# Measuring Power Consumption on IBM Blue Gene/Q

Sean Wallace,*† Venkatram Vishwanath,† Susan Coghlan,† 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, lan@iit.edu, and papka@anl.gov

*Abstract*—**In addition to pushing what is possible computationally, state-of-the-art supercomputers are also pushing what is acceptable in terms of power consumption. Despite hardware manufacturers researching and developing efficient system components (e.g., processor, memory, etc.), the power consumption of a complete system remains an understudied research area. Because of the complexity and unpredictable workloads of these systems, estimating the power consumption of a full system is a nontrivial task.**

**In this paper, we provide system-level power usage and temperature analysis of early access to Argonne's latest generation of IBM Blue Gene supercomputers, the Mira Blue Gene/Q system. The analysis is provided from the point of view of jobs running on the system. We describe the important implications these system level measurements have as well as the challenges they present. Using profiling code on benchmarks, we will also look at the new tools this latest generation of supercomputer provides and gauge their usefulness and how well they match up against the environmental data.**

## I. Introduction

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

Hardware manufacturers already recognize this problem, and, by leveraging better design choices and tradeoffs, have made significant strides toward mitigating this problem. Improvements to hardware alone will not be enough though, software will also have a big role to play namely in ensuring power is not wasted by dynamically managing power consumption across the system.

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

Fortunately, hardware manufacturers are starting to deploy various sensors on HPC systems to collect power-related data as well as providing relatively easy access to the data they collect. In this work, we will describe two power monitoring capabilities deployed on the Blue Gene/Q (BGQ): one is an environmental database and the other is profiling code built on 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 profiling code, on the other hand, provides 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.

In this paper, we compare the data collected from both sources in order better understand their usefulness. We provide analysis of data from one month of jobs run on the system to get a better idea of what trends this latest generation of supercomputer exhibits. To achieve this, we integrated a power-profiling library with an existing benchmark and compared the environmental data collected by the system to the data gathered by our profiling code as a means to gain a deeper understanding of what is possible with more data.

The rest of this paper is as follows: After reviewing the related work in Section II and IBM Blue Gene architecture as well as the environmental data collection of the system in Section III, we will provide detailed environmental analysis from our early experiences with BGQ in Section IV. We will also show a sample of data collected from the new and improved power-measuring capabilities of BGQ and provide comparison to the data found in the environmental database in Section V. Finally, we will provide our conclusions and future work in Section VI.

## II. 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 [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15].

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 [16], [17], [18].

Hennecke et al. [19] 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. [20] 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. [21] evaluated early access power monitoring capabilities on IBM BGQ utilizing the EMON API. While they did provide an in-depth analysis of the monitoring capabilities of BGQ, they did not analyze data from actual jobs that had run on the system.

## III. Blue Gene/Q Architecture and Environmental Data Collection

The Blue Gene/Q architecture is described in detail in [22]. Our analysis of power usage on BGQ is based on Argonne National Laboratory's 48-rack BGQ system called Mira. A rack of a BGQ 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 [23] (16 cores for applications, one core for system software, and one core inactive) with four hardware threads per core, with DDR3 memory. BGQ thus has 16,384 cores per rack.

In each BGQ 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 which 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; This is commonly referred to as the environmental database [24]. These sensors are found in locations such as service cards, node boards, the compute nodes themselves, 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 volume of data and stress that would be imparted on the database would exceed the server's processing capacity.

The Blue Gene database stores power consumption information (in watts and amperes) and temperature information (in degrees Celsius) for the following components that are of particular interest to us in this paper:

- *Bulk* - AC/DC converter on a rack—power consumption is measured in both the input and output directions
- *Node Board* - Temperature
- *Node* - Temperature
- *Link Card* - Temperature
- *Service Card* - Temperature
- *Coolant* - Temperature sensors located between the inlet and outlet pipes

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

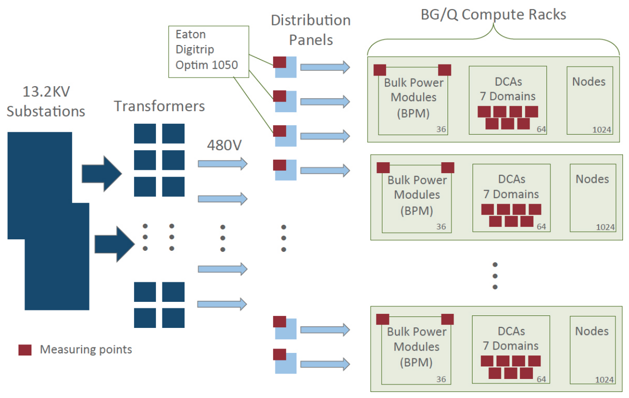

Fig. 1. Mira 480V compute power distribution system.

## IV. Analysis of Environmental Power Usage and Temperature on Blue Gene/Q

The data analyzed is a one month sample from September 2012, during which time part of the system was undergoing stability and short testing and the rest of the system was in maintenance. All of the results 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 calculation. For the results that present energy usage as a function of percentage of time, all of the jobs were normalized on their respective wall time.

Stability testing was performed on half of the system near the last week of the month. The rest of the data is the result of jobs that were run on very specific portions of the system to verify operation of components. Since the system wasn't fully operational for the entirety of the month, there were significantly more jobs that ran on 8 racks or less than there were jobs that ran on more than 8 racks. However, as will be discussed in greater detail in the following sections, the power consumption per rack is relatively stable from single-rack jobs to full-system jobs. While this data is not identical to real system usage, it's important to get an idea of what power consumption looks like for a wide ranging number of activities. As more data becomes available from the early science application runs, these results will be a great point for comparison.

TABLE I
NUMBER OF JOBS RUN ON $n$ NUMBER OF RACKS

| Number of Racks | Number of Jobs |
|---|---|
| 1 | 1,308 |
| 2 | 539 |
| 4 | 318 |
| 8 | 328 |
| 16 | 90 |
| 24 | 1 |
| 32 | 6 |
| 48 | 4 |
| **Total** | 2,594 |

## A. Power

Figure 2 shows a boxplot of input power consumption per rack for an average job. As discussed earlier, while this data set does not include a large number of greater than 8 rack job runs, there isn't a significant difference in the power consumption between smaller system runs and larger ones.

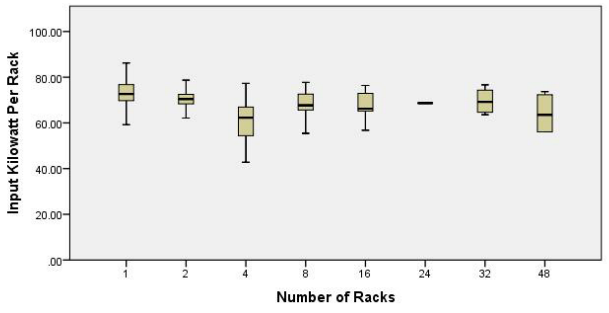

Fig. 2. Boxplot of kilowatt usage per rack. Shows no significant difference in jobs running on small or large number of racks. Average power consumption is 70.32 kW per rack.

Figure 3 shows the distribution of average kW/rack power consumption for the 2,594 jobs that were run during the month of September 2012. The histogram displays percentages partitioned separately by size ranging from single-rack jobs to full-system runs. Most jobs fall into the 65 to 75 kW per rack range, and very few jobs are at or above 85 kW per rack. While difficult to see in the histogram, the data shows that larger jobs (at or above 24 racks) tend to be in the 70 kW per rack bin.

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 a BPM conversion efficiency of about 94%, average AC input per rack would be roughly 74 kW.

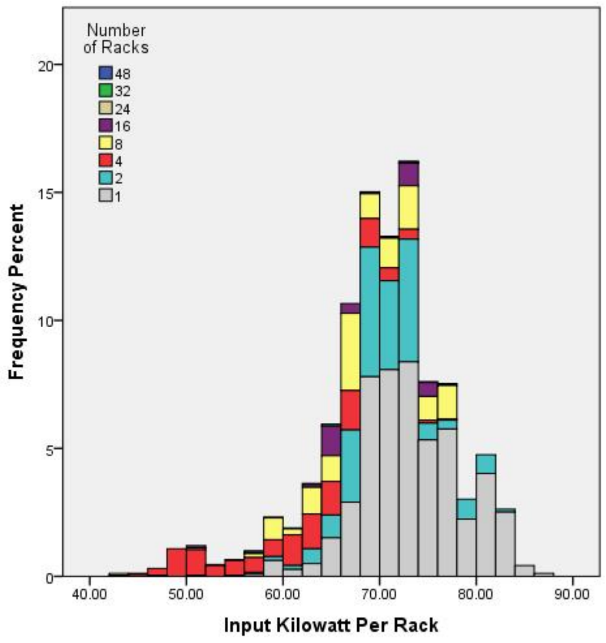

Fig. 3. Frequency distribution of average per-rack power consumption. Figure indicates most jobs fall into the 60-80 kW bins and all jobs are between 40 and 90 kW.

Figure 4 shows how the average input power measured at a given sensor on the BPMs changes as a function of a jobs execution time. The figure intriguingly shows that the power consumption of a job does slightly increase as a function of the amount of time it has been running. The first and last 5% are shown to be on average about 180 W less than the middle 90% and are likely influenced by the bootup and shutdown of the partition. Bootup times are mostly proportional to the size of the job to be run, but there is also further fluctuation due to environmental factors or otherwise. What's more, these bootup times haven't been precisely measured and analyzed over a large sample size, so any values we could have applied to further analyze the impact on power consumption would have been educated guesses at best. Surely these times are something that require further analysis in the future.

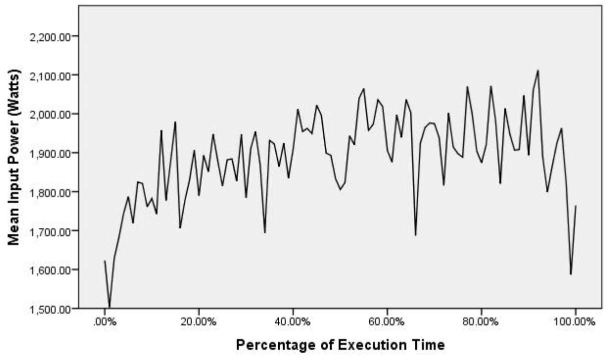

Fig. 4. Average power usage profile of a job running on system. Shows power on average does increase as a function of time. First and last 5% likely influenced by startup and shutdown times.

Figure 5 shows the efficiency of the BPMs as as a function of time. Efficiency is quite simply defined to be the ratio of output power to input power. Since AC power must be converted to DC power, there will always be some loss associated with the BPMs which begs the question, what impact does a running job have on efficiency? Coincidentally, that turns out to be a very difficult question to answer. As can be seen in the figure, there are several times during the execution of a given job when the efficiency goes over 100%.

As it turns out, BPM power monitoring was not implemented with the goal of reporting real-time conversion efficiency. This is mainly due to dissimilar sampling and averaging algorithms; input values have a much longer averaging window than output, so any transients will cause numbers to diverge from the actual conversion efficiency. What's more, the percentage of load on the BPMs also affects the percent error. The closer the load is to 100%, the less the percent error. More detailed information on the sampling and averaging parameters for the BPMs can be found in Table II.

These errors in reporting however do not mean that no useful information can be obtained from this data. With enough data, it is still possible to come to general conclusions about factors such as the conversion efficiency. In the case of this data, we calculated the conversion efficiency to be about 94% on average by taking the ratio of the input power to the output power for the entire month. As this percentage determines the amount of lost power as a result of conversion, the higher the value, the less power is wasted.

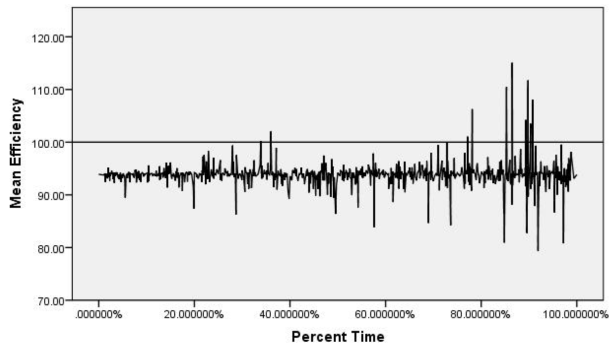

Fig. 5. AC/DC conversion efficiency of BPMs over time. As expected, the efficiency remains relatively constant over the course of execution for a given job. Because the sensors on the BPM are not designed for instantaneous readings, attempting to determine instantaneous efficiency readings is unrealistic and therefore results in incorrect values (i.e., over 100%). However, when looked at over a long period of time, it can be deduced that the BPMs maintain about 94% conversion efficiency.

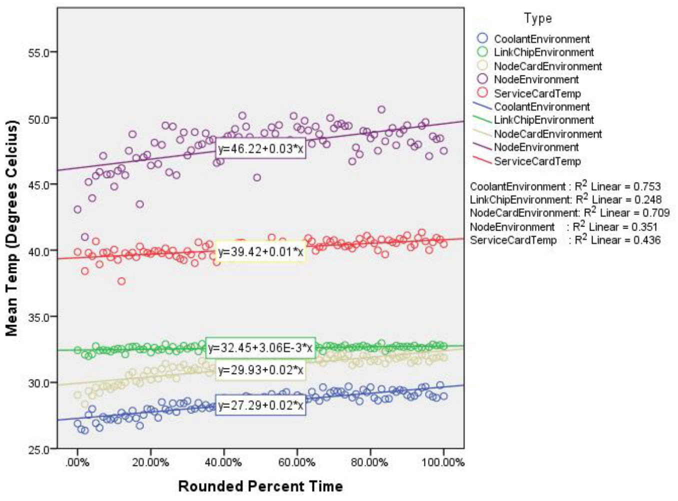

Fig. 6. Temperature fluctuation as a function of time. As expected, the system gets "hotter" as jobs run. Most sensors indicate a 2 to 3 degree increase of individual components. Bootup and shutdown times not taken into consideration because of fluctuation in those times, thus it is expected the temperature increase would be more dramatic than depicted.

## *B. Temperature*

In addition to power, temperature is also an important environmental factor that needs further analysis. On BGQ, there are a total of 5 temperature domains: coolant environment, link chip environment, node card environment, node environment, and service card environment.

Figure 6 shows how the temperature of each of the 5 temperature domains change over the course of a standard job's execution. Given the variance in bootup and shutdown times, which changes from job to job and is dependent on node allocation size, the bootup and shutdown times were not factored into these results. Also, because the scheduler tries to optimize utilization, jobs are frequently run back-to-back. It's likely that given the time to cool down adequately at idle, the temperature changes would be more dramatic. Except for the node environment, all other temperature domains are very strongly correlated with the regression line and all have a positive slope. This makes intuitive sense as the longer a job runs, the hotter the system on the whole gets. However, it's interesting that despite giving no consideration to cool down there is still an average of 2-3 degrees difference between the start and end of a job across almost all domains.

While this data is not directly correlatable to the data presented in Figure 4 due to differences in polling times for power and temperature sensors, it does show the same general trend that as a job continues to run, it both uses more power and as a result produces higher temperatures.

## V. Analysis of Blue Gene/Q EMON API Data

In this section, we will describe our power profiling code and discuss the results obtained by integrating this code into the MMPS benchmark [25]. We will also compare these power consumption results to those obtained from the environmental database at the BPM level.

On BGQ, IBM provides new interfaces in the form of an Environmental Monitoring (EMON) API that allows one to access power consumption data from code running on compute nodes, with a relatively short response time. However, the EMON API by itself is insufficient for our needs. The API only returns the total power consumption of all domains and does not contain any profiling functionality. To get past this limitation, we designed, MonEQ [26], a "power profiling

TABLE II
SAMPLING AND AVERAGING PARAMETERS FOR BPMS

| | **ADC-Sampling** | **Averaged Value** | **Update Reported Value** |
|---|---|---|---|
| Output Voltage | Every 500 $\mu$s | 160 ms (32 Values Every 5 ms) | Every 100ms |
| Output Current | Every 1ms | 8ms (8 Values Every 1ms) | Every 100ms |
| Input Voltage | Every 200 $\mu$s (On Prim $\mu$C) | 2,5s (38 Values - 6 Half Waves Averaged on Prim $\mu$C) | Every 100ms |
| Input Current | Every 200 $\mu$s (On Prim $\mu$C) | 2,5s (38 Values - 6 Half Waves Averaged on Prim $\mu$C) | Every 100ms |

library" that allows us to read the individual voltage and current data points. The exact domains and their corresponding BGQ IDs are displayed in Table III.

The EMON API is not without its faults, however. The power information obtained is *total* power consumption from the oldest generation. What's more, 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.

TABLE III
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 |

Our profiling code which utilizes the EMON API, and uses a timer implemented using the SIGALRM signal handler on each node board. This timer periodically invokes an "EMON system call" at an interval lower than the FPGA interval (i.e., 500ms), records the instantaneous power usage (Watts) across all available domains along with a timestamp, and proceeds to either dump this data to text files, or, populate a data structure available to the job. More details on the EMON API and the architecture of the node board are discussed in [21].

The `MMPS` benchmark helps us understand *how many messages can be issued per unit time*. It measures the interconnect messaging rate, which is the number of messages that can be communicated to and from a node within a unit of time. This gives a measure of the capability of the underlying implementation, both software and hardware, to process incoming and outgoing messages as quickly as possible. The selected message size is chosen very small so that the link bandwidth does not significantly influence the measurements. In this benchmark, a reference node communicates with all of its $k$ nearest neighbors that are located on the nodes one hop away on the torus. Each MPI task on the reference node communicates with the $k$ corresponding MPI tasks on the nearest neighbors by sending and receiving zero-length messages using non-blocking communications.

In our experimental setup, after successfully integrating our profiling code into the MMPS benchmark, the new profiled code was set to run on 512 nodes (one midplane) with 16 ranks per node for enough time to generate several data points from the BPMs. As previously discussed, the polling interval for the data inserted into the environmental database is quite long, and while this isn't a problem for our profiling code, in order to generate enough data at the environmental database level, we decided the application had to be run for a minimum of 20 minutes. While outside of the scope of this paper, the benchmark reported a maximum message rate of 9.87 million messages per second.

Figure 7 shows the resulting power consumption of the BPMs. Initially the node card is powered on but the compute nodes are not booted. This yields a standby power consumption of roughly 893 W for this experiment when the data is collected from the environmental database. When the partition is allocated by `mpirun`, the compute nodes are booted up causing an increase in node card power to about 1724 W, again reported from the environmental database. The figure clearly shows the point at which the system switches from standby, to running the application, and back to standby.

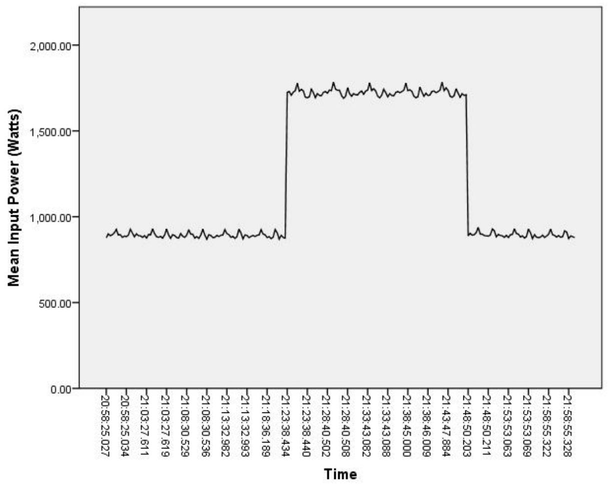

Fig. 7. Power usage as collected from the environmental database for the BPMs. Sharp increase/decrease shows point in time when application switches from stopped to running and back to stopped.

Figure 8 shows data obtained from our profiling code at the node card level (a summation of all 7 power domains). Since the current version of our profiling code only collects data when the application is in the running state, the times when the system is idle are not visible in the figure. However, similar to the data obtained from the BPMs, the data shows a very quick increase in power consumption at the node board level as the job starts to run. The power consumption then remains relatively stable as the program continues to execute. This data indicates that during program execution there was an average power consumption of 1,744 W per rank and thus, about 30 kW for the midplane. This means if we were to scale this benchmark to a full rack, we would expect the power usage to be about 60 kW. While this is about 10 kW less than we observed jobs to use per rack on average, this benchmark is designed to stress network communication and lacks other assets (such as high compute and memory utilization) commonly found in real-world applications which would surely increase total power consumption.

As expected, we see almost identical power usage numbers from our profiling code as we do in the environmental database. The slight difference between them is easily explained as a difference in polling intervals. As there are significantly more data points in the data obtained from our profiling code, it is the more "precise" of the two. Additionally, while the environmental database and our profiling code represent practically equivalent power usage, the higher level of detail afforded by our profiling code means we can draw much more precise conclusions on energy consumption especially as a function of time.

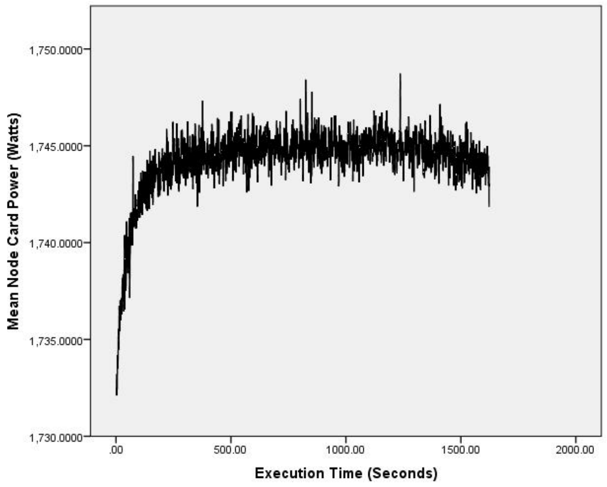

Fig. 8. Power usage of node card collected from profiling code. Because the EMON API allows for much faster sampling, the increase in power consumption when the job actually runs is not as dramatic as the figure from the BPM.

While there exists no data collected at the environmental database level for the network, given the nature of the MMPS benchmark it's worth showing what the power consumption looks like during execution. For this network-intensive benchmark, we combine the link chip core, optics, HSS network, and PCI express domains, which are all an integral part of the total network power consumption, in Figure 9. Although other domains such as the memory and the compute chip are certainly necessary to feed the network, they have been left out of this analysis to better focus on the components that directly influence network power usage. Unlike the data obtained from the node card, the network power consumption curve does not show a sharp jump when the application starts running. Instead, the power consumption remains within 1 W of the starting value. The average power consumption was calculated to be 360 W.

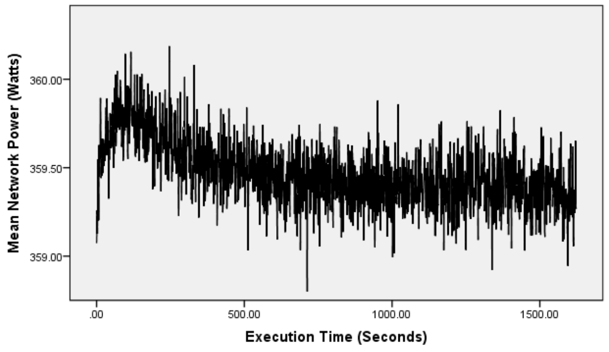

Fig. 9. Power usage of four network related domains collected from profiling code. Displays increasing then decreasing power curve at beginning of execution leveling out around 500 seconds. Stays within 1 W of starting value.

To better illustrate how the domains compare to each other as a result of running this benchmark, Figure 10 shows a pie chart of the relative percentages that each domain contributes to total power consumption. The heavy network activity is well demonstrated by the large portion of power that is used by the optics. Naturally, this offers a great point of comparison for other benchmarks designed to stress other portions of the system. With enough profiling, we could provide expected ranges across all of the domains for both times of high activity and low activity.

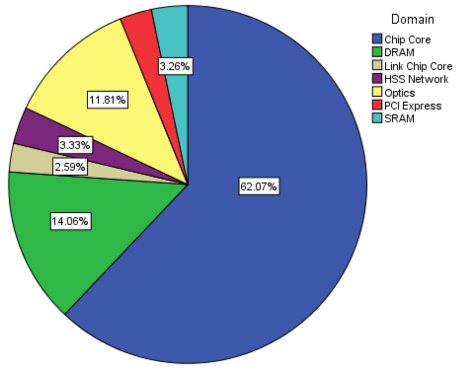

Fig. 10. Pie chart showing relative percentages of total power usage consumed by each of the 7 power domains. Intense network activity largely contributing to optics percentage.

## VI. Conclusions and Future Work

In this paper we evaluated the existing power monitoring capabilities of an IBM Blue Gene/Q system. While these capabilities were designed primarily for health and environmental monitoring, they are also incredibly useful for profiling applications and helping to make better design decisions.

We found that despite long polling intervals, the data stored in the environmental tables of the database can be useful for determining energy and temperature profiles of actual jobs running on the system. We also verified that our profiling library which utilizes the EMON API reports the same data as can be found in the environmental database at sub-second polling intervals and across the individual components of the system instead of just at the BPM level. More importantly, unlike the environmental data which is only available to privileged users, the data is directly accessible to the compute job running on the system.

Looking forward, we will continue to work with and improve our power-profiling code adding more features such as tagging for critical sections of code. Given the very low polling intervals, it is now possible to evaluate power consumption across multiple domains for specific sections of code as well as the application on the whole. This will be useful information to gather and study and will provide crucial details of applications never before possible.

## 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 authors thank Cheetah Goletz for providing the data analyzed in this paper as well as the rest of the ALCF application and operations support staff for their help. They also gratefully acknowledge the help provided by the application teams whose codes are used herein.

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 system.

## 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] 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, March 2010. [Online]. Available: http://doi.acm.org/10.1145/1740390.1740408

[6] 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, June 2007. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2007.1026

[7] 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.

[8] 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*, November 2011, pp. 1 –11.

[9] 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.

[10] 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*, August 31 - September 4 2009, pp. 1 –8.

[11] 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, November 2012. [Online]. Available: http://dx.doi.org/10.1007/s00450-011-0189-6

[12] 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.

[13] 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, November 2008.

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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, November 2012. [Online]. Available: http://dx.doi.org/10.1007/s00450-011-0192-y

[20] 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

[21] 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*, September 2012, pp. 36 –44.

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

[23] 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.

[24] 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

[25] V. Morozov, J. Meng, V. Vishwanath, J. Hammond, K. Kumaran, and M. Papka, "ALCF MPI benchmarks: Understanding machine-specific communication behavior," in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*, September 2012, pp. 19 –28.

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