Emulating the Power Consumption Behavior of Server Workloads using CPU Performance Counters

Norbert Schmitt
Universität Würzburg
norbert.schmitt@uni-wuerzburg.de

Jóakim v. Kistowski
Universität Würzburg
joakim.kistowski@uni-wuerzburg.de

Samuel Kounev
Universität Würzburg
samuel.kounev@uni-wuerzburg.de

***Abstract*—The accurate measurement of a server's power consumption when running realistic workloads enables characterization of its energy efficiency and helps to make better provisioning and workload placement decisions. Information on the energy efficiency of a server for a given target workload can greatly influence such decisions and thus the final energy efficiency of a cluster or data center. However, measuring energy efficiency and power consumption of server applications has become challenging as applications are often distributed or require work intensive configuration, setup, and specialized load drivers for reproducible testing. As a result, it may be not feasible to perform tests using the actual workload that is to be deployed.**

We introduce an approach to create small-scale workloads that emulate the power consumption-relevant behavior of an application by deliberately triggering specific power relevant performance counter events. These workloads can then be easily deployed on a target server for fast and efficient power characterization. We validate the proposed approach by approximating the power consumption behavior of different workloads at multiple load levels. We show that our approach is capable of producing small-scale workloads that reflect the power consumption behavior of their reference applications over multiple load levels with a minimum error of less than 1%.

I. Introduction

Energy efficiency of servers has become more and more significant over the past decades. In 2010, the US Environmental Protection Agency (EPA) estimated that 3% of the US energy consumption is caused by data center power draw [1]. Similarly, a New York Times study from 2012 found that data centers worldwide consume about 30 billion watts per hour [2].

Reducing this significant power consumption and improving the energy efficiency of servers requires the ability to accurately and reliably evaluate the efficiency of servers when running the applications they are expected to execute. Evaluation results can help to improve energy efficiency by enabling, i. a. better server purchasing and application placement decisions, as well as efficient configuration of hardware or software.

However, performing accurate measurements with real world applications is difficult. They are often distributed and load is driven by external requests. This makes the setup and configuration complex, time consuming, and difficult, especially since it can be hard to evaluate individual software components on their respective servers in isolation. Additionally, internal power meters might not measure losses in the power supply and RAPL counters only model the CPU and DRAM. External power meters on the other hand require stable loads for a period of time in order to provide accurate measurements [1]. Achieving stable loads for real world applications requires specialized external load drivers and may also limit configurations for which a test can be run. As a result, it is often not feasible to test the actual application.

We introduce an approach to create small-scale workloads that emulate the power consumption-relevant behavior of large-scale workloads by approximating their CPU performance counter (PC) profile. We analyze PCs and their relation to power consumption to construct a Performance counter Event Trigger (PET) framework. PET is designed to use PC measurements of third-party applications to construct a small-scale workload emulating the application's PC profile aiming at reproducing its power consumption-relevant behavior.

PET's small-scale workload can easily be locally executed and measured on a target server platform. It is not part of any larger distributed workload, requires no additional configuration and is designed to be a transactional workload runnable at different load levels for server and workload characterization.

The major contributions of this paper are:

- We characterize and analyze CPU PCs with respect to their relevance for server power consumption modeling.
- We show that emulating a workload's PC profile can lead to a similar or even identical power profile.
- We propose a light-weight approach for emulating a third party workload's power consumption-relevant behavior, enabling easier benchmarking of large-scale applications.

We validate our approach by its ability to accurately emulate power consumption-relevant behavior of a small-scale test application, a standard industry benchmark and a virtual network function, which, considering its heavy I/O load, would intuitively be expected to be difficult to model using CPU performance events. We analyze PET's ability to accurately reproduce the original workloads' PCs and power characteristics and compare multiple PC triggering options. We show that PET is capable of emulating the power consumption behavior of realistic workloads down to 0.19W (1%) mean deviation.

The remainder of this paper is structured as follows: Section II describes the measurement methodologies for both wall power and PC events. Section III presents related work. We describe our approach, including PC selection and composition, in Section IV. Finally, we evaluate the PC triggers and approach's accuracy in Section V, discuss possible limitations and future work in Section VI, and conclude the paper in Section VII.

II. Foundations

In this paper, we measure CPU performance counters and system AC power (wall power). Measurements of wall power require a specialized setup and measurement methodology with external measurement devices, whereas CPU counter measurements require system instrumentation.

A. Power Measurement Methodology

For this work, we use the SPEC Power Methodology [3]. Existing work has shown that this methodology ensures high accuracy for power measurements [4] and supports the characterization of system power over multiple load levels [5].

The measurement methodology's goal is achieving stable load at multiple load levels for steady-state power measurements on the system under test (SUT). To this end, it assumes transactional workloads. Throughput is used as the system-level performance metric for determining the current load level. In our context, a transaction is any computational unit for which a beginning and end can be defined so that throughput may be measured. Load levels are defined in terms of transaction rate. As a result, the maximum load level (100%) is the load level that achieves the maximum system performance in terms of throughput. A random exponentially distributed delay between successive transactions is added to achieve load level targets below 100%. The mean delay is chosen so that the target transaction rate corresponds to the selected load level.

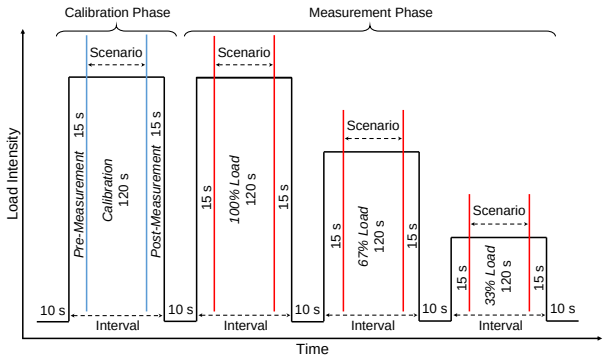

Fig. 1. Calibration and measurement phases. [6]

For stable measurements at the different load levels, measurement is organized into multiple phases, each consisting of one or multiple intervals, as illustrated in Fig. 1. First, a calibration phase executes one or multiple calibration intervals at maximum load that measures the SUT's maximum performance. Afterwards, the measurement phase measures a separate interval for each target load level. The exact number and target load levels depends on the specific test in question. Each interval includes short timed pre- and post- measurement intervals to avoid measurement during transient phases.

B. Performance Counters

Hardware manufacturers implement PCs in most processors to monitor a system's behavior by recording a variety of events, such as cache misses, branch mispredictions, and others. PCs can be read using either specialized software, performance monitoring utilities of the operating system (OS), or by directly accessing model specific registers of the CPU. Most counters log events on a system wide basis but some allow reading event subsets on a per socket or per core basis. In general, PCs can be partitioned in two groups. *Occurance events* that count how often an event has been observed and *duration events* that count the accumulated clock ticks during the occurance of an event [7] [8]. Reading PCs can have significant overhead, depending on the instrumentation [9]. Care must be taken when the third-party application is instrumented so the maximal throughput is not influenced, including a dedicated benchmarking network.

III. Related Work

PET generates performance events for efficiency benchmarking. Thus, we group related work in the following two categories.

A. Energy Efficiency and Benchmarking

A wide variety of work exists on energy efficiency benchmarking with a focus on CPU-intensive workloads. Yet, most contributions are focused on a broader scope, such as [6], in which workloads are distributed hierarchically from multiple machines down to simultaneous multithreading, or [10], which distributes virtual network functions efficiently while also taking software defined networking into account. [11] concentrates on the trade-off between the virtualization overhead and efficiency increase when using VMs to aggregate workloads on fewer physical machines with higher utilization. Yet, using less systems does not necessarily increase efficiency, as shown in [5], in which the efficiency of the SERT CPU workloads under different load levels is measured. [11] also explores the influence of different workloads on efficiency. The proposed solution tries to simulate workloads on a lower level without the need to approximate multiple connected systems.

A different point of view is presented in [4], showing that repeatable power measurements can be difficult to achieve and measurement results can vary, even with nominally identical CPUs. This work focuses on a specific component rather than a complete system, in which the CPU is only one of many components influencing the efficiency.

B. Power Estimation and Performance Counters

PCs are often used for software performance analysis or for compiler optimizations [12] [13]. For example, power consumption aware thread scheduling based on PCs is explored in [14] and [15]. Having a framework that can reliably trigger performance events can be used for validation and testing such implementations.

Modeling power consumption based on PCs is also a possible application. In [16], [17] and [18], models are developed that estimate power consumption as a function of PCs, dependent on the workload. [19] creates a model based on PCs but focuses on embedded devices with specialized CPU and memory architecture. [20] describes a tool based on PCs for the UltraSPARC platform which provides energy estimations.

These works show that PCs can be used for power estimations. It is therefore expected that the proposed framework for emulating the power consumption-related behavior of a workload by triggering counter events is a viable approach for power characterization.

[21] studies the accuracy of PC measurement facilities. In [22] two major deviations of PCs are identified, while the overhead of common PC implementations is researched in [9]. It is shown that the PAPI interface has significant overhead.

IV. Approach

The primary goal of our approach is the emulation of the power consumption-relevant behavior on a SUT by leveraging PCs as a basis for the emulation. To this end, PCs with relevance to power consumption must be identified by a real world reference workload. As an initial reference workload, we use a deep packet inspection (DPI) firewall, as it produces significant CPU load, but is also bound to other hardware components due to its intensive use of network I/O.

A. General Methodology

The system on which the measurements are performed is a HP DL20 Gen9 with $16\,\mathrm{GiB}$ of memory (single module), a Xeon E3-1230v5, one $500\,\mathrm{GiB}$ $7200\,\mathrm{rpm}$ disk drive, two dual port $1\,\mathrm{GBit/s}$ network adapters (BCM5720) and a standard non-redundant $290\,\mathrm{W}$ power supply. We use Debian with kernel version 3.16.0-4-amd64. Use of different hardware should not influence PET. We assume that each component uses a minimum amount of energy to be active and is consuming a specific amount of energy for each event occurrance. PET's goal is approximation of a software's power consumption characteristics using a PC approximation. Many hardware differences, such as active idle consumption, affect the original software and the approximation equally and have no bearing on the applicability of PET. System dependability might be an issue if the system is different in a major form, for example an extra L4 cache.

Our approach consists of the following four steps:

a) Identifying Relevant PCs: The firewall is measured at ten load levels, ranging from 10% to 100% in 10% steps. During the measurement, PCs and power consumption are recorded once per second. Relevant counters are identified by correlating an increase in counter events with the increase of power consumption over the different load levels using the Pearson correlation coefficient. A PC is considered relevant if its correlation with power consumption exceeds 0.8. It is assumed that some PCs are collinear, causing other counters to be triggered as side effects. Instead of not implementing counters causing side effects in PET, we chose to compensate side effects. Counters with dependencies or those requiring active management, such as CPU frequency, are excluded.

b) Implementing Event Trigger: As the next step, we implement event triggers designed to generate the respective performance events on demand by executing suitable artificial workloads. The accuracy of an event trigger can be influenced by implementation details and several competing trigger implementations are analyzed for each event under consideration. The implementations capable of reaching the defined target counter values with best accuracy are selected for PET. We also analyze potential side effects of our implementations by measuring the influence of event trigger implementations on other counters. Side effects must be considered, given that some PCs are not independent of each other. Recorded side effects are used to determine interactions/influences between performance events and their potential impact regarding the final power consumption emulation. We also consider background noise, which if not considered may lead to erroneous assumptions regarding the event trigger accuracy and side effects. Consequently, we measure the counter background noise on the SUT in an idle state for $120\,\mathrm{s}$.

c) Workload Composition: PET creates the final workload for power consumption behavior using one of three composition mechanisms. The mechanisms differ by how they account for the PC side effects that are caused by the concrete trigger implementations and possible collinearities. The first *naive* method ignores side effects and operates only on the target event count recorded for the workload. The second composition mechanism *accumulates* the side effects. If an event causes a side effect, its target value is multiplied with the number of side effects it is causing. All side effects on an event are summed and subtracted from the original target value. The third composition method uses *simulated annealing*, as described in [23]. We use simulated annealing, as PCs can have non-linear dependencies to a degree that makes an analytic solution of integrating side effects unfeasible. Simulated annealing is preferred to regular hill climbing as it is less prone to be caught in local extrema.

d) Trigger Pruning: Finally, we prune event triggers from the composition. In cases where some PCs reach values close to the system's background noise, it may be beneficial to remove them from the composition under the assumption that they have a low or negative impact on the system's power consumption. We make this assumption if the PC in question: *i)* is overcounting by at least one order of magnitude, taking into account the median over all load levels, *ii)* features a correlation with power consumption of less than 0.9 in the reference measurement or *iii)* features a median reference value below the background noise.

Not pruning transactions may lead to inaccuracies in both the power consumption as well as the target performance in the power measurement methodology. The power methodology is based on reaching target load levels by injecting respective target throughputs, which can only be done if the sizes of transactions within the workload are sufficiently small, so that steady state can be reached during measurement. However, triggers can only ever trigger a whole number of performance events. PET's goal is to preserve the ratio between different performance events for each transaction within the workload. Consequently, triggering events that occur in very small numbers causes all other events to have to be triggered in relatively high numbers, in turn causing transactions to grow in size.

This can cause instability in power measurements as large transactions are run at low frequencies, making it hard to ensure steady state during measurements.

B. Performance Counter Relevance to Power

In this first step of our approach, we analyze PCs with respect to their relevance for modeling a system's power consumption profile. The goal of this analysis is the selection of the most power relevant counters for inclusion in the PET framework.

Counters that are implemented in PET and that have a high correlation with system power include *L3MISS*, *L2MISS*, *READ*, *WRITE*, *INST*, *irq* and *ctxt*. Despite lower correlation (< 0.8), *L3HIT*s and *L2HIT*s are selected, considering that cache misses do correlate with power consumption. If a memory access misses L2, it could either hit or miss L3. It therefore seems reasonable to include *L3HIT* as L2 misses could directly generate L3 hits if needed. As the cache's content needs to be controlled to reliably trigger cache misses, *L2HIT* is also included in PET despite a correlation of 0.727. Correlation on the remaining events is deemed too low to be relevant or the event has a dependency on an already implemented counter.

C. Event Trigger Implementation

Triggers to generate performance events can be implemented in multiple ways. The following paragraphs describe our competing implementations for triggering counters.

To trigger cache hits, misses and memory access, an array of at least twice the L3 cache size and up to $2048\,\mathrm{MiB}$ is used. The array is traversed in a step size of 2, 4 and 6 times the cache line to determine if it has an influence on accuracy. A random number is added to each step size to account for the influence of hardware prefetching which can detect linear memory access. To determine if different instructions allow for a better accuracy, C, non-temporal SIMD intrinsics, and Assembler (ASM) for read, write and copy functionality are implemented. Each combination is tested with process owned memory, shared memory and mapped kernel memory marked as uncachable via the page attribute table. In case of uncachable memory, only fixed pointers are used.

Retired instructions are implemented by looping over an instruction adding a constant to a temporary variable. For each iteration, the counter is incremented by the amount of assembler instructions that build the loop.

To trigger a context switch, a new process or thread has to be created. We create a thread with an empty workload that is switched in and immediately joined to be switched out by the OS. Consequently, context switches are always triggered in groups of two.

We use the Boost open source library to generate interrupts by program the advanced programmable interrupt controller to throw a local timer interrupt after a deadline is reached. The deadline is set to the minimum value after which interrupts can be observed.

V. Evaluation

We evaluate our approach by measuring each trigger implementation by itself to determine its accuracy. Each trigger's most viable solution is incorporated into PET for use in the final workload compositions. This final framework is then evaluated for its ability to emulate the power consumption behavior of different server workloads with increasing complexity. For each of the workloads, measurements are taken for parallel and sequential versions with one, four and eight processes. As L2 cache misses are triggered as L3 hits, L2 misses are implicitly evaluated using L3 hits.

A. Accuracy of Performance Event Triggers

We evaluate the event trigger implementations based on their ability to reach pre-defined constant event counter numbers. The target ensures that the trigger implementation works and the deviation shows how accurate a trigger can modify its corresponding PC. Cache counters are evaluated based on their ability to accurately reach a target count of 1×10^6, memory access target count is 64×10^6 due to always reading or writing a complete cache line. For instructions retired, a value of 1×10^{10} is set as target count. Context switches and interrupts are set to 1×10^5. For parallelization, target values must be reached for each process. PCs are measured on an idle system for $120\,\mathrm{s}$ to establish the background noise.

a) L3 Cache Misses: Figure 2 shows that all read (red) and copy (blue) implementations reach a low deviation if used sequentially with step size 6 and random factor for higher memory sizes. For $512\,\mathrm{MiB}$, the ASM read implementation reaches the best deviation of -3.5%. The copy implementations perform equally well but exhibit higher side effects by writing and reading from memory. Write functions (green) generate almost no cache misses. The accuracy of the event trigger decreases without randomness, except for ASM that cannot be optimised by the compiler. Decreasing the step size reduces the accuracy for all implementations. Using shared memory instead of process memory yields similar behavior. Setting parts of memory as uncachable results in a negligible amount of L3 cache misses. We assume that accessing uncachable memory is not counted towards cache misses and is therefore not further investigated. The L3 cache miss trigger exhibits similar behavior when used in parallel with 4 and 8 processes. Using shared memory requires system calls but is otherwise comparable to using process memory and not used in PET. We can conclude that a step size of 6 works best overall and that the C and ASM implementations exhibit a higher overall accuracy than the SIMD implementation, with ASM exhibiting the most promising characteristics. It is included in PET without randomness and a step size of 6 together with a memory size of $512\,\mathrm{MiB}$, as a compromise between memory footprint and accuracy.

b) L3 Cache Hits and L2 Cache Misses: Cache hits are measured similarly as L3 cache misses, except we only evaluate a step size of 6 due to its promising characteristics. The results with a random step are presented in Table I. Results without randomness and write functionality are not presented

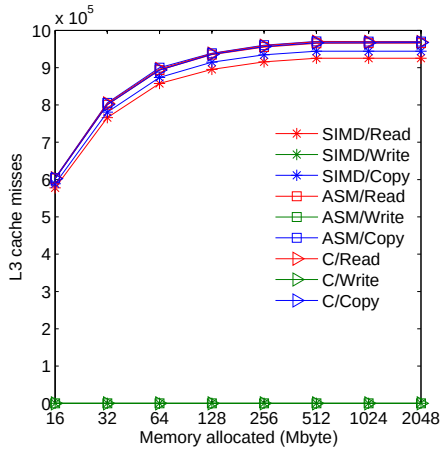

Fig. 2. L3 cache misses at different allocated memory sizes for one process, step size 6, random step and target value 1×10^6

Processes		ASM	SIMD	C
1	Read	923 975	966 181	971 336
	Copy	922 185	979 248	968 232
4	Read	3 391 786	3 408 932	3 537 446
	Copy	2 218 435	3 155 795	2 701 179
8	Read	5 253 885	4 047 238	4 199 528
	Copy	3 738 683	4 343 272	4 320 763

TABLE I
L3 CACHE HITS FOR PROCESS OWNED MEMORY WITH RANDOM STEP AND TARGET VALUES 1×10^6, 4×10^6 AND 8×10^6 BYTES

due to not reaching the target values by a large margin. Using shared memory results in similar behavior as observed with L3 cache misses. As with L3 cache misses, ASM and C implementations perform better than SIMD. With a deviation of -11.5%, the C read implementation performs best up to 4 processes in parallel. Increasing the process count above the number of physical cores (4) does not scale well. The C read trigger shows the most deterministic behavior overall and scales with the process count up to the physical core count and is used in PET.

c) L2 Cache Hits: The separate L2 cache hit trigger is not able to generate enough cache hits to reach the target value and not deemed sufficiently accurate to be included in PET.

d) Bytes Read from Memory Controller: As shown for L3 cache misses, a step size of 6 works best in triggering cache misses and investigations for other step sizes are omitted. The results in Table II show a large overcount on all implementations and memory sizes. Especially if parallelizing using 4 processes, the ASM read implementation comes close to the target value with only a small deviation. With 8 processes, deviations are larger with over -30% across all implementations. The ASM read function performs similarly as the C function and works well on lower process counts if combined with uncachable memory. ASM read is used in PET.

e) Bytes Written to Memory Controller: The bytes written counter behaves similar to Figure 2 using process owned memory. The implementation works well for memory sizes of 512 MiB and larger. Adding a random step size generally performs better due to circumventing hardware prefetching. Yet, implementations are not as accurate when parallelized. The event trigger struggles at approximating the target value

Processes	Implementation	Bytes read	Deviation
1	SIMD	64 002 560	0.004 %
	ASM	64 038 080	0.060 %
	C	64 025 024	0.039 %
4	SIMD	242.41×10^6	−5.31 %
	ASM	255.98×10^6	−0.01 %
	C	191.16×10^6	−25.33 %
8	SIMD	326.43×10^6	−36.24 %
	ASM	343.67×10^6	−32.88 %
	C	347.82×10^6	−32.07 %

TABLE II
BYTES READ RESULTS AND DEVIATION WITH UNCACHABLE MEMORY FOR TARGET VALUES 64×10^6, 256×10^6 AND 512×10^6 BYTES

in multi-process environments. Using shared memory yields the same characteristics and the uncachable memory did not perform as expected with poor accuracy.

f) Instructions Retired: Retiring instructions works with good accuracy and deviations of 0.05 %, 0.30 % and 0.52 % for 1, 4 and 8 processes respectively.

g) Context Switches: To test if our trigger implementations causes context switches in groups of two, we evaluate the implementation with different correction factors f by which the target event count is divided. The results using $f = 2.00$ of our assumption and $f = 1.00$ cause high deviations in all cases. We also test a correction factor of $f = 1.25$. This works well with one and four processes with deviations of 0.6 % and 0.2 %. With eight processes, a slightly higher deviation of −5.4 % can be observed. We therefore use $f = 1.25$ in PET.

h) Interrupts: The results show that the interrupt implementation performs well with only minor deviations of 0.3 %, 0.2 % and −0.2 % for 1, 4 and 8 processes.

B. PET

PET is evaluated using three workloads with increasing complexity on the same set of selected performance counters. The first is the synthetic workload *PI*, designed to only stress the CPU by approximating π using the Gregory-Leibniz series. The second workload, *XMLValidate* is a standard benchmark workload from the SPEC SERT [1] stressing the CPU and memory. For each transaction, XMLValidate randomly moves comments inside an XML document which is then validated against a XML schemata. The last and most demanding workload is a DPI firewall that also uses additional network interface cards (NICs) driven by external load generators.

For each load level, a pre-measurement, measurement and post-measurement time of 30 s, 120 s and 10 s is used. For each workload, we evaluate PET's three composition mechanisms to include the measured side effects. The mean and maximum deviation and coefficient of variation (CV) over all load levels is presented in our work. We also analyze the ability to accurately trigger the target performance counters and evaluate the results for each of the three different composition mechanisms.

a) PI Workload: When using a performance trigger composition that ignores side effects, power consumption behavior seems accurate on average. However, the maximum deviation is over 10% as shown in Table III. If accumulating side effects, the CV is higher but it deviates less from the target power

Measurement		Power Deviation Mean	Max.	CV
PI Full	Naive	2.95 %	14.96 %	6.94 %
	Accumulation	−1.93 %	−12.43 %	7.24 %
	Sim. An.	−42.05 %	−59.63 %	28.26 %
PI Pruned	Naive	−1.32 %	−4.39 %	1.79 %
	Accumulation	−0.62 %	−4.51 %	2.00 %
	Sim. An.	2.18 %	9.36 %	3.45 %
XMLVal. Full	Naive	−10.48 %	−38.93 %	27.69 %
	Accumulation	−4.32 %	−11.29 %	5.04 %
	Sim. An.	−52.57 %	−67.78 %	26.81 %
XMLVal. Pruned	Naive	−23.31 %	−39.48 %	29.82 %
	Accumulation	−4.36 %	−11.57 %	5.40 %
	Sim. An.	−18.85 %	−27.02 %	8.61 %
NFV Full	Naive	−6.63 %	−16.35 %	8.27 %
	Accumulation	−23.69 %	−40.19 %	14.33 %
	Sim. An.	−20.77 %	−35.97 %	12.33 %
NFV Pruned	Naive	−4.52 %	27.18 %	13.42 %
	Accumulation	−23.38 %	−39.83 %	14.28 %
	Sim. An.	−20.83 %	36.20 %	12.28 %

TABLE III
WORKLOAD EMULATION RESULTS WITH MEAN AND MAXIMUM DEVIATION AND CV

consumption on average and at its peak. Using simulated annealing to account for side effects does not perform as well and has a lower accuracy. Still a low deviation of below 5 % can be achieved on simple workloads. When pruning performance triggers, all triggers except instructions retired and context switches can be removed. Pruning event triggers results in an overall improvement for all three composition methods. For PI, accumulating side effects for the pruned trigger composition is determined as the best solution, as its average deviation is closest to the target power consumption behavior with only marginally higher maximum deviation and CV. The results show that the local, CPU heavy, PI workload can be approximated with good accuracy and only minor deviations from the target power behavior with our approach. It also shows that event trigger pruning can be beneficial for workloads that are focused on specific hardware components.

b) XMLValidate: For the XMLValidate workload accumulation works well and reaches deviations and a CV comparable to the PI workload. Simulated annealing and naive composition are not as accurate. L3 cache misses and hits, bytes read and written, and the retired instruction count are not pruned for all measurements. The interrupts are pruned from all configurations due to low correlation. Context switches are removed from accumulation due to a significant overcount. Using a more complex workload does not necessarily result in a higher deviation as XMLValidate shows. The accumulation measurement is in close proximity to the PI workload, with a mean deviation of below 5 %. Yet, pruning performance triggers does not necessarily yield better results. This is mainly due to the workload stressing more hardware components, which in turn leads to fewer event triggers that can be removed.

c) NFV: The DPI firewall stresses hardware parts otherwise not used by the previously employed CPU and memory bound workloads. Using the firewall, we evaluate the ability to emulate the power consumption-relevant behavior even though specific hardware components (NICs) remain unused. The results in Table III show that this can be achieved. The naive composition that ignores side effects works best, deviating the least from the target in both mean and maximum power consumption. Yet, PET slightly underestimates power consumption consistently. All compositions favor underestimation. This behavior is expected considering that PET does not utilize NICs. The accumulation and simulated annealing compositions do not achieve the naive method's good accuracy with higher deviations and CVs.

Only L3 cache misses can be pruned not resulting in an overall better approximation. This coincides with the XMLValidate workload. Accumulation and simulated annealing show only minor effects on the efficacy of pruning. Pruning improves the naive composition on average but at the cost of a higher CV and maximum deviation. We assume that the mean deviation would be close to the full measurement without an overestimation at the 90 % load level.

VI. Future Work

Most results using simulated annealing show that it does not improve results as assumed. One possible cause could be the parameterization of the simulated annealing algorithm which leaves room for improvements. Another option would be an energy function less suited for balancing PC events and their side effects. We still think annealing techniques can help improve the results but need further research for our use case.

Some of the less optimal results shown in the XMLValidate workload are most likely based on the *write* PC. XMLValidate exhibits large amounts of memory access which is the least accurate event trigger in PET. Further research in this direction is necessary to increase the accuracy of the trigger implementation, making PET more accurate on complex workloads.

To further increase accuracy and support a broader range of systems, other PCs for disk I/O or GPU could also be investigated in the future.

VII. Conclusion

In this paper, we introduce PET, a framework for creation of synthetic workloads that emulate the power consumption-relevant profile of other applications. PET is intended to be used to emulate workloads for which the setup of a power benchmarking infrastructure is either highly complex or infeasible. Specifically, we show PET's use in the context of benchmarking virtual network functions, which usually require a complex network setup.

PET emulates specific power consumption-relevant behaviors by triggering the performance counter events measured of the original workload. For this, it composes performance event trigger implementations into a synthetic workload, considering their power relevance and counter interdependencies. We show that PET can emulate the power consumption of workloads with an average deviation of less than 10% even if the original workload used additional hardware, such as network interface cards. It can reach a deviation of less than 1% when emulating workloads that do not employ such additional hardware.

REFERENCES

[1] K.-D. Lange and M. G. Tricker, "The Design and Development of the Server Efficiency Rating Tool (SERT)," in *ICPE 2016*, ser. ICPE '11. New York, NY, USA: ACM, 2011, pp. 145–150. [Online]. Available: http://doi.acm.org/10.1145/1958746.1958769

[2] C. Babcock, "NY Times data center indictment misses the big picture," 2012.

[3] Standard Performance Evaluation Corporation, "SPEC Power and Performance Benchmark Methodology," http://spec.org/power/docs/SPEC-Power_and_Performance_Methodology.pdf.

[4] J. von Kistowski, H. Block, J. Beckett, C. Spradling, K.-D. Lange, and S. Kounev, "Variations in CPU Power Consumption," in *ICPE 2016*. New York, NY, USA: ACM, March 2016.

[5] J. von Kistowski, H. Block, J. Beckett, K.-D. Lange, J. A. Arnold, and S. Kounev, "Analysis of the Influences on Server Power Consumption and Energy Efficiency for CPU-Intensive Workloads," in *ICPE 2015*, ser. ICPE '15. New York, NY, USA: ACM, February 2015.

[6] J. von Kistowski, J. Beckett, K.-D. Lange, H. Block, J. A. Arnold, and S. Kounev, "Energy Efficiency of Hierarchical Server Load Distribution Strategies," in *MASCOTS 2015*. IEEE, October 2015.

[7] *AMD64 Architecture Programmer's Manual Volume 2: System Programming*, Advanced Micro Devices Inc., April 2016. [Online]. Available: http://support.amd.com/TechDocs/24593.pdf

[8] *Intel® 64 and IA-32 Architectures Software Developer's Manual*, Intel Corporation, June 2016. [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf

[9] V. M. Weaver, "Self-monitoring Overhead of the Linux perf_event Performance Counter Interface," in *ISPASS 2015*. IEEE, March 2015, pp. 102–111.

[10] R. Bolla, R. Bruschi, C. Lombardo, and S. Mangialardi, "DROPv2: Energy-Efficiency through Network Function Virtualization," *IEEE Network*, vol. 28, no. 2, pp. 26–32, Apr. 2014.

[11] Y. Jin, Y. Wen, and Q. Chen, "Energy Efficiency and Server Virtualization in Data Centers: An Empirical Investigation," in *2012 IEEE Conference on Computer Communications Workshops*, Mar. 2012, pp. 133–138.

[12] S. Eyerman, L. Eeckhout, T. Karkhanis, and J. E. Smith, "A Performance Counter Architecture for Computing Accurate CPI Components," in *ASPLOS XII*, 2006, pp. 175–184.

[13] J. Cavazos, G. Fursin, F. Agakov, E. Bonilla, M. F. O'Boyle, and O. Temam, "Rapidly Selecting Good Compiler Optimizations using Performance Counters," in *CGO '07 Proceedings of the International Symposium on Code Generation and Optimization*, 2007, pp. 185–197.

[14] K. Singh, M. Bhadauria, and S. A. McKee, "Real Time Power Estimation and Thread Scheduling via Performance Counters," *ACM SIGARCH Computer Architecture News*, vol. 37, pp. 46–55, May 2009.

[15] F. Bellosa, "The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems," in *Proceedings of the 9th workshop on ACM SIGOPS*, 2000, pp. 37–42.

[16] W. L. Bircher and L. K. John, "Complete System Power Estimation Using Processor Performance Events," *IEEE Transactions on Computers*, vol. 61, no. 4, pp. 563–577, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1109/TC.2011.47

[17] A. Lewis, S. Ghosh, and N.-F. Tzeng, "Runtime Energy Consumption Estimation Based on Workload in Server Systems," in *Proc. of the 2008 Conf. on Power Aware Computing and Systems*, ser. HotPower'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855610.1855614

[18] C. Isci and M. Martonosi, "Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 36. Washington, DC, USA: IEEE Computer Society, 2003, pp. 93–. [Online]. Available: http://dl.acm.org/citation.cfm?id=956417.956567

[19] G. Contreras and M. Martonosi, "Power Prediction for Intel XScale® Processors Using Performance Monitoring Unit Events," in *Proceedings of the 2005 International Symposium on Low Power Electronics and Design*, ser. ISLPED '05. New York, NY, USA: ACM, 2005, pp. 221–226. [Online]. Available: http://doi.acm.org/10.1145/1077603.1077657

[20] I. Kadayif, T.Chinoda, M. Kandemir, N. Vijaykrishnan, M. Irwin, and A. Sivasubramaniam, "vEC: Virtual Energy Counters," in *PASTE'01*, 2001, pp. 28–31.

[21] D. Zaparanuks, M. Jovic, and M. Hauswirth, "Accuracy of performance counter measurements," in *ISPASS 2009*, Apr. 2009, pp. 23 –32.

[22] V. M. Weaver, D. Terpstra, and S. Moore, "Non-Determinism and Overcount on Modern Hardware Performance Counter Implementations," in *Performance Analysis of Systems and Software, 2013. ISPASS 2013. IEEE International Symposium on*, 2013.

[23] D. Henderson, S. H. Jacobson, and A. W. Johnson, *The Theory and Practice of Simulated Annealing*, F. Glover and G. A. Kochenberger, Eds. Boston, MA: Springer US, 2003. [Online]. Available: http://dx.doi.org/10.1007/0-306-48056-5_10