Deriving Power Models for Architecture-Level Energy Efficiency Analyses

Christian Stier[1], Dominik Werle[2], and Anne Koziolek[2]

[1] FZI Research Center for Information Technology, Karlsruhe, Germany
stier@fzi.de
[2] Karlsruhe Institute of Technology, Germany
{dominik.werle,koziolek}@kit.edu

Abstract. In early design phases and during software evolution, design-time energy efficiency analyses enable software architects to reason on the effect of design decisions on energy efficiency. Energy efficiency analyses rely on accurate power models to estimate power consumption. Deriving power models that are both accurate and usable for design time predictions requires extensive measurements and manual analysis. Existing approaches that aim to automate the extraction of power models focus on the construction of models for runtime estimation of power consumption. Power models constructed by these approaches do not allow users to identify the central set of system metrics that impact energy efficiency prediction accuracy. The identification of these central metrics is important for design time analyses, as an accurate prediction of each metric incurs modeling effort. We propose a methodology for the automated construction of multi-metric power models using systematic experimentation. Our approach enables the automated training and selection of power models for the design time prediction of power consumption. We validate our approach by evaluating the prediction accuracy of derived power models for a set of enterprise and data-intensive application benchmarks.

1 Introduction

Design-time quality analyses allow software architects to estimate quality characteristics of a designed system in early design phases and during software evolution. In the context of software systems, energy efficiency refers to the ratio of useful work the system performs and the energy it consumes, as Barroso et al. [1] outline. Energy efficiency is an essential quality characteristic as it determines a large portion of the deployed systems' operational cost. Power consumption accounts for over 15% of the Total Cost of Ownership (TCO) [2]. The usage profile of software determines the power consumption of servers on which it is deployed [3, 4, 5, 6]. Meaningful reasoning on the energy efficiency of software hence requires the consideration of both design and deployment of software architectures [7].

The consideration of energy efficiency at design time enables software architects to reason on the implications of design decisions on infrastructure sizing and operational cost. The energy efficiency of software architectures can be predicted using approaches as proposed by Brunnert et al. [5] and in our previous work [6]. The approaches utilize software performance models and power models to predict a system's power consumption at design time. Performance models predict performance and system metrics of a software system under a given workload. Power models then correlate the predicted system metric with power consumption of servers or individual hardware components to estimate power consumption. Using power models, previous work accurately predicts the effect of varying user workload [5] and architectural design decisions [6] on energy efficiency.

When extracting power models to evaluate the energy efficiency of a software system at design time, the implementation of the system is not yet fully available. Hence, the power models need to be trained on workloads for systems other than the system under design. Collecting representative measurements as training data is challenging as the relation between issued workload and values of the observed metrics is non-linear. A set of measurements hereby is representative if it allows to correlate the variance of power consumption with variances of system metrics. Individual workloads might not stress all the resources of the system under evaluation that impact its power consumption. In this case, individual workloads do not produce representative sets of measurements.

In early design phases and during software evolution, metrics such as throughput and utilization of processing units can be predicted with reasonable accuracy and modeling effort. Fine-grained system metrics such as the number of page faults per second are difficult to predict or require significant effort in refining the models. The effort in constructing fine-grained models should only be invested if it results in a significant increase in accuracy.

Existing approaches [5, 6] for the design time prediction of energy efficiency use a manual process for selecting a suitable power model for a system under investigation. The authors assume that the utilization of certain server components significantly correlate with power consumption. They do not systematically select these metrics for each server under investigation. Previous work on the automated construction of power models [3, 4] for run time estimation allow for an automated selection of metrics that correlate with power consumption. The approaches outlined in [3, 4] do not consider the tradeoff between accuracy and effort that is essential to the extraction of power models for design time consumption predictions. Rather, they produce power models that rely on low-level system metrics and hardware performance counters.

We propose a methodology for the automated extraction of power models for design time energy efficiency analyses. Our approach enables software architects to evaluate which system metrics are worth considering for a specific server type based on their expected impact on power consumption prediction accuracy. Our **C**ontributions are as follows:

C1: We define a profiling approach for automatically deriving power and system metric measurements based on representative workload combinations.

C2: We train a set of power models to identify the power models that most accurately predict our systems' power consumption.

C3: We outline a methodology for evaluating the effect of considering additional system metrics in the energy efficiency analysis.

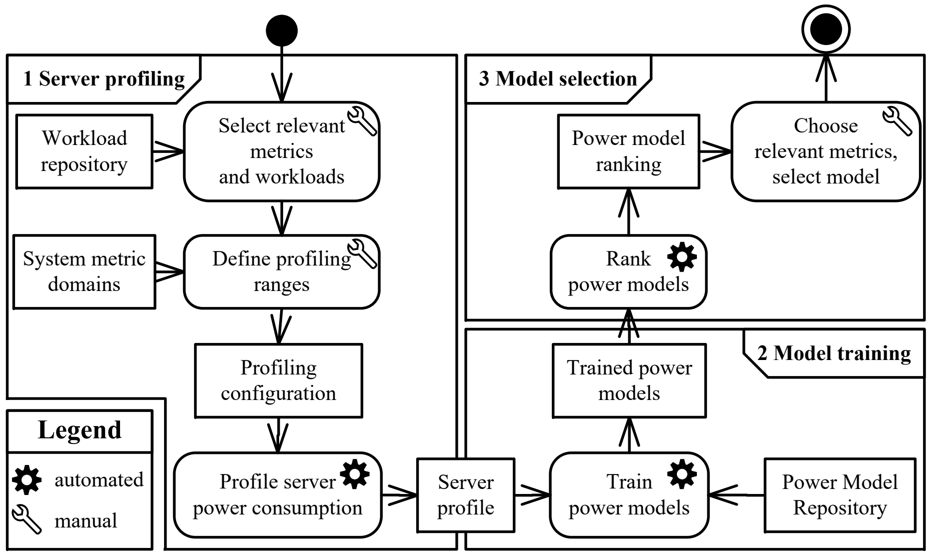

Fig. 1: Activity diagram overview of our power model extraction methodology

We evaluate our approach with application workloads different from the workloads used in profiling. The evaluation workloads cover an enterprise application workload, SPECjbb2015 [8], and a set of diverse Big Data application workloads contained in the HiBench benchmarking suite [9]. To evaluate the benefit gained by applying our profiling approach (C1) we compare the prediction accuracy of power models derived from measurements extracted using our approach with a baseline approach. The baseline approach subsumes a set of profiling approaches found in related work [3, 4]. We investigate the accuracy of power models constructed using our approach. The power models predict power consumption with an error of less than 4.9% for 19 of 27 considered models (C2). This confirms that we are able to construct power models that accurately predict the power consumption for application workloads not available at the time of profiling. We show that we correctly predict the accuracy gained by considering additional system metrics (C3).

This paper is structured as follows. Section 2 outlines our methodology. Section 3 outlines evaluation experiments and discusses their results. Section 4 discusses related work. Section 5 concludes and provides an outlook on future work.

2 Methodology

Figure 1 provides an overview of our methodology for deriving power models for architecture-level energy efficiency analyses. Our methodology consists of the three main steps *server profiling*, *model training*, and *model selection*. In server profiling, we automatically profile the power consumption for a set of system metrics to derive a representative server profile. We hereby consider a profile to be representative if it covers the typical values of system metrics of the system under the expected load. In model training, we construct a set of power models based on the server profile extracted in the first step. The final step *model selection* enables users to compare different power models and reason on the effect of system metrics on prediction accuracy. The following sections further elaborate on each of the three steps.

2.1 Server Profiling

In order to learn accurate power models for a server, we need representative measurements of the power consumption and relevant system metrics under different levels of utilization. A set of measurements is representative if it covers the typical behavior of the system under its expected workload. Using a single workload type to stress the server produces measurements that match only similar workload types. Thus, it is not sufficient to use an individual workload type as the foundation for learning power models. Different workload types need to be considered when learning power models. The measurements used to learn the models also need to cover different utilization levels for the model to be representative for possible workload mixes.

To the best of our knowledge, there does not exist an approach for targeting specific utilization levels for multiple resources using representative workloads.

We designed an approach for profiling the power consumption of a server under specific load levels. Our approach collects representative server profiles using workload mixes that use multiple resources. Our profiling approach controls the load intensity of a set of workloads to reach target values for a set of system metrics. This allows us to train power models that are representative of a large range of workloads and workload mixes. In order to validate our approach we implemented it upon the technical foundation of the Server Efficiency Rating Tool (SERT) [10, 11] framework. This enabled us to reuse industry-proven workloads for classifying server energy efficiency. The ENERGY STAR program of the U.S. Environmental Protection Agency (EPA) uses SERT and its workloads to classify server energy efficiency [12].

The following elaborates our approach. First, we provide an overview of implementation and prerequisites of our approach. Based on a running example, we discuss the workload intensity calibration and measurement performed as part of the approach.

Implementation We implemented our profiling approach atop the technical framework of SERT [10, 11]. SERT evaluates the energy efficiency of servers for a set of transactional workloads. In order to reach different throughput levels, SERT linearly scales the rate of transactions in the system based on the maximum transaction rate the system can process. SERT applies representative workloads for different resources, such as CPU and storage I/O, and scales them from idle to maximum utilization. However, SERT does not allow the parallel execution

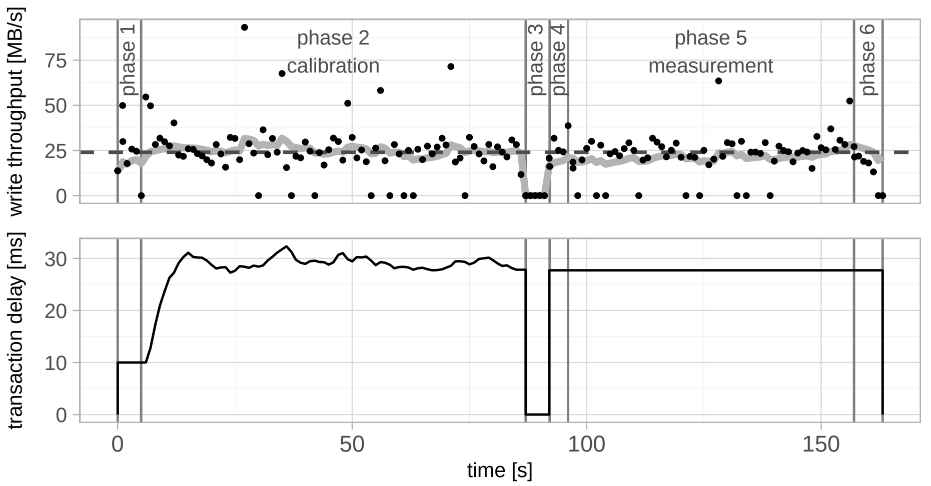

Fig. 2: Top: tp_{write} of exemplary run for target level $(u_{\text{cpu}}, tp_{\text{write}}) = (0.55, 24\,000)$. In gray: smoothed average of the measurements, and target value 24MB/s. Bottom: Transaction delays for the storage intensive workload.

of workloads that each stress different resources. Consequently, SERT does not produce a sample representative of the full combined domain of system metrics.

We implemented our profiling approach in a custom load driver. Our load driver controls the throughput by varying the delay time. It allows for the simultaneous execution of multiple workloads with different mean delay times.

Prerequisites The prerequisites subsume all activities highlighted as manual in Figure 1. Our approach requires the user to specify a set of target system metrics $M_{\text{profile}} = \{m_1, \dots, m_n\} \subseteq M$. M_{profile} is the set of system metrics targeted by the profiling. M is the domain of measurable system metrics. Example metrics are the average utilization of all CPU cores u_{cpu}, storage write throughput tp_{write} and storage read throughput tp_{read} in kilobytes per second.

The user defines a set of workload mixes used in the server profiling. A workload mix is a tuple $(w_1, \dots, w_n)$. w_j is a workload with a controllable load intensity parameter l, where there exists a monotonic relationship between l and measurements of m_j. An example element workload for u_{cpu} is the AES encryption workload $w_{\text{aes}} \in W_{u_{\text{cpu}}}$. The user defines a workload mix by selecting a workload w_j from a predefined set W_{m_i} for each $m_i \in M_{\text{profile}}$.

An individual user of our approach does not need to determine W_{m_i}. Rather, W_{m_i} has the role of a reusable repository. Once a monotonic relationship between l and the measurements m_i of a workload have been established for a workload w_{new}, any user can select from $W_{m_i}^{\text{new}} = W_{m_i} \cup \{w_{\text{new}}\}$.

```
state  : $thresholdReached \leftarrow$ false
input  : Current system metric value $u$, Target metric value $u_t$,
         Threshold metric value $u_{\text{thold}}$, Metric-specific alpha $\alpha_m$,
         Initial delay $currentDelay$
output : Delay to throttle workload $currentDelay$
if $\neg thresholdReached$ then
    if $u < u_{\text{thold}}$ then $thresholdReached \leftarrow true$;
    else $currentDelay \leftarrow 2 \cdot currentDelay$;
else
    $targetDelay \leftarrow currentDelay \cdot \frac{u}{u_t}$;
    $currentDelay \leftarrow currentDelay \cdot (1 - \alpha_m) + targetDelay \cdot \alpha_m$;
    if $\alpha_m > 0.1$ then $\alpha_m \leftarrow 0.9 \cdot \alpha_m + 0.01$;
```

Algorithm 1: Adaptive calibration policy for controlling the workload intensity.

Running Example In the following, we will explain the profiling process with reference to the example workload mix $(w_{\text{aes}}, w_{\text{rwrite}})$. w_{rwrite} is a workload executing random disk writes. We outline the profiling process using one of the target level tuples we used in our experiments. A target level describes the utilization level the profiling aims to observe for a workload run. The target level tuple we selected is $(u_{\text{cpu}},\ tp_{\text{write}}) = (0.55, 24\,000)$. Figure 2 illustrates the different steps involved in the profiling for this target level tuple. It shows measurements of tp_{write} and the load intensity of w_{rwrite} over time. The figure depicts the measured values in the upper graph. The lower graph shows the load intensity as the delay between two workload transactions. The figure shows the three phases of the calibration step (phases 1-3) and the three phases of the measurement step (phase 4-6) for $(u_{\text{cpu}},\ tp_{\text{write}}) = (0.55, 24\,000)$.

Workload Intensity Calibration The calibration phase has the goal of determining a suitable mean delay value for the transaction execution of every workload. We determine the mean value for each of the workloads in a workload mix in parallel. The calibrated mean delay value should result in a rate of transaction executions that induces the specified target level metric values.

Figure 2 shows the transaction delay for workload w_{rwrite}, together with metric values of tp_{write}. The depicted workload calibration for the storage intensive workload runs in parallel with the calibration for the CPU intensive workload. In a first step, our profiling framework initializes the workload and starts transactions at an initial rate (phase 1). Subsequently, the actual calibration process starts, in which the transaction rate is varied (phase 2). Algorithm 1 lists the algorithm used during calibration. The profiling framework executes the algorithm in every measurement interval. The algorithm tries to reach a sensible starting value for the system metric, e.g., 10 MB/s for HDD write throughput (lines 1–3). This avoids contention effects that occur for shared resources at low transaction delays.

After the threshold has been reached, the algorithm gradually approaches the target system metric value by determining the ratio between the current value u and the target value u_t. The algorithm determines the new target delay by multiplying this ratio with the current delay (line 5). We attenuate the adaption by considering the target delay with a weight α_m and the previous delay value with a weight $1 - \alpha_m$ in the calculation of the new delay value (line 6). The user can choose α_m for each metric. We set α_m to 0.2 for random writes, and 0.05 for sequential writes for the metric tp_{write}. In each run the algorithm continuously decreases α_m towards 0.1 (line 7). Consequently, the algorithm steers the transaction rate more directly in the beginning of the calibration process. After the calibration, the profiler stores the current transaction rate and stops all workloads for the idle phase 3.

The profiling framework executes the algorithm independently and simultaneously for each workload in the workload tuple. This enables the algorithm to adjust the load intensity based on interferences between the workloads. In the case of combining I/O-intensive with CPU-intensive workloads, the adjustment is necessary since most I/O-intensive workloads still utilize the CPU to perform operations on the read data, even though the CPU is not a potential bottleneck. The measured load would not match the target load for $(w_{\text{cpu}}, w_{\text{rwrite}})$ if we were to determine the delay time t_{cpu} that achieves the targeted average CPU utilization for (w_{cpu}) independently of the delay time t_{rwrite} for w_{rwrite}. Hence we need to determine delay times for the workload mix. The combined calibration allows us to achieve the utilization targets of both metrics with a parallel execution of the workload mix $(w_{\text{cpu}}, w_{\text{rwrite}})$.

Measurement Throughout the calibration and measurement phases our profiling framework takes equidistant measurements of relevant system metrics. Idle phase 1 reduces instabilities between the measurement of two target level tuples. Idle phase 3 and warmup phase 4 aim to avoid instabilities when transitioning between calibration and measurement. We consider measurements of system metrics and power consumption taken during the measurement phase to be representative values of a system in a stable state under the used workload.

In the pre-measurement step (phase 4), our framework starts all workloads in the workload mix using the calibrated transaction rate. The pre-measurement phase allows the system to stabilize and mitigates warm-up effects. Our load driver runs the system with the stable transaction rate for the measurement phase (phase 5). For technical reasons, the load driver continues to run the workload mix in a short post-measurement phase (phase 6). It then stops all workloads.

2.2 Model Training

We use power models to reason on the power consumption of the profiled servers. We construct the models by means of statistical learning techniques. The power models are trained using the power consumption profile extracted using our profiling approach discussed in Section 2.1. We utilize a *Power Model Repository* [6] to persist a set of recurring power model types. Each power model type is associated with a regression model formula. The power model type references the system metrics it requires as input. To apply a power model type to a profiled system, we instantiate it by training its non-parametrized regression model. This produces a regression model we can use to predict the power consumption of a software system deployed on the server.

In the scope of this paper we use an iterated reweighted least squares algorithm based on a robust M-estimator as implemented by Rousseeuw et al. [13] to train the regression models. The central advantage of robust regression techniques is their robustness towards outliers and anomalous measurements. While techniques for non-parametric regression have been applied to power modeling [4], we did not find conclusive evidence that they are more accurate than parametrized learning.

2.3 Model Selection

Power models can be used to reason on the power consumption at runtime and design time. Over the years, different power models have been proposed to model the relation between system metrics and power consumption of servers [14]. The accuracy of power models depends on the server under investigation and the workload executed on the system. When training power models for runtime use, the target workload for which we want to analyze the power consumption may already be fully known. In this case, we can measure the accuracy of trained power models under the expected workload mix. Based on the measured accuracy, we can select a suited power model.

At design time, the implementation of the target workload is not yet fully available. We can not select the most accurate power model based on measurements for the target workload. Still, we need to make an informed trade-off decision between the accuracy of a candidate power model and the effort required to predict its input metrics. As we cannot measure the power consumption of the target application, we need to reason on power model accuracy independent of the final implementation of the designed application.

There exist different model selection techniques based on statistical methods such as residual sum of squares, k-fold cross-validation and Akaike's Information Criterion (AIC). k-fold cross validation is commonly used in software performance engineering to evaluate the predictive quality of models. AIC is an information-theoretic measure that quantifies the information loss between the evaluated model versus the "unknown true mechanism" [15] that actually produced the data which the model was trained on. Stone [16] has shown AIC and k-fold cross-validation to be asymptotically equivalent. We apply AIC to determine whether we can increase prediction accuracy by considering additional metrics. We opted for AIC over k-fold cross due to its simplicity.

We evaluate a set of candidate power models we maintain in a Power Model Repository to find the model that most accurately describes the power consumption of the profiled server. We determine the rank of each power model based on its difference to the minimal AIC as described by Burnham and Anderson [15]: $\Delta_{\mathrm{AIC}} = \mathrm{AIC} - \mathrm{AIC}_{\mathrm{min}}$. If all models considering a set of metrics M with $m \in M$ are dominated by any model with the metric set $M \setminus \{m\}$, we deduce that there is no benefit in considering m. Should the consideration of m increase accuracy, we compare the difference in ranking between the best-performing model with metrics M and $M \setminus \{m\}$.

3 Evaluation

In our evaluation we investigated four Evaluation Questions (EQs):

EQ1: Do the power models we derive from our server profile accurately predict power consumption across different types of workload?

EQ2: Does the simultaneous profiling of CPU and HDD profiles increase the accuracy over profiling CPU and HDD in isolation?

EQ3: Does our approach produce server profiles that are better suited for training power models than other approaches?

EQ4: Does the AIC-based selection of power models accurately predict the effect of considering system metrics on prediction accuracy?

We evaluate EQ1 by analyzing the accuracy of power models from literature, which we trained using the server profile produced by our profiling approach. We investigate EQ2 by comparing the accuracy of power models trained on a profile from simultaneous profiling, and a profile from isolated profiling. To evaluate EQ3, we compare the server profiles produced by our approach against a server profile produced by a commonly used alternative approach. To investigate EQ4, we compare our AIC-based ranking with the actual accuracy of the power models for a set of workloads.

3.1 Setup

We used a PowerEdge R815 with four Opteron 6174 CPUs and 256 GB RAM. The server utilized a built-in storage RAID with six 900 GB $10,000$ RPM SAS. The server's resources were virtualized using XenServer 6.5. Profiling and power measurements were conducted within Ubuntu 14.04 VMs, with 48 virtual cores assigned to each VM. Only one VM was running at a single point in time. The SPECjbb2015 VM was assigned 32 GB RAM while the HiBench VM was allocated 16 GB RAM. Power monitoring was conducted using a ZES Zimmer LMG95 power meter connected to a dedicated notebook. The measurement data and analysis tooling used in our evaluation are available online[3].

We used the workloads *SequentialWrite*, *RandomWrite*, *XMLvalidate*, *CryptoAES* and *SOR* from the Server Efficiency Rating Tool (SERT) to profile our server under investigation. A detailed description of the used workloads can be found in the SERT design documents available for public review [11]. The profiling of each target level including warmup lasted around two and a half minutes. The framework collected around 60 power and system metric measurement samples per target level. The full profiling took approximately 38 hours.

3.2 Considered Power Models

We collected power models based on system metrics from literature. Table 1 contains an overview of the considered power models. The models range from simple linear regression models (1), only parametrized by CPU utilization, to

[3] https://sdqweb.ipd.kit.edu/wiki/Power_Consumption_Profiler

Table 1: Overview of considered power models

No.	Power Model	Considered Metrics
1	$P = c_0 + \sum_{m \in M} c_m u_m$	OS-level performance counters [3, 5, 17, 18], or only CPU utilization [19, 20]
2	$P = c_0 + \sum_{m \in M} (\sum_{l=1}^{l_{max}} c_l {u_m}^l)$	OS-level performance counters [18], or only CPU utilization [20]
3	$P = c_0 + \sum_{m \in M} \sum_{l=1}^{l_{max}} (e^{u_m} + c_l {u_m}^l)$	OS-level performance counters [18]
4	$P = c_1 \cdot e^{-(\frac{u_{\text{cpu}} - c_2}{\alpha_1})^2}$	CPU utilization [20]
5	$P = c_0 + c_1 u_{\text{cpu}} + c_2 u_{\text{cpu}}^{\alpha}$	CPU utilization [17, 19]
6	$P = c_0 + c_1 u_{\text{cpu}}^{\alpha}$	CPU utilization

multi-factorial models with exponential components (3, 4). As explained in Section 2.2 we extracted the power models using robust non-linear regression.

Previous work [17, 18] has shown that the prediction accuracy of system metric based power models can be increased by considering additional metrics. To evaluate the impact of metric selection on prediction accuracy we instantiated each of the multi-metric power models 1, 2, and 3 with CPU and storage metrics. Models 2 and 3 contain a complexity parameter l that defines the polynomial degree of the function. We instantiate 2 and 3 for values of $l = \{1, 2, 3\}$.

3.3 Prediction Accuracy of Power Models

To investigate whether our profiling approach produces server profiles that are suited for training power models, we used it to train the power models described in the previous section. If the models produced by the robust regression are accurate, we can deduce that our approach produces server profiles representative of the power consumption of the system under investigation.

We used a diverse set of workloads from the HiBench benchmarking suite [9] and SPECjbb2015 [8] to evaluate the prediction error of the power models. From the considered workloads, *K-means*, *TeraSort*, *DFSIOe*, *Page Rank* and *Nutch Indexing* were I/O intensive. All other benchmarks mostly stressed the CPU, or no resources at all in the case of *Sleep*.

Surprisingly, the models had a smaller prediction error when trained via measurements from separate profiling. For the considered workloads and power models, the results thus negatively answer EQ2. One potential reason for this is the large number of measurements with high utilization for multiple metrics from simultaneous profiling. The used regression approach minimizes the prediction error for the training set. However, the application workloads considered in the evaluation rarely stress CPU or HDD at the same point in time.

To assess the total accuracy of the models learned with our approach, we calculated the Mean Absolute Error for each workload. Overall, robust regression was able to train all types of power models to reach low prediction errors. Power models of type 1 with $M = \{u_{\text{cpu}}\}$, 5 and 6 had a median prediction error below 2.3%. Models of types 1, 3 for $l = 1$, and 4 suffered from poor prediction accuracies for utilization levels close to idle as observed for the *Sleep* workload.

Aside from *Sleep*, all power models achieved an error of at most 5.9% across all other workloads. The power model of type 3 with $l = 2$ and $M = \{u_{\text{cpu}}, u_{\text{read}}, u_{\text{write}}\}$ reached a maximum error of 4.7%. The power model 5 meets this maximum error. In total, 19 of 27 considered power models have a maximum prediction error of 5.9% across all workloads. From this we conclude that our approach produces representative server profiles that are well suited for training power models with high accuracy (EQ1).

3.4 Comparison of Profiling Approach with State of the Art

To evaluate the benefit of our profiling approach we compared it to state of the art profiling approaches. We replicate the behavior of state of the art approaches [3, 4] by monitoring the execution of SERT. As the SERT workloads individually stress the hardware components this matches the measurement procedure of state of the art approaches. We conducted a SERT run and collected measurements using the tooling described in Section 3.1.

The passive monitoring of SERT very rarely stressed storage to write more than 20 MB/s. Our profiling approach managed to reach write throughputs of up to 150 MB/s. This shows that the state of the art approach did not cover high write throughputs. Thus, the regression models trained on the resulting profile need to extrapolate for high write throughputs.

The power models built solely upon CPU utilization had high accuracy when trained using the profile from the SERT run. However, the models that consider both CPU utilization and storage throughput were significantly less accurate. Models 2 and 3 with $M = \{u_{\text{cpu}}, u_{\text{read}}, u_{\text{write}}\}$ deviated from the measured value by a factor of up to 70 for I/O-intensive workloads.

In conclusion, the profile obtained from monitoring SERT via a state of the art profiling approach can not be used to train multi metric power models. As our approach enabled us to train multi metric power models this confirms EQ3.

3.5 Impact of Metric Selection on Prediction Accuracy

We evaluated the impact of metric selection on prediction accuracy using the AIC-based ranking approach outlined in Section 2.3. Our intent was to evaluate whether the AIC-based ranking based on our server profile correctly predicted the effect of metric selection on prediction accuracy. For this, we compared the ranking with the prediction error of power models for our evaluation workloads. The ranking based on Δ_{AIC} indicated that the CPU-only models 6 followed by 5 had the highest likelihood of having the best prediction accuracy. Model 3 with $l = 3$ and $M = \{u_{\text{cpu}}, u_{\text{read}}, u_{\text{write}}\}$ followed third as the highest-placing model that considered storage metrics.

Since models 5 and 6 parametrized by both only CPU utilization outperformed all other models, we can deduce that considering storage metrics does not increase the prediction accuracy of trained power models for the models from Table 1. This was confirmed by the evaluation of error rates for the workloads outlined in Section 3.3. In the accuracy evaluation, model 6 had the lowest median prediction error. Considering storage write throughput did not reduce the average prediction error using our set of considered power models.

In conclusion, we were able to correctly predict the effect of considering additional metrics using the Δ_{AIC}-based ranking (EQ4). This indicates that the ranking is suited to the selection of a power model for consecutive use in design time predictions.

3.6 Threats to Validity

We conducted both profiling and measurements in a virtualized execution environment. This induces an overhead on the execution of both CPU and storage operations. We opted to perform the experiments in a virtualized environment as these environments are today's norm in the enterprise space. Benchmarks like SPECvirt [21] specifically target energy efficiency for virtualized environments. As with all models, power model abstract from system characteristics that can impact the power consumption. Examples for such system characteristics observed by Mccullough [18] are "hidden device states" and "significant variability" in power consumption of "identical components". Our approach does not consider these effects. Consequently, we can not quantify their significance to our findings.

Since we evaluated our approach for one specific server it cannot be guaranteed that our approach works for all server environments.

4 Related Work

Dayarathna et al. [14] provide an extensive overview of different power modeling techniques. The models covered by the survey range from manually created models to models trained using machine learning techniques. The following discusses a set of referenced modeling approaches that automate the creation or parametrization of their models.

Davis et al. [4] propose a methodology for automatically deriving power models based on OS-level performance counters. Their approach uses feature selection to identify the performance counters that strongly correlate with power consumption. Davis et al. use piecewise-defined regression power models. The profiling approach presented by the authors does not systematically vary load. Instead, it passively monitors the execution of a set of workloads to extract the measurement data needed to train the models. Davis et al. state that the workloads cover different load intensities and workload types which stress CPU, storage and network. However, their approach does not guarantee that measurements are collected for all relevant system metric levels and combinations.

Economou et al. [3] propose a profiling approach that individually stresses the hardware components of a server. Unlike our approach, it does not use hybrid workloads. Consequently, it does not support the investigation of interactions between multiple workloads on the measured system metrics. Section 3.4 had evaluated our approach against a profiling approach that replicated the behavior of the approaches by Davis et al. [4] and Economou et al. [3].

The PowerPack framework by Ge et al. [22] aims at profiling power consumption of distributed parallel applications. Like our work, Ge et al. investigate the effects of parallel job configurations on power consumption. In contrast to our work, PowerPack does not extract power models. Rather, it focuses on comparing the power consumption of the job configurations via measurements.

The Server Efficiency Rating Tool (SERT) [10] rates the energy efficiency of servers. It uses a set of workloads to stress the server under investigation. SERT varies the transaction rate of the workloads in order to assess the energy efficiency of the server at different load levels. Unlike our approach, SERT does not vary the workload to target system metric levels. SERT uses a hybrid workload based on the SSJ simulation library to assess efficiency for a mixed transactional workload. However, it does not assess the energy efficiency of workload combinations. This differs from our approach which simultaneously steers multiple workloads to reach target metric levels.

5 Conclusion

This paper presented an approach for the automated creation of power models using systematic experimentation. We outlined a methodology for deriving representative server profiles for training power models. Our approach allows for the creation of workload combinations from existing workloads. We presented an adaptive workload calibration policy that allows targeting system metric levels for combined workloads. We automatically parametrize a set of power models using the server profile produced by our profiling approach by means of robust nonlinear regression. To reason on the effect of considering additional system metrics on the power consumption prediction accuracy we rank the power models based on their Akaike's Information Criterion (AIC).

The evaluation investigated the applicability and accuracy of our approach by predicting the power consumption of a virtualized server system for a set of twelve benchmark applications, including the HiBench benchmarking suite [9] version 5.0 and SPECjbb2015 [8]. The evaluation showed that the power models parametrized by our approach accurately predicted the power consumption across all twelve applications (EQ1). We showed that separate profiling of CPU and HDD was sufficient to train accurate power models (EQ2). We compared a state of the art approach with our approach to determine whether our approach produced more representative server profiles for training power models (EQ3). The profile produced by our profiling was more representative of the system's power consumption. When we trained the power models based on the profiles collected using a state of the art approach, the model predictions deviated from the measured value by a factor of up to 70.

A comparison of our AIC-based ranking showed that we were able to estimate the effect of considering additional metric on prediction accuracy (EQ4). The most consistently accurate power model's prediction error ranged from 0.1% to 5.9%. Our AIC-based ranking had predicted this power model to have the highest likelihood of a high prediction accuracy. Four out of the six Pareto optimal power models from the evaluation had placed the highest in the ranking.

Our approach enables both software engineers to derive accurate power models of servers for design time predictions based on system metrics. It supports the combination of multiple workloads to create mixed system workloads. This enables engineers and operators to profile a server with workloads that more realistically match the behavior when hosting multiple collocated applications.

Our approach automates parametrization and ranking of power models. This reduces the effort for identifying a suitable power model for a given deployment environment. Engineers can choose from an extensible set of power models based on the system metrics they can predict. We ease reasoning on the effects of considering additional system metrics in power consumption analysis by ranking power models based on their estimated prediction accuracy.

In future work we will investigate how we can reduce the time needed for a profiling run. We plan to adaptively reduce the number of required measurement runs during profiling, reducing the total time required to create accurate power models for a server. To reason on the effect of adaptive server management policies we plan to include power consumption profiling for server reconfigurations. Examples for such reconfigurations are server shutdowns and bootups, as well as Virtual Machine migrations.

Acknowledgments

The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant agreement 610711. This work was also supported by the German Research Foundation (DFG) as part of the Research Training Group GRK 2153: "Energy Status Data – Informatics Methods for its Collection, Analysis and Exploitation".

References

1. Barroso, L.A., Clidaras, J., and Hölzle, U.: The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. Morgan & Claypool Publishers (2013)
2. Greenberg, A., Hamilton, J., Maltz, D.A., and Patel, P.: The Cost of a Cloud: Research Problems in Data Center Networks. ACM SIGCOMM Computer Communication Review 39(1), 68–73 (2008)
3. Economou, D., Rivoire, S., Kozyrakis, C., and Ranganathan, P.: Full-System Power Analysis and Modeling for Server Environments. In: Workshop on Modeling Benchmarking and Simulation (MOBS) (2006)
4. Davis, J.D., Rivoire, S., Goldszmidt, M., and Ardestani, E.K.: CHAOS: Composable Highly Accurate OS-based Power Models. In: Proceedings of the 2012 IEEE International Symposium on Workload Characterization (IISWC), pp. 153–163, Washington, DC, USA (2012)

5. Brunnert, A., Wischer, K., and Krcmar, H.: Using Architecture-level Performance Models As Resource Profiles for Enterprise Applications. In: Proceedings of the 10th International ACM Sigsoft Conference on Quality of Software Architectures. QoSA '14, pp. 53–62. ACM, Marcq-en-Bareul, France (2014)
6. Stier, C., Koziolek, A., Groenda, H., and Reussner, R.: Model-Based Energy Efficiency Analysis of Software Architectures. In: Proceedings of the 9th European Conference on Software Architecture. LNCS, Springer, Heidelberg (2015)
7. Jagroep, E.A., Werf, J.M. van der, Brinkkemper, S., Procaccianti, G., Lago, P., Blom, L., and Vliet, R. van: Software Energy Profiling: Comparing Releases of a Software Product. In: Proceedings of the 38th International Conference on Software Engineering Companion. ICSE '16, pp. 523–532. ACM, Austin, Texas (2016)
8. SPECjbb2015 Benchmark Design Document. Tech. rep., Gainesville, VA, USA: Standard Performance Evaluation Corporation (SPEC) (2015)
9. Huang, S., Huang, J., Dai, J., Xie, T., and Huang, B.: The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. In: 2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW), pp. 41–51
10. Block, H., Arnold, J.A., Beckett, J., Sharma, S., Tricker, M.G., and Rogers, K.M.: Server Efficiency Rating Tool (SERT) 1.0.2: An Overview. In: Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering. ICPE '14, pp. 229–230. ACM, Dublin, Ireland (2014)
11. Server Efficiency Rating Tool (SERT) Design Document 1.1.1. Tech. rep., Gainesville, VA, USA: Standard Performance Evaluation Corporation (SPEC) (2016)
12. *ENERGY STAR Program Requirements for Computer Servers — Partner Commitments*. U.S. Environmental Protection Agency. www.energystar.gov/ia/partners/prod_development/revisions/downloads/computer_servers/Program_Requirements_V2.0.pdf.
13. Rousseeuw, P., Croux, C., Todorov, V., Ruckstuhl, A., Salibian-Barrera, M., Verbeke, T., Koller, M., and Maechler, M.: *robustbase: Basic Robust Statistics*. R package version 0.92-6. 2016. CRAN.R-project.org/package=robustbase.
14. Dayarathna, M., Wen, Y., and Fan, R.: Data Center Energy Consumption Modeling: A Survey. IEEE Communications Surveys and Tutorials 18(1), 732–794 (2016)
15. Burnham, K.P., and Anderson, D.R.: Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach. Springer-Verlag, New York (2002)
16. Stone, M.: An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. Journal of the Royal Statistical Society, Series B 39, 44–47 (1977)
17. Rivoire, S., Ranganathan, P., and Kozyrakis, C.: A Comparison of High-level Full-system Power Models. In: Proceedings of the 2008 Conference on Power Aware Computing and Systems. HotPower'08, pp. 3–3. USENIX Association
18. McCullough, J., Agarwal, Y., Chandrashekhar, J., Kuppuswamy, S., Snoeren, A.C., and Gupta, R.: Evaluating the Effectiveness of Model-Based Power Characterization. In: Proceedings of the USENIX Annual Technical Conference, Portland, OR (2011)
19. Fan, X., Weber, W.-D., and Barroso, L.A.: Power Provisioning for a Warehouse-sized Computer. SIGARCH Computer Architecture News 35(2), 13–23 (2007)
20. Zhang, X., Lu, J., and Qin, X.: BFEPM: Best Fit Energy Prediction Modeling Based on CPU Utilization. In: 2013 IEEE Eighth International Conference on Networking, Architecture and Storage (NAS), pp. 41–49 (2013)
21. SPECvirt_sc$^{®}$ 2013, Last retrieved 2016-08-26. Standard Performance Evaluation Corporation (SPEC). www.spec.org/virt_sc2013/
22. Ge, R., Feng, X., Song, S., Chang, H.C., Li, D., and Cameron, K.W.: PowerPack: Energy Profiling and Analysis of High-Performance Systems and Applications. IEEE Transactions on Parallel and Distributed Systems 21(5), 658–671 (2010)