Paper

Server Workload Model Identification: Monitoring and Control Tools for Linux

Michał Karpowicz and Piotr Arabas

Institute of Control and Computation Engineering, Warsaw University of Technology, Warsaw, Poland
Research and Academic Computer Network (NASK), Warsaw, Poland

Abstract — Server power control in data centers is a coordinated process carefully designed to reach multiple data center management objectives. The main objectives include avoiding power capacity overloads and system overheating, as well as fulfilling service-level agreements (SLAs). In addition to the primary goals, server control process aims to maximize various energy efficiency metrics subject to reliability constraints. Monitoring of data center performance is fundamental for its efficient management. In order to keep track of how well the computing tasks are processed, cluster control systems need to collect accurate measurements of activities of cluster components. This paper presents a brief overview of performance and power consumption monitoring tools available in the Linux systems.

Keywords* — *cloud computing, energy efficiency, Linux, server performance metering.

1. Introduction

Data centers supporting both cloud services and high performance computing (HPC) applications consume enormous amounts of electrical energy. From 2005 to 2010 the energy consumed by data centers worldwide rose by 56%, which was accounted to be between 1.1% and 1.5% of the total electricity use in 2010. The growth of energy consumption rises operating costs of data centers but also contributes to carbon dioxide (CO_2) production. According to the analysis of current trends (gesi.org/SMARTer2020), the carbon dioxide emissions of the ICT industry are expected to exceed 2% of the global emissions, a level equivalent to the contribution of the aviation [1]. Energy usage in data centers grows rapidly with the climbing demand for cloud and HPC services. However, the growth rate of ICT cannot be sustained unless the power consumption problem is addressed [2]–[4]. In response to the created momentum new computing elements, i.e. CPUs/GPUs, memory units, disks, network interface cards (NICs), have been designed to operate in multiple (performance and idle) modes of differentiated energy-consumption levels (ACPI).

Although energy efficiency (FLOPS/watt) of ICT systems continues to improve, the rate of improvement does not match the growth rate of demand for computing capacity. Based on the projections of technology development it has been argued that continued scaling of available systems will eventually lead to a data center architecture consuming more than a gigawatt of electrical power (at exaflop level), a level that violates economic rationale for providing cloud or HPC services. Unless radically new energy-aware technologies are introduced, both in hardware and software domain, it will not be possible to meet DARPA's 20-megawatt exaflop goal (50 GFLOPS/watt) by year 2020 [3]. Limiting power consumption and related thermal emission has therefore become a key engineering problem.

In order to meet the challenging goals of cloud and high performance computing, advances in hardware layer development require immediate improvements in the design of cluster control software. In Section 2 a general structure and components of a data center monitoring and control system are presented. A brief description of resource allocation and performance control process is also given. In particular, the role of energy-efficient device controllers is indicated. Section 3 presents currently developed concepts of power measurement and control programming interfaces, both for data centers and wired IP networks. Basic performance metrics and benchmarking strategies are presented in Sections 4 and 5. Finally, in Section 6 the results of simple experiments are given to illustrate the discussed profiling and metering techniques.

2. An Overview of Data Center Management

Monitoring of cluster performance is fundamental for its efficient management. In order to keep track of how well the computing tasks are processed, cluster control systems need to collect accurate measurements of activities of cluster components. The collected measurements, including both data processing and power consumption metrics, provide feedback for management operations and serve as a basis for the design of new cluster control systems.

Figure 1 presents an overview of a cluster control system architecture. The racks, supplied with electric power by power distribution units (PDUs), are filled with blade servers. The racks are connected into a data center network with a hierarchy of switches (SW). The management layer is responsible for allocation of resources, job submission, adjustments of the interconnect settings, power budgeting and system monitoring. These tasks are executed by dedicated resource allocation and job management systems (RJMS) and system-wide energy management systems (SEM). The lower control layer, composed of operating sys-

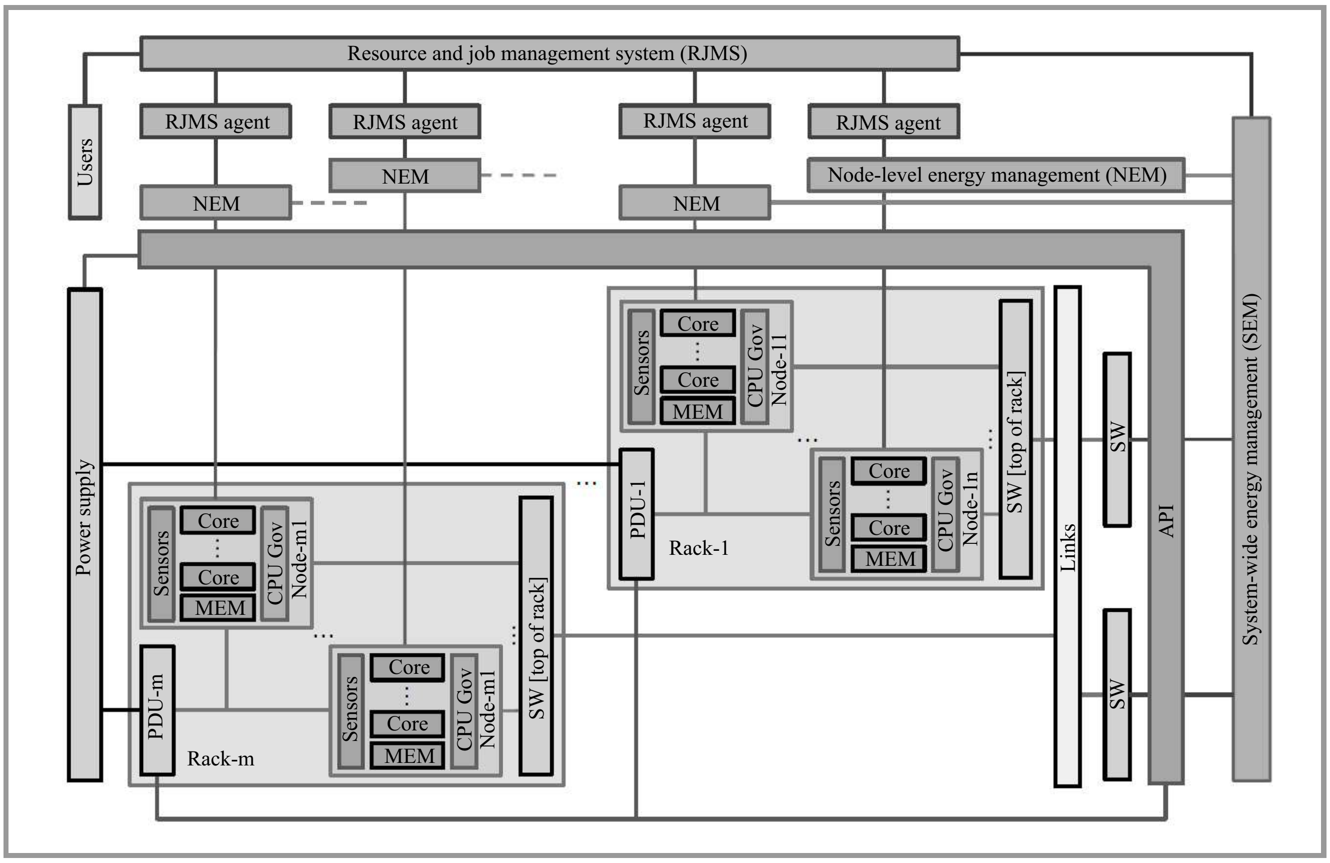

Fig. 1. An overview of cluster control system architecture.

tems controlling servers, is responsible for job execution and enforcement of resource usage constraints in computing nodes [5]–[7]. Available computing resources, including CPUs, memory and sockets, are exposed to the management systems via customized Abstract Programming Interfaces (APIs).

The challenging problem of data center control is to maximize the utilization of the resources subject to energy consumption and quality of service constraints. To solve this problem hardware-specific power saving capabilities of computing elements are used to keep the total power consumption of data center within the required power range. The resources that remain idle can be switched to a power saving (or sleeping) mode for a configurable time and restored to operating mode on demand. The resources performing operations can reduce their power consumption by dynamically adjusting their performance level [8]–[11].

3. Power Control Programming Interfaces

Power management capabilities of hardware layer are exposed in the form of Application Programming Interfaces (APIs). The foundations of power control APIs were built by the Advanced Configuration and Power Interface (ACPI) specification [12]. The specification defines hardware dependent energy saving (idle) and performance (active) states that can be adjusted on demand from the software level. This allows to control power saving and data processing efficiency according to a designed policy.

An attempt to design a vendor-neutral API dedicated for power measurement and control in HPC systems resulted in development of Power API specification [13], [14]. The Power API describes cluster as a collection of objects forming a discoverable hierarchy. Objects in the system are characterized by a set of attributes, which allow for measurement (reading attributes) and control (overwriting attributes) of their power saving capabilities. Functions providing gathering of statistics are provided for objects and groups of objects. Both ACPI and PAPI can be adapted to the Power API abstract model.

A similar approach is proposed by the ETSI Green Abstraction Layer (GAL) standard [15]. It describes a general concept of programming interface for energy state configuration of energy-aware telecommunication fixed network nodes. A hierarchical representation of a network device is proposed, which allows to control the available energy-aware states of its internal components. The innovation is not only in the described unification of control but also in the ability to query energy-aware capabilities of the components.

4. Performance Metrics and Benchmarks

Energy consumption management is a multi-objective optimization problem in which multiple performance and energy related metrics are considered [16]–[19]. Usually at least the following two objectives are considered:

- minimization of peak power consumption,
- maximization of energy-efficiency.

Limiting peak power consumption is critical to maintaining reliability of data center, avoiding power capacity overloads and system overheating, as well as fulfilling service-level agreements (SLAs). At the same time, since economically feasible power consumption levels are strongly correlated with the costs of electricity and power provisioning, it is important to maximize efficiency of operations performed in data center [2], [3], [20].
Energy-efficiency is defined as a number of operations performed per energy unit, i.e.:

$$\text{Energy efficiency} = \frac{\text{Computing performance}}{\text{Total energy consumed}}. \quad (1)$$

This universal metric has been in the center of research focused on energy-aware control of data processing systems. In order for the metric to be improved it is necessary to increase the number of operations performed per unit of energy consumed or to decrease the amount of energy required per operation. Based on the above observations various strategies of power management have been developed. Consequently, the metric has also been used in many benchmarking methodologies.
Basic industry-standard methodology for power and performance benchmarking of a computing server is SPECpower [21]–[23]. The benchmark measures power consumption of a server running an appropriately designed application (Java application server) at workload ranging from 10 to 100% of peak achievable level. Namely, a steady flow of work requests is submitted to the server under test to determine the number of requests that can be satisfied in a given time. The benchmark drivers request work at intermediate points between zero and the maximum throughput value. The related toolset can be used with other cluster-wide benchmarks.
Energy efficiency has been used as a default benchmarking metric. In the case of HPC systems (or batch processing systems) the performance metric is typically defined by the number of GFLOPS performed on average per watt while executing a selected benchmark [24], [25]. Transaction processing systems, composed of application and Web servers, as well as networks of routers have been evaluated in terms of served requests per watt during throughput-based benchmarks [26]–[28]. Dedicated tests reporting transaction throughput per watt have been developed for storage systems [29], [30] as well. Finally, from the perspective of data center management energy efficiency is viewed as a product of [17]:

- facility efficiency – the ratio of total amount of energy used by a data center facility to the energy delivered to computing equipment (PUE),
- server power conversion efficiency – the ratio of total server input power to its useful power consumed by the electronic components directly involved in the computation (SPUE),
- server's architectural efficiency – the ratio of computing performance metric to total amount of energy used by electronic components.

5. Power Monitoring and Profiling

Monitoring of power and energy consumption in computing clusters is a complex problem [7], [31]–[33]. Two general approaches can be distinguished that allow to perform the required measurements. The first one is based on power metering devices connected to the servers. Basic system-wide measurements are usually provided at rate ranging from 0.01 to several samples per second by power supply units (PSUs) and power distribution units (PDUs) through the Intelligent Platform Management Interface (IPMI) [34]. More accurate and detailed measurements, collected at high sampling rate and covering selected components of servers, may be provided by additional and dedicated metering devices [35], [36].
Whenever IPMI-based monitoring systems directly communicate with the Baseboard Management Controllers (BMC) or metering devices, there is no direct overhead on the observed servers caused by the measurements. Otherwise, perturbations of measurements should be expected. In practice IPMI is often used with server management software running under the local operating system. This allows to access hardware-specific function, exposed by available APIs, and conveniently deal with local measurements, control commands execution, error handling and alerting.
Monitoring, configuration and control of devices that support IPMI on Linux systems can be performed with `ipmitool` utility. A list of sensors visible in the system and their records can be viewed with commands:

```
$ ipmitool sensor
$ ipmitool sdr -v
$ ipmitool sdr elist full
```

Example below presents an outcome of IPMI-based monitoring:

```
# ipmitool sdr elist full
Fan1 RPM         | 30h | ok  |  7.1 | 3840 RPM
Fan2 RPM         | 31h | ok  |  7.1 | 3840 RPM
Fan3 RPM         | 32h | ok  |  7.1 | 3960 RPM
Fan4 RPM         | 33h | ok  |  7.1 | 3960 RPM
Fan5 RPM         | 34h | ok  |  7.1 | 3960 RPM
Fan6 RPM         | 35h | ok  |  7.1 | 3840 RPM
Inlet Temp       | 04h | ok  |  7.1 | 17 degrees C
Exhaust Temp     | 01h | ok  |  7.1 | 28 degrees C
Temp             | 0Eh | ns  |  3.1 | Disabled
Temp             | 0Fh | ns  |  3.2 | Disabled
Current 1        | 6Ah | ok  | 10.1 | 0.60 Amps
Current 2        | 6Bh | ok  | 10.2 | 0 Amps
Voltage 1        | 6Ch | ok  | 10.1 | 230 Volts
Voltage 2        | 6Dh | ok  | 10.2 | 240 Volts
Pwr Consumption  | 77h | ok  |  7.1 | 140 Watts
```

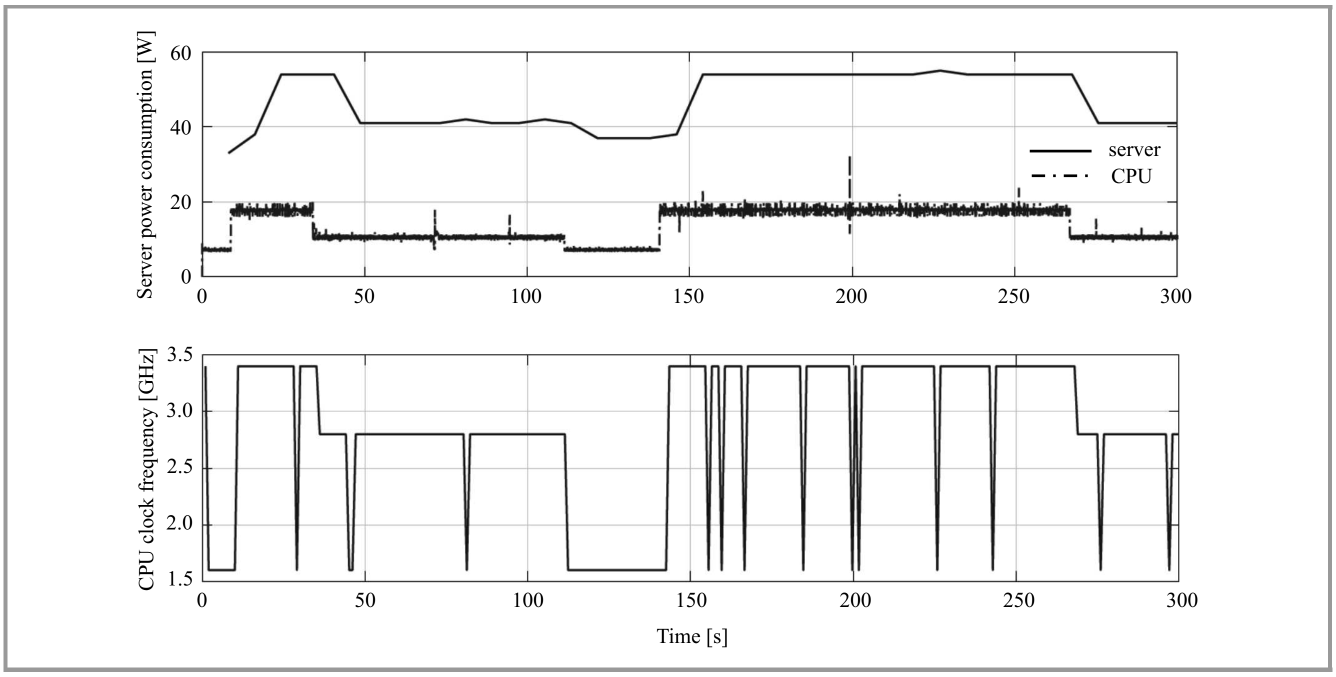

Fig. 2. Correlation of server/CPU power consumption (above) and CPU clock frequency.

The second approach to monitoring of power consumption exploits hardware-level counters provided by selected computing elements, including CPU, GPU and DRAM. The most commonly available counters are exposed through Intel's Running Average Power Limiting (RAPL) interface and NVIDIA's Management Library (NVML) functions [37], [38]. Performance counters allow to collect measurements at rate ranging from 100 to 1000 samples per second with high accuracy [39].

When used together with benchmarking probes of the operating system kernel a runtime estimation of power consumption and performance can be realized on a per-application basis. In the Linux system the required instrumentation is provided by the Performance Application Programming Interface (PAPI) and `perf_events` functions allowing to observe micro-architectural events (such as instructions completed per cycle and cache-misses) [27], [28], [40]–[43]. This paves the way for high resolution identification of data processing dynamics and its energy-efficiency, and potentially for the design of energy-aware application-specific server controllers. Novel auto-tuning systems are also developed that allow to introduce server energy control instructions into application source code [44].

It is important to point out that care must be taken when MSR-based measurements are used for performance benchmarking. Since readouts are taken on the system under test the measurements may be significantly perturbed by the measurement process itself, especially under high sampling rate. Figure 2 shows the results of measurements taken from `ipmi` system and MSRs of CPU.

Both methods of monitoring, briefly discussed above, are conveniently integrated by the resource allocation and job scheduling systems [7]. As a result it is not only possible to perform energy accounting and power profiling per job but also to setup system power-saving configuration for the purpose of job execution. Along with the scheduled batch of jobs appropriately defined control server control policies can be submitted to the computing nodes, thereby optimizing energy efficiency [45].

6. An Experimental Illustration

To demonstrate how benchmarking tools can be combined with performance and power consumption monitoring APIs several simple experiments are presented below.

Listing 1 illustrates how an average power consumption of CPU and DRAM can be calculated in Linux systems. Listing 2 illustrates how the desired RAPL MSRs (`address` variable) can be accessed from application level[1].

In order to get energy consumption information from the Intel's RAPL model specific registers (MSRs)[2] it is necessary to multiply increments of appropriate energy status counters, stored in `MSR_*_ENERGY_STATUS` registers, by scaled energy status unit, stored in `MSR_RAPL_POWER_UNIT` register. Energy status MSRs are updated approximately every 1 ms, with wraparound time of around 60 s when power consumption is high [37], [39].

The following listing shows how Linux `stress` microbenchmark can be analyzed with `perf` tool based on `perf_events` subsystem of the Linux kernel:

[1] Appropriate permissions should be setup to access `/dev/cpu/*/msr` interface.

[2] Intel's Sandy Bridge processors.

```
# perf stat -a\
  stress --cpu 32 --io 32 --vm 32 \
  --vm-bytes 512M --hdd 10 --timeout 30s

task-clock (msec) #   32.006 CPUs utilized
context-switches  #    0.122 K/sec
cpu-migrations    #    0.004 K/sec
page-faults       #    0.125 M/sec
cycles            #    2.154 GHz
instructions      #    0.75 ins per cycle
                  #    0.89 stalled cycles/ins
branches          #  430.045 M/sec
branch-misses     #    0.20% of all branches
```

Finally, in order to retrieve detailed measurements of performance and power consumption of selected parts of application source code, low-level performance and energy counters exposed via PAPI and RAPL MSRs can be used. For illustrative purposes the Linux `stress` micro-benchmark was appropriately adapted. The list of counters introduced into the source code of the benchmark included:

- total number of CPU cycles,
- reference clock cycles,
- number of completed instructions (INS),
- Level 1 instruction cache misses (ICM),
- Level 1 data cache misses (DCM),
- RAPL MSRs counting power consumption of CPU core and DRAM (W).

Listing 1: Simple power consumption monitoring script

```
#!/bin/bash

SAMPLING_RATE=1                  # seconds
MSR_PKG_ENERGY_STATUS="0x611"  # CPU energy
    counter
MSR_DRAM_ENERGY_STATUS="0x619" # DRAM energy
    counter

# Energy Status Units (ESU)
ESU=`echo "ibase=16;\
␣␣␣␣␣1/2^$(rdmsr␣-X␣0x606␣-f␣12:8)" | bc -l`
# Calculate number of CPU energy status
# counter incremants during sampling period
ESPKG=`a=$(rdmsr -X $MSR_PKG_ENERGY_STATUS);\
       sleep $SAMPLING_RATE; echo "ibase=16;\
␣␣␣␣␣␣␣$(rdmsr␣-X␣$MSR_PKG_ENERGY_STATUS)-$a"|bc`
# Calculate DRAM energy status
# counter incremants during sampling period
ESDRAM=`a=$(rdmsr -X $MSR_DRAM_ENERGY_STATUS);\
        sleep $SAMPLING_RATE; echo "ibase=16;\
␣␣␣␣␣␣␣␣$(rdmsr␣-X␣$MSR_DRAM_ENERGY_STATUS)-$a"|
    bc`
# Calculate power consumption [W]
CPUPOW=`echo "$ESPKG␣*␣$ESU" | bc -l`
DRAMPOW=`echo "$ESDRAM␣*␣$ESU" | bc -l`
echo CPU: $CPUPOW W
echo DRAM: $DRAMPOW W
```

Listing 2: Example of RAPL MSR read function

```
int read_msr(int cpu, unsigned int address,
    uint64_t *value)
{
  int   err = 0;
  char  msr_path[32];
  FILE *fp;

  sprintf(msr_path, "/dev/cpu/%d/msr", cpu);
  err = ((fp = fopen(msr_path, "r")) == NULL);
  if (!err) err = (fseek(fp, address, SEEK_CUR)
       != 0);
  if (!err) err = (fread(value, sizeof(uint64_t)
      , 1, fp) != 1);
  if (fp != NULL) fclose(fp);
  return err;
}
```

Figures 3 and 4 present the results of experiments in which the server under test[3] was forced to execute CPU and memory intensive benchmarking loops consisting of randomly generated number of iterations. In addition, the experiments were conducted for two different CPU frequency scaling governors, `intel_pstate powersave` and `intel_pstate performance` [46]. The results show a relation between the CPU frequency, power consumption, number of completed instructions and L1 cache misses. It can be seen that energy-efficiency of instructions is high when the CPU frequency is reduced. The same pattern can be observed with L1 cache misses, i.e. probability of a cache miss rises with the number CPU cycles performed. Integration of collected data allows to retrieve aggregated results of experiments.

Based on the obtained results interesting observations can be made regarding operations performed by the operating system. In particular, it is possible to study efficiency of CPU frequency control policy implementation. In order to increase the number of computing operations performed per watt, thereby maximizing Eq. (1), it is necessary to reduce the amount of time the processor spends running idle loops or stall cycles [16]. Therefore, energy-efficiency maximizing CPU controllers should implement a workload following policy dynamically adjusting CPU performance state (ACPI P-state) to the observed short-term CPU utilization or application-related latency metrics. This control concept is indeed implemented in the currently distributed CPU frequency governors of the Linux kernel, namely `intel_pstate` and `cpufreq_ondemand`.

Given a CPU workload estimate the `intel_pstate` governor, used in the presented experiments, applies PID control rule to keep the workload at the default reference level of 97%. In comparison, the `ondemand` governor calculates CPU frequency according to the following policy. If the observed CPU workload is higher than the upper-threshold value then the operating frequency is increased to the maximal one. If the observed workload is below the lower-threshold value, then the frequency is set to the lowest level at which the observed workload can be supported. For a discussion of optimal CPU frequency control policy design problem see e.g. [45], [47]. Identification of server data processing dynamics is discussed e.g. in [27], [28].

[3] DELL PowerEdge R720, 2 $\times$ Intel Xeon E5-2670 2.60 GHz, 20 MB cache, 12 $\times$ 16 GB RDIMM 1600 MHz, Linux kernel ver. 4.4.

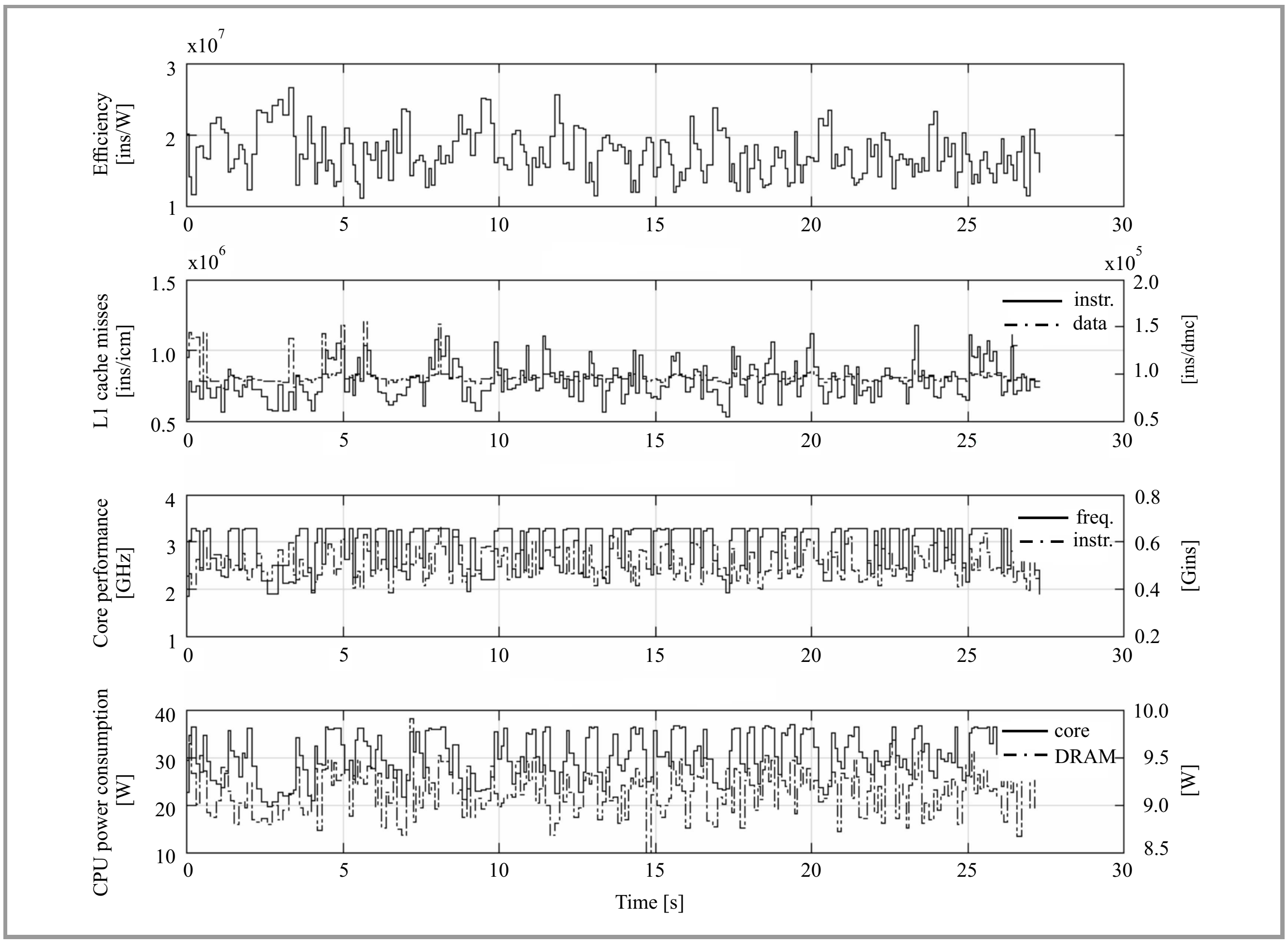

Fig. 3. Server performance and energy-efficiency trace (`intel_pstate powersave`).

7. Summary

Currently developed techniques of server power control exploit increasing possibilities provided by high-resolution sensors of modern computing hardware and software. This paper presents a brief overview of performance and power consumption monitoring tools available in Linux systems. It is argued that the measurements collected at high sampling rate can be used to develop maximally informative power consumption metrics and accurate dynamical processing models for the purpose of energy-aware design of server controllers.

This research was partially supported by the National Science Centre (NCN) under the grant no. 2015/17/B/ ST6/ 01885.

References

[1] J. Koomey, *Growth in Data Center Electricity Use 2005 to 2010*. Oakland, CA: Analytical Press, 2011.

[2] B. Subramaniam and Wu-chun Feng, "Towards energy-proportional computing for enterprise-class server workloads", in *Proc. 4th ACM/SPEC Int. Conf. Perform. Engin. ICPE 2013*, Prague, Czech Republic, 2013, pp. 15–26.

[3] J. Dongarra *et al.*, "The international exascale software project roadmap", *Int. J. High Perform. Comput. Appl.*, vol. 25, no. 1, pp. 3–60, 2011.

[4] S.-Y. Jing, S. Ali, K. She, and Y. Zhong, "State-of-the-art research study for green cloud computing", *The J. of Supercomput.*, vol. 65, no. 1, pp. 445–468, 2013.

[5] V. K. Vavilapalli *et al.*, "Apache hadoop yarn: yet another resource negotiator", in *Proc. 4th Ann. Symp.on Cloud Comput. SOCC'13*, Santa Clara, CA, USA, 2013, pp. 5:1–5:16, 2013.

[6] S. Jha, J. Qiu, A. Luckow, P. Mantha, and G. C. Fox, "A tale of two data-intensive paradigms: applications, abstractions, and architectures", in *Proc. 3rd IEEE Int. Congr. on Big Data BigData 2014*, Anchorage, AK, USA, 2014, pp. 645–652.

[7] Y. Georgiou, T. Cadeau, D. Glesser, D. Auble, M. Jette, and M. Hautreux, "Energy accounting and control with SLURM resource and job management system", in *Distributed Computing and Networking*, M. Chatterjee *et al.*, Eds., *LNCS*, vol. 8314, pp. 96–118. Springer, 2014.

[8] E. Niewiadomska-Szynkiewicz, A. Sikora, P. Arabas, M. Kamola, M. Mincer, and J. Kołodziej, "Dynamic power management in energy-aware computer networks and data intensive computing systems", *Future Gener. Comp. Syst.*, vol. 37, pp. 284–296, 2014 (doi: 10.1016/j.future.2013.10.002).

[9] M. P. Karpowicz, P. Arabas, and E. Niewiadomska-Szynkiewicz, "Energy-aware multilevel control system for a network of Linux software routers: design and implementation", *IEEE Syst. J.*, vol. PP, no. 99, pp. 1–12, 2015
(doi: 10.1109/JSUST.20152489244).

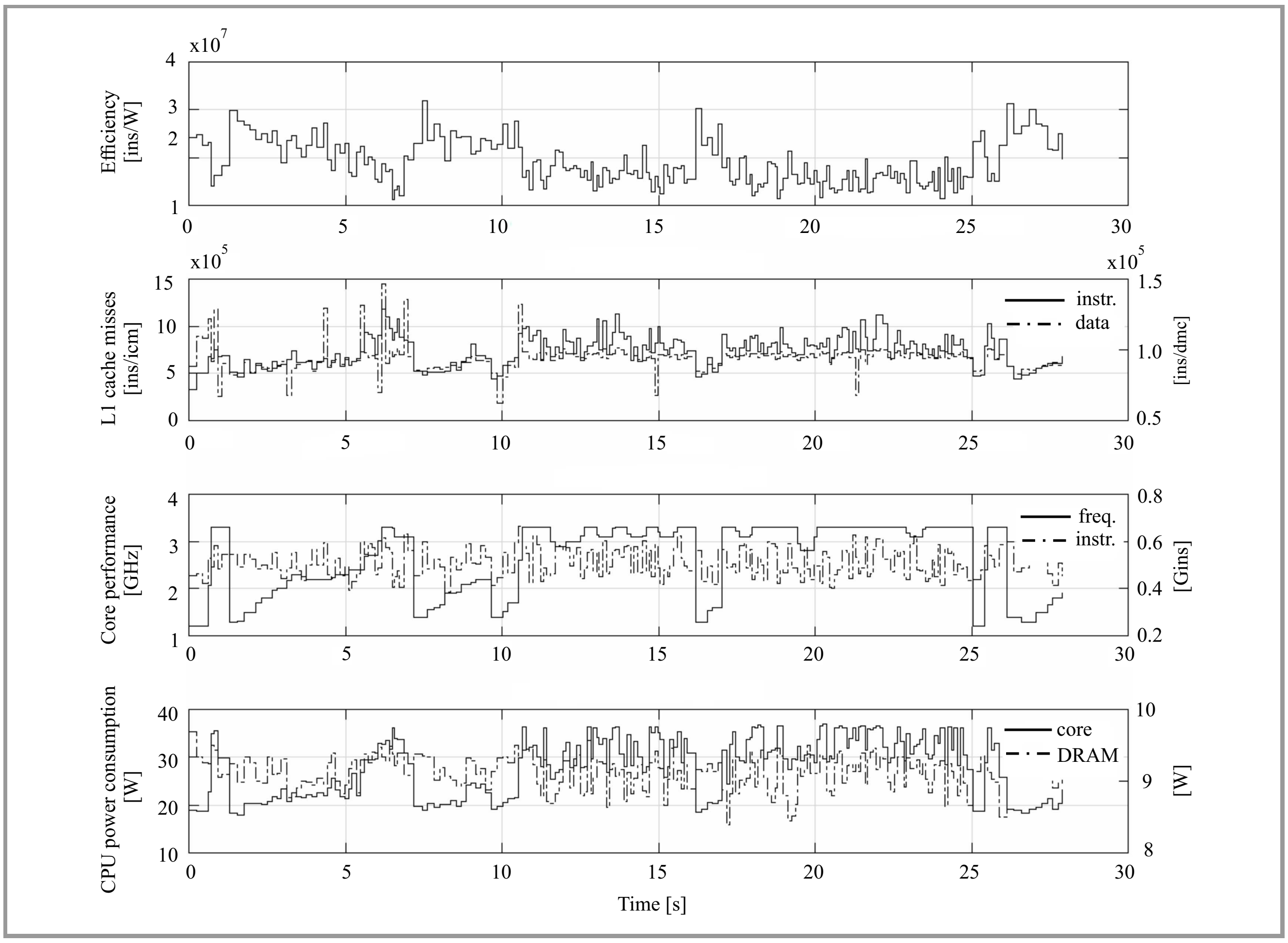

Fig. 4. Server performance and energy-efficiency trace (`intel_pstate performance`).

[10] P. Jaskóła, P. Arabas, and A. Karbowski, "Simultaneous routing and flow rate optimization in energy–aware computer networks", *Int. J. Appl. Mathem. & Comp. Sci.*, vol. 26, no. 1, pp. 231–243, 2016.

[11] A. Karbowski and P. Jaskóła, "Two approaches to dynamic power management in energy-aware computer networks – methodological considerations", in *Proc. of Feder. Conf. Comp. Science and Inform. Syst. FedCSIS 2015*, Łódź, Poland, 2015, vol. 5, pp. 1177–1182.

[12] ACPI Specification Document [Online]. Available: www.acpi.info

[13] J. H. Laros III, D. DeBonis, R. Grant, S. M. Kelly, M. Levenhagen, S. Olivier, and K. Pedretti, "High performance computing-power application programming interface specification", Tech. Rep. SAND2014-17061, Sandia National Laboratories, 2014.

[14] D. DeBonis *et al.*, "A power API for the HPC community", Sandia Report SAND2014-17061, Sandia National Laboratories, 2014.

[15] R. Bolla *et al.*, "Green Abstraction Layer (GAL): power management capabilities of the future energy telecommunication fixed network nodes", Techn. Rep. ES 203 237, ETSI, 2014.

[16] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller, "Energy management for commercial servers", *Computer*, vol. 36, no. 12, pp. 39–48, 2003.

[17] L. A. Barroso, J. Clidaras, and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, 2nd ed. Morgan & Claypool Publ., 2013.

[18] L. Wang and S. U. Khan, "Review of performance metrics for green data centers: a taxonomy study", *The J. Supercomput.*, vol. 63, no. 3, pp. 639–656, 2013.

[19] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J.-M. Pierson, and A. V. Vasilakos, "Cloud computing: survey on energy efficiency", *ACM Comput. Surveys*, vol. 47, no. 2, pp. 33:1–33:36, 2015.

[20] B. Subramaniam, W. Saunders, T. Scogland, and Wu-chun Feng, "Trends in energy-efficient computing: A perspective from the Green500", in *4th Int. Green Comput. Conf. IGCC 2013*, Arlington, VA, USA, 2013, pp. 1–8.

[21] Standard Performance Evaluation Corporation (SPEC), SPEC Power and Performance Benchmark Methodology [Online]. Available: www.spec.org/power_ssj2008/

[22] K.-D. Lange, "Identifying Shades of Green: The SPECpower Benchmarks", *IEEE Computer*, vol. 42, no. 3, pp. 95–97, 2009.

[23] D. Molka, D. Hackenberg, R. Schöne, T. Minartz, and W. E. Nagel, "Flexible workload generation for HPC cluster efficiency benchmarking", *Comp. Science-Res. & Develop.*, vol. 27, no. 4, pp. 235–243, 2012.

[24] J. J. Dongarra, P. Luszczek, and A. Petitet, "The LINPACK Benchmark: past, present and future", *Concurr. & Comput.: Practice and Experience*, vol. 15, no. 9, pp. 803–820, 2003.

[25] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. J. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing", *IEEE Trans. Parall. & Distrib. Syst.*, vol. 22, no. 6, pp. 931–945, 2011.

[26] S. Bradner and J. McQuaid, "RFC 2544: Benchmarking methodology for network interconnect devices", Mar. 1999.

[27] P. Arabas and M. Karpowicz, "Server power consumption: measurements and modeling with MSRs", in *Challenges in Automation, Robotics and Measurement Techniques*, R. Szewczyk, C. Zieliński, and M. Kaliczyńska, Eds., *Advances in Intelligent Systems and Computing*, vol. 440, pp. 233–244. Springer, 2016.

[28] M. P. Karpowicz and P. Arabas, "Preliminary results on the Linux libpcap model identification", in *Proc. 20th Int. Conf. Methods & Models Autom Robot MMAR 2015*, Międzyzdroje, Poland, 2015, pp. 1056–1061.

[29] SNIA Emerald, SNIA Emerald Power Efficiency Measurement Specification [Online]. Available: www.snia.org

[30] Storage Performance Council (SPC), Storage Performance Council SPC Benchmark 2/Energy Extension [Online]. Available: www.storageperformance.org

[31] D. Hackenberg *et al.*, "Power measurement techniques on standard compute nodes: A quantitative comparison", in *Proc. IEEE Int. Symp. on Perform. Anal. Syst. & Software ISPASS 2013*, Austin, TX, USA, 2013, pp. 194–204.

[32] J. Mair, D. Eyers, Z. Huang, and H. Zhang, "Myths in power estimation with performance monitoring counters", *Sustain. Comput.: Inform. & Syst.*, vol. 4, no. 2, pp. 83–93, 2014.

[33] M. E. Mehdi Diouri *et al.*, "Assessing power monitoring approaches for energy and power analysis of computers", *Sustainable Comput.: Informatics and Syst.*, vol. 4, no. 2, pp. 68–82, 2014.

[34] Intel Intelligent Power Node Manager [Online]. Available: www.intel.com

[35] D. Hackenberg *et al.*, "HDEEM: high definition energy efficiency monitoring", in *Proc. IEEE Energy Efficient Supercomput. Worksh. SC14*, New Orleans, LA, USA, 2014, pp. 1–10.

[36] M. F. Dolz, M. R. Heidari, M. Kuhn, T. Ludwig, and G. Fabregat, "ARDUPOWER: A low-cost wattmeter to improve energy efficiency of HPC applications", in *Proc. 6th Int. Green Comput. Conf. & Sustain. Comput. Conf. IGSC 2015*, Las Vegas, NV, USA, 2015, pp. 1–8.

[37] Intel IA-64 and IA-32 Architectures Software Developer's Manual [Online]. Available: www.intel.com

[38] NVML API Reference Manual, 2012 [Online]. Available: http://developer.nvidia.com

[39] T. Ilsche, D. Hackenberg, S. Graul, R. Schöne, and J. Schuchart, "Power measurements for compute nodes: improving sampling rates, granularity and accuracy", in *Proc. 6th Int. Green Comput. Conf. and Sustain. Comput. Conf. IGSC 2015*, Las Vegas, NV, USA, 2015.

[40] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore, "Measuring energy and power with PAPI", in *Proc. 41st Int. Conf. Parallel Proces. Worksh. ICPPW 2012*, Pittsburgh, PA, USA, 2012, pp. 262–268.

[41] L. Taniça, A. Ilic, P. Tomás, and L. Sousa, "Schedmon: A performance and energy monitoring tool for modern multi-cores", in *Euro-Par 2014: Parallel Processing Workshops*, *LNCS*, vol. 8806, pp. 230–241. Springer, 2014.

[42] Performance Application Programming Interface (PAPI) [Online]. Available: icl.cs.utk.edu/papi

[43] Unofficial Linux Perf. Events Performance Counter Weg Page [Online]. Available: web.eece.maine.edu/∼vweaver/projects/perf_events

[44] M. Gerndt, E. César, and S. Benkner, Eds., *Automatic Tuning of HPC Applications*. Shaker Verlag, 2015.

[45] M. P. Karpowicz, "Energy-efficient CPU frequency control for the Linux system", *Concur. & Computat.: Pract. & Exper.*, vol. 28, no. 2, pp. 420–437, 2016 (cpe.3476).

[46] D. Brandewie, "Intel P-state driver" [Online]. Available: www.kernel.org/doc/

[47] M. P. Karpowicz, P. Arabas, and E. Niewiadomska-Szynkiewicz, "Design and implementation of energy-aware application-specific CPU frequency governors for the heterogeneous distributed computing systems", *Future Generation Computer Systems*, available online, 2016 (doi: 10.1016/j.future.2016.05.011).

Michał Karpowicz received his Ph.D. in 2010. He is an Assistant Professor of Computer Science at the Research and Academic Computer Network (NASK) and the Warsaw University of Technology. His research interests focus on stochastic control theory, control engineering, game theory and network optimization.
E-mail: M.Karpowicz@elka.pw.edu.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland

Piotr Arabas received his Ph.D. in Computer Science from the Warsaw University of Technology, Poland, in 2004. Currently he is an Assistant Professor at the Institute of Control and Computation Engineering at the Warsaw University of Technology. Since 2002 with Research and Academic Computer Network (NASK). His research area focuses on modeling computer networks, predictive control and hierarchical systems.
E-mail: P.Arabas@elka.pw.ewu.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland

Institute of Control and Computation Engineering
Warsaw University of Technology
Nowowiejska st 15/19
00-665 Warsaw, Poland