How to Measure Energy Consumption in Machine Learning Algorithms

Eva García-Martín[1(✉)], Niklas Lavesson[1,2], Håkan Grahn[1], Emiliano Casalicchio[1,3], and Veselka Boeva[1]

[1] Blekinge Institute of Technology, Karlskrona, Sweden
eva.garcia.martin@bth.se
[2] Jönköping University, Jönköping, Sweden
[3] Sapienza University of Rome, Rome, Italy

Abstract. Machine learning algorithms are responsible for a significant amount of computations. These computations are increasing with the advancements in different machine learning fields. For example, fields such as deep learning require algorithms to run during weeks consuming vast amounts of energy. While there is a trend in optimizing machine learning algorithms for performance and energy consumption, still there is little knowledge on how to estimate an algorithm's energy consumption. Currently, a straightforward cross-platform approach to estimate energy consumption for different types of algorithms does not exist. For that reason, well-known researchers in computer architecture have published extensive works on approaches to estimate the energy consumption. This study presents a survey of methods to estimate energy consumption, and maps them to specific machine learning scenarios. Finally, we illustrate our mapping suggestions with a case study, where we measure energy consumption in a big data stream mining scenario. Our ultimate goal is to bridge the current gap that exists to estimate energy consumption in machine learning scenarios.

Keywords: Machine learning · Green computing · Computer architecture · Energy efficiency

1 Introduction

Machine learning algorithms have been increasing their predictive performance significantly during the past years. This is visible in tasks such as object recognition, where deep learning algorithms are beating human classifiers [17]. However, this has occurred at a high cost of computation and energy. In order to achieve such accurate models, the amount of computation (number of operations per second) has been increasing exponentially, with a 3.5 month-doubling time[1]. This has a direct impact on energy consumption.

[1] https://blog.openai.com/ai-and-compute/.

This work is part of the research project "Scalable resource-efficient systems for big data analytics" funded by the Knowledge Foundation (grant: 20140032) in Sweden.

C. Alzate et al. (Eds.): ECML PKDD 2018 Workshops, LNAI 11329, pp. 243–255, 2019.
https://doi.org/10.1007/978-3-030-13453-2_20

The challenge addressed in this paper is related to energy estimation and energy measurement. While machine learning researchers are starting to focus on the amount of energy consumption of their algorithms [7], estimating energy consumption is non-trivial [16]. Our goal is to bridge the gap that currently exists between computer architecture, in terms of estimating energy consumption, and machine learning by proposing to the machine learning community different approaches to model energy consumption. We present a survey with the key approaches to estimate energy from different angles (e.g. architecture level, instruction level). Moreover, from the different models that are presented, we suggest which are more suitable for the learning task at hand. We conclude the paper with a case study, where we put in practice the mentioned suggestions, and we estimate the energy consumption of a specific data mining task, with streaming data and real-time constraints.

The contributions of this paper are summarized as follows:

- We present a survey of the state-of-the-art and key methodologies to estimate power and energy consumption, shown in Sect. 4
- We suggest different approaches to estimate energy for several machine learning scenarios, shown in Sect. 5
- We present a case study of applying one of the modeling approaches in a stream mining scenario, shown in Sect. 6
- We bridge the gap between machine learning and computer architecture for estimating energy consumption.

To the best of our knowledge, this is the first study that proposes direct ways to estimate energy consumption for machine learning scenarios. We believe that the best way to move forward, given the current trend, is by a close collaboration between researchers in machine learning and computer architecture. We contribute by encouraging machine learning researchers to reduce the energy consumption of their computations, and proposing ways on how to achieve that. This is a preliminary study that plans to be extended by digging deeper into the methodologies and mapping them to more machine learning scenarios (Sect. 7).

2 Energy and Power Consumption

This section aims to give a background explanation on energy and power consumption for the machine learning audience. We explain general formulations on power, time, and energy, and specific clarifications on how software programs consume energy.

Energy efficiency in computing usually refers to a hardware approach to reduce the power consumption of processors, or ways to make processors handle more operations using the same amount of power [20].

Power is the rate at which energy is being consumed. The average power during a time interval T is defined as [35]:

$$P_{avg} = \frac{E}{T}, \tag{1}$$

where E, energy, is measured in joules (J), P_{avg} is measured in watts (W), and time T is measured in seconds (s). We can distinguish between dynamic and static power. Static power, also known as leakage power, is the power consumed when there is no circuit activity. Dynamic power, on the other hand, is the power dissipated by the circuit, from charging and discharging the capacitor [11,18]:

$$P_{dynamic} = \alpha \cdot C \cdot V_{dd}^2 \cdot f \tag{2}$$

where α is the activity factor, representing the percentage of the circuit that is active. V_{dd} is the voltage, C the capacitance, and f the clock frequency measured in hertz (Hz). Energy is the effort to perform a task, and it is defined as the integral of power over a period of time [11]:

$$E = \int_0^T P(t)dt \tag{3}$$

Energy consumption is usually the key variable to consider, since it directly translates to money spent on computations, and battery life of devices.

Finally, we conclude with an explanation of how programs consume energy. The total execution time of a program is defined as [11]:

$$T_{exe} = IC \times CPI \times T_c \tag{4}$$

where IC is the number of executed instructions, CPI (clocks per instruction) is the average number of clock cycles needed to execute each instruction, and T_C is the machine cycle time. The total energy consumed by a program is:

$$E = IC \times CPI \times EPC \tag{5}$$

where EPC is the energy per clock cycle, and it is defined as

$$EPC \propto C \cdot V_{dd}^2 \tag{6}$$

The value CPI depends on the type of instruction, since different instructions consume different amounts of energy. That is why measuring time does not give a realistic view on the energy consumption, because there are instructions that can consume more energy due to a long delay (e.g. memory accesses), or others that consume more energy because of a high requirement of computations (floating point operations). Both could obtain similar energy consumption levels, however, the first one would have a higher execution time than the last one.

3 Challenge: Measuring Energy Consumption

Measuring the energy consumption of a computer program is challenging, since there are many variables involved, e.g. cache hits, cache misses, DRAM accesses, etc. There are different ways to measure the energy consumption of a computer [16]. Traditional empirical ways to measure power are by using power

meters at different places, ranging from a wall outlet [9], to direct measurements at the motherboard. This approach outputs the real power consumption at a specific time, however, it does not provide information of where the power is consumed. More fine-grained approaches to measuring energy consumption are using simulators or using performance monitoring counters. A computer consumes energy or power by making use of its hardware components. The exact energy consumption depends on the component's use, and the energy consumption of each component. Since this is a complicated and challenging approach, power models to estimate the overall energy consumption are proposed.

This survey focuses on approaches that use either simulated hardware, or performance monitoring counters (PMC). The advantages of using simulators is that the researcher is able to have a detailed view of how each hardware component is being accessed by a specific program. It also allows for instrumentation, which can give energy consumption of different functions of a program. The main disadvantage of using simulators resides on the added overhead, which makes it unfeasible for obtaining real-time energy measurements.

PMC are available in almost all modern processors [25]. They provide the ability to count microarchitectural events of the processor at run time [16]. Since they are available for each core, and they can output many different statistics, such as instructions per cycle (IPC) and cache accesses, many researchers have used them to create power models. Moreover, Intel has created an energy model called RAPL, that estimates the energy consumption based on PMC values [8, 28]. We have used this interface to estimate the energy consumption in our case study.

4 Methods to Estimate Energy Consumption

This section presents different approaches to model power and energy consumption. The scope of this survey includes papers analyzed by [16], and [23]. We plan to extend this survey to include more estimation approaches, as presented in a recent work [26], where the authors portray significantly more estimation techniques. We have classified the papers based on three categories: type, technique, and level.

Type refers to the type of scientific modeling approach, either empirical or analytical. Analytical models estimate the power consumption for more than one type of processor, based on mathematical equations that represent the power behavior of the different parts of the processor. Empirical models are based on empirical observations of one kind of processor, and they are not usually applicable to other processor models.

Technique refers to approaches to model the energy, either by direct measurement or by simulation [30]. The authors of [30] define it as the ways to obtain the activity factors of the processor. The activity factors are those statistics or indicators of how each architectural block is being used or accessed. For instance, information on the instructions per cycle is one type of activity factor. A simulation approach to obtain the activity factors is based on simulating

a specific hardware and instrumenting the execution on that platform. On the other hand, obtaining the measurements directly can be done by accessing PMC. While simulators introduce significant overhead, PMC measurement allows for real-time energy monitoring, useful in machine learning learning contexts (e.g. data stream mining). However, models based on PMC are less portable than models based on simulations [12].

Finally, *level* refers to the granularity level of the model. We differentiate between architecture or instruction level. Architecture level models breakdown the energy consumption into the different elements of the processor and memory. For instance, they show the amount of energy consumed by the cache, the DRAM, etc. Instruction level models breakdown the energy consumption by giving an energy cost to each instruction [34]. This is useful to optimize software, since the model outputs which instructions are being the energy hotspots.

Table 1. Energy estimation models

Model	Type	Technique	Level
[22]	Analytical	Simulation	Architecture
[14]	Empirical	PMC	Architecture
[19]	Empirical	PMC	Architecture
[5]	Analytical	Simulation	Architecture
[36]	Analytical	Simulation	Architecture
[21]	Empirical	Simulation	Architecture
[1]	Empirical	PMC	Architecture
[2]	Empirical	PMC	Architecture
[13]	Empirical	PMC	Architecture
[12]	Empirical	PMC	Architecture
[15]	Empirical	PMC	Architecture
[27]	Empirical	PMC	Architecture
[31]	Analytical	PMC	Architecture
[29]	Analytical	Simulation	Instruction
[34]	Analytical	Simulation	Instruction
[23]	Empirical	PMC	Both
[32]	Analytical	PMC	Architecture
[30]	Empirical	PMC	Instruction
[8]	Empirical	PMC	Architecture

Type: = Analytical or empirical.
Technique: Performance monitoring counters (PMC), or simulation
Level: Instruction or architecture.

Table 1 summarizes the studied papers and classifies them based on the categories explained above: type, technique, and level. Table 2 clusters the papers into 6 different categories.

Table 2. Models clustered in 6 different categories, based on the features of the model

Category	Energy estimation model features	Papers
1	Analytical, Simulation, Architecture	[5,22,36]
2	Empirical, PMC, Architecture	[1,2,8,12–15,19,23,27]
3	Analytical, Simulation, Instruction	[29,34]
4	Empirical, PMC, Instruction	[23,30]
5	Analytical, PMC, Architecture	[31,32]
6	Empirical, Simulation, Architecture	[21]

Category 1 presents papers [5,22,36], which introduce analytical models, based on simulation and at an architecture level. [5] and [36] are simulators that estimate the CPU power consumption. Both are based on SimpleScalar [6], a widely used microarchitecture simulator. [22] is a state-of-the-art simulator that provides power, area, and timing models for multicore, multithreaded, and manycore architectures.

Category 2 presents papers [1,2,8,12–15,19,23,27], which introduce empirical models, based on performance counters and at an architecture level. [2,14] are able to model both dynamic and static power of the processor. The authors of [19] on the other hand do not differentiate between static and dynamic power in their model. [12,13,15,19,23] use statistical correlations between performance counters and hardware components. [1] also correlates the energy consumption but with the processor internal events to estimate and limit the processor's temperature. [1,12,27] provide models for online (real-time) power prediction. [8] presents the Intel RAPL interface, although Intel has not published in detail how they obtain their energy model.

Category 3 is composed of empirical models at the instruction level based on simulations [29,34]. [34] presented the first approach to estimate the energy cost for each type of instruction. [29] extends this approach with a more fine-grained model that is able to give accurate information during an instruction-level simulation.

Category 4 presents empirical instruction level models, based on performance counters [23,30]. [30] build an instruction level model of the Xeon Phi using microbenchmarks.

Category 5 presents analytical architecture level models based on performance counters [31,32]. [31] uses multiple linear regression between performance counters and hardware, to estimate per-core power consumption. [32] estimates the static and power consumption.

Finally, category 6 is composed by an empirical architecture level model based on simulation [21]. The authors predict power via statistical models.

Table 3. Energy consumption model suggestion for machine learning scenarios.

Goal			Dataset		Learning	
Optimize code	Optimize HW	Measure	Big data	Small datasets	Online learning	Offline learning
[23,29, 30,34].	[1,2,5,8, 12–15,19, 21–23,27, 31,32,36]	[1,2,5,8, 12–15,19,21–23,27,29–32,34,36]	[1,2,8, 12–15,19,23, 27,30–32]	[1,2,5,8, 12–15,19,21–23,27,29–32,34,36]	[1,2,8, 12–15,19,23, 27,30–32]	[1,2,5,8, 12–15,19,21–23,27,29–32,34,36]

Optimize Code: Instruction-level. Optimize HW: Architecture level. Measure: All models apply.
Big Data: not based on simulation. Small datasets: all models apply.
Online learning: PMC, but not instruction level (for real-time measurements)
Offline learning: all models apply.

5 Model Suggestion for Machine Learning Scenarios

After having described different techniques to model power consumption, we now suggest different models based on the type of machine learning scenario. We differentiate between 3 characteristics of a possible scenario: (i) goal, (ii) dataset size, and (iii) online or offline learning. We have created Table 3 to summarize which models fall into each category.

5.1 Goal

Goal refers to the goal of the researcher for using the model. We differentiate between the following three goals: optimizing software code, optimizing hardware platforms, and simply measuring energy consumption. If the goal is to optimize software code towards energy efficiency, we recommend to use **instruction-level** modeling techniques. As explained by [34], instruction-level approaches give an insight of which are the instructions responsible for the highest energy consumption. With this knowledge, the researchers can focus on mapping those instructions to software code and optimize that part of the program to reduce the overall energy consumption. Papers surveyed that focus on instruction-level modeling are: [23,29,30,34].

On the other hand, if the goal is to optimize hardware, for instance to create accelerators for deep neural networks [7], the authors can focus on **architecture-level** estimation techniques. These techniques give an understanding of which hardware components are being used in a specific computation. Papers surveyed with an architecture-level modeling focus are: [1,2,5,8,12–15,19,21–23,27,31,32, 36].

Finally, to just measure energy consumption, any of the estimating methodologies can be applied, since there are no constraints on optimizing a specific part of the system.

5.2 Dataset Size

The next characteristic refers to the characteristic of the dataset, if it is a small dataset or a large-scale dataset (e.g. Big Data). If we encounter a Big Data scenario, we recommend to model the energy consumption using techniques that focus on direct measurements. Since *big data* and *small data* are terms that are complicated to quantify and depend also on the context, we classify *big data* as any dataset that is unfeasible to train using a simulator. Simulation approaches introduce a significant overhead, making it difficult to analyze the energy consumption of scenarios with large-scale datasets. On the other hand, direct measurements are suitable for these scenarios. Modeling approaches suitable for large-scale datasets are: [1,2,8,12–15,19,23,27,30–32]. Small datasets can use the same models as suggested for big datasets, plus the ones based on simulations.

5.3 Online or Offline Learning

Our final suggestion depends on the type of learning of the task, and refers also to simulation or direct measurement techniques. As was explained before, one of the many advantages of using performance counters (direct measurements) is the ability to measure the energy consumption in real-time. This is very useful in systems such as data centers that are constantly optimizing performance (in terms of operations/watt). It is also useful in streaming scenarios, such as sensor networks or mobile devices. Real-time measurements allows for real-time energy optimization based on current load. While all models are suitable for offline learning, since there are no specific requirements on this part, only a set of those are also suitable for online learning scenarios, thus for real-time measurements [1,2,8,12–15,19,23,27,30–32].

In addition to Tables 1, 2, and 3, we portray Table 4. Table 4 gives a more clear view on possible modeling choices, based on the categories defined in Table 2. For example, we can conclude that models characterized by analytical, simulation and architecture (category 1) are suitable for small datasets and offline learning problems while the models in category 2 can be considered applicable for almost any machine learning problem.

6 Case Study: Energy Estimation in Big Data Stream Mining

We present a case study where we use the suggestions from Sect. 5, and apply them to a specific use case. Our task for this use case is to measure energy consumption in a big data stream mining scenario. For large-scale datasets we need to choose a modeling approach that does not introduce overhead to obtain the energy measurements, otherwise is not feasible when there are too many instances. For handling a stream of data, we need to have energy estimations in real time.

Table 4. Machine learning suggestions mapped to the categories from Table 3

Cat	Goal			Dataset		Learning	
	Opt code	Opt HW	Measure	Big data	Small datasets	Online learning	Offline learning
1		[5,22,36]	[5,22,36]		[5,22,36]		[5,22,36]
2		[1,2,8,12–15,19,23,27]	[1,2,8,12–15,19,23,27]	[1,2,8,12–15,19,23,27]	[1,2,8,12–15,19,23,27]	[1,2,8,12–15,19,23,27]	[1,2,8,12–15,19,23,27]
3	[29,34]		[29,34]				[29,34]
4	[23,30]		[23,30]	[23,30]		[23,30]	[23,30]
5		[31,32]		[31,32]	[31,32]	[31,32]	[31,32]
6		[21]	[21]		[21]		[21]

6.1 Experimental Design

In relation to the characterization explained above and the suggestions from Sect. 5:

- **Goal:** Compare the energy consumption of the Very Fast Decision Tree (VFDT) [10], with the Hoeffding Adaptive Tree (HAT) [4]. The VFDT and HAT are decision tree classification algorithms that can analyze potentially infinite streams of data. These algorithms obtain very similar levels of accuracy in comparison to offline decision trees. HAT is an extension of the VFDT that can handle concept drift (change in the input data stream) by using the ADWIN algorithm [3]. The goal of this case study is to show: (i) how to measure energy consumption in real time using one of the models proposed in this study; (ii) compare the energy consumption between the VFDT and HAT algorithms; (iii) compare the accuracy of the VFDT and HAT algorithms.
- **Dataset size:** Large-scale dataset with 1M instances.
- **Online or offline learning:** Online learning with real-time accuracy and energy measurements.

We have used the random tree and SEA synthetic generators, available in the stream mining framework scikit-multiflow [24]. The random tree synthetic dataset is typically used in data stream mining scenarios and was first introduced by the authors of the VFDT. The idea is to create a tree that randomly splits the features (attributes) and sets a label for the leaf. The tree is then traversed to create the different instances, based on the features at the nodes and the class labels at the leaves. The SEA synthetic dataset was introduced by [33] to test abrupt concept drift. The algorithms are run using the scikit-multiflow framework. The accuracy was evaluated using the prequential evaluator, which trains and tests the model every certain number of instances.

Table 5. Experimental results from 10 runs. Algorithms: Very Fast Decision Tree (VFDT), Hoeffding Adaptive Tree (HAT). Datasets: SEA and Random Tree generator. Measurements: average of: Accuracy, Total energy, processor energy, DRAM energy. Total energy = processor energy + DRAM energy

Alg	Dataset	Acc (%)	Tot energy (J)	Proc energy (J)	DRAM energy(J)
HAT	RandomTree	0.65	10487.04	10254.63	232.42
HAT	SEA	0.91	3612.32	3517.08	95.25
VFDT	RandomTree	0.89	5720.73	5555.49	165.24
VFDT	SEA	1.00	2006.00	1943.80	62.20

Based on the type of task, and looking at Table 4, the models that match the set {Measure, Big data, Online learning}, belong to categories: 2, 4, and 5. Thus, by looking at Table 2, these models are either: {empirical, PMC, architecture}; {empirical, PMC, instruction}; or {analytical, PMC, architecture}.

Based on these suggestions, we have chosen a model based on PMC, the Intel RAPL interface [8,28]. The reason for this choice, is that it matches the requirements, and it has a tool available to make direct energy estimation measurements. The tool, Intel Power Gadget[1], access the performance counters of the processor that are related to energy consumption. Intel has not published the details of how they estimate the energy consumption from the different performance counter statistics, that is why we do not provide extensive details. We could have also used the following models: [1,2,12–15,19,23,27,30–32].

6.2 Results and Analysis

The results of the experiment are shown in Table 5. We have evaluated the accuracy and energy consumption of running the VFDT and HAT under two synthetically generated datasets. We can see how the HAT algorithm consumes significantly more energy than the VFDT. This is understandable since the HAT algorithm performs more operations than the VFDT algorithm to be able to handle concept drift. We also output the energy consumed by the DRAM and the processor. Most of the energy is consumed by the processor, while the DRAM consumes three to five percent of the total energy.

We can see very similar accuracy values between both algorithms in these datasets. However, taking a look at the SEA dataset, we can see how the VFDT obtains higher accuracy than the HAT. These are unexpected results, since the HAT was developed to handle concept drift, and for this example the VFDT shows higher accuracy. We plan to investigate this further in future works.

This case study shows how our suggestions on using different estimation approaches depending on the machine learning task can be used. Our recommendation is to use an approach that has a tool available, since that simplifies

[1] https://software.intel.com/en-us/articles/intel-power-gadget-20.

the measurements, and encourages the researcher to look into energy consumption, not just predictive performance of the algorithm. In a real situation, once the researcher has an understanding of the energy consumption of their algorithm, the next step can involve more fine-grained results of the energy consumed either at the instruction level, with instruction-level energy models, or at the architecture level, with architecture-level energy models.

7 Conclusions and Future Work

Energy consumption is a key variable when designing machine learning algorithms. However, although some research is being conducted to reduce the computations of deep learning tasks [7], most of the research focuses on increasing the predictive performance of algorithms. One of the key challenges that exist nowadays is to measure the energy consumption of programs.

To address this challenge, we presented a survey of different approaches to estimate energy consumption. We differentiate between approaches to optimize hardware, and approaches to optimize software. Moreover, we presented suggestions to use different approaches for specific machine learning scenarios. Finally, we created a case study to illustrate our modeling approach, and a straightforward way to measure the energy in a data stream mining scenario by using performance counters. We believe that a way to improve energy efficiency in machine learning is by making energy estimation modeling approaches accessible to the machine learning community, our ultimate goal. Our case study has validated that it is possible to measure energy consumption in a machine learning scenario, proposing several ways to obtain the energy consumption values. We believe that this study brought the machine learning and computer architecture communities a step closer, in particular to achieve energy efficiency in machine learning.

This is a preliminary survey that plans to be extended to include more estimation methodologies. We intend to dig deeper into the methodologies, and match them with the requirements of specific classes of machine learning algorithms, to create recommendations of (energy model/machine learning task) that are more specific.

References

1. Bellosa, F., Weissel, A., Waitz, M., Kellner, S.: Event-driven energy accounting for dynamic thermal management. In: Proceedings of the Workshop on Compilers and Operating Systems for Low Power, COLP 2003, vol. 22 (2003)
2. Bertran, R., Gonzalez, M., Martorell, X., Navarro, N., Ayguade, E.: Decomposable and responsive power models for multicore processors using performance counters. In: Proceedings of the 24th ACM International Conference on Supercomputing, pp. 147–158. ACM (2010)
3. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining, pp. 443–448. SIAM (2007)

4. Bifet, A., Gavaldà, R.: Adaptive learning from evolving data streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) IDA 2009. LNCS, vol. 5772, pp. 249–260. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03915-7_22
5. Brooks, D., Tiwari, V., Martonosi, M.: Wattch: a framework for architectural-level power analysis and optimizations, vol. 28. ACM (2000)
6. Burger, D., Austin, T.M.: The simplescalar tool set, version 2.0. ACM SIGARCH Comput. Archit. News **25**(3), 13–25 (1997)
7. Chen, Y.H., Krishna, T., Emer, J.S., Sze, V.: Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE J. Solid-State Circuits **52**(1), 127–138 (2017)
8. David, H., Gorbatov, E., Hanebutte, U.R., Khanna, R., Le, C.: RAPL: memory power estimation and capping. In: 2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED), pp. 189–194. IEEE (2010)
9. Devices, E.E.: Watts up pro (2009)
10. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of 6th SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80 (2000)
11. Dubois, M., Annavaram, M., Stenström, P.: Parallel Computer Organization and Design. Cambridge University Press, Cambridge (2012)
12. Economou, D., Rivoire, S., Kozyrakis, C., Ranganathan, P.: Full-system power analysis and modeling for server environments. In: International Symposium on Computer Architecture-IEEE (2006)
13. Gilberto, C., Margaret, M.: Power prediction for intel XScale processors using performance monitoring unit events power prediction for intel XScale processors using performance monitoring unit events. In: ISLPED, vol. 5, pp. 8–10 (2005)
14. Goel, B., McKee, S.A.: A methodology for modeling dynamic and static power consumption for multicore processors. In: IPDPS, pp. 273–282 (2016)
15. Goel, B., McKee, S.A., Gioiosa, R., Singh, K., Bhadauria, M., Cesati, M.: Portable, scalable, per-core power estimation for intelligent resource management. In: 2010 International Green Computing Conference, pp. 135–146. IEEE (2010)
16. Goel, B., McKee, S.A., Själander, M.: Techniques to measure, model, and manage power. In: Advances in Computers, vol. 87, pp. 7–54. Elsevier (2012)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
18. Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative Approach. Elsevier, Amsterdam (2011)
19. Joseph, R., Martonosi, M.: Run-time power estimation in high performance microprocessors. In: Proceedings of the 2001 International Symposium on Low Power Electronics and Design, pp. 135–140. ACM (2001)
20. Koomey, J., Berard, S., Sanchez, M., Wong, H.: Implications of historical trends in the electrical efficiency of computing. IEEE Ann. Hist. Comput. **33**(3), 46–54 (2011)
21. Lee, B.C., Brooks, D.M.: Accurate and efficient regression modeling for microarchitectural performance and power prediction. In: ACM SIGOPS Operating Systems Review, vol. 40, pp. 185–194. ACM (2006)
22. Li, S., Ahn, J.H., Strong, R.D., Brockman, J.B., Tullsen, D.M., Jouppi, N.P.: McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In: 2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO-42, pp. 469–480. IEEE (2009)

23. Mazouz, A., Wong, D.C., Kuck, D., Jalby, W.: An incremental methodology for energy measurement and modeling. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, pp. 15–26. ACM (2017)
24. Montiel, J., Read, J., Bifet, A., Abdessalem, T.: Scikit-multiflow: a multi-output streaming framework. CoRR abs/1807.04662 (2018). https://github.com/scikit-multiflow/scikit-multiflow
25. Mucci, P.J., Browne, S., Deane, C., Ho, G.: PAPI: a portable interface to hardware performance counters. In: Proceedings of the Department of Defense HPCMP Users Group Conference, vol. 710 (1999)
26. O'brien, K., Pietri, I., Reddy, R., Lastovetsky, A., Sakellariou, R.: A survey of power and energy predictive models in HPC systems and applications. ACM Comput. Surv. (CSUR) **50**(3), 37 (2017)
27. Rajamani, K., Hanson, H., Rubio, J., Ghiasi, S., Rawson, F.: Application-aware power management. In: 2006 IEEE International Symposium on Workload Characterization, pp. 39–48. IEEE (2006)
28. Rotem, E., Naveh, A., Ananthakrishnan, A., Weissmann, E., Rajwan, D.: Power-management architecture of the intel microarchitecture code-named sandy bridge. IEEE Micro **32**(2), 20–27 (2012)
29. Sami, M., Sciuto, D., Silvano, C., Zaccaria, V.: An instruction-level energy model for embedded vliw architectures. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **21**(9), 998–1010 (2002)
30. Shao, Y.S., Brooks, D.: Energy characterization and instruction-level energy model of Intel's Xeon Phi processor. In: Proceedings of the 2013 International Symposium on Low Power Electronics and Design, pp. 389–394. IEEE Press (2013)
31. Singh, K., Bhadauria, M., McKee, S.A.: Real time power estimation and thread scheduling via performance counters. ACM SIGARCH Comput. Archit. News **37**(2), 46–55 (2009)
32. Spiliopoulos, V., Sembrant, A., Kaxiras, S.: Power-sleuth: a tool for investigating your program's power behavior. In: 2012 IEEE 20th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 241–250. IEEE (2012)
33. Street, W.N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 377–382. ACM (2001)
34. Tiwari, V., Malik, S., Wolfe, A., Lee, M.T.C.: Instruction level power analysis and optimization of software. In: Chandrakasan, A.P., Brodersen, R.W. (eds.) Technologies for Wireless Computing, pp. 139–154. Springer, Boston (1996). https://doi.org/10.1007/978-1-4613-1453-0_9
35. Weste, N., Harris, D.: CMOS VLSI Design: A Circuits and Systems Perspective, 4th edn. Addison-Wesley, USA (2010). ISBN 0321547748, 9780321547743
36. Ye, W., Vijaykrishnan, N., Kandemir, M., Irwin, M.J.: The design and use of simplepower: a cycle-accurate energy estimation tool. In: Proceedings of the 37th Annual Design Automation Conference, pp. 340–345. ACM (2000)