DiG: Enabling Out-of-Band Scalable High-Resolution Monitoring for Data-Center Analytics, Automation and Control (Extended)

Antonio Libri · Andrea Bartolini · Luca Benini

arXiv:1806.02698v2 [cs.DC] 17 Jul 2019

Abstract Data centers are increasing in size and complexity, and we need scalable approaches to support their automated analysis and control. Performance counters and power consumption are their key "vital signs". State-of-the-Art (SoA) monitoring systems provide built-in tools to collect performance measurements, and custom solutions to get insight on their power consumption. However, with the increase in measurement resolution (in time and space) and the ensuing huge amount of measurement data to handle, new challenges arise, such as bottlenecks on the network bandwidth, storage and software overhead on the monitoring units. To face these challenges we propose a novel monitoring platform for data centers, which enables real-time high-resolution profiling (*i.e.*, all available performance counters and the entire signal bandwidth of the power consumption at the plug - sampling up to $20\,\mu s$ - with an error below $1\,\%$) and analytics, both at the edge (node-level analysis) and on a centralized unit (cluster-level analysis). The monitoring infrastructure is completely out-of-band, scalable, technology agnostic and low cost, and it is already installed in a SoA high-performance compute cluster (*i.e.*, D.A.V.I.D.E. - 18th in Green500 November 2017).

Antonio Libri
D-ITET, ETH Zurich, Zurich, Switzerland
E-mail: a.libri@iis.ee.ethz.ch

Andrea Bartolini
DEI, University of Bologna, Bologna, Italy
E-mail: a.bartolini@unibo.it

Luca Benini
D-ITET, ETH Zurich, Zurich, Switzerland
DEI, University of Bologna, Bologna, Italy
E-mail: lbenini@iis.ee.ethz.ch, luca.benini@unibo.it

1 Introduction

Data centers and High Performance Computing (HPC) systems are becoming increasingly complex and the need for novel methods to support their automation, analytics and control is garnering considerable attention [29]. In this direction, industry and academia researchers are pushing toward the use of Artificial Intelligence (AI) and Machine Learning (ML) techniques to address non-trivial challenges such as efficient management of computational / infrastructure resources, detection of anomalies and failures, and predictive maintenance. As an example, Duplyakin et al. [13] have shown how to get high-confidence predictions of the time-to-completion and energy consumption of scientific applications (which can help for a more efficient usage of the resources) via Active Learning techniques applied to regression problems. Other examples are based on unsupervised learning, such as [5] which introduces methods for real-time anomaly detection on streaming data (useful for an early warning about problems in the system and the hosted applications), and [33] that shows a way to detect malware using hardware features.

All these techniques exploit low-level monitoring of the hardware of the data-center infrastructure (*i.e.*, application and system performance, and related power and energy consumption). In particular, depending on the target use-case, some features can reveal more information than others: a highly flexible monitoring has to collect as many metrics as possible. On the other hand, this implies to face three main bottlenecks related to
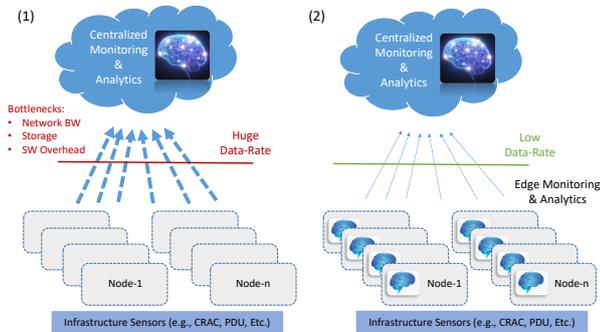
the large amount of monitoring data produced: (i) overhead on the network's bandwidth, (ii) overhead on the data storage capacity (to save measurements for post-processing analysis) and (iii) overhead on the software tools that have to handle the measurements (in real-time and offline).

Fig. 1 Data-center monitoring design and bottlenecks.

This is depicted in Figure 1 (left), which shows that the node dedicated to the monitoring software stack has the complete view of the status of the cluster (and thus can exploit measurements for ML analysis), but has to deal with the above mentioned bottlenecks. To give an example, Ilsche et al. developed a high resolution power monitoring system (*i.e.*, HAEC [20]) that supports a sampling rate of $500\,kS/s$ (kilo Samples per second) on 4 custom sensors. In general, high resolution power monitoring instrumentation is the current trend for both industrial and academia HPC facilities and data centers [20, 17, 24]. This is useful to appreciate the power consumption of application phases, but of course the finer the granularity the more difficult is to scale to a large number of nodes in a machine. For instance, instrumenting with HAEC the supercomputer *Sunway TaihuLight* - 2nd in Top500 of June 2018 and that includes around 41 thousand computing nodes [14] - would require a data collection network bandwidth of around $82\,GS/s$, with obvious overheads on software and storage to handle it.

An intuitive solution is to bring some of the "monitoring intelligence" to the edge and codesign the monitoring infrastructure to leverage data analysis between distributed monitoring agents and a centralized unit. This is represented in Figure 1 (right), which shows distributed smart monitoring agents that can carry out real-time analysis per node (*e.g.*, feature extraction, ML inference, etc.) and share information with the centralized monitoring at a much lower rate (*e.g.*, detection of an anomaly in a node, plus other measurements at a lower rate needed for cluster level analysis). In this distributed architecture, each monitoring agent has the complete knowledge of the status of its node, while the centralized monitoring unit has the complete view of the cluster, thus can carry out analysis at a higher level.

Current State-of-the-Art (SoA) monitoring solutions allow to collect measurements in-band and out-of-band by means of built-in tools (*e.g.*, Amester [31] or RAPL [22], which expose hardware performance counters) or custom sensors (*e.g.*, HDEEM [17] or HAEC [20], which provide fine grain power measurements), where the benefit of the out-of-band solution is no overhead on the computing resources. However, to the best of our knowledge, there is not yet a monitoring infrastructure for compute clusters and high performance machines that provides a flexible way to analyze all possible features (*i.e.*, all available hardware performance counters and the entire signal bandwidth of the node's power consumption). This paper, which builds upon and extends our previous publication [25], focuses on a novel scalable and high resolution monitoring infrastructure for data centers and HPC systems. The system is completely out-of-band and provides a highly flexible environment to work both at the edge and at cluster-level for data center analytics, automation and control.

Contributions of the work:

1. design of an out-of-band monitoring infrastructure - we named it DiG (*i.e.*, Dwarf in a Giant) - that exploits edge monitoring agents and centralized cluster-level monitoring for data centers analytics, automation and control. The platform design provides a highly flexible environment to tackle different challenges. We designed a custom power sensor at the plug to monitor the power consumption at high resolution, covering the entire signal bandwidth (sampling up to $20\,\mu s$) with a measurements precision below $1\,\%$ (σ) (therefore also suitable for the most rigorous power measurement requirements to benchmark a computing system in Top500 [16]). The system allows to interface with existing out-of-band telemetry (*e.g.*, Amester [31], IPMI [21]), but also with in-band built-in tools if required (*e.g.*, RAPL [22]). All the measurements are synchronized at sub-microseconds precision to obtain a detailed picture over time of the nodes and cluster state. We adopted a scalable and lightweight interface to the centralized monitoring (*i.e.*, MQTT [19]) to support large-scale computing centers. The monitoring infrastructure is technology agnostic (*i.e.*, already tested on different architectures, such as Intel, ARM and IBM) and low cost (*i.e.*, the custom power sensor does not require any motherboard redesign).

2. we report (i) the performance of the monitoring agents (*i.e.*, measurements granularity, precision,

synchronization, software overhead and scalability), along with (ii) an extensive campaign of ML inference benchmarks running on the dedicated embedded computers and based on deep Residual Networks (a.k.a. ResNets [18]), to obtain an assessment of their real-time inference capabilities, and (iii) an extensive set of tests based on frequency-domain analysis to show the capability of the high resolution monitoring to unveil high-frequency components directly related to the computation activity and including also two use-cases that can be used for anomaly detection.

3. we validated and calibrated our high-resolution power measurements, and provide detailed information on accuracy and precision (best in class w.r.t SoA data centers monitoring systems); moreover, we integrated the monitoring infrastructure in a SoA HPC cluster (*i.e.*, D.A.V.I.D.E. [6] - 18th in Green500 November 2017) that is already in production and available to the users community since more than one year.

4. We provide detailed information of both hardware and software architectures for the whole monitoring infrastructure (Sections 2.1, 2.2, 2.3 describe the edge infrastructure, while the cluster-level software - namely ExaMon - can be downloaded from [3]).

Outline: Section 2 presents the monitoring architecture. Its performance is analyzed in Section 3, together with several case studies of frequency-domain analysis on the high-resolution power measurements. We report related works in Section 4 and conclude the paper in Section 5.

2 Monitoring System Architecture

One of the main challenges we faced during the monitoring system design was to make it suitable for different hardware architectures and low cost. With this goal, we targeted only what is missing on today's built-in monitoring solutions [20]: a custom power sensor that allows high resolution monitoring. We placed it at the node power source to completely avoid motherboard re-design or modification. We then interfaced it with a dedicated low-cost embedded computer (one per node) that is suitable for monitoring applications.

A second challenge was to make the system highly flexible in terms of monitoring capabilities. With this goal, we interfaced the embedded computer with built-in tools to get per-component monitoring (*i.e.*, hardware performance counters) and have the complete knowledge of the status of the node. This information, along with the high resolution power monitoring, can

reveal not only insights on application behavior but also patterns on performance / failures of components (*e.g.*, fans, HDDs).

Finally, we exploited a scalable and lightweight interface (*i.e.*, MQTT) to send information to a centralized monitoring unit and perform cluster-level analytics. Figure 2 shows the main components of the monitoring system that will be described in this section.
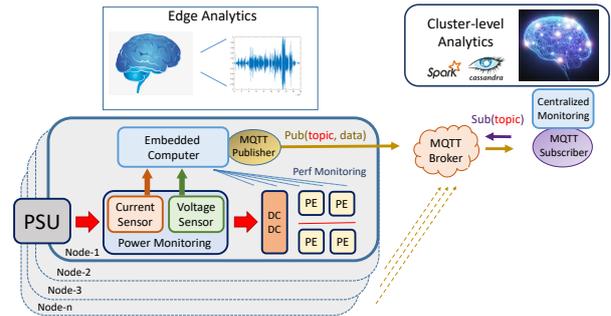

Fig. 2 Sketch of the monitoring system architecture.

2.1 High-Resolution Power Sensing

To provide high resolution measurements of the nodes' power consumption we placed a power sensing module between the Power Supply Unit (PSU) and the DC-DC converters that provide power for all the processing elements (PE) / electrical components within the node. Figure 3 shows the schematic of the power sensing module. We use a voltage divider based on high precision resistors to measure the voltage and a current transducer to measure the current. Their outputs are then connected to the ADC integrated in the embedded monitoring board, via first-order low-pass filters needed to counter aliasing effects. Indeed, due to the high operating frequencies of data centers / HPC nodes, the power consumption is highly dynamic, and therefore an anti-aliasing filter is required. We have chosen a voltage divider as it provides a simple but effective solution to properly scale the voltage in input to the ADC without any additional hardware (*e.g.*, an isolated power supply would be needed if using active components).

For the current transducer, we tested two configurations: one based on a Hall Effect (HE) sensor and one based on a current mirror and shunt resistor. Thanks to the high output linearity, both solutions obtain satisfactory results. We tested the first configuration with Intel Xeon E5-2600 (Haswell) and ARM Cavium TunderX architectures, while the second one on a cluster based on OpenPOWER IBM Power8 [6]. In particular,
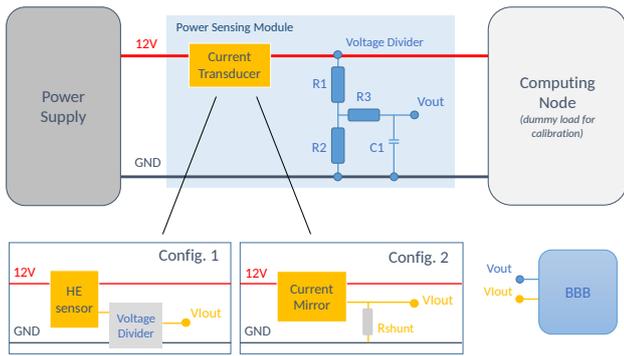
Fig. 3 High-Resolution power sensing module.

the HE sensor is the *Allegro MicroSystems ACS770* [7]. It can measure currents in the range 0–100 A with low-intrusiveness, good linearity and high precision sensitivity (40 mV/A). It has a typical bandwidth of 120 kHz and an internal conductor resistance of 100 µΩ, which implies a negligible power loss. All these features make it suitable for integration with standard PSUs used on Intel and ARM architectures.

For the configuration based on current mirror and shunt resistor, we exploited the same current mirror already used by the Baseboard Management Controller (BMC) on IBM Power8 to measure at a coarse grain the total node power consumption. This provides a twofold benefit: measuring currents in a wider range (0–250 A) and avoiding extra cost for current sensing components. Finally, to avoid a possible measurement accuracy degradation due to heating effects on the resistors of the voltage dividers, we have chosen resistors with equal Temperature Coefficient of Resistance (TCR), and placed them close to each other (similar temperature on both) and far from external sources of heating (to avoid uneven heating effects).

2.2 Embedded Monitoring and Edge Analytics

We selected a Beaglebone Black (BBB) [34] as embedded computing board as it provides several interesting features off-the-shelf: (i) it includes a 12-bit Successive Approximation Register (SAR) ADC needed for the power-sensing module, (ii) hardware support for Precision Time Protocol (PTP) which allows sub-microsecond measurements synchronization [27, 26], and (iii) an ARM Cortex-A8 processor with NEON technology, useful for DSP processing and edge ML inference (*e.g.*, by leveraging the ARM NN SDK [2], which enables efficient translation of existing neural network frameworks, such as TensorFlow, to ARM Cortex-A CPUs). Moreover, the Sitara AM335x chip used in BBB includes two programmable real-time units

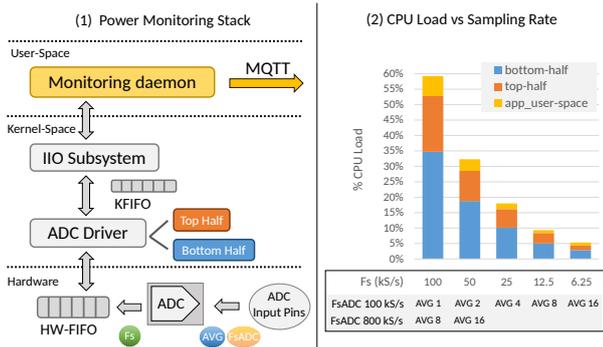(PRUs) useful for real-time acquisition and extra processing on-board.

It should be noted that standard computing servers already integrate an embedded system used for real-time monitoring and management, namely the BMC. This is usually a closed platform with no access to the firmware, but thanks to the recent OpenBMC project [31] few vendors started to release it open-source. We decided to do not use the BMC for this purpose as (i) we needed to add extra hardware to integrate an ADC and interface it with the custom power sensor (the BBB already includes an ADC), (ii) it does not provide hardware support for PTP, and (iii) it is based on an old ARM processor family (*i.e.*, ARM11 [9]) which is not a good choice for edge analytics. Moreover, (iv) it is a critical component to ensure safe working conditions of the nodes, thus it is more convenient to do not overwhelm its limited processing resources with our monitoring / edge-analytics software stack.

Figure 4 reports the embedded monitoring stack design. The bottom layer represents the ADC hardware module. We exploit the ADC *continuous sampling mode* to continuously sample the two input channels (*i.e.*, current and voltage), average and store them in a hardware FIFO that is managed by a kernel driver. By tuning the ADC sampling frequency (FsADC) and the hardware averaging (AVG), it is possible to set the frequency (Fs) at which samples enter the software layers.

When the hardware FIFO reaches a watermark on the number of samples acquired (we set it to 16), an interrupt is raised and the samples are flushed into the main memory (kernel FIFO) by the ADC driver. In particular, the ADC driver involves two routines (*IRQ handlers*): the *top half*, which monitors the watermark on the hardware FIFO, and the *bottom half* that is used to flush the samples into the kernel FIFO.

Finally, the power measurements are exposed via the Industrial I / O (IIO) Subsystem API to a user-space monitoring daemon, which is in charge of converting data from integer to Watt and associate them with a timestamp. This daemon also collects node's performance measurements from hardware performance counters via built-in tools (*e.g.*, IPMI [21], Amester [31] and RAPL [22]). In this way, we can perform edge analytics on a target use-case (*e.g.*, ML inference for anomalies detection) and send the results, together with the power and performance measurements at a lower rate, to the centralized monitoring unit for cluster-level analytics.

It should be noted that by using the continuous sampling mode, the hardware guarantees a negligible uncertainty on the acquisition time of consecutive samples, ensuring correctness of the energy computation at a fine granularity [17]. Moreover, to make negligible

Fig. 4 Embedded power monitoring stack (left) and software overhead (right).

both overhead and uncertainty introduced by the timestamp's function call, we only generate timestamps at each flush of the kernel FIFO, and not for every sample (timestamps for every sample are then derived accordingly to the number of samples acquired).

2.3 Centralized Monitoring Unit and Cluster-Level Analytics

We exploit centralized monitoring, based on the open-source ExaMon monitoring platform [3, 11], to carry out cluster-level analytics with data coming from multiple nodes. To send data from the distributed monitoring agents to the centralized monitoring unit, we adopted MQTT [19], which is a robust, lightweight and scalable protocol, already used for large-scale systems both in industry and academia (*e.g.*, Amazon, Facebook, [19, 11]). Figure 2 outlines its publish / subscribe communication model, where the publishers (running in the embedded computers) send measurements to a broker, along with a topic that corresponds to the monitored metric (*e.g.*, power consumption). The broker resides in the centralized monitoring unit together with the subscriber. The latter is used to filter and collect the data that is interested in, and expose them to a Big Data engine, namely Apache Spark [36]. The measurements are also stored in a scalable database - Apache Cassandra [1] - enabling ML analytics both in streaming and batch mode.

3 Experimental Results

This section starts by reporting the performance of the monitoring agents, along with the validation of the high-resolution power measurements from the point of view of accuracy and precision. It then presents a set of monitoring use cases based on Fourier analysis to show the capability of the high-resolution monitoring

in revealing fine grain computation activity. Finally, we report a campaign of ML inference benchmarks, based on ResNets running on the embedded monitoring platform, to show an example of the capability of DiG on carrying out edge ML analytics.

3.1 Monitoring Agents Performance

Performance Measurements: To provide completely out-of-band monitoring, we integrated our infrastructure in a SoA OpenPOWER computing cluster, namely D.A.V.I.D.E. [6, 10], that consists of 45 nodes (3 racks with 15 nodes each) based on IBM Power8. In this system we take advantage of its out-of-band telemetry and collect via Amester through the On Chip Controller (OCC) [31] 242 metrics per-component every 10 s (*e.g.*, the performance of Core, Cache, FAN, etc.), and via IPMI 89 metrics per-component every 5 s. All these measurements are sent to the centralized monitoring for cluster-level analytics, but can also be exploited on-board for real-time edge analysis on a target use-case (*e.g.*, detection of anomalies).

Power Measurements: To cover the full power consumption bandwidth at the node plug (*i.e.*, tens of microseconds, observed with a professional oscilloscope - *Keysight DS0X3054T* - attached to the power sensor) and at the same time to avoid overwhelming the CPU with the data acquisition routine, we tested several sampling rates (Fs) that we report in Figure 4.2. In particular, the best trade-off corresponds to a sampling rate of $800\,\mathrm{kS/s}$ per channel and hardware averaging every 16 samples. This is equivalent to $50\,\mathrm{kS/s}$ (*i.e.*, $20\,\mu s$) and allows to obtain several benefits: (i) to cover the entire signal bandwidth, (ii) to keep the CPU load below 35 % and (iii) to reach a precision below 1 % (σ) of uncertainty (a.k.a. oversampling and averaging method [35]). It is noteworthy that this precision makes our system suitable for the most rigorous requirement needed to benchmark an HPC system in Top500 [16].

Finally, we send the measurements to the centralized monitoring for cluster-level analytics at the rates of 1 s and 1 ms, while measurements at higher resolution can be analyzed directly at the edge.

Software overhead: The entire BBB CPU load of the monitoring software stack (power and performance) is below 46 %. In particular, performance monitoring requires roughly 11 % and the power monitoring around 35 %. We have run also some benchmarks to evaluate the computing capabilities of the embedded platform: we can perform (i) real-time Power Spectral Density (PSD) analysis of the high resolution power measurements (*e.g.*, useful for feature extraction [28]), in a time

window of around 40 ms with roughly 7 % of CPU usage, and (ii) ML inference via TensorFlow on these PSDs, implementing ResNet with 8 layers and channels $\{8, 8, 16, 32\}$ and respecting a real-time constraint of 30 ms per spectrogram (a detailed analysis can be found in Section 3.4). Moreover, it must be noted that we did not use any optimization to run ML inference (*e.g.*, ARM NN SDK [2] or TensorFlow Light [4]), which means these results can be further improved.

Synchronization: To ensure accurate and precise timestamping of the measurements, we exploit PTP hardware obtaining sub-microsecond synchronization (*i.e.*, smaller than the sampling period) across multiple nodes and embedded monitoring devices. Moreover, results in [26] show that with a proper setting of NTP, it is possible to reach synchronization with an uncertainty of a few microseconds in today's data centers and HPC clusters.

Scalability: To benefit from a scalable interface to the centralized monitoring unit, we exploit Mosquitto [11] which is a Linux implementation of MQTT that consists of a single thread process. Our tests show that Mosquitto broker (*i.e.*, the bottleneck of the network) running in an Intel Xeon E5-2600 (Haswell) can handle up to 16 publishers that send data every millisecond using just 30 % of a core, and of course, it is possible to increase the number of brokers if needed. In our current configuration of the monitoring system integrated in the D.A.V.I.D.E. HPC machine, we use one broker for all performance counters and three brokers (one per rack) for the power measurements, with no particular issue for the users / system admins of the data center since November 2017. Moreover, we tested MQTT with a similar configuration on all 516 computing nodes of GALILEO at CINECA, Italy (Intel Xeon E5-2630v3 processors), proving that this interface is suitable for even larger scale systems.

3.2 Power Measurements Validation

To verify the accuracy and precision of the high resolution power measurements, we attached the DiG power sensing module to a dummy load (depicted in Figure 3) and calibrated the current and voltage sensors against a high-precision reference multimeter. In this way, we can also determine the conversion factors (offset and gain) for both voltage and current needed for computing power consumption in Watt. Figure 5 reports the results of the current measurements after calibration for both configurations, namely HE Sensor and current mirror plus shunt resistor. In particular, the x-axis corresponds to the different input loads that we applied to the power sensing module accordingly to the allowed

current range (*i.e.*, 0–100 A for the HE Sensor and 0–200 A for the shunt resistor), while the y-axis reports the current measured by DiG (dots) and a linear regression on the measurements (straight line). As can be seen from the plot, all the DiG measurements well match the input load, so the curve is linear across the full range (*i.e.*, *coefficient of determination* $R^2_{Shunt} = 0.9999$ and $R^2_{HE} = 0.9997$), except around zero, which is anyway not a problem as compute nodes never work in this low range of currents. We carried out the same procedure for the voltage measurements and obtained similar results.
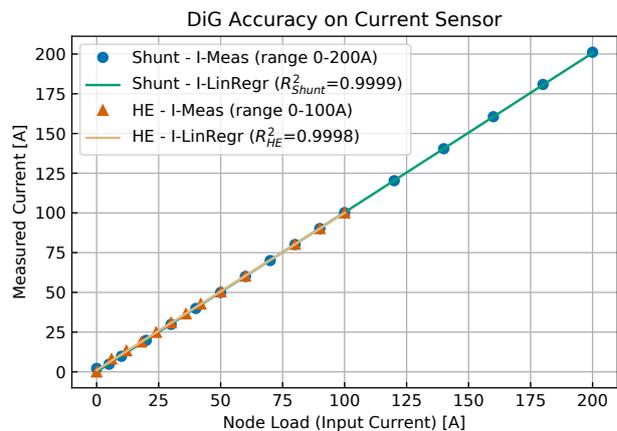

Fig. 5 Accuracy of the DiG current sensors.

After calibrating DiG, we can evaluate the precision on the power measurements (*i.e.*, standard deviation - σ - and coefficient of Variation - CV). With this goal, we can start by quantifying the precision of each ADC channel (current and voltage) independently, and use the propagation of uncertainty theorem for computing the uncertainty of the resulting power [8]. Indeed, given the measured current and voltage with uncertainties, $I \pm \sigma_I$ and $V \pm \sigma_V$ (where I and V correspond to the measured average value), under the assumption they are not correlated, the uncertainty on the power measurements is:

$$\sigma_P \approx \sqrt{I^2 \sigma_V^2 + V^2 \sigma_I^2} \qquad (1)$$

Table 1 reports the resulting precision at 50 kS/s for three different server operating conditions: idle, medium load and maximum load (*e.g.*, to give an idea in an Intel Xeon E5-2600 these conditions corresponds to roughly 180 W, 600 W and 1200 W, respectively). The precision for around 68.3 % of the power measurements (σ) is bounded between 1.73–3.96 W, for the minimum (idle) and maximum load, respectively, and increases of a factor of 3 when considering 99.7 % of the samples

Table 1 DiG Precision based on dynamic software average.

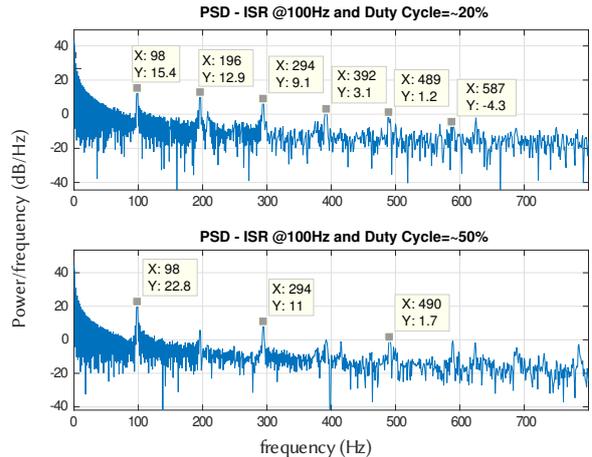|	Idle σ (CV)	Mid-Load σ (CV)	Max Load σ (CV)
50 kS/s	1.73 W (0.96%)	2.58 W (0.43%)	3.96 W (0.33%)
25 kS/s	0.5 W (0.28%)	1.26 W (0.21%)	2.28 W (0.19%)
1 kS/s	0.47 W (0.26%)	1.14 W (0.19%)	2.16 W (0.18%)
1 S/s	0.32 W (0.18%)	1.02 W (0.17%)	2.04 W (0.17%)

(5.2–11.88 W for 3σ). Of course, the CV follows the opposite trend: it decreases when the power consumption increases (from 0.96 % in idle to 0.33 % for maximum workload).

The table reports also the precision when sampling at lower rates - by applying a software average -, namely 25 kS/s, 1 kS/s and 1 S/s. As can be seen, σ drastically improves to a few watt precision already at 25 kS/s (even at the maximum load). With the goal to dynamically increase the DiG precision on the power measurements when required, the monitoring daemon can be set to dynamically switch to a lower sampling rate (*e.g.*, to 25 kS/s) by averaging in software the power samples if it is monitoring low currents for a certain time period. Thanks to this trade-off we can always keep the monitoring precision below a pre-set threshold (up to subwatt precision), which makes DiG suitable to be used in production environments as a high-precision HPC energy monitoring solution.

3.3 Feature Extraction Benchmarking

This section aims at showing the capability of the high resolution monitoring in unveiling high-frequency components directly related to the computation activity. We exploit Fourier analysis as an example of feature extraction technique for time series that is suitable for deep learning algorithms (*e.g.*, Deep Neural Networks - DNNs [28]). Future works can extend this real-time analysis targeting specific use-cases (*e.g.*, anomaly detection in workloads) and (i) exploit it, together with performance counters, as input data for DNN models running inference in the monitoring agents [28]; or (ii) just send it with lossy compression algorithms (data are sparse, as shown by the following tests) to the centralized monitoring for cluster-level analytics. We note that due to the limitations of SoA power monitoring support for computing nodes, up until now this kind of analysis could not be performed in production data centers.

We start the evaluation with a synthetic benchmark on the computing node, consisting of an Interrupt Service Routine (ISR) that we run every ~10 ms (roughly 100 Hz) with different sets of instructions. In particular,

Fig. 6 PSD of an ISR at 100 Hz with different sets of instructions.

the first set of instructions corresponds to a duty cycle of 20 % in power consumption (*i.e.*, 2 ms of workload and 8 ms of sleep), while the second set to 50 %. Figure 6 shows the PSD for the two cases, computed in a time window of 40 ms. According to Fourier analysis, the set of instructions with duty cycle 20 % (top) shows the fundamental at around 100 Hz plus all its harmonics, while the one at 50 % (bottom) correctly reports only the fundamental and the odd harmonics (even harmonics are not completely null due to the not exact 50 % duty cycle). This example shows that our monitoring can really capture spectral properties of different workloads in execution.

The second set of benchmarks, reported in Figure 7, aims at demonstrating distinctive frequency-domain signatures of real bottlenecks (*e.g.*, CPU or memory limitations) and scientific applications. Goal of this test is not to analyze in depth the reasons behind the peaks, but instead to show that different patterns emerge in the power spectrum with different workloads, which can be used as input features for DNN algorithms.

In particular, comparing the first four plots we can clearly see four different patterns (peaks highlighted with dark / light circles to indicate stronger / weaker magnitude): the first plot portrays the PSD of the computing node in idle and reveals five main peaks (dark red circles) plus other weaker peaks (light red circles) spread in the entire bandwidth 0–12 kHz (richer activity in 0–4 kHz); notice that these main peaks persist also in all other benchmarks; the second and third plot depicts respectively a memory bound and a CPU bound synthetic benchmark, where the former is bound in the SDRAM, while the latter is stuck in the CPU *'front-end'* process (*i.e.*, phase where instructions are fetched and decoded into operations - it differs from the CPU *'back-end'* process where instead the required compu-

tation is performed); these two benchmarks report a rich activity up to 6 kHz and are almost flat for frequencies above; moreover, they can be clearly distinguished from their pattern (three main peaks in the tested memory bound application vs. ten main peaks in the CPU bound). It is noteworthy that, while measuring only a power consumption coarse-grain value would not be enough to discern any difference between the two bottlenecks, DiG (w.r.t SoA monitoring systems) can detect spectral components associated to different usage of architectural resources in the two benchmarks. Finally, the fourth plot shows a real scientific application (*i.e.*, Quantum Espresso - QE [15]) and reports four main peaks more with respect to idle (dark yellow circles) and rich activity in all the spectrum.

With the next three plots in Figure 7, we demonstrate that we can appreciate the activity of short regions of code. More in detail, we report the PSD of the computing node when it is running sets of instructions at a desired frequency and duty cycle (*i.e.*, pulse train of instructions where we alternate high load computation phases with idle phases). The black circles in the plots highlight we can capture the activity of software routines running roughly every 150 μs, 110 μs and 90 μs, with a respective duration of 75 μs, 55 μs and 45 μs (*i.e.*, routines at 6.5 kHz, 9 kHz and 11 kHz with 50 % duty cycle between idle and computation).

Finally, we report in the last three plots of Figure 7 two use-cases of anomaly detection in the computing nodes. In particular, the first plot corresponds to a case of misconfiguration, where we disabled in the system the dynamic tick. This is enabled by default in Linux OS in order to potentially make the system more energy efficient (*i.e.*, the kernel can save power when idle because it does not have to wake up regularly just to service the timer tick). Comparing the PSD of the system in idle with the dynamic tick enabled (first plot) with the one without dynamic tick we can clearly see the peak at 1 kHz (frequency of the static tick) and all its harmonics (at multiples of the fundamental) till 11 kHz. We envision this kind of high resolution monitoring along with edge ML analytics to help on catching anomalies in next generation of data centers.

Another interesting scenario for detection of anomalies is related to the detection of cyber-attacks. Network security is a crucial challenge in data centers and cloud infrastructures, to prevent attackers from getting access into the system and steal sensitive data or illegally use computing resources [32,12]. Before intruding into the system attackers need to gather information about the target machine and its running services, and thus about vulnerabilities that can be exploited. This is called scan-
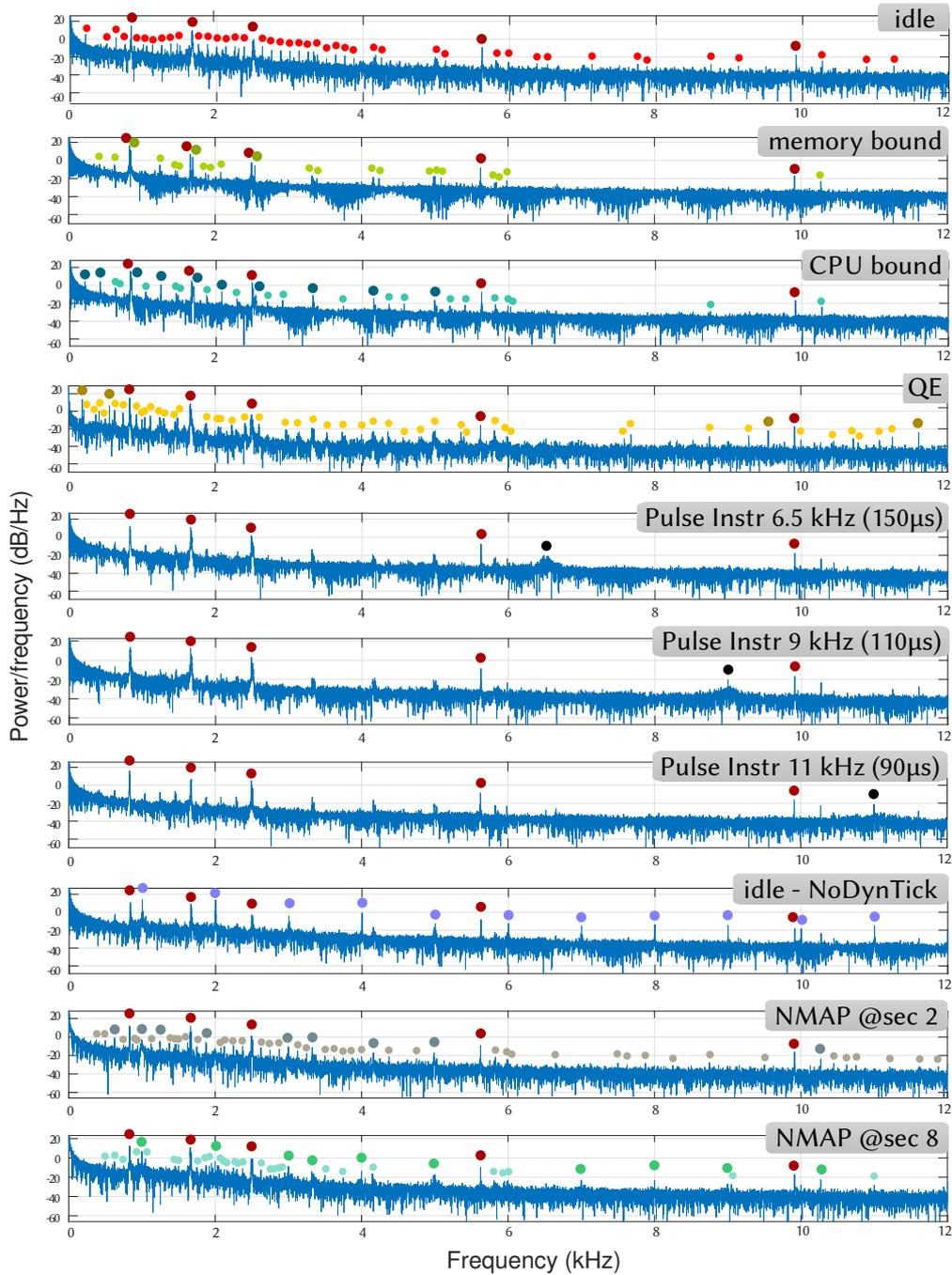
ning phase, and one of the most popular tools for port scanning is *NMAP* [30].

The use case scenario is an attacker that tries to collect information about the front-end node of data centers / clouds to get access into the local network. Thus we run NMAP from a remote computer (outside the local network) with the OS detection mode enabled, which means we want to understand open ports, running services and OS of the front-end node. The scanning attack requires around 10 seconds and the last two plots of Figure 7 show the PSD of the DiG power measurements when monitoring the front-end node. With the goal to show that different phases of NMAP correspond to different patterns of PSD, which are also different from the PSD of the system in idle, we report the attacked node at second 2 and second 8, on the first and second plot, respectively. As can be seen, results show 2 different patterns: the plot at second 2, with regards to plot at second 8, reports main peaks only up to 5 kHz and is almost flat for the frequencies above; instead, the plot at second 8 reports main peaks spread in the entire bandwidth (*i.e.*, 1, 2, 3, 3.5, 4, 7, 8, 9, 10.5 kHz). This pattern recognition analysis, based on high resolution power measurements, can be used in future works to exploit ML classifiers running on the edge and correlate this information with SoA signature-based IDS (*e.g.*, SNORT) to help on preventing from intrusions.

3.4 Edge ML Benchmarking

In this section we show the performance of ResNets [18] running on the embedded monitoring platform with different layers and sizes on a dataset of pre-computed PSDs of 2048 points in frequency each (*i.e.*, 4096 B per spectrogram, which correspond to a time window of 40 ms), to show an example of the capability of DiG on carrying out edge ML inference. In particular, we use TensorFlow inference compiled for ARM architecture in order to exploit the NEON SIMD accelerator.

Figure 8 reports the results of the benchmarks, where the x-axis corresponds to the chosen batch size (*i.e.*, number of test images per iteration) and the y-axis to the processing time per image (milliseconds). Results show that the best trade-off is between batch sizes 3 and 5, which can give an improvement up to around 10 ms w.r.t. using the same ResNet with no batch mode enabled. For bigger batch sizes the improvement is negligible (in the best case up to 1 ms). Moreover, as a matter of comparison we have run a ResNet with 8 layers and channels {8, 8, 16, 32} both exploiting the NEON accelerator (gray triangles) and without using it (purple triangles), and results show a 3.8× improvement when using NEON, which is a relevant processing time

Fig. 7 Example of PSD patterns of real bottlenecks and applications.

when handling high resolution measurements and live analysis.

Finally, considering a time constraint of 40 ms for running real-time PSDs on the edge (4096 B per spectrogram), results suggest that ResNet with 8 layers and channels $\{8, 8, 16, 32\}$ (and below), but also 14 layers and channels $\{4, 4, 8, 16\}$ (and below), are suitable for running on high resolution power measurements, while larger ResNet size and layers can be used together with

performance measurements acquired at a lower rate. It should be noted that in our benchmarks we did not use any optimized ML framework for our embedded platform, such as ARM NN SDK [2] or TensorFlow Light [4], which we believe would further improve these results.
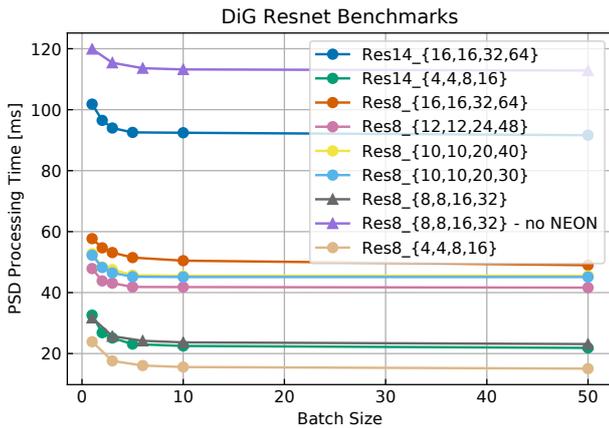
Fig. 8 ResNet benchmarking with TensorFlow on BBB.

4 Related Work

Existing off-the-shelf methods to measure power and performance of computing nodes in data centers rely on in-band or out-of-band telemetry depending on the technology vendors. In particular, an example of in-band solution is Intel RAPL [22], while examples of out-of-band solutions are IBM Amester [31] and the two standards IPMI [21] and Redfish [23] (*i.e.*, new protocol for managing data centers hardware, that fixes the security vulnerabilities of IPMI [31]). All these built-in tools allow a fine grain per-component monitoring (*i.e.*, based on hardware performance counters and up to 1 s for IPMI, 1 ms for RAPL and 250 μs for Amester [22, 31,17]), but not high-resolution power monitoring (*i.e.*, covering the entire signal bandwidth - tens of microseconds).

Pushed by the growing interest in fine-grained power monitoring, industry and academia researchers are providing custom solutions for data centers. Examples are HDEEM [17], PowerInsight [24] and HAEC [20]. The first two systems provide power consumption measurements up to millisecond time resolution, while the last one has a much more fine grain insight, with a sampling rate up to 500 kS/s. All these custom solutions focus on only monitoring the power consumption (*i.e.*, no performance knowledge) and send all the measurements to a centralized monitoring unit for analysis. Thanks to them, new opportunities for research on energy efficiency and other challenges are now possible, but going toward high resolution measurements this kind of monitoring design entails scalability issues (*e.g.*, as discussed in [20], HAEC is suitable for high resolution monitoring in just a node, but not for an entire cluster).

Comparison with SoA: In our system (*i.e.*, DiG) we combined all these features to enable research on several challenges for analytics, automation and control of data centers, with a highly-flexible monitoring platform: (i) we work completely out-of-band (*i.e.*, no impact / perturbation on the computing resources); (ii) we collect all performance counters and (iii) the full power bandwidth at the plug (*i.e.*, sampling at 50 kS/s) (iv) with high precision (*i.e.*, below 1 % - σ); (v) we provide highly synchronized measurements (*i.e.*, sub-microsecond) for a detailed correlation of the activities within the cluster; (vi) we leverage the monitoring between edge and a centralized unit, by exploiting dedicated embedded computers to collect measurements (they have complete knowledge of the status of their node) and have the possibility to carry out both edge and cluster-level analytics; (vii) the system is scalable (thanks to our flexible design, based on edge monitoring agents and a robust and scalable protocol - MQTT - to the centralized monitoring, where we analyze data at a lower rate), (viii) technology agnostic (*i.e.*, tested on ARM, Intel and IBM) and (ix) low cost (*i.e.*, no motherboard redesign required).

5 Conclusion

This work reports on the design of a novel monitoring infrastructure - namely DiG - that enables real-time high-resolution profiling and analytics of data centers, for their automation and control. Main design choices include complete out-of-band monitoring of power and performance, with a dedicated embedded computer per node to perform edge analysis, and a custom power sensor at the plug for high-resolution and high-precision measurements. We report (i) architecture design choices of both hardware and software, (ii) monitoring platform performance, (iii) an extensive set of tests based on Fourier analysis to show the high resolution monitoring insights and (vi) a campaign of benchmarks of ML inference, running on the embedded computer, to provide an overview of the real-time edge analytics capabilities of DiG.

References

1. Apache cassandra. URL `http://cassandra.apache.org/`
2. Arm NN SDK. URL `https://developer.arm.com/products/processors/machine-learning/arm-nn`
3. Examon HPC Monitoring. URL `https://github.com/EEESlab/examon`

4. TensorFlow Lite. URL https://www.tensorflow.org/mobile/tflite/

5. Ahmad, S., Lavin, A., Purdy, S., Agha, Z.: Unsupervised real-time anomaly detection for streaming data. Neurocomputing **262**, 134 – 147 (2017). DOI https://doi.org/10.1016/j.neucom.2017.04.070

6. Ahmad, W.A., Bartolini, A., Beneventi, F., Benini, L., Borghesi, A., Cicala, M., Forestieri, P., Gianfreda, C., Gregori, D., Libri, A., Spiga, F., Tinti, S.: Design of an energy aware petaflops class high performance cluster based on power architecture. In: 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 964–973 (2017). DOI 10.1109/IPDPSW.2017.22

7. Allegro MicroSystems: Thermally Enhanced, Fully Integrated, Hall Effect-Based High Precision Linear Current Sensor IC with 100 μΩ Current Conductor. ACS770xCB Datasheet Rev. 4, 2015

8. Arras, K.: Technical Report EPFL-ASL-TR-98-01 R3 (1998). URL https://infoscience.epfl.ch/record/97374/files/TR-98-01R3.pdf

9. ASPEED: AST2500 Advanced PCIe Graphics & Remote Management Processor. AST2500 Datasheet

10. Bartolini, A., Borghesi, A., Libri, A., Beneventi, F., Gregori, D., Tinti, S., Gianfreda, C., Altoè, P.: The D.A.V.I.D.E. big-data-powered fine-grain power and performance monitoring support. In: Proceedings of the 15th ACM International Conference on Computing Frontiers, CF '18, pp. 303–308. ACM, New York, NY, USA (2018). DOI 10.1145/3203217.3205863. URL http://doi.acm.org/10.1145/3203217.3205863

11. Beneventi, F., Bartolini, A., Cavazzoni, C., Benini, L.: Continuous learning of hpc infrastructure models using big data analytics and in-memory processing tools. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2017, pp. 1038–1043 (2017). DOI 10.23919/DATE.2017.7927143

12. Defense, R.C.T.: Cloud Security Trends, +17 Tips to Fortify Your Public Cloud Computing Environment (2017). https://veristor.com/wp-content/uploads/2017/11/RedLock_CloudSecurityTrends_Veristor.pdf

13. Duplyakin, D., Brown, J., Ricci, R.: Active learning in performance analysis. In: 2016 IEEE International Conference on Cluster Computing (CLUSTER), pp. 182–191 (2016). DOI 10.1109/CLUSTER.2016.63

14. Fu, H., Liao, J., Yang, J., Wang, L., Song, Z., Huang, X., Yang, C., Xue, W., Liu, F., Qiao, F., Zhao, W., Yin, X., Hou, C., Zhang, C., Ge, W., Zhang, J., Wang, Y., Zhou, C., Yang, G.: The sunway taihulight supercomputer: system and applications. Science China Information Sciences **59**(7), 072001 (2016). DOI 10.1007/s11432-016-5588-7

15. Giannozzi, P., Baroni, S., Bonini, N., Calandra, M., Car, R., Cavazzoni, C., Ceresoli, D., Chiarotti, G.L., Cococcioni, M., Dabo, I., Corso, A.D., de Gironcoli, S., Fabris, S., Fratesi, G., Gebauer, R., Gerstmann, U., Gougoussis, C., Kokalj, A., Lazzeri, M., Martin-Samos, L., Marzari, N., Mauri, F., Mazzarello, R., Paolini, S., Pasquarello, A., Paulatto, L., Sbraccia, C., Scandolo, S., Sclauzero, G., Seitsonen, A.P., Smogunov, A., Umari, P., Wentzcovitch, R.M.: Quantum espresso: a modular and open-source software project for quantum simulations of materials. Journal of Physics: Condensed Matter **21**(39), 395502 (2009)

16. Group, E.W.: Energy efficient high performance computing power measurement methodology (v.2.0 RC 1.0) (2017). https://eehpcwg.llnl.gov/assets/sc17_bof_methodology_2_0rc1.pdf

17. Hackenberg, D., Ilsche, T., Schuchart, J., Schöne, R., Nagel, W.E., Simon, M., Georgiou, Y.: Hdeem: High definition energy efficiency monitoring. In: Energy Efficient Supercomputing Workshop (E2SC), 2014, pp. 1–10 (2014). DOI 10.1109/E2SC.2014.13

18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)

19. Hunkeler, U., Truong, H.L., Stanford-Clark, A.: Mqtt-s – a publish/subscribe protocol for wireless sensor networks. In: Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on, pp. 791–798 (2008). DOI 10.1109/COMSWA.2008.4554519

20. Ilsche, T., Schöne, R., Schuchart, J., Hackenberg, D., Simon, M., Georgiou, Y., Nagel, W.E.: Power measurement techniques for energy-efficient computing: reconciling scalability, resolution, and accuracy. Computer Science - Research and Development (2018). DOI 10.1007/s00450-018-0392-9

21. Intel, Hewlett-Packard, NEC, Dell, Rep., T.: IPMI Specification, V2.0, Rev. 1.1 (2013)

22. Khan, K.N., Hirki, M., Niemi, T., Nurminen, J.K., Ou, Z.: Rapl in action: Experiences in using rapl for power measurements. ACM Trans. Model. Perform. Eval. Comput. Syst. **3**(2), 9:1–9:26 (2018). DOI 10.1145/3177754

23. Kumari, P., Saleem, F., Sill, A., Chen, Y.: Validation of redfish: The scalable platform management standard. In: Companion Proceedings of the10th International Conference on Utility and Cloud Computing, UCC '17 Companion, pp. 113–117. ACM, New York, NY, USA (2017). DOI 10.1145/3147234.3148136

24. Laros, J.H., Pokorny, P., DeBonis, D.: Powerinsight - a commodity power measurement capability. In: Green Computing Conference (IGCC), 2013 International, pp. 1–6 (2013). DOI 10.1109/IGCC.2013.6604485

25. Libri, A., Bartolini, A., Benini, L.: 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 (2018)

26. Libri, A., Bartolini, A., Cesarini, D., Benini, L.: Evaluation of ntp/ptp fine-grain synchronization performance in hpc clusters. In: 2nd Workshop on AutotuniNg and aDaptivity AppRoaches for Energy efficient HPC Systems (ANDARE 2018) (2018)

27. Libri, A., Bartolini, A., Magno, M., Benini, L.: Evaluation of synchronization protocols for fine-grain hpc sensor data time-stamping and collection. In: 2016 International Conference on High Performance Computing Simulation (HPCS), pp. 818–825 (2016). DOI 10.1109/HPCSim.2016.7568419

28. Lin, S., Liu, N., Nazemi, M., Li, H., Ding, C., Wang, Y., Pedram, M.: Fft-based deep learning deployment in embedded systems. In: 2018 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 1045–1050 (2018). DOI 10.23919/DATE.2018.8342166

29. Liu, Z., Kettimuthu, R., Foster, I., Beckman, P.H.: Toward a smart data transfer node. Future Generation Computer Systems **89**, 10 – 18 (2018). DOI https://doi.org/10.1016/j.future.2018.06.033

30. Lyon, G.F.: Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure, USA (2009)

31. Rosedahl, T., Broyles, M., Lefurgy, C., Christensen, B., Feng, W.: Power/Performance Controlling Techniques in OpenPOWER. In: J.M. Kunkel, R. Yokota, M. Taufer, J. Shalf (eds.) High Performance Computing, pp. 275–289. Springer International Publishing, Cham (2017)

32. Tahir, R., Huzaifa, M., Das, A., Ahmad, M., Gunter, C., Zaffar, F., Caesar, M., Borisov, N.: Mining on someone else's dime: Mitigating covert mining operations in clouds and enterprises. In: M. Dacier, M. Bailey, M. Polychronakis, M. Antonakakis (eds.) Research in Attacks, Intrusions, and Defenses, pp. 287–310. Springer International Publishing, Cham (2017)

33. Tang, A., Sethumadhavan, S., Stolfo, S.J.: Unsupervised anomaly-based malware detection using hardware features. In: A. Stavrou, H. Bos, G. Portokalidis (eds.) Research in Attacks, Intrusions and Defenses, pp. 109–129. Springer International Publishing, Cham (2014)

34. Texas Instruments: BeagleBone Black System Reference Manual. Rev. C.1, 2014

35. Villa-Angulo, C., Hernandez-Fuentes, I.O., Villa-Angulo, R., Donkor, E.: Bit-resolution improvement of an optically sampled time-interleaved analog-to-digital converter based on data averaging. IEEE Transactions on Instrumentation and Measurement 61(4), 1099–1104 (2012). DOI 10.1109/TIM.2011.2179335

36. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I.: Apache spark: A unified engine for big data processing. Commun. ACM 59(11), 56–65 (2016). DOI 10.1145/2934664