Understanding Power Measurement Capabilities on Zaius Power9

Bo Li
Virginia Tech
Blacksburg, Virginia, USA
Email: bxl14074@vt.edu

Edgar A. León
Lawrence Livermore National Laboratory
Livermore, California, USA
Email: leon@llnl.gov

Kirk W. Cameron
Virginia Tech
Blacksburg, Virginia, USA
Email: cameron@cs.vt.edu

***Abstract*—Power and energy are first-class operating concerns for data centers, emerging supercomputers, and future exascale machines. The power and energy measurement capabilities in emerging systems is critical to understand and optimize power usage according to application characteristics. In this work, we describe our evaluation of the power monitoring capabilities of the Zaius Power9 server. We highlight existing limitations of this system and report on the available power domains, measurement granularity, and sampling rate. We provide empirical power profiles that stress the memory system and the compute capabilities. Furthermore, we demonstrate high-level insights of a scientific proxy-application through its power consumption. Our goal is to provide an empirical study for the benefit of developers and researchers planning on utilizing the power capabilities of this state-of-the-art architecture.**

***Index Terms*—Power measurement; HPC; Zaius; Power9.**

I. Introduction

Power and energy are becoming first-class operating concerns for emerging supercomputers and future exascale machines. The implications of power and energy concerns for supercomputers have a broad impact ranging from managing power by utility companies to pursuing optimizations for power and energy consumption, in addition to performance, by system and application developers. One set of optimizations may include shifting power from hardware components not in the critical path to those components in the critical path of an application. Therefore, the power and energy measurement capabilities in emerging systems is critical to understand and optimize power usage according to application characteristics.

On the path to exascale computing, the U.S. Department of Energy will field two new supercomputers in 2018 featuring IBM Power9 processors, NVIDIA Volta GPUs, and the InfiniBand interconnect. For example, *Sierra*, hosted at Lawrence Livermore National Laboratory (LLNL), is expected to provide 125 petaflops within a 12 megawatt power budget. While it is possible to rely on third party solutions to monitor and profile power consumption [1]–[3], the Power9 processors enable fine-grained power measurements through an on-chip controller. In this paper, we describe our experience with power monitoring on the Zaius Power9 server. While the Zaius server targets data centers, we expect the lessons learned from this study to be useful in future power studies on supercomputers like Sierra.

The paper is organized as follows. First, Section II introduces the power measurement capabilities of interest available on the Power9 processor. Section III describes the experimental setup including the testbed platform. Then, Section IV establishes a performance baseline using several microbenchmarks. This is followed by Section V, where we present a set of experiments to understand power consumption and its limitations focusing on two use cases: the cache hierarchy and the behavior of an application. Finally, Section VI summarizes our findings and describes future work.

II. Power Monitoring Overview

The IBM Power9 processor embeds an On-Chip Controller (OCC) to monitor system information such as the power consumption of CPU, memory, and GPUs as well as thermal data [4]. The OCC works with other components including the Autonomic Management of Energy Component (AMEC), the Analog Power Subsystem Sweep (APSS), and the Baseboard Management Controller (BMC) to read system data and to control system settings. For example, the power of system can be capped in which case the OCC monitors the power sensors and throttles the CPU and memory subsystem accordingly.

There are two ways to monitor power consumption: in-band and out-of-band [5], [6]. The out-of-band method collects the power data without the intervention of the main processor. The BMC can communicate with the OCC to get the power and sensors data, which is collected periodically by the OCC. This method allows the profiling of power consumption of a host from another system connected on the same network. Some example sensors collected by the OCC include the power consumption of each processor, the temperature of each core, the temperature of each memory DIMM, and the power consumption of GPUs.

While the out-of-band method requires the support of several hardware components including the BMC, the in-band method only relies on sampling the OCC. The OCC driver periodically copies the sensor data to main memory and makes it accessible as *hwmon* sensors. Most modern systems have sensor chips on the motherboard to support the monitor of system status (e.g., temperature, fan speed, and voltage) and use the hwmon kernel module to interact with those sensors. The OCC takes 8 ms to update a block of sensors to main memory and up to 80 ms to update all of the available sensors. For example, the OCC takes 8 ms to read processor core data and 64 ms to read DIMM memory data. One can use *lm_sensors* [7] to access these sensors. In this work, we use lm_sensors because it works with the hwmon kernel module to read hardware sensor data from user space.

 ISBN: 978-1-61208-655-2

III. Experimental Setup

In this work, we investigate the power measurement capabilities of the Zaius Power9 server. We conducted experiments to investigate the power measurement interfaces; characterize performance (i.e., computation, memory bandwidth, and memory latency) and power consumption using benchmarks; and demonstrate how one may apply these monitoring capabilities to a proxy application from the U.S. Department of Energy.

A. Testbed platform

Our testbed includes a Zaius Power9 server designed by Rackspace. The server has two Power9 LaGrange processors with 12 SMT-4 cores each and a total of 96 hardware threads. There are 73 different CPU clock frequencies ranging from 2.0 to 3.2 GHz. Each core has 32 KB L1 cache (data and instruction), 512 KB shared L2 cache, and 10 MB shared L3 cache. The server has 128 GB of DDR4 memory.

We use the in-band method to measure power because the Zaius board lacks an APSS. Some important features that rely on the APSS include out-of-band power profiling, setting a power cap, and measuring DRAM power.

In this work, all the power measurements were collected in-band. To avoid application interference by the power monitoring thread, we bind this thread to one of the processors and the application tasks to the other processor (there are two processors). While we could simply dedicate a core for the monitoring thread, we want to avoid any interference that could result by using the shared memory bus. In future work, we will assess this other configuration and quantify the associated overheads. The monitoring thread, thus, measures the power consumption of the second processor where the application runs.

We also measured the overhead of polling the in-band power sensors. It takes about 17 ms for each query, which means the highest sampling rate of in-band power measurement is 17 ms.

B. Benchmarks and Mini-Applications

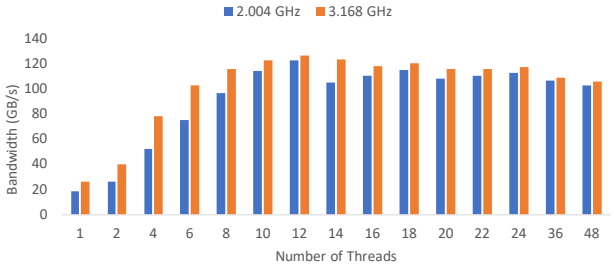
LMBench [8] is a benchmark suite used for analyzing memory speed. We employ STREAM and LAT_MEM_RD in LMBench to measure memory bandwidth and latency, respectively. We use the OpenMP version of STREAM and the single-thread version of LAT_MEM_RD.

LULESH [9], the Livermore Unstructured Lagrange Explicit Shock Hydrodynamics mini-application, provides a simplified source code that contains the data access patterns and computational characteristics of larger hydrodynamics codes at LLNL. It uses an unstructured hexahedral mesh with two centerings and solves the Sedov problem. Because of its smaller size, LULESH allows for easier and faster performance tuning experiments on various architectures.

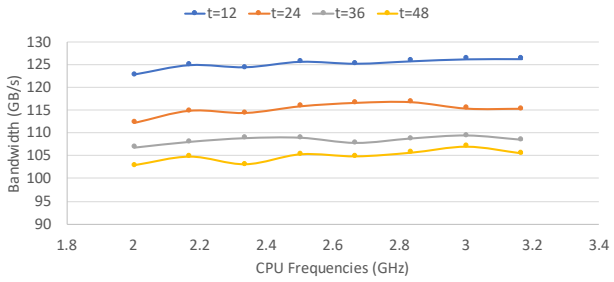
DGEMM from the APEX benchmark suite [10] is a simple double-precision dense-matrix multiplication code. We use it to capture floating-point computational rate.

IV. Performance Characterization

STREAM measures memory throughput and represents the practical peak bandwidth of the memory system. The performance of STREAM is dependent on a number of parameters including CPU clock frequency and the number of threads. Figure 1 shows the impact of thread concurrency and CPU frequency. In Figure 1a, we measured the throughput of STREAM-ADD (performs add operations on memory arrays) as a function of thread concurrency for two CPU frequencies. The results show that, for both CPU frequencies, STREAM obtains the highest throughput, i.e., 126 GB/s, by running with 12 threads. As we further increase the thread count, the throughput gets worse due to memory resource contention. Figure 1b shows that CPU clock frequency has a small impact on memory throughput regardless of the concurrency level.



(a) STREAM-ADD under various thread concurrencies.



(b) STREAM-ADD under various CPU frequencies.

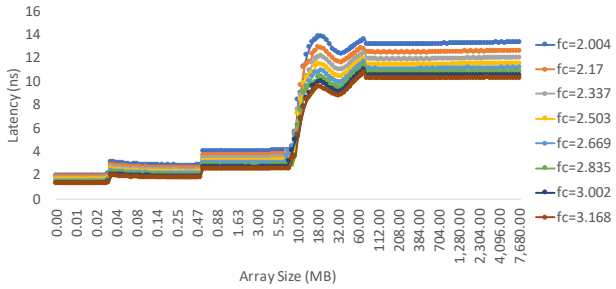
Figure 1. Memory bandwidth under different configurations.

LAT_MEM_RD captures memory latency by measuring the time it takes to access data residing in different levels of memory. It controls the memory level to access by varying the size of the input array and stride. Input arrays of small size fit into cache resulting in faster access latency. Main memory accesses occur when the array size is too large to fit in cache. By measuring the time to access different array sizes, LAT_MEM_RD shows the empirical latency of L1, L2, L3, and main memory.

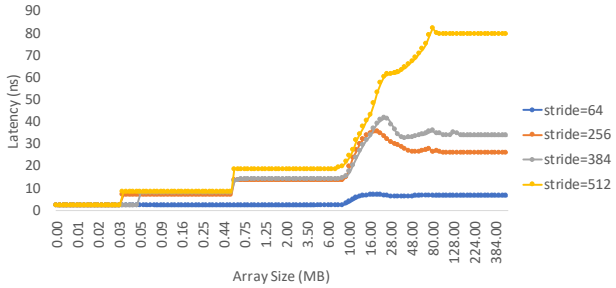
Figure 2 illustrates the latency (in nanoseconds) of accessing the different caches and main memory. Figure 2a shows memory latency for multiple CPU frequencies. As CPU frequency affects how much time one CPU cycle takes, higher CPU frequency leads to lower latency. Focusing on a single CPU frequency, we observe that there are four groups of latency (*steps*) corresponding to the L1, L2, L3, and main memory. We

 ISBN: 978-1-61208-655-2

used an array size of 8 GB, to ensure main memory accesses, and a stride size of 128 bytes, to match the cache line size of the Power9 processor.



(a) Memory latency under different CPU frequencies (f_c). Array size is 8 GB and stride size is 128 bytes.



(b) Memory latency under different stride sizes. CPU frequency is 2.004 GHz and array size is 500 MB.

Figure 2. Impact on hierarchical memory latency.

Figure 2b illustrates the impact of stride size on latency. When the stride size is small, one cache line can potentially satisfy multiple data requests, thus the overall latency of multiple data accesses is lower. When the stride size is large enough, data load requests may cause more cache misses, thus the overall data access latency is higher.

V. Understanding Power Consumption

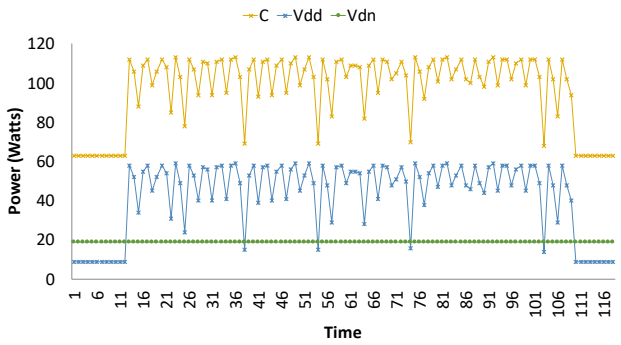
We rely on the OCC to measure power consumption at runtime. The granularity of power measurement is at the processor (socket) level. Since the Zaius system does not have an APSS, we can only measure core power (Vdd) and nest power (Vdn). Nest power mainly includes the on-chip interconnect and the memory controller. To account for the remaining processor power (cache, I/O, etc.), a fixed value is added to Vdd and Vdn to estimate total processor power (C):

$$Processor\_power = Vdn + Vdd + 35Watts \quad (1)$$

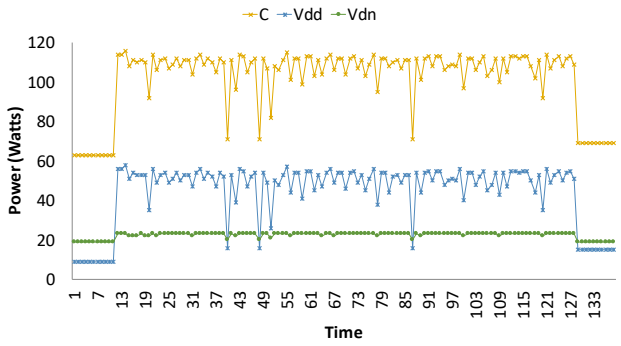
The fixed 35 Watts value is defined in the Machine Readable Workbook (MRW), an XML description of the machine specified by the system administrator. In addition, without an APSS power draw of main memory or GPUs are not easily accessible.

We use lm_sensors to query instant power consumption in user space. We make direct system calls to read the power sensors and get the power data back as standard output. By issuing the lm_sensors call multiple times and averaging its elapsed time, we obtained the overhead of querying sensor data: about 17 milliseconds. We wrapped the system call with a sleep timer, which can be used to change the power sampling rate. For example, if we set the sleep timer to be 100 ms, the sample rate of power profiling will be 117 ms/sample. If we set the timer to be 0 ms, we can get the highest sample rate: 17 ms/sample. If not explicitly stated, we set the sleep timer to 100 ms.

First, we profile the power consumption of DGEMM, a compute intensive benchmark, and STREAM, a memory intensive benchmark. Their power profiles are shown in Figure 3. There are two processors on our test system. As mentioned before, we bind the power monitoring process to the first processor and run the codes on the second processor to avoid application interference by the monitoring process. As shown in Figure 3, when the benchmarks start to execute the core power, Vdd, increases significantly. Interestingly, even though STREAM is memory intense, the peak core power consumption is the same as DGEMM. STREAM, however, stresses other components. For example, the nest power, Vdn, is higher and shows more dynamic variation than DGEMM. This is because the memory controller is part of Vdn. If we could measure the power consumption of main memory, we would expect to see pronounced differences between DGEMM and STREAM.



(a) DGEMM.



(b) STREAM.

Figure 3. Power profile of compute-intense and memory-intense benchmarks.

 ISBN: 978-1-61208-655-2

We also observed low points in Vdd power. This could be the result of several factors. For example, the processor has to wait for data from main memory. An idle processor consumes significantly less power than a busy processor. Other factors reducing the computational intensity of the code would affect the power consumption of the processor. Finally, we note that the total power consumption of the processor, C, follows the same pattern as Vdd because of the constant factor shown in (1) and the small or null variations in Vdn power.

A. Cache Hierarchy Power Draw

The relationship between power and memory access patterns is important for power and energy efficiency. Figure 4 shows the power profile of LAT_MEM_RD, which accesses the different levels of the memory hierarchy as a function of time. In order to stress the power consumption when the code accesses the levels of memory, we ran ten instances of LAT_MEM_RD concurrently. As the figure shows, as the benchmark accesses progress from L1 to L3, the core power consumption, Vdd, decreases. This is because load operations become slower (see Figure 2) and the code becomes less compute bound. When the application starts to access main memory, the core power continues to decrease while the nest power increases, as we would expect.

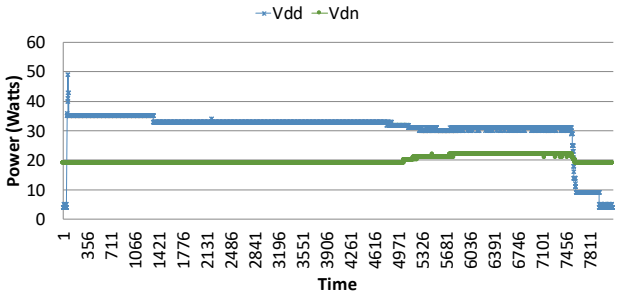


Figure 4. Power profile of the memory hierarchy.

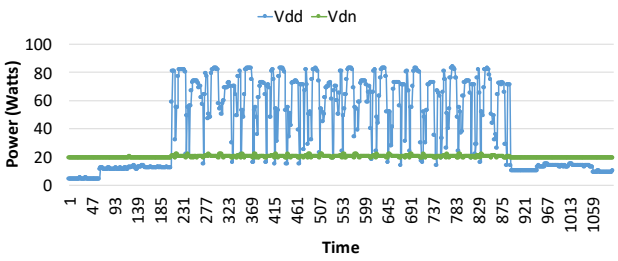


Figure 5. Power profile of LULESH.

B. Application Behavior and Power Draw

The capability of profiling power consumption for scientific applications is important for gaining insights into their behavior and identifying optimization areas [5], [9]. Figure 5 shows the power profile of LULESH with the following parameters: 48 OpenMP threads, 10 iterations, and 240 input elements. In order to capture finer power characteristics, we set the power sampling rate to the highest, i.e., 17 ms. From the power profile, we can identify 10 curves with similar patterns. They map to the 10 iterations of LULESH that do the similar computations. Within each iteration, we observe dynamic variations in core power consumption due to the different code phases of LULESH within an iteration, some of which are compute bound and others memory bound [9].

The core power profile, Vdd, also shows that the initialization and completion stages have significant lower power consumption due to the less intensive compute and memory workload. In future work, we may instrument the application to perform function-level power profiling.

VI. Summary and Future Work

In this work, we evaluated the power monitoring capabilities of a state-of-the-art Zaius Power9 server. The Power9 processor is an important architecture in both data centers and high-performance computing markets. Although there are many sensors to monitor power and energy in the Power9 processor, we were limited to only three power domains because of the lack of certain components on the Zaius board such as the APSS. Additionally, in-band power monitoring was our only option and with this a 17ms sampling rate, which may be too coarse to analyze small code fragments within an application. Despite these limitations, we were able to characterize the performance and power of this system under different configurations using a number of benchmarks to stress the compute and memory capabilities. We observed that the core power dominates the behavior of the total processor power because of a constant factor used in its calculation. Also, changes in nest power are subtle even when exercising the cache hierarchy. The power profile of LULESH represented the iterative nature of the code, as well as changes in phases based on their memory and compute utilization.

In future work, we plan to evaluate policies to shift power between the processor, memory, and the GPUs. To this end, we will investigate other ways to communicate with the OCC on the Zaius board to get access to the power of the memory system and other sensors of interest.

Acknowledgment

We thank Adi Gangidi from Rackspace and Martha Broyles, Chris Cain, and Todd Rosedahl from IBM for their support and assistance. Prepared by LLNL under Contract DE-AC52-07NA27344. LLNL-CONF-748981.

References

[1] J. H. Laros, P. Pokorny, and D. DeBonis, “PowerInsight – a commodity power measurement capability,” in *International Green Computing Conference*, June 2013.

[2] B. Li, H. C. Chang, S. Song, C. Y. Su, T. Meyer, J. Mooring, and K. W. Cameron, “The power-performance tradeoffs of the Intel Xeon Phi on HPC applications,” in *IEEE International Parallel Distributed Processing Symposium Workshops*, May 2014.

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

 ISBN: 978-1-61208-655-2

[4] T. Rosedahl, M. Broyles, C. Lefurgy, B. Christensen, and W. Feng, "Power/performance controlling techniques in OpenPOWER," in *High Performance Computing*. Springer International Publishing, 2017, pp. 275–289.

[5] R. E. Grant, J. H. Laros, M. J. Levenhagen, S. L. Olivier, K. Pedretti, H. L. Ward, and A. J. Younge, "Evaluating energy and power profiling techniques for HPC workloads," in *International Green and Sustainable Computing Conference*, Oct. 2017.

[6] R. E. Grant, M. Levenhagen, S. L. Olivier, D. DeBonis, K. T. Pedretti, and J. H. L. III, "Standardizing power monitoring and control at exascale," *IEEE Computer*, vol. 49, no. 10, pp. 38–46, Oct. 2016.

[7] (2018, May) The lm-sensors package. [Online]. Available: https://github.com/groeck/lm-sensors

[8] L. W. McVoy and C. Staelin, "lmbench: Portable tools for performance analysis." in *USENIX annual technical conference*, 1996, pp. 279–294.

[9] E. A. León, I. Karlin, and R. E. Grant, "Optimizing explicit hydrodynamics for power, energy, and performance," in *International Conference on Cluster Computing*, ser. Cluster'15. Chicago, IL: IEEE, Sep. 2015.

[10] (2018, Jan.) APEX benchmarks. [Online]. Available: http://www.nersc.gov/research-and-development/apex/apex-benchmarks

 ISBN: 978-1-61208-655-2