within a range of 60-1,800 seconds), and while a shorter polling interval would be ideal, the resulting volume of data alone would exceed the server's processing capacity.

The Blue Gene environmental database stores power consumption information (in watts and amperes) in both the input and output directions of the BPM.

Figure 1 depicts how power is distributed from the power substation to the Mira BG/Q system. It shows the various measurement points along this path where we can monitor the power consumption.

III. Analysis of Environmental Power Usage on Blue Gene/Q

We analyzed six months of data starting in December 2012 and ending in May 2013. The number and rack utilization breakdown of this data is presented in Figure 2. All of the results in this section are presented from the point of view of an "average" job. That is, for a given metric, the average across all jobs was used for analysis. For the results that present

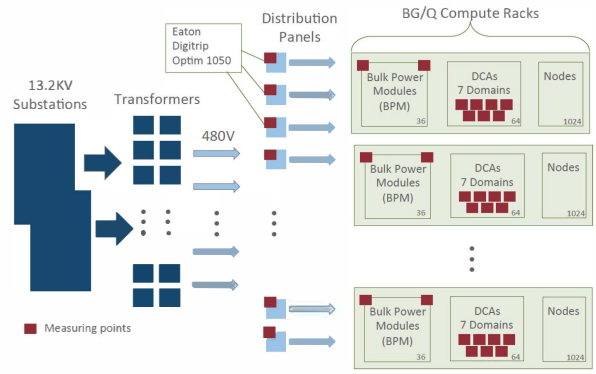

Fig. 1. Mira 480V compute power distribution system.

energy usage as a function of percentage of time, all of the jobs were normalized on their respective wall time.

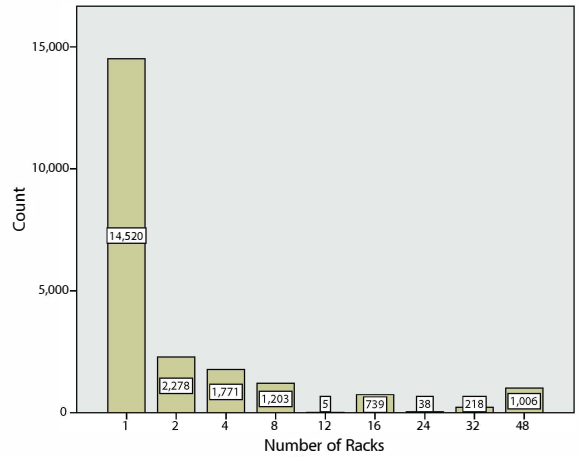

Fig. 2. Number of jobs run on n number of racks.

Figure 3 shows a boxplot of input power consumption per rack for an average job. The results show there is not a significant difference in power consumption between smaller system runs and larger ones.

Figure 4 shows the distribution of average kW/rack power consumption for the 21,778 jobs that were run during the span of six months. The histogram displays percentages partitioned separately by size ranging from single-rack jobs to full-system runs. Most jobs fall into the 60-74 kW per rack range, and very few jobs are at or above 80 kW per rack. The data shows that larger jobs (at or above 24 racks) tend to be in the 52 kW per rack bin. It is not yet known why larger jobs tend to use less power. One theory is that larger jobs are generally more prone to load imbalances and wait times due to communication which tends to decrease power consumption.

Note that this is the sum of the 48 V DC output of the BPMs so it does not include AC/DC conversion loss within the BPMs. Given our measured BPM conversion efficiency of about 94% over the six month span, the total average AC input

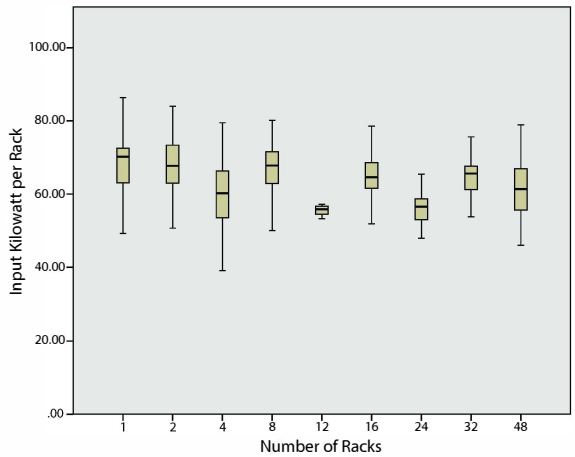

Fig. 3. Boxplot of kilowatt usage per rack. Shows no significant difference in jobs running on small or large number of racks. Average power consumption is 66.91 kW per rack.

per rack would be roughly 71 kW.

These results are different from those presented in our last work [8]. Average per-rack power consumption has dropped about 5% for all jobs measured and about 35% in the case of large jobs. In the case of the larger jobs, these results need to be put into context; in our previous work we had very few large run jobs to analyze. However, the reduction in energy consumption across the board is an interesting point. But, given the diversity of jobs run during this period, it's impossible to say without further analysis why the average power consumption might be lower.

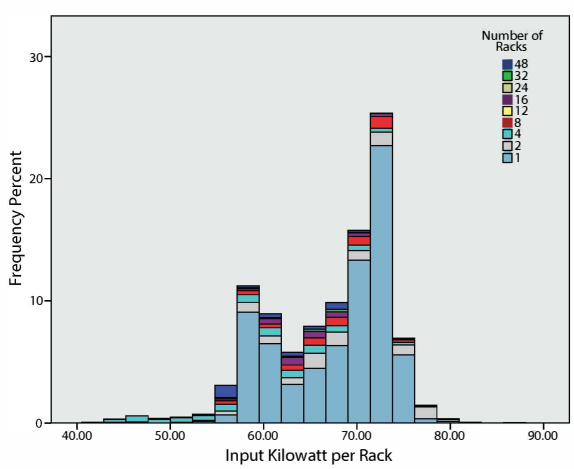

Fig. 4. Frequency distribution of average per-rack power consumption. Figure indicates most jobs fall into the 60-74 kW bins and all jobs are between 40 and 90 kW.

IV. MonEQ: A User-Level Power Profiling Library for Application Developers

For BG/Q, IBM provides new interfaces in the form of an environmental monitoring API called EMON that allows one to access power consumption data from code running on compute nodes, with a relatively short response time. However, EMON by itself is insufficient for our needs. The power information obtained using EMON *total* power consumption from the oldest generation. Furthermore, the underlying power measurement infrastructure does not measure all domains at the exact same time. This may result in some inconsistent cases, such as the case when a piece of code begins to stress both the CPU and memory at the same time. In this case, we might not see an increase in power for both domains in the same generation of data if there is a gap in time between when the CPU and memory power are measured. There is active research by IBM to improve the power monitoring systems, so this problem may change in the future.

To get past this limitation, we designed, MonEQ [9], a "power profiling library" that allows us to read the individual voltage and current data points. The exact domains and their corresponding BG/Q IDs are displayed in Table I.

In its default mode, MonEQ will pull data from the EMON API every 560 ms, which is currently the lowest polling interval possible; however users have the ability to set this interval to whatever valid value is desired. With the value of the polling interval set, MonEQ then registers to receive a `SIGALRM` signal at that polling interval. When the signal is delivered, MonEQ calls down to the EMON API and records the latest generation of power data available in an array local to agent rank of the node card.

Listing 1. Simple MonEQ Example

```
int status, myrank, numtasks, itr;

status = MPI_Init(&argc, &argv);

MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

/* Setup Power */

status = MonEQ_Initialize();


/* User code */

/* Finalize Power */

status = MonEQ_Finalize();

MPI_Finalize();
```

One limitation of the EMON API is that it can only collect data at the node card level (every 32 nodes). At first this might seem rather significant, however given the scale at which most applications are run (i.e., some number of racks) and the nature of large scale jobs, having 32 data points per rack is usually

sufficient. This limitation is part of the design of the system; as such, it is not possible to change the number of nodes for which data is collectible in software.

If the "automatic" mode of MonEQ has not been disabled, the collection will continue on in this fashion collecting power data for all of the domains from the initialization call (usually called shortly after `MPI_Init`) to the finalize call (usually called shortly before `MPI_Finalize`). If desired, the `MPI_Init` and `MPI_Finalize` calls can be overloaded to include automatic calls to MonEQ making integration completely automatic.

Listing 2. Simple MonEQ Example With Tags

```
int status, myrank, numtasks, itr;

status = MPI_Init(&argc, &argv);

MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

/* Setup Power */

status = MonEQ_Initialize();

/* User code */


status = MonEQ_StartPowerTag("work_loop");

/* Work loop */

status = MonEQ_EndPowerTag("work_loop");

/* User code */

/* Finalize Power */

status = MonEQ_Finalize();

MPI_Finalize();
```

As soon as finalize is called on MonEQ, it proceeds to dump all of the data it has collected in its working array to CSV files. A simple example of how MonEQ is implemented is shown in Listing 1. Before collecting data the user can specify how the output files should be partitioned according to two modes. In the first mode, the number of CSV files will depend on the number of node cards utilized by the application (i.e., one per node card). In the second, the user can specify to have output files generated based on the ranks of the MPI program. Thus, if it was desirable to have the power data from ranks 1-10 reported in one file and ranks 11-32 in another, that is possible with MonEQ. It should be noted this does not get around the "one data point per node card per unit of time" limitation. If this feature is utilized, MonEQ will bunch ranks running on a node card together just the same as it always would, but upon call of the finalize method data is aggregated for the appropriate ranks into a single file. That is, if ranks 1-32 are running on a node card and the user wants files for ranks 1-10 and 11-32, both files will contain the same data.

If automatic collection is disabled, power data is no longer dumped to a flat file at the finalize call. Instead, the application being profiled by MonEQ can access the data being collected through a set of abstracted calls. As one might expect, this data is now available to the running application during the execution of the program. Thus, if desired, a program could alter itself during execution based on its power profile. While this is possible, we have not performed any experimentation of this nature in this work.

Oftentimes application developers have logically and functionally distinct portions of their software which is of primary interest for profiling. To address this, we have implemented a tagging feature. An example of how tagging is implemented is shown in Listing 2. This feature allows for sections of code to be wrapped in start/end tags which inject special markers in the output files for later processing. In this way, if an application had three "work loops" and a user wanted to have separate profiles for each, all that is necessary is a total of 6 lines of code. Better yet, because the injection happens after the program has completed, the overhead of tagging is almost negligible.

Regardless if tagging is utilized, it's possible to end up with a substantial amount of data (a full system run for 5 minutes at the default polling interval will produce 1,536 files and a total of about 830,000 data points). To address this we wrote two more pieces of software: a reducer and an analyzer. The reducer, as the name suggests, takes as input the output files generated by MonEQ and reduces them into a single CSV file. If tags are utilized, this is when they are parsed; the reducer will create a CSV file for each tag, an overall file containing all of the data points, and an "untagged" file containing all of the data points that did not fall between tags. Once the output has been reduced it can then be more easily analyzed. Because the data is still in its original form, if a user wishes to analyze the data themselves, they may do so. However, if the user has no desire to analyze the data themselves, they can utilize our analyzer. Utilizing gnuplot and R, it will produce power utilization graphs as a function of time for all of the reduced files as well as descriptive statistics. Examples of this output can be found in the following subsections.

TABLE I
NODE BOARD POWER DOMAINS

Domain ID	Description
1	Chip Core Voltage
2	Chip Memory Interface and DRAM Voltage
6	HSS Network Transceiver Voltage Compute+Link Chip
7	Chip SRAM Voltage
3	Optics
4	Optics + PCIExpress
8	Link Chip Core

V. CASE STUDIES

Benchmarks are a crucial part of understanding the capabilities of a supercomputing system. The Argonne Leadership Computing Facility (ALCF) staff have a variety of benchmarks

designed to test both the capabilities and limitations of each system they deploy. Some of these benchmarks are designed based on real science applications that are commonly run on the system while others are designed simply to push components to their maximum capability.

In this work we will provide detailed analysis of three benchmarks: MPI bisection, `DGEMM`, and `XSBench`. Each are intended to test a very specific portion of the system and as such serve as excellent test platforms for profiling. For each we will show two figures: the first will be a breakdown of the various domains and how much they contributed to the total power consumption; the second a power profile from MonEQ from beginning to end of execution.

A. MPI Bisection

The bisection bandwidth benchmark [10] studies *the minimum bisection bandwidth*, defined as the minimum delivered MPI bidirectional bandwidth over all possible bisections. The *minimum bisection* is constructed by cutting a partition of nodes into two equally-sized, non-overlapping sections in such a way that the resulting cut yields the smallest number of links connecting both sections. This benchmark measures the total bi-directional bandwidth by sending and receiving messages using non-blocking communication between pairs of nodes in different sections.

This benchmark was run on 4 racks with 16 ranks per node and produced an aggregate bandwidth of 35.48 GB/s.

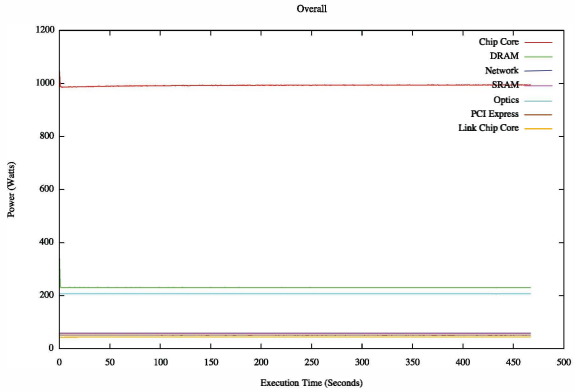

Fig. 5. Power profile of MPI bisection benchmark run for about 8 minutes. Shows consistent power usage for across all domains for the duration of the run.

Figure 5 shows a complete run of the bisection benchmark. As a consequence of only providing intense network activity, most of the domains are relatively close to idle power consumption for the duration of the run. Even despite this network activity, the domains that contribute the most to the power consumption of the network are very close to what they are at idle. Based on this result it is fair to say that the domains which contribute the most to communication power consumption have low variance in their power profiles. Despite this, it can be seen in Figure 6 that certain domains, such as the optics and network, contribute enough extra power consumption to cause a noticeable difference in percentage.

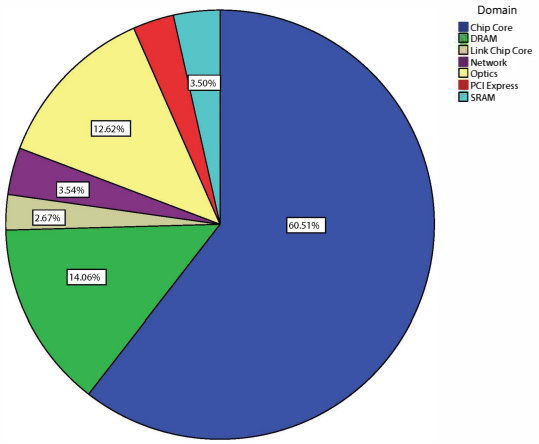

Fig. 6. Domain breakdown of MPI bisection benchmark. Shows slightly elevated consumption in optics and network domains.

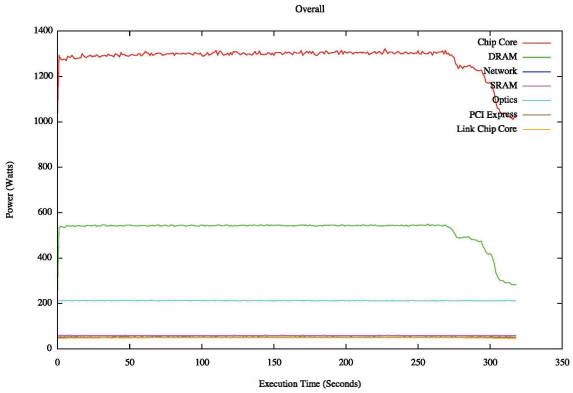

Fig. 7. Power profile of DGEMM benchmark run for about 5 minutes. Shows increased chip core and DRAM power usage for duration of execution tapering off as program cleans up at end of execution.

B. DGEMM

While the MPI bisection benchmark provides an excellent example of intense network activity, it does not provide a good picture of the power consumption of the memory in the system. A very well known memory intensive benchmark, `DGEMM` [11], was therefore chosen for profiling. `DGEMM`, a matrix multiplication microbenchmark, is designed to overload the caches present on the CPU and randomly access data in very large matrices therefore causing massive DRAM activity.

This benchmark was run on 4 racks with 16 ranks per node with an aggregate of 426.203 TFlops.

In contrast to the MPI bisection benchmark, `DGEMM` provides a more interesting overall picture in Figure 7. Immediately it can be seen that both the chip core and DRAM domains are using more power in `DGEMM`, which is also reflected in the domain breakdowns in Figures 6 and 8. It can also be seen that

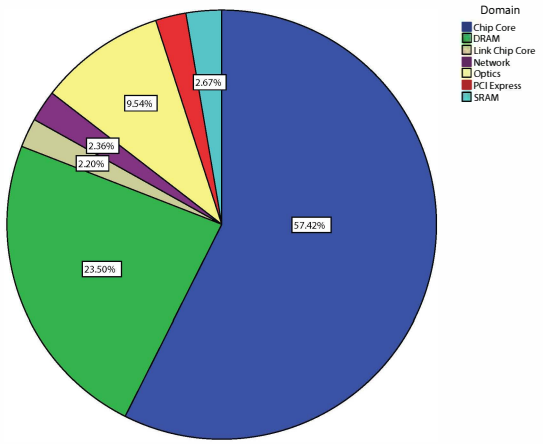

Fig. 8. Domain breakdown of DGEMM benchmark. Shows more significant power consumption in both the chip core and DRAM domains.

towards the end of execution of DGEMM both the chip core and DRAM taper off back to a level closer to idle. This is because the last thing this benchmark does before terminating is free any resources it used during simulation.

C. XSBench

XSBench [12] is a simple application that executes only the most computationally expensive steps of Monte Carlo particle transport, the calculation of macroscopic cross sections, in an effort to expose bottlenecks within multi-core, shared memory architectures.

In a particle transport simulation, every time a particle changes energy or crosses a material boundary, a new macroscopic cross section must be calculated. The time spent looking up and calculating the required cross section information often accounts for well over half of the total active runtime of the simulation. XSBench uses a unionized energy grid to facilitate cross section lookups for randomized particle energies. There are a similar number of energy grid points, material types, and nuclides as are used in the Hoogenboom-Martin benchmark. The probability of particles residing in any given material is weighted based on that material's commonality inside the Hoogenboom-Martin benchmark geometry.

The end result is that the essential computational conditions and tasks of fully featured Monte Carlo transport codes are retained in XSBench, without the additional complexity and overhead inherent in a fully featured code. This provides a much simpler and clearer platform for stressing different architectures, and ultimately for making determinations as to where hardware bottlenecks occur as cores are added.

This benchmark was run on 4 racks with 16 OpenMP threads per MPI rank. The result of the benchmark was an average of 513,749 lookups per MPI rank per second.

As with the bisection and DGEMM benchmarks, an overview of XSBench is presented in Figure 9 and the breakdown of the domains and their respective power usage is provided in

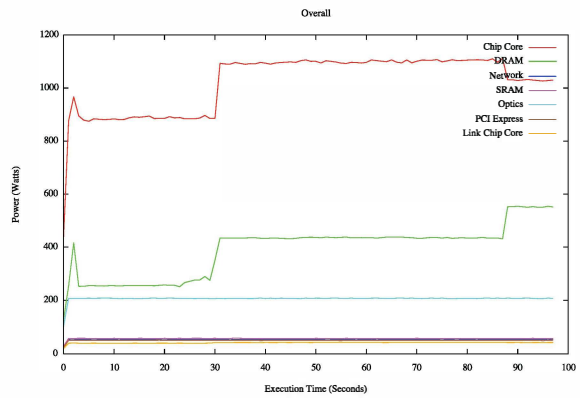

Fig. 9. Power profile of XSBench benchmark run for about 2 minutes. Shows two points of interest at 30 seconds when program starts generating data and 90 seconds when it switches from generation to random table lookups.

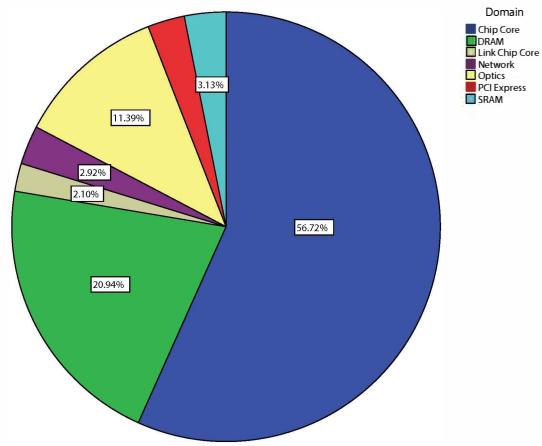

Fig. 10. Domain breakdown of XSBench benchmark. Shows elevated percentages in both chip core and DRAM domains.

Figure 10. As can be seen clearly, XSBench has two points of interest. The first 30 seconds are spent setting up the program, at which point data generation starts. This is expressed as an uptick in both the chip core and DRAM domains. The second point of interest, at about 90 seconds into the execution, is when XSBench switches from generating the data in the matrices to performing the random table lookups described above. These lookups result in a decrease in chip core power usage and an increase in the DRAM power usage, as expected.

As discussed in our previous works, one feature we wanted to add to MonEQ was the ability to tag specific portions of code that we have since implemented in this work. In Figure 11 we show the result of placing tags around the "work loop" which is the portion of the benchmark that performs the random table lookups.

Because of the nature and design of these benchmarks, it's not difficult to look at the code and figure out why there might be a significant change in the power profile of a given

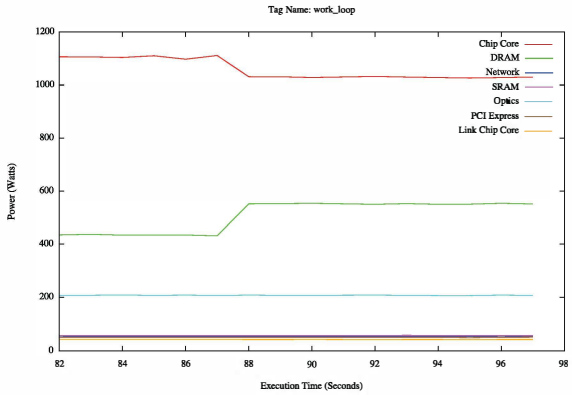

Fig. 11. Tagged "work loop" section of XSBench benchmark. Shows greater detail for portion of profiled code.

domain at a certain point in time. However, the ability to wrap tags around newly profiled applications can provide an easy way to determine which portions of the application constitute dramatic shifts in power usage.

VI. Related Work

Research in energy-aware HPC has been active in recent years, and existing studies have mainly focused on the following topics: power monitoring and profiling energy-efficient or energy-proportional hardware, dynamic voltage and frequency scaling (DVFS) techniques, shutting down hardware components at low system utilizations, power capping, and thermal management. These studies however focus on evaluating power consumption of individual hardware components and neglect to consider the system as a whole [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23].

From the system-level perspective, power consumption of HPC systems has increasingly become a limiting factor as running and cooling large computing systems comes with significant cost [24], [25], [26].

Hennecke et al. [27] provided an overview of the power measuring capabilities of Blue Gene/P (BGP). The measured power consumption a production workload of HPC applications and presented the integration of power and energy. However, no in-depth analysis on the accuracy was presented in that study.

Alam et al. [28] measured power and performance results of different kernels and scientific applications on BGP. They also compared these results to other large-scale supercomputers such as Cray's XT4. They concluded that while BGP has good scalability and better performance-per-watt characteristics for certain scientific applications, XT4 offers higher performance per processor.

Yoshii et al. [29] evaluated early access power monitoring capabilities on IBM BG/Q utilizing the EMON API. While they did provide an in-depth analysis of the monitoring capabilities of BG/Q, they did not analyze data from actual jobs that had run on the system as at the time only access to a single rack of BG/Q was available. This work also was limited to profiling only one MPI rank per node. We have overcome this in our monitoring mechanism and can profile any number of MPI ranks per node. Typical applications use 8-16 MPI ranks per node and MonEQ can now be used to profile applications on the BG/Q system. We have also defined an API for applications to use. Additionally, features include tagging to profile selective code fragments. Our earlier work didn't provide any of these features. Previously no tools were available for analysis and the task of dealing with the numerous output files was left up to the user. Our current work includes several utilities to analyze and plot the profiled data. Finally, the current work has demonstrated scalable performance up to 128K cores of BG/Q.

In our previous work [8] we further evaluated the power monitoring capabilities on BG/Q by providing an in depth comparison between the environmental data and data obtainable from the EMON API. We seek to further our experiments in this work by providing both updated analysis of the environmental data (now that actual science applications have been running for a few months) and further analysis of the EMON API on more benchmarks.

VII. Conclusions and Future Work

In this paper we provided an updated analysis of the power monitoring capabilities of an IBM Blue Gene/Q supercomputer. While these capabilities were designed primarily for monitoring of system health and environmental monitoring, they are also incredibly useful for profiling applications and helping to make better design decisions. We found that compared to our previous results, average power usage per job has decreased, though it is not possible to say exactly why.

We also profiled, using our power monitoring library MonEQ, several well understood benchmarks to better understand what power profiles they exhibit. As expected, these benchmarks, which are designed to stress very specific portions of the system, had power profiles to match. In this work we looked at three benchmarks: MPI bisection, `DGEMM`, and `XSBench`, which are designed to stress communication, memory, and computation abilities of the system. Our profiling mechanism, MonEQ, was easily integrated into these benchmarks and able to provide sub-second, accurate, and precise power profiles, which have not been studied before.

Looking forward, we will continue to work with and improve our power-profiling code. Given enough power profiles, it might be eventually possible to match real science applications to well understood benchmarks allowing for potential improvement in both efficiency as well as performance.

Acknowledgments

This research has been funded in part and used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357. The research has been funded in part by the Center for Exascale Simulation of Advanced Reactors

(CESAR) Co-Design center, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

The authors would also like to thank Paul Coteus and Christopher M. Marroquin from IBM for their help in clarifying results as well as providing essential information of the inner workings of the EMON system. The authors thank Vitali Morozov, Kalyan Kumaran, and the ALCF staff for their help. They also gratefully acknowledge the help provided by the application teams whose codes are used herein.

References

[1] "The Top500 List," November 2012. [Online]. Available: http://www.top500.org/list/2012/11/

[2] DOE, "Architectures and technology for extreme scale computing," December 2009.

[3] "The Green500 List," November 2012. [Online]. Available: http://www.green500.org/lists/green201211

[4] C. Bekas and A. Curioni, "A new energy aware performance metric," *Computer Science - Research and Development*, vol. 25, pp. 187–195, 2010. [Online]. Available: http://dx.doi.org/10.1007/s00450-010-0119-z

[5] IBM, "Introduction to Blue Gene/Q," 2011. [Online]. Available: http://public.dhe.ibm.com/common/ssi/ecm/en/dcl12345usen/DCL12345USEN.PDF

[6] R. Haring, M. Ohmacht, T. Fox, M. Gschwind, D. Satterfield, K. Sugavanam, P. Coteus, P. Heidelberger, M. Blumrich, R. Wisniewski, A. Gara, G.-T. Chiu, P. Boyle, N. Chist, and C. Kim, "The IBM Blue Gene/Q compute chip," *Micro, IEEE*, vol. 32, no. 2, pp. 48 –60, march-april 2012.

[7] G. Lakner and B. Knudson, *IBM System Blue Gene Solution: Blue Gene/Q System Administration*. IBM Redbooks, June 2012. [Online]. Available: http://www.redbooks.ibm.com/abstracts/sg247869.html

[8] S. Wallace, V. Vishwanath, S. Coghlan, Z. Lan, and M. Papka, "Measuring power consumption on IBM Blue Gene/Q," in *The Ninth Workshop on High-Performance, Power-Aware Computing, 2013 (HPPAC'13)*, Boston, USA, May 2013.

[9] "MonEQ: Power monitoring library for Blue Gene/Q," https://repo.anl-external.org/repos/PowerMonitoring.

[10] V. Morozov, K. Kumaran, V. Vishwanath, J. Meng, and M. Papka, "Early experience on the IBM Blue Gene/Q system," *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2013)*, May 2013.

[11] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff, "A set of level 3 basic linear algebra subprograms," *ACM Trans. Math. Softw.*, vol. 16, no. 1, pp. 1–17, Mar. 1990. [Online]. Available: http://doi.acm.org/10.1145/77626.79170

[12] J. Tramm and A. R. Siegel, "Memory Bottlenecks and Memory Contention in Multi-Core Monte Carlo Transport Codes," *Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013 (SNA + MC 2013)*, Oct. 2013.

[13] B.-G. Chun, G. Iannaccone, G. Iannaccone, R. Katz, G. Lee, and L. Niccolini, "An energy case for hybrid datacenters," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 1, pp. 76–80, Mar. 2010. [Online]. Available: http://doi.acm.org/10.1145/1740390.1740408

[14] V. W. Freeh, D. K. Lowenthal, F. Pan, N. Kappiah, R. Springer, B. L. Rountree, and M. E. Femal, "Analyzing the energy-time trade-off in high-performance computing applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 6, pp. 835–848, Jun. 2007. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2007.1026

[15] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. Cameron, "PowerPack: Energy profiling and analysis of high-performance systems and applications," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 5, pp. 658 –671, may 2010.

[16] I. Goiri, K. Le, M. Haque, R. Beauchea, T. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "GreenSlot: Scheduling energy consumption in green datacenters," in *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, nov. 2011, pp. 1 –11.

[17] K. Kant, M. Murugan, and D. Du, "Willow: A control system for energy and thermal adaptive computing," in *Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International*, may 2011, pp. 36 –47.

[18] J. Laros, K. Pedretti, S. Kelly, J. Vandyke, K. Ferreira, C. Vaughan, and M. Swan, "Topics on measuring real power usage on high performance computing platforms," in *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, 31 2009-sept. 4 2009, pp. 1 –8.

[19] O. Mammela, M. Majanen, R. Basmadjian, H. Meer, A. Giesler, and W. Homberg, "Energy-aware job scheduler for high-performance computing," *Comput. Sci.*, vol. 27, no. 4, pp. 265–275, Nov. 2012. [Online]. Available: http://dx.doi.org/10.1007/s00450-011-0189-6

[20] V. Patil and V. Chaudhary, "Rack aware scheduling in HPC data centers: An energy conservation strategy," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, may 2011, pp. 814 –821.

[21] Q. Tang, S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 11, pp. 1458 –1472, nov. 2008.

[22] T. V. T. Duy, Y. Sato, and Y. Inoguchi, "Performance evaluation of a green scheduling algorithm for energy savings in cloud computing," in *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, april 2010, pp. 1 –8.

[23] Z. Xu, Y.-C. Tu, and X. Wang, "Exploring power-performance tradeoffs in database systems," in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, march 2010, pp. 485 –496.

[24] N. Rasmussen, "Calculating total cooling requirements for data centers," White Paper, April 2011.

[25] W. chun Feng and K. Cameron, "The Green500 list: Encouraging sustainable supercomputing," *Computer*, vol. 40, no. 12, pp. 50 –55, dec. 2007.

[26] E. Pakbaznia, M. Ghasemazar, and M. Pedram, "Temperature-aware dynamic resource provisioning in a power-optimized datacenter," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, march 2010, pp. 124 –129.

[27] M. Hennecke, W. Frings, W. Homberg, A. Zitz, M. Knobloch, and H. Böttiger, "Measuring power consumption on IBM Blue Gene/P," *Comput. Sci.*, vol. 27, no. 4, pp. 329–336, Nov. 2012. [Online]. Available: http://dx.doi.org/10.1007/s00450-011-0192-y

[28] S. Alam, R. Barrett, M. Bast, M. R. Fahey, J. Kuehn, C. McCurdy, J. Rogers, P. Roth, R. Sankaran, J. S. Vetter, P. Worley, and W. Yu, "Early evaluation of IBM BlueGene/P," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, ser. SC '08. Piscataway, NJ, USA: IEEE Press, 2008, pp. 23:1–23:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=1413370.1413394

[29] K. Yoshii, K. Iskra, R. Gupta, P. Beckman, V. Vishwanath, C. Yu, and S. Coghlan, "Evaluating power-monitoring capabilities on IBM Blue Gene/P and Blue Gene/Q," in *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, sept. 2012, pp. 36 –44.