Evaluating the Energy Measurements of the IBM POWER9 On-Chip Controller

Hannes Tröpgen
Center for Information Services and High Performance Computing (ZIH)
Technische Universität Dresden
Dresden, Germany
hannes.troepgen@tu-dresden.de

Mario Bielert
Center for Information Services and High Performance Computing (ZIH)
Technische Universität Dresden
Dresden, Germany
mario.bielert@tu-dresden.de

Thomas Ilsche
Center for Information Services and High Performance Computing (ZIH)
Technische Universität Dresden
Dresden, Germany
thomas.ilsche@tu-dresden.de

ABSTRACT

Dependable power measurements are the backbone of energy-efficient computing systems. The IBM PowerNV platform offers such power measurements through an embedded PowerPC 405 processor: The *On-Chip Controller* (OCC). Among other system-control tasks, the OCC provides power measurements for several domains, such as system, CPU, and GPU. This paper provides a detailed description and an in-depth evaluation of these OCC-provided power measurements. For that, we describe the provided interfaces themselves and experimentally verify their overhead (3.6 µs to 10.8 µs per access) and readout rate (24.95 Sa/s). We also study the consistency of the reported sensor readouts across the measurement domains and compare it to externally measured data. Furthermore, we estimate the internal sampling rate (1996 Sa/s) by provoking aliasing errors with artificial workloads, and quantify the errors that such aliasing could introduce in practice (for power consumption of processors 12 % in our experimental worst-case scenario). Given these insights, practitioners using the IBM PowerNV platform can assess the quality of the embedded measurements, permitting sought-after energy efficiency improvements.

CCS CONCEPTS

• **Hardware → Platform power issues**; **Chip-level power issues**; *Post-manufacture validation and debug*; Energy metering.

KEYWORDS

On-Chip Controller; POWER9; Power Measurements; Energy Efficiency

ACM Reference Format:
Hannes Tröpgen, Mario Bielert, and Thomas Ilsche. 2023. Evaluating the Energy Measurements of the IBM POWER9 On-Chip Controller. In *Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3578244.3583729

ICPE '23, April 15–19, 2023, Coimbra, Portugal

ACM ISBN 979-8-4007-0068-2/23/04…$15.00
https://doi.org/10.1145/3578244.3583729

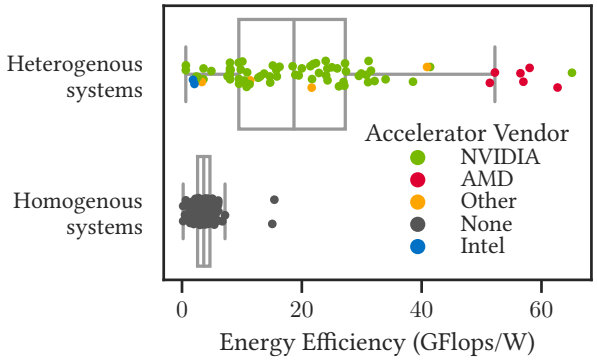

Figure 1: Energy efficiency of heterogenous and homogenous *High Performance Computing* (HPC) Clusters according to the GREEN500 list compiled from [29].

1 INTRODUCTION

The growing demand for accelerated computing, especially in machine learning and *artificial intelligence* (AI), leads to the development of heterogenous architectures comprising prevalent multi-purpose processors and accelerators. For example, processors of the IBM PowerNV platform bundle a high-core-count processor with several NVIDIA accelerators connected through NVLink. IBM geared this architecture towards scalable and data-intensive workloads. Even though such heterogenous systems are generally more energy-efficient (see Figure 1) when used for suitable tasks, there are still parameters that influence the effective energy efficiency, e. g., voltage/frequency selection [27]. Consequently, tuning such parameters to optimize energy efficiency depends on reliable power measurements.

This paper thoroughly evaluates the embedded power measurement interface on a POWER9 system as one representative of the PowerNV platform. We describe the available measurement interfaces and their characteristics, such as readout latency and resolution. We present the functional measurement domains and examine their accuracy. Lastly, we use artificial workloads to examine the behavior of the internal measurement setup. We want to establish how reliable the embedded power measurements are, and ultimately allow for informed decisions regarding the energy efficiency of applications running on the PowerNV platform.

The remainder of this paper is structured as follows: In the next Section 2, we introduce the background for power measurements and give an overview of the PowerNV processors. Their power measurement interface itself is then described in Section 3, and the interface's update rate and readout overhead are shown in Section 4. After that, we focus on the measured values themselves and examine their accuracy in Section 5, where we also demonstrate our measurement setup suitable for application tracing and profiling. In Section 6 we experimentally determine the internal sampling rate of the measurement. The final Section 7 summarizes our work and sketches an outlook for future research directions.

Artifacts, including our raw results and programs used to obtain them, can be found online [39].[1] This artifacts archive consists of multiple subdirectories, which are referred to throughout this paper with the prefix `artifacts/`, e. g.:

The data used for Figure 1 can be found in `artifacts/green500`.

2 RELATED WORK

This section gives an overview of existing work. First, we cover available measurements of the power consumption of computing systems in general. Second, we describe prior work on the investigated architecture and its On-Chip Controller.

2.1 Power Measurements of Computing Systems

While dedicated precision power analyzers can provide excellent accuracies and sampling rates, their cost and space requirements are prohibitive for systems with multiple compute nodes. However, several components in a modern data center power distribution provide power measurements out-of-the-box. Some *power distribution units* (PDUs) offer revenue-grade energy metering, e.g., Raritan PDUs with a 1 % accuracy per ISO/IEC 62053-21 [30]. In Section 5, we use a monitored IBM PDU[2] that provides power readouts via the *Simple Network Management Protocol* (SNMP). To the best of our knowledge, there is no specification of the accuracy or quality of its power measurements.

Many server nodes also offer power monitoring via the *Baseboard Management Controller* (BMC), typically measured at the *power supply unit* (PSU). The most prevalent protocols to read the power measurement data and other sensor data are the *Intelligent Platform Management Interface* (IPMI) [19] and the more recent Redfish [9]. The data can be read in-band or out-of-band via a network connection to the BMC and typical readout rates are 1 Sa/s (*1 sample per second*) or lower. Such measurements are not always reliable, for example, Hackenberg et al. [14] have shown that Dell's implementation of power measurements via IPMI exhibits severe aliasing errors despite a documented 1 % accuracy.

Moreover, many modern server processors provide power or energy measurements at the CPU level. The most prominent example is the *Running Average Power Limiting* (RAPL) mechanism originally developed by Intel [33] but now also implemented by AMD. Intel's documentation [18, Chapter 14.10] describes that the RAPL registers provide an energy counter that is updated at ~1 kSa/s. However, Lipp et al. [26] report higher update rates of up to 20 kSa/s for certain domains. Naively computing the average power consumption for short code regions may lead to inaccuracies since RAPL does not provide an update counter and thus the actual measurement duration is unknown. Hähnel et al. [16] have demonstrated how to overcome this limitation and use RAPL to accurately measure regions in the order of a few milliseconds. Several studies have shown that the quality highly depends on the specific implementation in the micro-architecture. In particular, when the energy counter is implemented with a model rather than a physical measurement, the values can have biases towards certain workloads (see [7, 14, 15, 34, 35]).

Several projects have demonstrated scalable approaches to add more sophisticated measurement infrastructure to compute clusters. Examples are ArduPower [8], PowerSensor 2 [31], [17], and DiG [24]. The available readout rates for these solutions range from 1 kSa/s to 50 kSa/s, but some use higher internal sampling rates for increased accuracy.

2.2 POWER8, POWER9 & The On-Chip Controller

In their intro to the POWER8 processor, Fluhr et al. [11] place emphasis on the requirement to optimize energy and consequently introduce the *On-Chip Controller* (OCC) and sketch its monitoring and control capabilities. The OCC is an embedded PowerPC 405 processor running a *real-time operating system* (RTOS) [11, Sec. II]. The OCC's documentation states its main goals are to "keep the system [thermally & power] safe" [5, Sec. 1.2], while leaving the P-state selection up to the *operating system* (OS). Additionally, the collected sensor data should be provided for external display [5, Sec. 1.2].

The OCC persists across the PowerNV platform from its launch with POWER8 to the latest POWER10 processors. However, the focused features shifted for the POWER9 processors: Gonzalez et al. [12, 13] describe it as a "scale-out (SO)" [12, Sec. II] processor, highlighting its *input/output* (IO) capabilities, namely a total 300 GB/s accelerator, 192 GB/s PCIeGen4 and 230 GB/s memory bandwidth. They also sketch the general architecture including the power domains, thereby explaining some of the OCC's sensors.

The OCC itself runs a firmware of the same name, available under the terms of the Apache 2 license [28]. Its architecture, interfaces, and capabilities are sketched in Section 3.

Several submissions for OpenPOWER Summits cover the OCC, including Rosedahl [32] with a general introduction, and Bhat [2] with a sketch of possible applications.

3 MEASURING POWER USING THE ON-CHIP CONTROLLER

The OCC manages the hardware sensors of the system and reports their readouts to the higher levels. In total, it can report data for 75 types of sensors [5, Sec. 11.3].[3] While the existence of an individual sensor is not guaranteed, over 300 individual sensors were available during all of our tests. These sensors report various measurement values, including temperature, voltage, current, and frequency, but also more abstract formats, e. g., processor "utilization" [5, Sec.

[1] https://github.com/tud-zih-energy/2023-power9-occ
[2] Model number: 01KL833(46M4002)

[3] 31 of the 75 sensor types may only be collected out-of-band through *Automated Measurement of Systems for Temperature and Energy Reporting* (AMESTER [1]) [5, Sec. 11.3.3].

Table 1: OCC power sensors (without APSS), excerpt from [5, Sec. 11.3.2.2] with our domain descriptions

Name	reported once per …	Sampling interval (ms)
↪ Domain		
`PWRSYS`	system	0.5
↪ bulk power consumption, measured after PSU output		
`PWRGPU`	processor	0.5
↪ GPUs connected to this processor		
`PWRMEM`	processor	0.5
↪ memory connected to this processor		
`PWRPROC`	processor	0.5
↪ this processor itself (without attached components)		
`PWRVDD`	processor	1.0
↪ processor cores, see V_{dd} in [12, Fig. 1]		
`PWRVDN`	processor	1.0
↪ processor nest, see V_{dn} in [12, Fig. 1]		

11.3.2.4], or event counts. The scope varies from sensor to sensor, most report on the core or processor level. Table 1 lists all supported power sensors.

The OCC also reports up to 16 further power sensors from the *Analog Power Subsystem Sweep* (APSS), whose assignment is system-dependent and thus not considered in any of the experiments. Some, but not all APSS-supported sensors are exposed by the OCC: IO, storage, or fans power consumptions are only available through the APSS sensors [5, Sec. 6.3.1].

The processor powers domains V_{dd} (cores) and V_{dn} (nest) are reported by the OCC, although they do not fully cover the entire processor. Besides the power of core and nest voltages, the POWER9 processor distinguishes voltages for e. g., caches, memory, and other IO [13].

On Linux systems, the OCC's sensor data is available through two in-band interfaces: `hwmon` and the OCC main memory interface. Additionally, the *OCC Poll Response* interface allows for individual sensors to be polled, e. g., by the BMC [5, Sec. 2 & 6]. We did not consider the Poll Response interface in our experiments due to security considerations: The Poll Response interface is only accessible for the BMC and *Host Thermal Management* (HTMGT), and requires granting write privileges for users to their respective memory regions for communication. These same interfaces grant administrative permissions, which is undesirable for regular users. Granting access purely to the OCC main memory interface/`hwmon` is less intrusive, as read-only access is sufficient.

Overall the functionality is similar to those of classical BMCs via IPMI/Redfish. The power measurements with six different kinds of measurement locations are comparatively sophisticated.

3.1 The Linux Kernel Interface `hwmon`

`hwmon` [25] is the standard interface for *hardware monitoring* of the Linux kernel. Sensor data may be queried independently of the underlying hardware by reading files under `/sys/class/hwmon` (*sysfs*), using the library `libsensors`, or the program `sensors`. The reported values are provided by different drivers, depending on the hardware [38, `Documentation/hwmon/hwmon-kernel-api.rst`, `Documentation/hwmon/sysfs-interface.rst`].

The Linux kernel reads the list of available sensors and creates the corresponding `hwmon` entries [38, `drivers/hwmon/ibmpowernv.c`]. When accessing a sensor through `hwmon`, the corresponding callbacks read the requested value from the OCC main memory interface [37, `hw/occ-sensor.c`, l. 246 ff.]. Even though `hwmon` indirectly uses the same data source as all other programs reading sensor data on PowerNV, through its more generalized structure, it can not represent all aspects of the values: E. g., the OCC provides the sensor data in a buffer, which is updated regularly, consequently providing a timestamp of the last update [5, Sec. 11.3.1.3.1]. `hwmon` employs callbacks [38, `Documentation/hwmon/hwmon-kernel-api.rst`, l. 147 ff.] and hence treats the values as being read on demand in real-time, ignoring said timestamp under this assumption.

A patch series exposing more details from the OCC to `hwmon` has been discussed on the Linux kernel mailing list [21], but was ultimately not implemented.

Table 2: Power sensor data reported by the OCC main memory interface [5, Sec. 11.3.1.3]

name	size (byte)	description
`gsid`	2	global sensor ID
`timestamp`	8	512 MHz-based timestamp
`sample`	2	measured value
`accumulator`	8	continuous sum
`update_tag`	4	number of samples stored in accumulator

3.2 The OCC Main Memory Interface

The OCC is connected to the main memory and periodically writes data for its sensors through this connection. Note that the OCC documentation [4, 5] does not use a consistent name for this interface, calling the corresponding chapter *OCC Main Memory Sensor Data*, however not using this term anywhere else. We will use the term *OCC main memory interface* to refer to the interface described here.

The Linux kernel exposes the memory region to which the OCC writes sensor data to the userspace *as-is* at `/sys/firmware/opal/exports/occ_inband_sensors`. [38, `arch/powerpc/platforms/powernv/opal.c`, l. 881 ff.] Every OCC (i. e., every processor) creates one *Sensor Data Block* of 150 kB containing data for this particular processor. Additionally, the first block also contains data for the entire system, e. g. the bulk power consumption. Every block contains two data buffers (*ping* and *pong buffer*), which are used in an alternating fashion, such that always at least one buffer is not being written to and hence contains valid data [5, Sec. 11.3].

The data reported for power sensors is shown in Table 2. The resolution of an individual sample for all power sensors is 1 W. Notably, the OCC format [28, `src/occ_405/sensor/sensor_info.c`] would support a more fine-grained resolution—which is not used for any power sensor. Even though the documentation promises an update rate of 1 or 2 kSa/s for power sensors [5, Sec. 11.3.2.2], the

OCC main memory interface update is only triggered every 8 ms [28, `src/occ_405/amec/amec_slave_smh.c`].[4] Hence, the update rates of 1 or 2 kSa/s only apply to the accumulator, the exposed individual samples have a much lower update rate. The 1 or 2 kSa/s that make up this accumulator are not stored separately, only their sum, i. e., the accumulator may be retrieved. This leads to a theoretical resolution of 1 mJ (1 kSa/s) or 0.5 mJ (2 kSa/s) for the energy. Due to this structure, in the following, we will distinguish between the *external* update rate at which the interfaces expose their data, and the *internal* update rate at which the OCC operates, e. g., updates timestamps and the accumulator.

4 INTERFACE PROPERTIES

In this section, we aim to measure the behavior of the interfaces themselves, namely the readout latency and external update rate of `hwmon` and the OCC main memory interface. The remaining system is not taken into account.

4.1 Setup

All experiments were performed on our local *High Performance Computing* (HPC) cluster *taurus*.

The jobs are distributed through the batch system and (exclusively) run on one of the 32 POWER9 nodes. All 32 nodes are *AC922* systems (code name *Newell*, formerly *Witherspoon*) by IBM. Every node holds two POWER9 processors (model `02CY209`, code name *Monza*) with 22 cores each, resulting in a total of 176 threads with four-way *simultaneous multithreading* (SMT) enabled. Each processor has a *thermal design power* (TDP) of 250 W; the nominal frequency is 2.8 GHz (up to 3.1 GHz possible). The machines use 256 GB DDR4 main memory with a design bandwidth of 170 GB/s. They are primarily used for their six NVIDIA V100 (*Volta*) GPUs, which are entirely ignored in these experiments. The power is supplied by two 2.2 kW *power supply units* (PSUs) following the standard 80+ Platinum [6]. The PSUs run on 230 V *alternating current* (AC). The nodes run *Red Hat Enterprise Linux Server*, release 7.6 (*Maipo*) as the operating system with a version 4.14.0 Linux kernel. The OCC version is the commit `9047e57`, skiboot version `v6.5.3-29-g74a7a87a`. The OCC main memory interface is configured to be readable for non-root users.

This experiment's code and results are included in `artifacts/sampling_frequency_external_interface`.

4.2 Approach

To observe the external update rate and readout latency of the interfaces, the system is used as-is, i. e., no specially crafted workloads are used. The two interfaces (`hwmon`, OCC main memory interface) are observed separately, one after the other.

The measurement program collects 2^{24} samples in a loop using the respective interface and saves each read value together with a timestamp to an in-memory buffer. After the execution is finished, this buffer is dumped into a file. The program takes approx. 1 min per run, chosen as a trade-off between accuracy and total experiment time.

[4] We could not recreate these update rates in our experiments, neither the 8 ms OCC main memory interface update rate (see Section 4.3), and 2 kSa/s sample rate given in the documentation only with a measurable error (see Section 6).

For `hwmon` the readout consists of a single read to the sysfs sensor file; for the OCC main memory interface the exposed sensor data is copied into a buffer, from which the desired value is extracted following the specification [5, Sec. 11.3], similar to an implementation presented by Bhat [3]. This extraction includes a lookup at which address the respective sensor is stored. Although the specification does not guarantee that this sensor address stays constant [5, Sec. 11.3], during test runs, it never changed. Hence, we also include an optimized version to read the OCC main memory interface, which breaks the specification by only reading the sensor address once and then only accessing the region for the respective sensor (instead of reading the entire file and always checking where the sensor data is located). All methods allow us to monitor only one measurement domain at a time. We use the bulk power of the system.

We inspect the produced dumps for two separate properties: First, we discuss the readout latency, i. e., the duration of a single access to the respective interface. This ignores the reported data and purely uses the timestamps in the dumps. In the second step, we examine the reported data itself to determine the *external update rate*, i. e., at which interval new data is exposed by the OCC.

4.3 Results

The (mean) readout latency is the mean duration of a single interface access, i. e., the mean time between two successive interface readouts' timestamps. The readout latency is 4.3 µs for `hwmon`, 10.8 µs for the OCC main memory interface (normal)/3.8 µs (optimized) as Figure 2 details. Both `hwmon` and the normal OCC main memory interface readout exhibit a single large spike in Figure 2: Their readout latency is practically constant. (Tiny secondary spikes can be seen for both methods, which we consider negligibly small. They are most likely caused by the scheduler briefly halting the execution of the experiment.) This is not the case for the optimized access to the OCC main memory interface. Here, Figure 2 shows two spikes. These originate in the two used data buffers (ping and pong buffer, see Section 3.2). If only one buffer contains valid data, the overhead is lower (3.6 µs mean). If both buffers contain valid data, the readout routine checks which buffer has a newer timestamp. This additional check increases the overhead to 4.8 µs (mean), which was required for 16 % of the samples in this experiment.

On average, the value provided by the interface changed every 40.08 ms for both the OCC main memory interface and `hwmon`: The external update rate is 24.95 Sa/s. To compensate for a sensor reporting the same value in a new measurement interval again, only value changes within a 60 ms window have been respected. Due to its nature, the accumulator always changes between two readouts and is not affected by this.

The resolution of this experiment is limited by the duration of a single sample collection, i. e., the readout latency. As this is three orders of magnitude smaller than the external update rate, its determination remains accurate enough for the purposes of this experiment.

This difference in the orders of magnitude also makes the 60 %-gap between reading `hwmon` vs. reading the OCC main memory interface negligible: One readout per update interval introduces approximately less than 0.03 % overhead, considering a single thread.

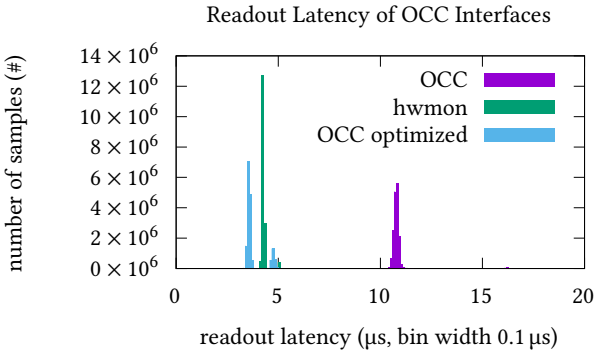

Figure 2: Separation between two readouts of the OCC interfaces

As discussed in Section 3.2, updates are expected exactly at multiples of 8 ms, i. e., here every 40 ms. However, with 40.08 ms the measurement deviates from that by approx. 0.2 %: The same difference occurs when measuring the internal sampling rate of the OCC (see Section 6).

Li et al. [23, Sec. II, III-A] performed similar measurements on Google's Zaius platform. They use `hwmon` and report a duration of 17 ms to query the sensors. They do not describe the process of obtaining these in detail, hence the factor of ~1300 between their and our results can't be explained here. Moreover, their mentioned properties of the sensors do not align with our specifications at hand: They mention "processor core data"[5] as an example with an update rate of 8 ms. In a reiteration of this experiment monitoring the power consumption of a single processor, the external update rate remained at approx. 40 ms.

5 CONSISTENCY ACROSS DIFFERENT MEASUREMENTS

To verify the values themselves, we compare them across different power measurement sources. Figure 3 shows the monitoring domains available to us along the power delivery path from the PDUs to the consumers. First, the PDUs themselves report AC power output values via SNMP (cf. Section 2). Next, the BMC reports both per-PSU input power as well as bulk power of the node. The OCC also reports the bulk (total) power value. Note that this value may not necessarily correspond to a single physical measurement point but could possibly be computed as a sum of multiple sensors at multiple voltages. Finally, the OCC measurement points are detailed by Table 1.

To quantify the accuracy, multiple data sources for the same measurement domain are necessary. The only domain with more than one data source in our setup is the total/bulk power (reported by OCC and BMC). In initial tests, we observed that the reported bulk power from the BMC matches the corresponding values from the OCC—but they can both plausibly come from the same sensors. Therefore, we can not quantify the accuracy for any of the OCC's reported power domains.

Lacking multiple sensors for the same measurement domain, we compare sensors of neighboring measurement domains: (1) OCC-reported bulk power against the PSU input, and (2) OCC-reported bulk power against the sum of individual components' power consumptions. Since these comparisons do not cover the same measurement domain and even include a voltage conversion, a difference is expected. That difference, as already shown in Figure 3, could contain both conversion losses as well as unaccounted consumers, e.g., small fans.

Nevertheless, this comparison demonstrates plausibility and would reveal workload biases. Furthermore, the processor measurement domains comprise multiple voltages and also contain conversion losses and possibly unaccounted voltages, but we do not evaluate them in detail. We confirmed that the PDU outlet and PSU input powers match closely.

5.1 Measurement Setup

For examination of the OCC's response to certain workloads, the setup is much more similar to typical application profiling: The desired workload is defined using the synthetic workload generator roco2 [14], which provides Score-P [22] user instrumentation.[6] The OCC's values are loaded via the *IBM PowerNV Score-P Plugin*, outlined in the following section. The execution produces an OTF2 trace file [10], which records OCC-reported values and the sections of the workload. The raw trace and its processed forms are included in `artifacts/psu_comparison`.

5.2 Score-P PowerNV Plugin

To record the sensor readouts reported by the OCC, we developed a metric plugin for the Score-P plugin interface [36], the *IBM PowerNV Score-P Plugin*.[7] This plugin reads the OCC main memory interface and records all available power sensors at a configurable interval into an OTF2 trace. For every OCC power sensor (see Table 1) the plugin records the current sample, the timestamp, and the total energy based on the accumulator, as well as the number of samples in this accumulator.

According to the documentation [5, Sec. 11.3.2.2], the accumulator has a sampling rate of 1 or 2 kSa/s, depending on the sensor. Therefore, between two readouts of the OCC main memory interface (which are 40 ms apart), multiple samples are collected internally, i.e., 40 or 80 samples, respectively. These samples are not exposed individually, but their sum is reported as the accumulator. By tracing the changes in the number of accumulated samples and the accumulator itself, the average power consumption can be computed:

$$power\ from\ energy(t_1, t_2) = \frac{accumulator(t_2) - accumulator(t_1)}{sample\ count(t_2) - sample\ count(t_1)}$$

As this equation uses *energy* (here the accumulator), the result is referred to as *power from energy*.

[5] Notably, according to the current OCC specification [5], the smallest reported power domain is the processor. Data for individual cores are not available.

[6] When tracing an application, roco2 would be replaced by the instrumented application.

[7] https://github.com/score-p/scorep_plugin_ibmpowernv

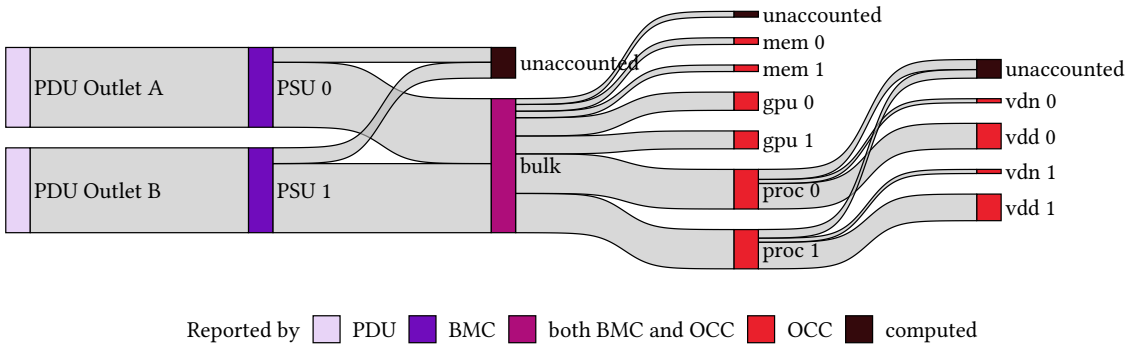

Figure 3: Power delivery scheme and monitored power domains. Colors indicate the data source. Band heights are scaled relatively based on a configuration from Section 5.3.

The resulting traces contain two metrics tracking the current power consumption in W: The *direct samples*, a single sample reported every 40 ms by the OCC, and the *power from energy*, the average power consumption of the last 40 ms based on 40 or 80 samples.

5.3 Approach

To gain a fine-grained profile of the sensor behavior, we test a wide range of power levels. These levels are achieved by running seven workloads[8] defined by roco2 [14] on 1, 2, ..., 44 cores (with four threads per core). During run-time, each workload executes for 60 s to create a stable environment, and to circumvent problems in the synchronization with the external data source. This particular configuration is bundled as an example with roco2 [14] under the name *P9 Longrun*. During execution, the power consumption as reported by the two PSUs was continuously collected and stored in the trace.

The tested system has two 2.2 kW power supplies, whose power budget is not exhausted even with both processors under full load, where the bulk power consumption is approximately 1 kW.[9] One such configuration is shown in Figure 3, here the kernel `memory write` running on all cores draws a total 1055 W for both PSU inputs combined.

5.4 Results

Figure 4 compares the power consumption reported by the PSUs to the power consumption reported by the OCC. Modeling the PSUs' efficiency using a quadratic fit[10] yields plausible results: To this regression the workloads have a 0.2 % *mean absolute percentage error* (MAPE) and 1.7 W *mean absolute error* (MAE). Across all workloads, the efficiency is 77 %. This low efficiency is rooted in the load on the PSUs with 458 W to 859 W OCC-reported bulk power, which is well below the design capacity of the two 2.2 kW PSUs.

[8]These workloads are `busy wait`, `compute`, `matmul`, `memory copy`, `memory read`, `memory write`, and `sine`.
[9]Their larger design capacity is due to the six *graphics processing units* (GPUs) of the node, which are entirely ignored for this test.
[10]From our experience, PSUs do not exhibit linear efficiency. In this particular value range a linear regression would be sufficient, but still has approx. twice the error with 0.4 % MAPE, 3.2 W MAE.

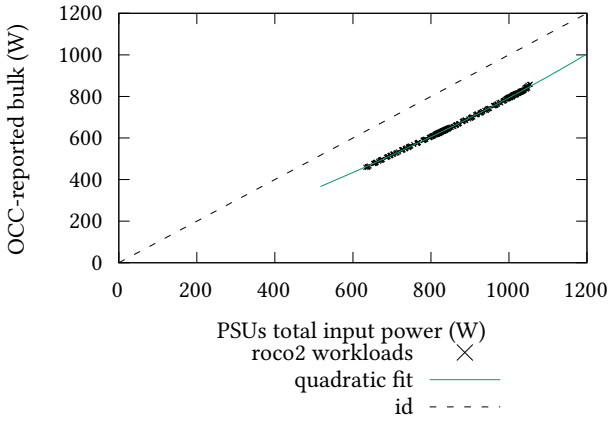

Figure 4: PSU input power vs. bulk power reported by OCC

This experiment also uncovered a discrepancy in the OCC-reported bulk power: Recalculating the bulk power consumption by adding the reported power consumptions of GPUs, CPUs, and memories has an inconsistent difference when compared to the reported bulk power of the system.[11] This discrepancy is shown in Figure 5, where the re-calculated sum vs. the reported bulk power consumption show 3.8 % MAPE, 25.5 W MAE. The cause of this is not clear; one or multiple components not measured individually, but included in the bulk power could be responsible for this difference.

The bulk powers reported by the OCC and the BMC match very well (MAPE 0.2 %, MAE 1.3 W). The OCC and BMC could use the same underlying data source, as the BMC could query the OCC using its Poll Response interface [5, Sec. 1.6]. Further investigation of this possibility is not possible with the used setup.

Even though OCC and BMC-reported values match, this experiment is no verification of the OCC-reported values: We conclude that the OCC-reported values are plausible, but may not verify their correctness.

[11]This is marked as *unaccounted* in Figure 3.

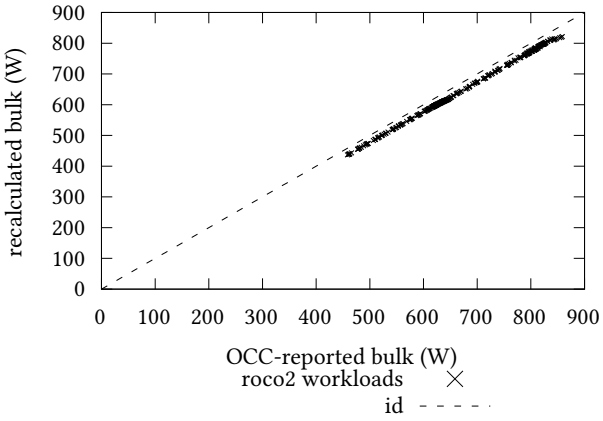

Figure 5: Bulk power as reported by OCC vs. re-calculated sum

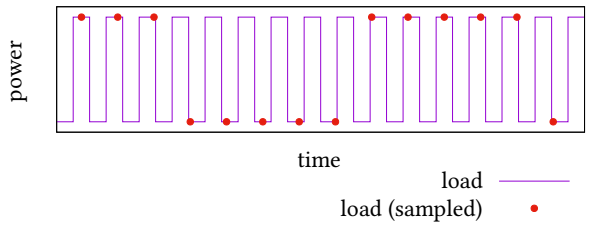

Figure 6: Concept of aliasing during sampling

6 INTERNAL SAMPLING RATE

The OCC provides an accumulator for its power sensors but does not expose individual samples that contribute to its value. In this section, we measure the internal sampling rate of the accumulator.

6.1 Setup & Approach

The setup is identical to Section 4, but lacking the PSU observation. For simplicity, here we use a single sensor: The power consumption of the first processor (*proc 0*).[12]

We assume that the OCC internally samples with 2 kSa/s (as the corresponding "sample time" is given with 500 µs in the specification [5, Sec. 11.3.2.2]), and that these samples are added without any further processing into the accumulator. Based on these assumptions, we design a workload that idles when it is being sampled and creates a high load when it is not sampled (or vice versa), effectively provoking aliasing. For this, the frequencies of sampling and workload have to match perfectly—which, in practice, they do not: A slight mismatch of frequencies causes this effect to shift over time, i. e., first only idle is sampled, after some time this shifts and only high load (work) is sampled, shifting back after some more time etc. However, throughout this shift, there remains only approx. 1 sample per period. This is sketched in Figure 6.

This aliasing affects the traces indirectly: Due to the update rate of the interface (approx. 40 ms), the OCC does not expose all of these individual samples. Only the mean of all accumulated samples since the last interface update is stored as *power from energy* (see Section 5.2); here the mean across the last 80 samples is recorded to the trace.

Typically, some of the 80 samples are captured during the low power level, and some during the high power level, resulting in the power from energy-values hovering in between these two levels. In particular, as half the time is spent idle and half working, the power from energy-values are the mean of the low and high power levels. Hence, the sketch in Figure 6 would produce a stable power from energy level, as every 5 samples the sampled power level changes between high and low level.

The power from energy deviates from the mean only if almost all of the 80 samples between two interface readouts are captured during idle (or almost all during work). I. e., the aliasing only becomes apparent in the traces when workload and sampling rate are almost equal.

Combining this deviation from the mean of power from energy with the shift from sampling only high to sampling only low—due to small frequency deviations described above (see Figure 6)—yields typical aliasing patterns: The power from energy slowly alternates between a higher and a lower level. If we observe such a pattern, we know the sampling rate and workload frequency are almost equal. Based on the frequency of power from energy alternating between the two levels, and the relation of this alternating pattern to the (known) frequency of the underlying workload changing between idle and work, we can derive the sampling rate of the accumulator.

To produce such an effect, we employed an alternating synthetic workload around 0.5, 1, 2, and 2.05 kHz. Half of each period is spent idling, the other with computing.[13] This configuration is bundled as an example with roco2 [14] under the name *P9 Highlow*. The resulting traces and scripts for post-processing are provided in `artifacts/sampling_frequency_internal_accumulator`.

This particular workload creates another indicator for the aforementioned aliasing: The OCC main memory interface provides the latest individual sample (*direct sample*), which have a larger spread compared to the power from energy: As the power from energy is based on an average across 80 samples, which contains samples during idle *and* work, they are evening out—opposed to the single sample directly provided by OCC, which is captured during *either* idle *or* work[14]. In a non-aliasing scenario, all direct samples exhibit a larger spread compared to the power from energy.

During aliasing, all samples within the 80-sample window that power from energy uses are captured during idle (or all during work). Consequently, as all 80 samples record the same power level, the mean is identical to the single direct sample: When aliasing occurs, the spread of power from energy and direct samples is pretty much identical.

[12] The approach can be applied to other sensors as long as a workload can be generated.

[13] The commands are selected to create low and high power levels. *Idle* corresponds to setting medium thread priority [20, Sec. 3.2, p. 838], high load is created by a naive vector dot product implementation. Setting the thread priority is also used by the Linux kernel to idle on PowerNV [38, `arch/powerpc/kernel/idle.c`].

[14] or during the transition between those, but for this approach we consider this transition period to be negligibly short

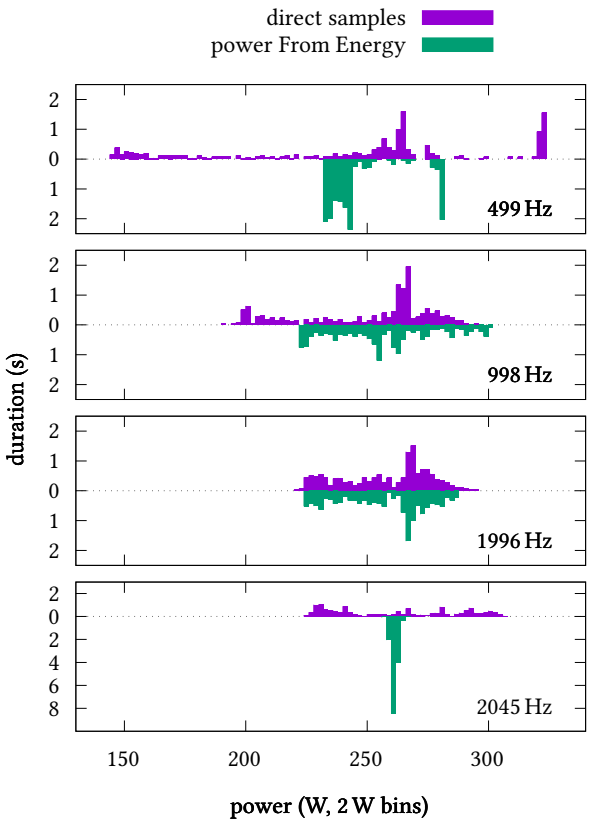

Figure 7: Spread of direct samples versus power from energy for workloads of different frequencies

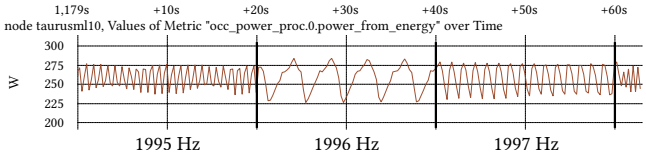

Figure 8: Power from energy for the first processor during workloads alternating with 1995 Hz **to** 1997 Hz

6.2 Results

Exactly this effect on the spread of power from energy and direct samples can be observed: For a workload with 499, 998, and 2045 Hz the spread of direct samples and power from energy values show a clear difference—which is lacking for a workload at 1996 Hz, as Figure 7 shows.

This indicates that aliasing is occurring for the 1996 Hz workload and only one sample per period (of the workload) is collected, as discussed above. The aliasing pattern itself, i. e., the power from energy shifting between a high and a low level is shown in Figure 8. (Note that the apparent slow shift between high and low power levels is purely a measuring artifact: The underlying workload still loops with 1995 Hz to 1997 Hz.)

Table 3: Internal OCC sampling rate, computed from the results in Figure 8

$f_{pattern}$ (Hz)	$f_{workload}$ (Hz)	$f_{sampling}$ (Hz)
$24 \div 19.4\,\text{s} \approx$ 1.24	1995	1996.24
$4 \div 16.5\,\text{s} \approx$ 0.24	1996	1996.24
$14 \div 18.3\,\text{s} \approx$ 0.77	1997	1996.23

We use the pattern visible in Figure 8 to compute the accumulator sampling rate. For that, we manually read the frequencies of the emerging aliasing pattern from Figure 8 and apply $|f_{sampling} - f_{workload}| = f_{pattern}$, which yields the results shown in Table 3. The internal sampling rate is approx. 1996 Sa/s, which is 0.2 % slower than the 2 kSa/s noted in the specification [5, Sec. 11.3.2.2]—the same 0.2 % as for the update rate of the interface noted above (see Section 4.3).

This leads us to hypothesize that the clocks of the system and the OCC diverge. The experiment also recorded the timestamps and number of accumulated samples reported by the OCC: Over the approx. 30 min total experiment duration the clock deviates by less than one *part per million* (PPM) from the expected 512 MHz,[15] whereas the accumulator reports 1996.16 collected samples per second.

6.3 Potential Errors and Implications

For this experiment, the power from energy slowly shifts between 225 W and 285 W as visible in Figure 8. By further adjusting the frequency of the workload closer to the internal sampling rate, it would be possible to stretch out this pattern further. This may potentially hide its periodic behavior—in such a worst-case scenario only the low or only the high level might be sampled. Assuming a 255 W true average power consumption in the middle between both levels (as the workload is split evenly between idle and work), a sampling of only low (or only high) levels would yield a 12 % error in comparison. Such an error occurs only under very specific circumstances and represents a worst-case scenario. Hence, for practical applications the error is likely much lower.

One should prefer the accumulator-based power from energy to measure power consumption, as its sampling rate of ~2 kSa/s yields more precise results compared to the individual samples reported with ~25 Sa/s at every interface update (i. e., every ~40 ms). The magnitude of the observed error depends on the ability of the monitored component to change the power consumption within one sampling period. Observing GPUs or the entire system may result in different errors, depending on the particular setup.

In general, the observed 0.2 % deviation from the advertised 2 kSa/s internal sampling rate is hardly relevant to measuring applications: As the OCC reports the number of accumulated samples (see Table 2), the computation of the power from energy (see Section 5.2) remains correct. For practical applications in general we do not expect a relevant impact in accuracy introduced through aliasing.

[15] The trace is recorded on the same system; but the accuracy of neither the trace nor the system is validated. For this particular experiment, a single update cycle of the interface takes 40 ms, which corresponds to 20 PPM.

7 SUMMARY AND FUTURE WORK

In this paper, we presented a detailed description of the *On-Chip Controller* (OCC) of PowerNV platform processors. We described the available power sensors (see Table 1) and the interfaces that can be used to retrieve their readouts. We measured that such readouts take between 3.8 µs to 10.8 µs (mean) depending on the interface, and that new values are provided every 40.08 ms (24.95 Sa/s)—which is 0.2 % slower than we expected. Then, we compared the OCC measurement with an external measurement and presented our Score-P plugin to integrate these sensor readouts into OTF2 traces. This comparison of values collected from PSUs, BMC, and OCC did not verify the correctness of the OCC-reported data, but still confirms its plausibility. Furthermore, we discovered an emerging aliasing effect for workloads with a frequency matching the OCC's internal sampling rate, through which we produce 12 % error in carefully crafted experiments. This aliasing also exposes that the internal sampling rate is 1996 Sa/s, which is 0.2 % slower than specified in the documentation—similar to the external update rate of the interfaces.

The used workload generator only supports CPUs, hence the sensor behavior under load is only experimentally verified for CPUs. By extending the workload generator to GPUs these experiments could cover more load scenarios. All sensors, including those for GPUs, are processed by the same mechanisms of the OCC. Hence, we expect no discrepancies to the general behavior described in the paper. In particular, the principle for provoking the 12 % error of the measured energy for CPUs through aliasing can be applied to GPUs as well, but the magnitude of this error heavily depends on the installed hardware. To confirm the values reported by the OCC additional sensors are required. This can be achieved by manually instrumenting the node to measure the power consumption of the entire system, CPUs, and GPUs externally.

While in this paper we discussed the entire measurement pipeline of the OCC and quantified worst-case errors, additional measurements, e. g., using external hardware, could further strengthen the confidence in the out-of-the-box power measurements provided by the OCC.

ACKNOWLEDGMENTS

This work is supported in part by the German National High Performance Computing (NHR@TUD). The authors are grateful to the Center for Information Services and High Performance Computing at TU Dresden for providing the Power9 Systems used in the measurements and the support during them.

REFERENCES

[1] Sheldon Bailey, Charles Lefurgy, Andrew Jeffery, Stewart Smith, and Markus Hilger. 2020. AMESTER. https://github.com/open-power/amester

[2] Shilpasri G Bhat. 2016. Enabling Instrumentation Using Programmable on-chip Components to Monitor Sensors. https://openpowerfoundation.org/enabling-instrumentation-using-programmable-on-chip-components-to-monitor-sensors

[3] Shilpasri G Bhat. 2018. Openpower based Inband OCC sensors. https://github.com/shilpasri/inband_sensors

[4] Martha Broyles. 2016. *OCC Firmware Interface Specification for Open Power* (1.3 ed.). IBM, Hopewell Junction, New York. https://raw.githubusercontent.com/open-power/docs/master/occ/OCC_OpenPwr_FW_Interfaces.pdf

[5] Martha Broyles. 2019. *OCC Firmware Interface Specification for POWER9* (0.24 ed.). IBM, Hopewell Junction, New York. https://raw.githubusercontent.com/open-power/docs/master/occ/OCC_P9_FW_Interfaces.pdf

[6] CLEAResult. 2022. What is 80 PLUS certified? https://www.clearesult.com/80plus/program-details

[7] Spencer Desrochers, Chad Paradis, and Vincent M. Weaver. 2016. A Validation of DRAM RAPL Power Measurements. In *Proceedings of the Second International Symposium on Memory Systems* (Alexandria, VA, USA) *(MEMSYS '16)*. Association for Computing Machinery, New York, NY, USA, 455–470. https://doi.org/10.1145/2989081.2989088

[8] Manuel F. Dolz, Mohammad Reza Heidari, Michael Kuhn, Thomas Ludwig, and Germán Fabregat. 2015. ArduPower: A Low-cost Wattmeter to improve Energy Efficiency of HPC Applications. In *2015 Sixth International Green and Sustainable Computing Conference (IGSC)*. 1–8. https://doi.org/10.1109/IGCC.2015.7393692

[9] DTMF. 2018. Redfish Scalable Platforms Management API Specification. https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.6.0.pdf

[10] Dominic Eschweiler, Michael Wagner, Markus Geimer, Andreas Knüpfer, Wolfgang E. Nagel, and Felix Wolf. 2012. Open Trace Format 2: The Next Generation of Scalable Trace Formats and Support Libraries. In *Applications, Tools and Techniques on the Road to Exascale Computing (Advances in Parallel Computing, Vol. 22)*. 481–490. https://doi.org/10.3233/978-1-61499-041-3-481

[11] Eric J. Fluhr, Steve Baumgartner, David Boerstler, John F. Bulzacchelli, Timothy Diemoz, Daniel Dreps, George English, Joshua Friedrich, Anne Gattiker, Tilman Gloekler, Christopher Gonzalez, Jason D. Hibbeler, Keith A. Jenkins, Yong Kim, Paul Muench, Ryan Nett, Jose Paredes, Juergen Pille, Donald Plass, Phillip Restle, Raphael Robertazzi, David Shan, David Siljenberg, Michael Sperling, Kevin Stawiasz, Gregory Still, Zeynep Toprak-Deniz, James Warnock, Glen Wiedemeier, and Victor Zyuban. 2015. The 12-Core POWER8™ Processor With 7.6 Tb/s IO Bandwidth, Integrated Voltage Regulation, and Resonant Clocking. *IEEE Journal of Solid-State Circuits* 50, 1 (2015), 10–23. https://doi.org/10.1109/JSSC.2014.2358553

[12] Christopher Gonzalez, Michael Floyd, Eric Fluhr, Phillip Restle, Daniel Dreps, Michael Sperling, Rahul Rao, David Hogenmiller, Christos Vezyrtis, Pierce Chuang, Daniel Lewis, Ricardo Escobar, Vinod Ramadurai, Ryan Kruse, Juergen Pille, Ryan Nett, Pawel Owczarczyk, Joshua Friedrich, Jose Paredes, Timothy Diemoz, Saiful Islam, Donald Plass, and Paul Muench. 2018. The 24-Core POWER9 Processor With Adaptive Clocking, 25-Gb/s Accelerator Links, and 16-Gb/s PCIe Gen4. *IEEE Journal of Solid-State Circuits* 53, 1 (2018), 91–101. https://doi.org/10.1109/JSSC.2017.2748623

[13] Christopher Gonzalez, Eric Fluhr, Daniel Dreps, David Hogenmiller, Rahul Rao, Jose Paredes, Michael Floyd, Michael Sperling, Ryan Kruse, Vinod Ramadurai, Ryan Nett, Saiful Islam, Juergen Pille, and Donald Plass. 2017. 3.1 POWER9™: A processor family optimized for cognitive computing with 25Gb/s accelerator links and 16Gb/s PCIe Gen4. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 50–51. https://doi.org/10.1109/ISSCC.2017.7870255

[14] Daniel Hackenberg, Thomas Ilsche, Robert Schöne, Daniel Molka, Maik Schmidt, and Wolfgang E. Nagel. 2013. Power measurement techniques on standard compute nodes: A quantitative comparison. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 194–204. https://doi.org/10.1109/ISPASS.2013.6557170

[15] Daniel Hackenberg, Robert Schöne, Thomas Ilsche, Daniel Molka, Joseph Schuchart, and Robin Geyer. 2015. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. IEEE. https://doi.org/10.1109/ipdpsw.2015.70

[16] Marcus Hähnel, Björn Döbel, Marcus Völp, and Hermann Härtig. 2012. Measuring energy consumption for short code paths using RAPL. *ACM SIGMETRICS Performance Evaluation Review* 40, 3 (dec 2012), 13–17. https://doi.org/10.1145/2425248.2425252

[17] Thomas Ilsche, Robert Schöne, Joseph Schuchart, Daniel Hackenberg, Marc Simon, Yiannis Georgiou, and Wolfgang E. Nagel. 2018. Power Measurement Techniques for Energy-Efficient Computing: Reconciling Scalability, Resolution, and Accuracy, In SICS Software-Intensive Cyber-Physical Systems. *Computer Science - Research and Development*. https://doi.org/10.1007/s00450-018-0392-9

[18] Intel Corporation. 2021. *Intel® 64 and IA-32 ArchitecturesSoftware Developer's Manual*.

[19] Intel Corporation, Hewlett-Packard, NEC, and Dell. 2013. *IPMI Specification*. Technical Report. https://www.intel.com/content/www/us/en/servers/ipmi/ipmi-second-gen-interface-spec-v2-rev1-1.html

[20] International Business Machines and OpenPOWER Foundation. 2020. *Power ISA™*. https://files.openpower.foundation/s/XXFoRATEzSFtdG8

[21] Edward A. James, Guenter Roeck, and Rob Herring. 2017. [PATCH v3 00/12] hwmon: Add On-Chip Controller hwmon driver. https://lore.kernel.org/lkml/1511222021-562-1-git-send-email-eajames@linux.vnet.ibm.com thread on linux kernel mailing list.

[22] Andreas Knüpfer, Christian Rössel, Dieter an Mey, Scott Biersdorff, Kai Diethelm, Dominic Eschweiler, Markus Geimer, Michael Gerndt, Daniel Lorenz, Allen Malony, Wolfgang E. Nagel, Yury Oleynik, Peter Philippen, Pavel Saviankou, Dirk Schmidl, Sameer Shende, Ronny Tschüter, Michael Wagner, Bert Wesarg, and Felix Wolf. 2012. Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope,Scalasca, TAU, and Vampir. In *Tools for High Performance Computing 2011*, Holger Brunst, Matthias S. Müller, Wolfgang E. Nagel, and Michael M. Resch (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 79–91.

[23] B Li and E A Leon. 2018. Understanding Power Measurement Capabilities on Zaius Power9. (March 2018). https://www.osti.gov/biblio/1466115
[24] Antonio Libri, Andrea Bartolini, and Luca Benini. 2019. DiG: Enabling Out-of-Band Scalable High-Resolution Monitoring for Data-Center Analytics, Automation and Control. In *The 2nd International Industry/University Workshop on Data-center Automation, Analytics, and Control*.
[25] Linux Kernel Contributors. 2021. Linux hwmon Subsystem. https://hwmon.wiki.kernel.org/
[26] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. 2021. PLATYPUS: Software-based Power Side-Channel Attacks on x86. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE.
[27] Xinxin Mei, Ling Sing Yung, Kaiyong Zhao, and Xiaowen Chu. 2013. A measurement study of GPU DVFS on energy conservation. In *Proceedings of the Workshop on Power-Aware Computing and Systems*. 1–5.
[28] OCC Firmware Contributors. 2020. OCC Firmware. https://github.com/open-power/occ
[29] PROMETEUS Professor Meuer Technologieberatung und -Services GmbH. 2022. Green500 List - June 2022. https://www.top500.org/lists/green500/2022/06/
[30] Raritan. 2020. Technical Specifications / Engineering SubmittalsRaritan Model Number: PX3-5871I2U-F1N2. https://d3b2us605ptvk2.cloudfront.net/product-selector/pdus/PX3-5871I2U-F1N2/PX3-5871I2U-F1N2-spec.pdf
[31] John W. Romein and Bram Veenboer. 2018. PowerSensor 2: a Fast Power Measurement Tool. In *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. https://doi.org/10.1109/ISPASS.2018.00020
[32] Todd Rosedahl. 2015. On Chip Controller (OCC) Overview. *Tech. rep* (2015). https://cdn.openpowerfoundation.org/wp-content/uploads/2015/03/RosedahlTodd_OPFS2015_IBM_-031615.pdf
[33] Efraim Rotem, Alon Naveh, Avinash Ananthakrishnan, Eliezer Weissmann, and Doron Rajwan. 2012. Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge. *IEEE Micro* 32, 2 (3 2012), 20–27. https://doi.org/10.1109/MM.2012.12
[34] Robert Schöne, Thomas Ilsche, Mario Bielert, Andreas Gocht, and Daniel Hackenberg. 2019. Energy Efficiency Features of the Intel Skylake-SP Processor and Their Impact on Performance. arXiv:1905.12468 [cs.DC] accepted for publication.
[35] Robert Schöne, Thomas Ilsche, Mario Bielert, Markus Velten, Markus Schmidl, and Daniel Hackenberg. 2021. Energy Efficiency Aspects of the AMD Zen 2 Architecture. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. 562–571. https://doi.org/10.1109/Cluster48925.2021.00087
[36] Robert Schöne, Ronny Tschüter, Thomas Ilsche, Joseph Schuchart, Daniel Hackenberg, and Wolfgang E. Nagel. 2017. Extending the Functionality of Score-P Through Plugins: Interfaces and Use Cases. In *Tools for High Performance Computing 2016*, Christoph Niethammer, José Gracia, Tobias Hilbrich, Andreas Knüpfer, Michael M. Resch, and Wolfgang E. Nagel (Eds.). Springer International Publishing, Cham, 59–82.
[37] SkiBoot Contributors. 2021. SkiBoot. https://github.com/open-power/skiboot
[38] Linus Torvalds et al. 2017. Linux. https://kernel.org
[39] Hannes Tröpgen, Mario Bielert, and Thomas Ilsche. 2023. *Dataset Related to "Evaluating the Energy Measurements of the IBM POWER9 On-Chip Controller"*. https://doi.org/10.5281/zenodo.7670506