Pinpoint the Joules: Unifying Runtime-Support for Energy Measurements on Heterogeneous Systems

Sven Köhler[1], Benedict Herzog[2], Timo Hönig[3], Lukas Wenzel[1],
Max Plauth[1], Jörg Nolte[4], Andreas Polze[1] and Wolfgang Schröder-Preikschat[2]

[1] Hasso Plattner Institute for Digital Engineering, University of Potsdam, Germany
[2] Friedrich-Alexander University Erlangen-Nürnberg (FAU), Germany
[3] Ruhr University Bochum (RUB), Germany
[4] Brandenburg University of Technology Cottbus–Senftenberg (BTU), Germany
{firstname.lastname}@{hpi,cs.fau,rub,b-tu}.de

***Abstract*—For the design and operation of today's computer systems, power and energy requirements are highest priorities. Unlike performance analyses, however, power and energy measurements of heterogeneous systems are difficult to conduct. Especially at the system-software level, performing power and energy measurements remains challenging. Yet, such measurements are essential to improve software components for low power and high energy-efficiency.**

In this paper, we analyze and discuss the power and energy characteristics of several heterogeneous systems with up to 20 cores (160 hardware threads) and 1 TB of main memory. For the analyzed systems, we outline challenges regarding power and energy measurements and show ways to overcome limitations (i.e., sampling constraints). To improve the current state of the art in power and energy measurements at the system-software level, we present the design and implementation of PINPOINT, an energy-profiling tool which unifies different power and energy measurement interfaces.

***Index Terms*—Power and Energy Measurements, Heterogeneous Systems, System Software**

I. Introduction

Over the past years, power and energy demand have become primary design criteria and critical operating resources for computing systems. This applies to all types of systems, from embedded systems to desktop computer systems, high-performance computers, and supercomputers.

Until recently, the focus of system development has been on performance. However, as technologies have progressed (i.e., parallel computing on multi- and many-core processors) to address technologic limitations (i.e., breakdown of Dennard Scaling, pervasion of dark silicon [1]), power and energy requirements became the crucial points in computer systems' design. Due to the diversity of different computing system types (and their varying purposes), power and energy are important for various reasons. For small and embedded systems available power and energy resources must be handled with care to maximize the operating time (i.e., battery life) of the computing systems [2], [3]. This is due to the limited progress of battery technologies and the mediocre increase of energy density of energy storage over the last decades [4]. For large, high performance and cluster computing systems, however, power and energy became limiting factors as to thermal stress and external constraints (i.e., dependencies at the grid level) [5], [6]. The current Top-500 #1 system *Fugaku* can demand over 28 MW peak power [7], which easily exceeds the power demand of entire towns. The corresponding challenges arise by run-time requirements which demand dynamic adaptation of system characteristics (i.e., change between low and high power operations) *on the go* [8].

To improve the performance of computing systems, considerations about the system *performance* [9], [10] commonly is measured in the time dimension. For example, by specifying the execution time that is required to complete a single operation or a compound task. In the network domain, *throughput* [11] (e.g., rate at which operations are completed within a period) and *latency* [12] (e.g., delay of processing time of certain operations) are also seen on the background of *time*. For different types of systems, performance analysis procedures are similar: hardware-level timers are exposed via the operating system as programming interfaces. Software developers can use these facilities (Linux high resolution timers, for example) accordingly to analyze and improve performance, throughput, and latency of their program code at run-time.

However, there are no alike, standardized programming interfaces available in order to analyze or improve *power* and *energy* characteristics of computing systems. Such easy-to-use interfaces would be especially important as energy and power efficiency not always correlate with performance [13]. Instead, each and every hardware platform provides different means (if any) to obtain power values during run-time. As a result, there are no unified methods of performing measurements to determine the system power demand for the majority of platforms. Power measurements in combination with time measurements can be used to reflect about the energy demand (power over a period of time) of a system [14]. Thus, energy demand analyses of computing systems suffer from the same fact that unified programming interfaces are unavailable.

The increasing amount of computing systems' heterogeneity (i.e., CPUs, GPUs, TPUs[1]) make the situation even more

[1] tensor processing unit, accelerator application-specific integrated circuits which improve the execution of neural networks

difficult: to provide a holistic view on the power and energy demand of a computing systems, various different power-measurement interfaces must be used and combined. The manifold variants to measure power and energy—that are diverse in character and control—make it inherently difficult to improve not only performance properties, but also power and energy properties of software.

As reliable energy models [15], [16] are commonly unavailable and became difficult to establish for many of hardware components, we investigate the characteristics of different (on-board and external) power and energy measurement technologies and use them for the analysis on nine heterogeneous computing systems. Based on our findings, we describe the design and implementation of a Perf-Inspired Energy Profiling Tool (PINPOINT) [17], an advanced software tool that eases research work that targets improving power and energy efficiency of software for heterogeneous computing systems.

With this paper we make the following main contributions:

- an in-depth analysis and discussion of hardware and software interfaces for power and energy measurements
- the design & implementation of PINPOINT, an advanced software tool for power and energy demand analyses
- an analysis and evaluation of power and energy characteristics for several platforms and accelerators (i.e., GPUs)

The paper is structured as follows. First, we outline power and energy characteristics of nine different hardware platforms (Section II). We include embedded systems (i.e., ARM Cortex-A, and -M) that have become increasingly popular in heterogeneous HPC systems [18], [19] and supercomputers [20] as well as high-performance systems (i.e., AMD Ryzen, Intel Xeon, and IBM POWER8) to discuss individual, important differences. In Section III, we discuss details on power characteristics of the hardware platforms and outline their individual interfaces for power measurements. We share our gained knowledge on how to perform power and energy measurements on the discussed hardware platforms and describe the design and implementation of our contribution, a platform-agnostic software tool for unified power and energy measurements (PINPOINT) [17] in Section IV. We further present and discuss evaluation results of several measurement series and show operational details of the platform-agnostic software tool for unified power and energy measurements (Section V). Related work is discussed in Section VI, and Section VII concludes the paper.

II. Platforms

This section introduces several hardware platforms that we have analyzed for this paper. Besides high-performance computing systems from AMD, Intel, and IBM, we also include hardware accelerator hardware (i.e., Tensor Processing Units, GPUs) and embedded systems (i.e., ARM SoCs). Such compute units have become increasingly popular in heterogeneous HPC systems [18], [19] and supercomputers [20] over the last years. With this diverse selection of platforms we stress the strong deviations and differences among the heterogeneous systems that must be considered for power and energy measurments. Table I outlines the requirements and preconditions to measure each platform's power or energy demand and shows a system overview and provides details about CPU, memory, and available accelerators hardware, if applicable.

1) Coral USB Accelerator (Edge TPU): The advent of machine learning (ML) techniques lead to an ever-rising need for more processing power. Usually, the training process (i.e., *learning*) is more compute-intensive than the inference process (i.e., *execution*). Therefore, ML developers started to accelerate their training utilizing GPUs and ASICs like *Tensor Processing Units* (TPUs). However, IoT scenarios require not only efficient training, but also energy-efficient inference supported by accelerators [21]. The Coral USB accelerator hosts an Edge TPU, a specialized ASIC for power-efficient execution of ML models (i.e., neural networks). The accelerator communicates with a host PC via a USB-C connection controlled by an ARM Cortex M0+ microcontroller. Additionally, the USB connection powers the accelerator. The M0+ controls the communication channel and is not used for inference. According to the vendor, the Edge TPU executes ML models with up to 4 tera-operations per second (TOPS) using $0.5\,\mathrm{W}$ per TOPS [22]. Unfortunately, the board provides no means to retrieve the drawn power. To measure the energy demand of the accelerator, we intercept the USB connection and measure the power draw. Therefore, we use the LTC2991 voltage and current monitor, as described in Section III-B2.

2) NXP Kinetis FRDM-KL02Z: The NXP Freedom KL02Z board runs an ARM Cortex M0+ microcontroller with a slightly higher clock rate than the usual of $48\,\mathrm{MHz}$ [24]. It has access to $4\,\mathrm{kB}$ of SRAM and $32\,\mathrm{kB}$ of flash memory. For communication purposes, the board is equipped with an nRF24L01+ transceiver from Nordic Semiconductor, which transmits and receives in the $2.4\,\mathrm{GHz}$ ISM band [30]. This setup resembles a low-power sensor node with flexible communication abilities. We measure the power demand of the whole setup (i.e., board, microcontroller, and transceiver) by connecting the board's power supply to the MeasureAlot measurement device, as described in Section III-B1.

3) Microchip SAMA5D3 Xplained: The Microchip SAMA5D3 Xplained board hosts an ARM Cortex A5 microprocessor running with up to $536\,\mathrm{MHz}$ [25]. The memory composition is very flexible as several memory interfaces are available. In total, the memory in our configuration consists of two $64\,\mathrm{kB}$ SRAM modules running at CPU frequency, two $128\,\mathrm{MB}$ DDR2 modules, and $256\,\mathrm{MB}$ SLC flash as persistent memory. One advantage of this board is the extensive number of pins, where the power demand of different components can be measured separately. In fact, the board supports isolated measurements of the CPU core and embedded memories[2], the DDR2 interface, and several other peripheral components. This allows fine-grained and detailed power and energy demand measurements and

[2]The incorrect JP1 routing, described in the board's user guide, was fixed before any measurement: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11269-32-bit-Cortex-A5-Microcontroller-SAMA5D3-Xplained_User-Guide.pdf

TABLE I: Details and characteristics of different hardware platforms. The table shows the utilized CPUs (including number of cores and hardware threads), clock rate, and memory configuration. If available, additional accelerators are shown.

	Device	CPU	Cores/ Threads	Speed	Memory		Accel.
[23]	**Coral USB**	Cortex M0+	1/1	32 MHz	2 kB	SRAM	1x TPU
[24]	**FRDM-KL02Z**	Cortex M0+	1/1	48 MHz	4 kB	SRAM	—
					8 MB	SDRAM	
[25]	**SAMA5D3**	Cortex A5	1/1	536 MHz	256 MB	DDR2	—
[26]	**Odroid-C2**	Cortex A53	4/4	2.00 GHz	2 GB	DDR3	—
[27]	**Jetson TX2**	Cortex A57	4/4	2.00 GHz	8 GB	LPDDR4	1x iGPU
		Denver 2	2/2	2.00 GHz			
—	**Desktop PC**	Ryzen 3950X	16/32	3.50 GHz	32 GB	DDR4	—
[28]	**HPE BL460c**	2×Xeon E5	12/24	2.00 GHz	128 GB	DDR3	—
[29]	**IBM S824L**	2×POWER8	20/160	3.42 GHz	1 TB	DDR3	2x dGPU
							1x FPGA

accurate attribution of induced energy demand to specific hardware components. We measure the power draw using the LTC2991 voltage and current monitor as described in Section III-B2.

4) Odroid-C2: The Odroid-C2 [26] is a low-cost Single-Board Computer (SBC) based on the Amlogic S905 SoC. The S905 builds around an ARM Cortex-A53 quad-core CPU clocked at up to 2.0 GHz. It is optimized for energy efficiency, as the A53 is the energy-efficient counterpart to the more powerful A57 in *big.LITTLE* CPUs. At release, one advantage of the C2 was the *aarch64* Linux image, which could use the 64-bit *ARMv8* CPU, compared to legacy 32-bit images of other low-cost SBCs. Like the majority of SBCs, the C2 equips no power monitoring facilities. To cover the C2 in our analysis, the INA260 and MCP39F511N power monitors are employed as elaborated in Section III-B.

5) Jetson TX2: Ranging from autonomous driving to portable game consoles, NVIDIA Jetson compute modules target embedded use cases that require decent GPU performance in a constrained power envelope. The Jetson TX2 is based on the Tegra X2 SoC, which uses a Heterogeneous Multi-Processing (HMP) approach to implement the 64-bit *ARMv8* ISA. It comprises two NVIDIA Denver 2 CPU cores and four ARM Cortex-A57 CPU cores to enable different power and performance profiles. On the GPU side, it provides 256 CUDA cores based on the Pascal microarchitecture. For an embedded platform, the TX2 provides a fast 128-bit memory interface clocked at up to 1866 MHz. The board is equipped with two INA3221 triple-channel power monitors. For all measurements, the system operates in the unrestricted performance and energy profile (i.e., Max-N).

6) Ryzen-based Desktop PC: The third Ryzen series shows promising performance results for reasonable prices for both desktop and server applications. Therefore, we analyze a custom-built PC based on an AMD Ryzen 9 3950X with 16 cores and 32 hardware threads running at a base clock rate of 3.5 GHz and maximum boost rates of 4.7 GHz [31]. The RAM configuration comprises two 16 GB DDR4 memory modules. An important feature is a semi-compatible RAPL interface available for third-generation Ryzen processors [32]. The contents of the RAPL model-specific registers (MSRs) are identical to the Intel RAPL interface. However, the MSR numbers differ. This allows a fine-grained on-board power evaluation of the CPU package, specific cores, and CPU subsystems.

7) HPE ProLiant BL460c Gen8 Server: The BL460c is a blade server that is equipped with two Intel Xeon E5-2620 CPUs clocked at up to 2.0 GHz. Like most servers, it only exposes coarse-grained power draw readings via the Intelligent Platform Management Interface (IPMI). Therefore, only the energy demand of the CPUs can be quantified at a sufficient level of detail via the RAPL interface, because the power rating of the blade center exceeds the limits of all external measurement approaches presented in this paper. Section III-A provides details about the IPMI and RAPL approaches.

8) Power System S824L: With a maximum rated power of 2.3 kW, the IBM Power System S824L features two POWER8 CPUs with ten cores and eight buffered DDR3 memory channels, each. Providing a per-channel memory bandwidth of 24 GB/s, one of the system's key strengths is its high total memory bandwidth of up to 384 GB/s. The Simultaneous Multithreading (SMT) level of the POWER8 CPUs can be configured to provide up to eight hardware threads per core, resulting in a maximum of 160 hardware threads over 20 cores. Furthermore, it is equipped with an NVIDIA Tesla K80 GPU and a Nallatech N250S accelerator card featuring a Xilinx Kintex UltraScale KU060 FPGA, making it a good candidate to study heterogeneous system architectures. System-wide power measurements are only supported through the coarse-grained IPMI interface and two external MCP39F511N power monitors. The latter power monitors are subject to our custom system configuration and, thus, are commonly unavailable for alike POWER8 systems elsewhere.

III. Measuring Power and Energy

In this section we discuss the necessary different methods for measuring the power demand of the hardware platforms that we presented in Section II. We distinguish between *on-board measurement facilities* (i.e., methods that are integrated with the individual hardware platform and *external measurement facilities* (i.e., methods that use standalone devices). Each measurement device is best suited for an individual field of

application and we discuss the individual assets and drawbacks (i.e., different sampling rates and measuring accuracy).

A. On-Board Measurement Facilities

1) NVIDIA Jetson TX2 module: NVIDIA equips the Jetson TX2 module with two triple channel INA3221 [33] power monitors. The monitors report voltage and current values of the power rails by averaging the last 512 samples from the continuously probed data with a precision of 5% [34]. From these values, the power draw is computed and exposed via Linux `sysfs`-interface under `/sys/bus/i2c/drivers/ina3221x/`. Individual measuring points are provided for several subsystems such as CPU, GPU, DDR memory, and the remainder of the SoC. Additionally, the total power draw of the entire compute module is measured. Since the exact conversion time settings of the INA3221 are unspecified, we can only estimate that the effective sampling frequency is in the order of $20\,\mathrm{Hz}$.

2) Running Average Power Limit (RAPL): Starting with Sandy Bridge CPUs, Intel integrated the *Running Average Power Limit* (RAPL) interface in its CPUs [35]. The main objective of RAPL is limiting the power drawn by the CPU. These limits are helpful, for example, for data centers to control the power consumption at rack level and maximize its power supply utilization. However, the RAPL interface also provides a means to measure the energy demand of the CPU at different granularities (e.g., whole CPU package, cores, integrated GPU). The energy demand is determined by detailed energy models with access to many CPU-internal performance counters. The energy model is enhanced by actual energy measurements, depending on the specific CPU architecture and model. The RAPL *Energy Status* registers contain the cumulative energy consumption and are updated every $1\,\mathrm{ms}$. During high load the registers wrap around after around $60\,\mathrm{s}$, depending on the CPU's maximal power demand. Starting with the third-generation of Ryzen processors, AMD provides a semi-compatible RAPL interface (i.e., same register contents, but different register locations) [32].

The main advantages of RAPL are its convenient usage, little overhead, and accurate measurements. However, it is not possible to generate fine-grained information about the power demand over time, but only about the energy demand.

3) Baseboard Management Controller: Using the baseboard management controllers available in servers, most vendors expose basic power consumption measurements via IPMI. Even though the implementation can vary from vendor to vendor, we found that with roughly $1\,\mathrm{Hz}$, the sampling rate is usually very low. As such, these measurements are not suitable for detailed energy profiling tasks.

B. External Measurement Facilities

1) MeasureAlot: The usual approach of energy demand measurements consists of sampling the power demand over a period of time and eventually integrating over the power demand to calculate the energy demand. However, this may lead to inaccuracies if the power demand differs significantly between two samples (i.e., power-demand peaks). Increasing the sampling rate, which usually requires expensive measurement devices, reduces these inaccuracies, but cannot eventually avoid them.

Our custom-built *MeasureAlot* device avoids this problem by not sampling at all [13]. Instead, it uses a current mirror to duplicate the drawn current to charge a capacitor with known capacity. If the capacitor is fully charged, the current mirror is redirected to charge a second capacitor. In the meantime, the first capacitor is discharged and can be used again once the second capacitor is fully charged. The energy demand can be calculated by multiplying the number of capacitor charges with the energy required to charge the capacitor once.

The MeasureAlot device provides general-purpose input/output (GPIO) pins to a) start and stop a measurement or b) trigger an intermediate result while continuously measuring. The GPIO pins allow for a low-overhead measurement control. Results are buffered on the device and can be retrieved via a USB connection during or after a measurement process.

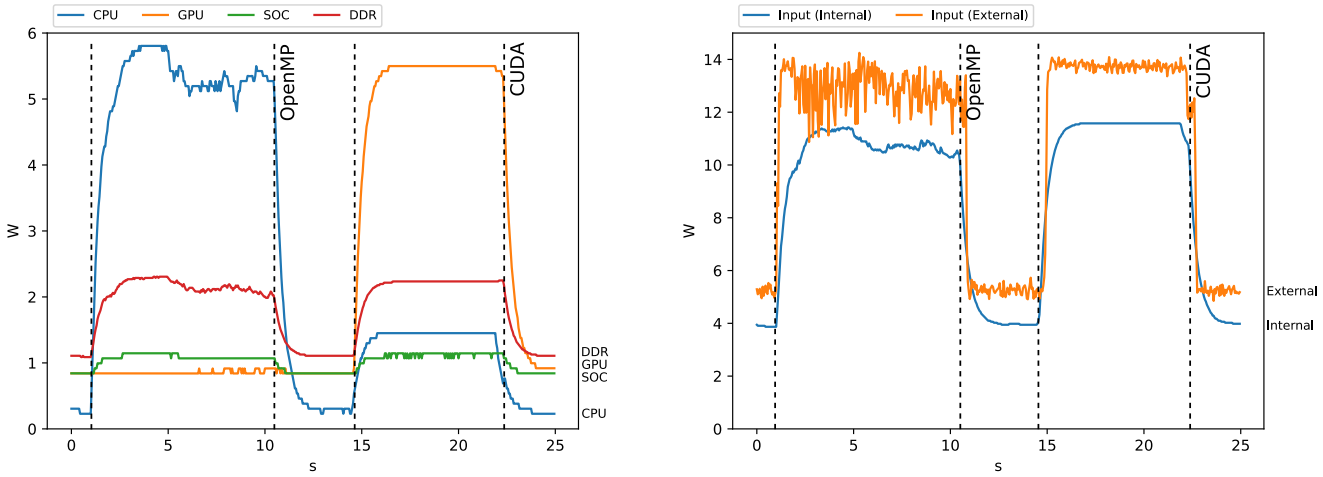
This approach has the advantage to include all power peaks, independent of their duration, and thus provide precise measurements. However, this approach provides only the energy demand between two points in time and no fine-grained information about the power demand. Additionally, due to the design of the measurement device, the energy demand is at least doubled during measurements.

2) Linear Technology LTC2991: We use an LTC2991 temperature, voltage, and current monitor with up to four channels for a variety of small hardware platforms [36]. The LTC2991 sits on a DC1785B board, which provides a (replaceable) shunt resistor for each measurement channel. The provision of different shunts allows simultaneous measurements of hardware components with different power demands, that is, relatively high and low power demand, respectively. The LTC2991 utilizes a 14-bit analog-to-digital converter (ADC) with $1\,\%$ accuracy and a sampling rate of $250\,\mathrm{Hz}$. If more than one device-under-test (DUT) is connected to the measurement board, the ADC multiplexes between all connected channels. Hence, the per-channel sampling rate depends on the number of connected DUTs.

We use an Atmega328P connected to the DC1785B to integrate the power measurements and provide a USB interface to a host. The host retrieves the measurement data, which is either power or energy demand, over the USB connection.

3) Texas Instruments INA260: To enable use cases ranging from continuous monitoring of a small SBC cluster down to fine-grained tracing of a workloads executed on individual cluster nodes, we designed a custom power measurement system based on the INA260 single-channel voltage and current monitor chip [37]. The system consists of one or more modules with four INA260 chips each sharing a common input voltage rail. Each channel can theoretically sustain a current of $15\,\mathrm{A}$ at $30\,\mathrm{V}$, but based on the board layout, we estimate the total load across all four channels should not exceed $20\,\mathrm{A}$.

The monitor chips are connected to an I2C-Bus, which can be shared across up to four modules. The fastest single-



(a) Internal power measurement facilities with multiple measuring points.

(b) Internal and external measurement techniques for total power draw at input.

Fig. 1: Sampled power draw on Jetson TX2 during the execution of a heat dispersion simulation workload first using OpenMP, then CUDA. In (a), the internal power measurement infrastructure is used, yielding individual measurement points for CPU, GPU, DDR memory, and the remainder of the SoC. In (b), we measure the total power draw using the internal measurement facilities and the external MCP39F511N measurement method. Different Y-axis scales are used to present more details.

channel measurement is achieved with minimal voltage and current conversion times, resulting in a sampling rate of $(2 \cdot 0.14\text{ms})^{-1} = 3.6\,\text{kHz}$. With 16 active channels, the data rate constraint on the I2C bus allows for a sample rate of 925 Hz per channel. At the highest accuracy, each channel produces samples at a rate of $(2 \cdot 8.2\text{ms})^{-1} = 61\,\text{Hz}$.

4) Microchip MCP39F511N: A common problem to many systems is the absence of convenient access to the power supply of specific components or subsystems. For these systems, often the only practical way to measure the power demand is to intercept the system's power supply and measure the power drawn from the wall socket. The Microchip MCP39F511N is a dual-channel power measurement device that measures the power demand between the wall socket and a system [38]. A USB connection allows the easy retrieval of measurement data either by an external control host or by the measured device itself. The latter case introduces overhead during the measurement (i.e., retrieving the measurement data adds to the energy demand). However, it allows the measuring device to gain knowledge about itself and hence allows self-awareness.

The MCP39F511N allows up to 15 A at a maximum of 230 V, which is sufficient for most desktop and server platforms. The power demand is measured over a 2 mΩ shunt with a 24-bit delta-sigma analog-to-digital converter (ADC). This allows an accuracy of 0.5 %. The sampling rate is phase-locked to the line frequency and allows a configurable number of samples per line-frequency cycle. Usually, we decided to use four samples per cycle, which results in a sampling rate of 200 Hz to 240 Hz (depending on the country's line frequency). Due to small changes in the line frequency, the time between measurement samples may not be equidistant, which is why a timestamp for each power sample is recorded. By integrating the power demand over all measurement samples, the energy demand can be calculated.

IV. Energy Profiling with PINPOINT

In stark contrast to the availability of standardized performance measurement tools, comparable software infrastructures do not exist for power and energy measurements. As to the existence of manyfold power and energy measuring methods (cf. Section III) a corresponding unification at the system-software level is necessary, for example to provide standardized programming interfaces that implement power and energy measurements.

Inspired by the *perf* [39] performance analysis tools in Linux, we implemented Perf-Inspired Energy Profiling Tool (PINPOINT) to fill this gap. PINPOINT collects and processes power measurements and enables users to determine the energy demand of arbitrary applications. On startup, PINPOINT detects available power measurement data sources by probing known kernel interfaces and device ports from user space. Our current prototype implementation supports the Microchip MCP39F511N (which we use in conjunction with systems like our POWER8 installation, see Section II) and the NVIDIA Jetson's INA3221 power monitors. We designed PINPOINT to easily integrate and support additional power and energy measurement methods.

Like *perf*, PINPOINT offers many switches that users can leverage to optimize their measurement setup. Options include a list of power measurement data sources to be used, the number of runs the measurement should be repeated, the desired sampling interval, as well as forerun and trail times to be included in the analyses. By default, the raw energy demand

TABLE II: Available measurement facilities for the different hardware platforms.

Device	Jetson	RAPL	IPMI	M.Alot	LTC2991	INA260	MCP.
Coral USB	—	—	—	—	✓	✓	—
FRDM-KL02Z	—	—	—	✓	✓	✓	—
SAMA5D3	—	—	—	—	✓	✓	—
Odroid-C2	—	—	—	—	✓	✓	✓
Jetson TX2	✓	—	—	—	—	✓	✓
Desktop PC	—	✓	—	—	—	—	✓
HPE BL460c	—	✓	✓	—	—	—	—
IBM S824L	—	—	✓	—	—	—	✓

per power data source is reported. However, a switch can be supplied to report the Energy Delay Product (EDP) [40] for the workload under test. Additionally, the series of sampled power draws can be recorded or live-streamed over the network for off-site analysis and visualization. Finally, the source code of PINPOINT has been made freely available on GitHub [17].

Internally, PINPOINT takes a modular approach towards the diverse set of possible data sources: new data sources can be implemented against a stable interface for data acquisition and auto-detection routines, which are bound to PINPOINT at compile time. The actual measurements are performed by a sampling component, which spawns the workload, takes timestamps and synchronizes the data acquisition across one or more data sources. Finally there is the evaluation component that takes care of postprocessing the acquired data. It orchestrates multiple workload runs collecting statistical information, and can either output raw power samples or perform a numerical lower Darboux integration to determine consumed energy and the energy-delay product.

To achieve a close synchronization between workload execution and power measurements, PINPOINT runs alongside the measured workload. Even though this setup might theoretically interfere with the workload under test, the tool sleeps between taking samples and does not perform heavy computations during workload execution. Therefore, the degree of self-interference caused by this setup is minimal and can be neglected in practice. When external measurement techniques are employed, the tool can also be executed on a different machine than the workload under test. In that case, timestamps are used to align start and end of the workload with the power samples. However, this approach still requires a minimal overhead on the device under test for capturing timestamps and depends on very accurate clock synchronization between machines.

V. Evaluation

Having presented a variety of different, heterogeneous hardware platforms and measurement facilities in Sections II and III, respectively, this section provides an insight into how both worlds can be brought together. First, it is essential to determine which measurement facilities can be used in conjunction with each individual hardware platform. Second, we demonstrate how PINPOINT (cf. Section IV) ties hardware platforms and measurement facilities together.

To address the first issue, Table II provides an overview of the measurement facilities suitable for the different hardware platforms. The sparsity of the result matrix presented in Table II clearly communicates the unsatisfactory situation that a wide range of external measurement facilities are required to cover all platforms. On the positive side, a number of platforms are covered by more than one measurement technique.

For the evaluation of PINPOINT, we analyze the energy demand of CPU and GPU based execution of a heat dispersion simulation as an exemplary workload. The multi-threaded, CPU-based version is implemented using OpenMP, whereas the GPU-based version is implemented using CUDA. Operating on a 1000×1000 matrix of single-precision floating-point cells, $10\,000$ timesteps of the simulation are executed. Using a low-power compute platform and a high-performance compute node to represent the two ends of the spectrum of heterogeneous compute platforms, we perform our tests on a NVIDIA Jetson TX2 compute module (see Section II-5) and an IBM Power System S824L server (see Section II-8).

Figure 1 illustrates the raw power draw series measured with PINPOINT during the execution of first the CPU and then the GPU-based implementations on the Jetson TX2 board. Throughout all measurements, the board operates in the unrestricted performance and energy profile (i.e., Max-N). In Figure 1a, the internal INA3221 power monitors quantify the power consumption of individual subsystems, including CPU, GPU, DDR memory, and the remainder of the SoC. Being able to observe the power consumption of different subsystems provides valuable insights into the detailed system behavior.

To unequivocally determine which implementation requires less energy on the TX2, the power integral is computed. As Listing 1 demonstrates, PINPOINT fully automates this tedious process, as it takes care of capturing measurement data and postprocessing tasks. Using the Jetson-internal power monitor facilities, we measured that the CPU-based implementation with OpenMP consumed $95.93\,\mathrm{J}$, whereas the GPU version with CUDA consumed $86.93\,\mathrm{J}$, confirming that GPU-based execution indeed is more energy-efficient on the TX2 for the workload at hands.

Figure 1b compares power measurements on the same Jetson TX2 board gathered via two different measurement techniques: The internal series reports the total power draw of the Jetson's internal power sensors, whereas the MCP series are taken externally at the wall power level with the Microchip MCP39F511N power meter. The MCP measurement setup also includes the switching power supply of the TX2. Even though the basic shapes correspond, there are some interesting differences between the curves. As expected, the external measurement yields consistently higher power draws than the internal sensors. The difference is about $2\,\mathrm{W}$, which is plausible as the internal sensors capture only the actual module power consumption, while the external measurement also includes the carrier board, fan and power supply.

```
tx2$> pinpoint -i 50 -r 10 ./heat 1000 1000 10000 in.csv
[interval: 50ms, before: 0ms, after: 0ms, runs: 10]

       47436.04 mJ CPU    ( +- 4.65% )
       8084.43 mJ  GPU    ( +- 5.22% )
       9904.58 mJ  SOC    ( +- 4.26% )
       19315.06 mJ DDR    ( +- 3.58% )
       95933.73 mJ INPUT  ( +- 4.63% )

       8.91673945 seconds time elapsed ( +- 5.92% )
```

Listing 1: Example call of PINPOINT, inspecting the OpenMP workload's mean energy demand on the Jetson TX2 with the internal measurement facilities across 10 runs using a sampling interval of 50 ms.

```
s824l$> pinpoint -i 50 -r 10 ./heat 1000 1000 10000 in.csv
[interval: 50ms, before: 0ms, after: 0ms, runs: 10]

       1499510.65 mJ INPUT ( +- 6.19% )

       1.53317275 seconds time elapsed ( +- 6.19% )
```

Listing 2: Example call of PINPOINT, inspecting the OpenMP workload's mean energy demand on the IBM Power System S824L with the external MCP39F511N measurement method across 10 runs using a sampling interval of 50 ms.

Besides the offset between curves, there is an interesting difference between the curve details: the external measurement produces about 100 mW noise, whereas the internal sensors yield much smoother results. This effect can be explained by the configuration that NVIDIA employs for the INA3221 monitors, where the moving averages from the last 512 samples are reported [34]. While this configuration reduces noise, the external measurement curve responds directly with a sharp slope at the start of the example workloads, while the internal curve exhibits a significantly dampened response. Also, the external measurements show more clearly that the power draw during GPU-based execution is much more stable compared to CPU-based execution.

On the TX2, the additional power draw caused by our test workload exceeds the idle power draw significantly. A strikingly different picture is given by the power draw measurements of the IBM Power System S824L, as presented in Figure 2. On this system, the idle power draw exceeds the additional power draw caused by the workload execution significantly. On closer look, large amounts of the idle power draw in the S824L can be attributed to power-hungry components such as a large RAID array consisting of twelve SAS 15K hard drives, as well as a large amount of main memory.

The power draw measurements of the S824L have been performed using the external MCP39F511N measurement method, with two units and two channels each to cover all four power supplies of the S824L. Unfortunately, we were unable to use a second power draw measurement method, as the time resolution of the IPMI method turned out to be too coarse on this machine.

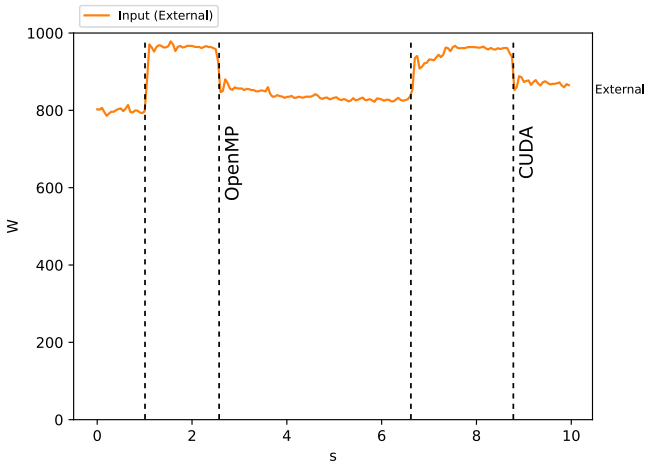


Fig. 2: Sampled power draw on IBM Power System S824L during the execution of a heat dispersion simulation workload first using OpenMP, then CUDA. The total power draw is measured using the external MCP39F511N measurement method.

In contrast to the TX2, the measurements presented in Figure 2 suggest that on the S824L, CPU-based execution of the heat dispersion simulation workload is faster and also more efficient compared to GPU-based execution. Using PINPOINT, we were able confirm this observation, as the CPU implementation consumed 1499.51 J, whereas the GPU version consumed 1982.84 J.

These examples demonstrated in our evaluation show that care should be taken for choosing the most efficient execution target for a certain workload. Also, the variety in measurement characteristics indicates that it is useful to incorporate multiple data sources in the same experimental setup, in order to gain additional insights and also validate different sources against each other.

VI. Discussion and Related Work

Determining or estimating the power and energy demand of computing systems at the software level has received increased attention [41], [42], [43], [44] as to new emerging technologies that require low power and energy-efficient operations. However, power measurement facilities in HPC systems are as system-specific as they are in the selection of platforms discussed in Section II. For example, the measurement methodology described by Laros et al. [45] is specific to Cray XT systems. The GEOPM framework [46] proposes a cluster-level power management architecture, with a focus on extensible job scheduling and power capping mechanisms. Being rooted in x86 system architectures, the intended variation points concentrate more on workload management and instrumentation instead of power measurement mechanisms. Nevertheless there are efforts to adapt GEOPM to different hardware platforms [47]. In contrast to GEOPM, PINPOINT operates on the node- instead of cluster-level and is focused on

supporting a wide variety of power measurement mechanisms without requiring workload instrumentation.

Grant et al. [48] conclude that for detailed energy demand attribution in scenarios where no application code knowledge can be assumed, external measurements are the preferred collection method. However, they also agree that corresponding measurement facilities are only available in selected platforms as they require specialized measurement hardware. This observation supports our impression that there is a large gap between coarse-grained external measurement methods at the level of power distribution units that do not allow breakdowns to the component level, and internal measurement methods such as the RAPL counters that have a limited scope and are platform-dependent.

Rieger et al. have recently presented a case study on the state of the art of assessing software energy consumption [49]. While numerous tools exist in the Android smartphone ecosystem, the few approaches available that are targeting desktop, server, or high-performance computing systems are platform-dependent. As a follow-up work, Rieger et al. have published a short paper in which they formulate the goal of platform-independent, method-exact energy measurement techniques [50]. However, they only use a single measurement technique and the aspect of platform-independence is only brushed at best. Their main contribution is that they evaluate two techniques for attributing energy measurements to methods in the application code, namely linear scaling and dynamic time warping. They conclude that both methods provide unsatisfactory results.

Measuring and improving the power and energy characteristics of HPC applications has been subject to many research works [51], [52] that rely on external hardware equipments (i.e., multi meters, power meters) which often provide mixed interfaces. Due to its adaptability and extensibility by design, our proposed tool-based measurement infrastructure PINPOINT makes existing measuring interfaces available via a generic programming interface.

To interface with existing software techniques, we consider that using established performance measurement methods (e.g., Linux perf [39]) are well-suited to integrate power and energy measurement analysis techniques, too. In contrast to performance analytics [9], [10], however, this paper has shown that necessary programming interfaces and analysis methods are manyfold and as heterogeneous as the underlying hardware platforms. We therefore advocate the design and implementation of unified and platform-agnostic power and energy measurement infrastructures to facilitate power and energy measurements at the software level. In the context performance analytics, the PAPI project [53] offers a well established and platform agnostic API to expose and access hardware performance counters. PAPI is aimed at internal performance counters, whereas PINPOINT integrates external measurement methods as well. Though it has not been a focus of this work, an integration of the PINPOINT backend into PAPI is an interesting line of future work.

An interesting aspect that has been beyond the scope of this work—and therefore is subject to future research—are methods for profiling the amount of energy required to transfer data [54], also in combination with a fine-grained system-level power demand analysis driven by PINPOINT.

VII. Conclusion

Power and energy demand of software and hardware components has emerged as a primary design criterion for all types of heterogeneous computing systems. On the one hand, battery capacities of uninterrupted power supplies (UPS) and embedded systems must be used efficiently—every joule counts—and on the other hand, high performance systems must adhere to thermal limits and their individual cooling capacities. During operation, the systems therefore must adhere to certain power and energy requirements. To improve the understanding of power and energy characteristics for different types of platforms, we discussed specific peculiarities of heterogenous systems, discussed different measurement methods, and discussed our gained insights and experiences. To advance the analyzability of systems we further propose PINPOINT, a platform-agnostic software tool for unified power and energy measurements. Compared to previous approaches [41], [42], [43], [44], PINPOINT provides a generic programming interface at the system-software level which enables power and energy measurements on heterogeneous systems independent of the available measurement backend(s).

Acknowledgment

This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 146371743 – TRR 89 “Invasive Computing” as well as the individual research grants SCHR 603/10-2 and NO 625/7-2.

References

[1] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” in *Proceedings of the 2011 ACM/IEEE International Symposium on Computer Architecture*, 2011, pp. 365–376.

[2] J. H. Snyder, J. B. McKie, and B. N. Locanthi, “Low-power software for low-power people,” in *Proceedings of the 1994 IEEE Symposium on Low Power Electronics*, 1994, pp. 32–35.

[3] J. R. Lorch and A. J. Smith, “Software strategies for portable computer energy management,” *IEEE Personal Communications*, vol. 5, no. 3, pp. 60–73, Jun. 1998.

[4] J. Li, Z. Du, R. E. Ruther, S. J. An, L. A. David, K. Hays, M. Wood, N. D. Phillip, Y. Sheng, C. Mao *et al.*, “Toward low-cost, high-energy density, and high-power density lithium-ion batteries,” *JOM*, vol. 69, no. 9, pp. 1484–1496, 2017.

[5] W.-c. Feng, X. Feng, and R. Ge, “Green supercomputing comes of age,” *IEEE IT Professional*, vol. 10, no. 1, pp. 17–23, 2008.

[6] T. Minartz, T. Ludwig, M. Knobloch, and B. Mohr, “Managing hardware power saving modes for high performance computing,” in *Proceedings of the 2011 IEEE International Green Computing Conference and Workshops*, 2011, pp. 1–8.

[7] TOP500.org. (2020) Fugaku Supercomputer. Acc. 2020-08-13. [Online]. Available: https://www.top500.org/system/179807/

[8] T. Hönig, C. Eibel, A. Wagenhäuser, M. Wagner, and W. Schröder-Preikschat, “How to make profit: Exploiting fluctuating electricity prices with albatross, a runtime system for heterogeneous HPC clusters,” in *Proceedings of the 2018 International Workshop on Runtime and Operating Systems for Supercomputers*, 2018, pp. 1–9.

[9] L. Svobodova, "Computer system performance measurement: Instruction set processor level and microcode level," Stanford, CA, USA, Tech. Rep., 1974.

[10] D. J. Lilja, *Measuring computer performance: a practitioner's guide*. Cambridge University Press, 2005.

[11] G. Jin and B. Tierney, "Netest: A tool to measure the maximum burst size, available bandwidth and achievable throughput," in *Proceedings of the 2003 IEEE International Conference on Information Technology: Research and Education*, 2003, pp. 578–582.

[12] A. Wolman, G. Voelker, and C. A. Thekkath, "Latency analysis of TCP on an ATM network," in *Proceedings of the Winter 1994 USENIX Technical Conference*, 1994, pp. 1–14.

[13] T. Hönig, H. Janker, C. Eibel, O. Mihelic, R. Kapitza, and W. Schröder-Preikschat, "Proactive energy-aware programming with PEEK," in *Proceedings of the 2014 USENIX Conference on Timely Results in Operating Systems*, 2014, pp. 1–14.

[14] B. Herzog, T. Hönig, W. Schröder-Preikschat, M. Plauth, S. Köhler, and A. Polze, "Bridging the gap: Energy-efficient execution of software workloads on heterogeneous hardware components," in *Proceedings of the 2019 ACM International Conference on Future Energy Systems (e-Energy '19)*, 2019, p. 428–430.

[15] P. Gschwandtner, M. Knobloch, B. Mohr, D. Pleiter, and T. Fahringer, "Modeling CPU energy consumption of HPC applications on the IBM POWER7," in *Proceedings of the 2014 Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2014, pp. 536–543.

[16] B. Acun, E. K. Lee, Y. Park, and L. V. Kale, "Support for power efficient proactive cooling mechanisms," in *Proceedings of the 2017 IEEE International Conference on High Performance Computing*, 2017, pp. 94–103.

[17] Sven Köhler. (2020) PINPOINT: Perf-Inspired Energy Profiling Tool. Acc. 2020-08-13. [Online]. Available: https://github.com/osmhpi/pinpoint

[18] D. Yokoyama, B. Schulze, F. Borges, and G. Mc Evoy, "The survey on ARM processors for HPC," *The Journal of Supercomputing*, vol. 75, no. 10, pp. 7003–7036, 2019.

[19] M. Plauth and A. Polze, "Are low-power SoCs feasible for heterogenous HPC workloads?" in *Proceedings of the 2016 Euro-Par: Parallel Processing - International Conference on Parallel and Distributed Computing (Workshops)*. Springer, 2016, pp. 763–774.

[20] N. Rajovic, P. M. Carpenter, I. Gelado, N. Puzovic, A. Ramirez, and M. Valero, "Supercomputing with commodity CPUs: Are mobile SoCs ready for HPC?" in *Proceedings of the ACM/IEEE International Conference on High Performance Computing, Networking, Storage and Analysis (SC'19)*, 2013, pp. 1–12.

[21] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, and et. al., "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 2017 ACM/IEEE Annual International Symposium on Computer Architecture*, 2017, p. 1–12.

[22] Google Coral. (2020) Edge TPU Performance Benchmarks. Acc. 2020-06-12. [Online]. Available: https://coral.ai/docs/edgetpu/benchmarks/

[23] ——. (2020) USB Accelerator Datasheet. Acc. 2020-06-12. [Online]. Available: https://coral.ai/static/files/Coral-USB-Accelerator-datasheet.pdf

[24] Freescale. (2013) FRDM-KL02Z User Manual. Acc. 2020-06-12. [Online]. Available: https://www.nxp.com/docs/en/user-guide/FRDMKL02ZUM.pdf

[25] Microchip. (2016) SAMA5D3 Series Datasheet. Acc. 2020-06-12. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11121-32-bit-Cortex-A5-Microcontroller-SAMA5D3_Datasheet_B.pdf

[26] Hardkernel. (2015) ODROID-C2 User Manual. Acc. 2020-06-12. [Online]. Available: https://magazine.odroid.com/wp-content/uploads/odroid-c2-user-manual.pdf

[27] NVIDIA. (2020) Jetson TX2 Series Module Datasheet. Acc. 2020-06-12. [Online]. Available: http://developer.nvidia.com/embedded/dlc/jetson-tx2-series-modules-data-sheet

[28] HPE. (2015) ProLiant BL460c Gen8 Server Blade QuickSpecs. Acc. 2020-06-12. [Online]. Available: https://www.hpe.com/h20195/v2/GetDocument.aspx?docname=c04123239

[29] IBM. (2014) Power System S824L Technical Overview and Introduction. Acc. 2020-06-12. [Online]. Available: https://www.redbooks.ibm.com/redpapers/pdfs/redp5139.pdf

[30] Nordic Semiconductor. (2008) nRF24L01+ Product Specification. Acc. 2020-06-12. [Online]. Available: https://infocenter.nordicsemi.com/pdf/nRF24L01P_PS_v1.0.pdf?cp=10_4_0_0

[31] AMD. (2020) Ryzen 9 3950X Product Information. Acc. 2020-06-12. [Online]. Available: https://www.amd.com/en/products/cpu/amd-ryzen-9-3950x

[32] ——. (2020) Processor Programming Reference (PPR) for AMD Family 17h. Acc. 2020-06-12. [Online]. Available: https://www.amd.com/system/files/TechDocs/54945_3.03_ppr_ZP_B2_pub.zip

[33] Texas Instruments. (2016) INA3221 Datasheet. Acc. 2020-06-12. [Online]. Available: https://www.ti.com/lit/ds/symlink/ina3221.pdf

[34] NVIDIA. (2020) Jetson TX2 Series Thermal Design Guide. Acc. 2020-08-25. [Online]. Available: http://developer.nvidia.com/embedded/dlc/jetson-tx2-series-thermal-design-guide

[35] Intel. (2020) 64 and IA-32 Architectures Software Developer's Manual (Volume 3). Acc. 2020-06-12. [Online]. Available: https://software.intel.com/content/dam/develop/public/us/en/documents/325384-sdm-vol-3abcd.pdf

[36] Linear Technology. (2020) LTC2911 Datasheet. Acc. 2020-06-12. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/2991ff.pdf

[37] Texas Instruments. (2016) INA260 Datasheet. Acc. 2020-06-12. [Online]. Available: http://www.ti.com/lit/ds/symlink/ina260.pdf

[38] Microchip. (2018) MCP39F511N Datasheet. Acc. 2020-06-12. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/20005473B.pdf

[39] V. M. Weaver, "Self-monitoring overhead of the linux perf_event performance counter interface," in *Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software*, 2015, pp. 102–111.

[40] R. Gonzalez and M. Horowitz, "Energy dissipation in general purpose microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 9, pp. 1277–1284, 1996.

[41] F. Chang, K. I. Farkas, and P. Ranganathan, "Energy-driven statistical sampling: Detecting software hotspots," in *Proceedings of the 2002 International Workshop on Power-Aware Computer Systems*. Springer, 2002, pp. 110–129.

[42] W. Baek, Y.-J. Kim, and J. Kim, "ePRO: A tool for energy and performance profiler for embedded applications," in *Proceedings of the 2004 IEEE International SoC Design Conference*, 2004, pp. 482–485.

[43] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app? fine grained energy accounting on smartphones with Eprof," in *Proceedings of the 2012 ACM European Conference on Computer Systems*, 2012, pp. 29–42.

[44] S. Schubert, D. Kostic, W. Zwaenepoel, and K. G. Shin, "Profiling Software for Energy Consumption," in *Proceedings of the 2012 IEEE International Conference on Green Computing and Communications*, 2012, pp. 515–522.

[45] J. H. Laros III, K. T. Pedretti, S. M. Kelly, W. Shu, K. B. Ferreira, J. V. Dyke, and C. T. Vaughan, *Energy-Efficient High Performance Computing: Measurement and Tuning*. Springer London, 2013.

[46] J. Eastep, S. Sylvester, C. Cantalupo, F. Ardanaz, B. Geltz, A. Al-Rawi, F. Keceli, and K. Livingston, "Global extensible open power manager: a vehicle for hpc community collaboration toward co-designed energy management solutions," *Supercomputing PMBS*, 2016.

[47] M. Puzović, V. Elisseev, K. Jordan, J. Mcdonagh, A. Harrison, and R. Sawko, "Improving performance and energy efficiency on openpower systems using scalable hardware-software co-design," in *High Performance Computing*, R. Yokota, M. Weiland, J. Shalf, and S. Alam, Eds. Cham: Springer International Publishing, 2018, pp. 411–417.

[48] R. E. Grant, J. H. Laros III, M. Levenhagen, S. L. Olivier, K. Pedretti, L. Ward, and A. J. Younge, "Evaluating energy and power profiling techniques for HPC workloads," in *Proceedings of the 2017 IEEE International Green and Sustainable Computing Conference*, 2017, pp. 1–8.

[49] F. Rieger and C. Bockisch, "Survey of approaches for assessing software energy consumption," in *Proceedings of the 2017 ACM International Workshop on Comprehension of Complex Systems*. Association for Computing Machinery, 2017, p. 19–24.

[50] ——, "Evaluating techniques for method-exact energy measurements: Towards a framework for platform-independent code-level energy measurements," in *Proceedings of the 2020 ACM Symposium on Applied Computing*, ser. SAC '20, 2020, p. 125–128.

[51] B. Rountree, D. K. Lowenthal, B. R. De Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, "Adagio: making DVS practical for complex HPC applications," in *Proceedings of the 2009 ACM International Conference on Supercomputing*, 2009, pp. 460–469.

[52] G. Da Costa and J.-M. Pierson, "DVFS governor for HPC: Higher, faster, greener," in *Proceedings of the 2015 Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2015, pp. 533–540.

[53] P. J. Mucci, S. Browne, C. Deane, and G. Ho, "Papi: A portable interface to hardware performance counters," in *Proceedings of the department of defense HPCMP users group conference*, vol. 710, 1999.

[54] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie, "Quantifying the energy cost of data movement in scientific applications," in *Proceedings of the 2013 IEEE International Symposium on Workload Characterization*, 2013, pp. 56–65.