Model-Based Energy Efficiency Analysis of Software Architectures

Christian Stier[1], Anne Koziolek[2], Henning Groenda[1], and Ralf Reussner[2]

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

Abstract. Design-time quality analysis of software architectures evaluates the impact of design decisions in quality dimensions such as performance. Architectural design decisions decisively impact the energy efficiency (EE) of software systems. Low EE not only results in higher operational cost due to power consumption. It indirectly necessitates additional capacity in the power distribution infrastructure of the target deployment environment. Methodologies that analyze EE of software systems are yet to reach an abstraction suited for architecture-level reasoning. This paper outlines a model-based approach for evaluating the EE of software architectures. First, we present a model that describes the central power consumption characteristics of a software system. We couple the model with an existing model-based performance prediction approach to evaluate the consumption characteristics of a software architecture in varying usage contexts. Several experiments show the accuracy of our architecture-level consumption predictions. Energy consumption predictions reach an error of less than 5.5% for stable and 3.7% for varying workloads. Finally, we present a round-trip design scenario that illustrates how the explicit consideration of EE supports software architects in making informed trade-off decisions between performance and EE.

1 Introduction

Software architects design enterprise software systems to meet quality requirements in multiple dimensions, e.g., performance and reliability. In designing a software system they have to make trade-off decisions to address contradictory goals of stakeholders. System users are interested in having sufficient Quality of Service (QoS) at an acceptable price. System providers aim to reduce the cost incurred from hosting the software system. Power consumption is a major operating cost factor. It is responsible for over 15% of a data center's Total Cost of Ownership (TCO) [9].

The power consumption of servers varies strongly depending on the *load* induced on the servers. If the utilization of a server increases (e.g. because more software components are deployed to it, or because more users use its services), its power consumption increases.

In addition to the amount of power consumed by the servers, the costs induced by power consumption also include the costs of power distribution equipment. The degree by which system operators consolidate load depends not only on QoS requirements but also on the available power distribution infrastructure. Power distribution infrastructure in data centers is organized in a hierarchical manner [1]. Power distribution units

(PDUs) distribute power to racks which in turn provide power to the connected servers. As servers rarely all simultaneously reach peak utilization, the power distribution infrastructure is usually over-subscribed [1]. Whether this over-subscription will lead to problems depends upon the workload mix and the deployed software components.

The software system's architecture impacts these power consumption costs: First, design decisions may influence the power consumption directly, for example the decision how to distribute the system to how many servers, or the decision what architecture-level communication style to use [19]. Second, to assess the costs of the expected power consumption, architects additionally need to consider what power distribution infrastructure is needed and when power is consumed. Thus, software architects need to consider the influence of architecture design on power consumption to make informed trade-off decisions between the QoS of offered services and power consumption costs.

Software engineering approaches are yet to reach an abstraction suitable for supporting the design of energy efficient software systems on an architectural level. Seo et al. perform energy consumption analysis for specific architectural styles [19]. Their work cannot be applied to predict the impact of other design decisions on EE. Brunnert et al. analyze the EE of software systems on the basis of average-case analysis [6]. This is not sufficient to determine the peak power consumption the deployed software causes on the infrastructure. Approaches from the embedded systems and cloud/grid computing domain make limiting assumptions on the behavior of users [12, 14] or disregard parametric dependencies between software components [7, 12].

In this paper, we propose an approach for analyzing the EE of software architectures. It accounts for variations in user load and allows to identify periods with high power consumption. Thereby it not only enables software architects to determine whether the planned deployment of a software system meets average-case QoS and energy consumption goals, but also whether the system violates QoS and consumption limits during temporary workload spikes.

The contributions of this paper are as follows. First, it outlines an approach for *modeling the power consumption* characteristics of software systems on the architectural design layer. Second, an *analysis methodology for energy efficiency*, i.e. as a trade-off between power consumption and performance, is presented. Our approach extends an architecture-level approach for performance prediction by power consumption predictions. It does not require software component-specific consumption annotations. Rather, it uses component-independent consumption characterizations of the deployment environment and component-specific performance annotations to reason on the consumption characteristics of a deployed software system.

In our evaluation, we investigated the accuracy of the power consumption predictions for a media hosting system (evaluation question Q1). The architectural power consumption predictions reached an average-case error of less than 5.5%. We evaluated whether the approach could accurately predict power consumption trends (Q2). Furthermore, we showcased the potential benefits of applying power consumption analysis as part of the architecture design by investigating a design decision (Q3) and a deployment decision (Q4) that affect both performance and power consumption in a nontrivial manner. Our approach predicted the absolute effect of using an alternative encoder with an error lower than 18.9% (Q3). A deployment decision scenario illustrated the effect

that power infrastructure sizing has on the degree by which a system can be horizontally scaled (Q4)).

The paper is structured as follows: Section 2 provides foundations on power models used for power consumption predictions and Palladio. Section 3 presents the state of research and highlights the gaps. Section 4 introduces the architecture of the running example. Section 5 presents our power consumption model and 6 our analysis methodology. Finally, Section 7 presents the evaluation results and Section 8 concludes.

2 Foundations

This section presents methodologies and concepts that our approach is built upon. Section 2.1 discusses models to estimate the power consumption of hardware components. Palladio's Architectural Description Language (ADL) and performance prediction approach are presented in Section 2.2.

2.1 Power Models

Power models estimate the power consumption of hardware components or sets of hardware components. Power models correlate power consumption with measurable metrics. They are constructed based on power measurements, which are collected as part of a benchmark. A wide variety of power models exists. They range from models that rely on system-level metrics [8, 11], i.e., CPU utilization, to models that consider performance counters [5] and other hardware internals [10].

Fan et al. [8] propose and evaluate a linear as well as a non-linear regression-based power model for predicting the power consumption of single servers. Their models correlate power consumption solely with CPU utilization. Non-linear power models exceed the accuracy of linear power model across a wide range of different workloads, as is confirmed by Rivoire et al. [18]. Nevertheless, existing power consumption predictions most commonly build upon linear power models [6, 7, 17].

2.2 Palladio

Palladio [4] supports the analysis of quality characteristics of a component-based software architecture. Software architectures are specified in the *Palladio Component Model (PCM)* ADL. Palladio evaluates the performance characteristics of an architecture either using analytical approaches or Discrete Event Simulation (DES) [4].

PCM is a meta-model for modeling component-based software architectures and the factors that influence the performance of a software system. PCM consists of different views that abstract distinct modeling concerns. The central view of PCM is *Component Specification*. In the Component Specification view the interfaces and behavior of services offered by each component are described. The behavior of a service is specified in a *Resource Demanding Service Effect Specification (RDSEFF)*. The RDSEFF correlates the input parameters of a service call with the demands it issues on resources such as CPU or HDD. Furthermore, the RDSEFF specifies parametric dependencies of the service's performance on the performance of service calls to its required components. The

Assembly Model view defines an assembly of component instances. Figure 1b provides an example Assembly and RDSEFF instance. Besides the description of components and their assembly, the views include the definition of the system's deployment environment and usage context. The *Resource Environment* view describes the deployment environment of components. It describes the performance characteristics of the available compute and network infrastructure. The *Allocation* view maps the components in the assembly to the deployment environment. PCM's *Usage Model* separately models the behavior and arrival rate of users that interact with the system.

The advantage of PCM's performance abstraction over the direct specification of performance models using formalisms such as Queueing Networks (QNs) lies in its composability. The performance model of each component is defined as a set of RDSEFFs of its provided services. The RDSEFFs of a component are parametrized over the component's assembly, deployment and usage, so that it can be reused for different contexts in which the component itself shall be reused (e.g. in a different system, a different deployment, or a different usage profile). Figure 1b shows an example RDSEFF parametrized over the size of the input values in bytes. Once the assembly, deployment and usage of the components in the architecture have been specified, the Palladio tool composes the performance models and analyzes it via simulative or analytical approaches from the domain of queueing theory [4].

3 Related Work

Over the years many design patterns [16] have been proposed to increase the EE of software systems. Little work, however, has been done on quantitatively evaluating the EE of software systems on an architectural level at design time.

Seo et al. [19] evaluate the impact of architectural communication styles on energy consumption. The authors outline consumption models for specific communication styles such as client-server and publish-subscribe. Their approach disregards power consumption resulting from computation as the authors argue that application behavior is independent from the communication style. While the approach proposed by Seo et al. can be applied to compare the power consumption of specific communication styles, it consequently cannot be leveraged to evaluate the overall energy consumption of a software system.

Meedenyia et al. [14] propose a multi-objective architecture optimization approach for embedded component-based systems. The authors focus on the trade-off between energy consumption and reliability. Their approach annotates each component with an estimate of the energy consumption incurred by calling one of its services. Meedeniya et al. assume that all calls to a component consume the same amount of energy. The authors do not differentiate energy consumption for different input parameters. This limits the applicability of their approach to enterprise software systems where the resource demand of services largely depend upon its input parameters.

Brunnert et al. [6] capture performance and power consumption characteristics of software systems for systematic capacity planning. Their approach predicts energy consumption using linear power models [8]. Non-linear power models are not supported. The authors evaluate energy consumption via an average-case system analysis. Their

approach does not support identifying peaks in power consumption and the violation of consumption constraints.

The previously discussed approaches [6, 14, 19] focus on EE analysis of software systems on an architectural level. A number of consumption modeling and prediction approaches have been developed to evaluate the power consumption of cloud and High Performance Computing (HPC) systems.

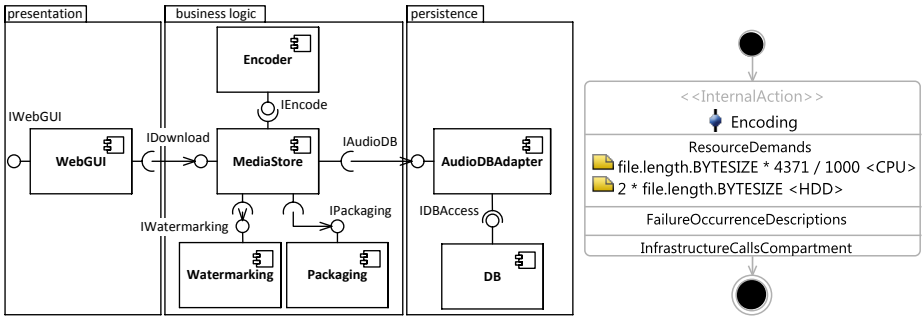
The cloud data center simulator *CloudSim* developed by Calheiros et al. [7] supports the prediction of power consumption in data centers using power models. Virtual Machines (VMs) and not software components form the central first-level entities. CloudSim does not consider dependencies in the behavior of VMs. The resource demand of each VM is described as a fixed function over time. *DCWorms* by Kurowski et al. [12] is a simulator aimed at performance and energy consumption predictions for HPC systems. Their model assumes that all computational tasks have predetermined durations. This assumption is often violated for systems outside of the HPC domain.

Basmadjian et al. [2] outline a power consumption estimation methodology for servers. Their approach estimates a data center's power consumption by aggregating the consumption of individual resources such as CPUs and fans. Since the authors assume all resources of one type to follow the same power model, their approach is unsuited for evaluating the energy consumption of heterogeneous computer systems.

4 Running Example

Media Store is a reference implementation of a light-weight media hosting service. Users can download and upload media files using its services. Media Store has been used to empirically validate Palladio's applicability for design-time performance predictions [13]. The accuracy of Palladio's performance predictions has also been evaluated for a specification of Media Store in PCM [4].

Figure 1a shows the Assembly view on Media Store. The assembly consists of a GUI frontend (*WebGUI*), a business logic layer for organizing users and media files (*MediaStore*), and a persistence layer. For the sake of simplicity, we deploy all three Media Store layers on one high-capacity server.



(a) MediaStore architecture (b) Encoding Service RDSEFF

Fig. 1: Media Store architecture and RDSEFF of Encoding service offered by Encoder

The *MediaStore* component offers download and upload services. It re-encodes media files to a specific bit rate using the Encoder component. The RDSEFF characterization of the Encoder's download service is depicted in Figure 1b. It correlates the file's input size with the number of CPU cycles needed to process one byte.

5 Power Consumption Model

In order to reason on a software system's EE at the architecture level, it is necessary to describe its consumption characteristics on a suitable abstraction level. ADLs like PCM enable the design-time analysis of QoS characteristics such as performance [4]. Quality analysis approaches built upon ADLs rely on a characterization of fundamental factors that impact the system's QoS in the set of considered quality dimensions. Existing ADLs that describe the power consumption characteristics abstract from fundamental characteristics of power distribution infrastructure design [6]. Although more comprehensive abstractions of power distribution infrastructure have been proposed [2, 15], they have thus far not been integrated with an architecture-level approach for the design of software systems.

This section outlines our proposed modeling of a software system's power consumption characteristics as part of an ADL. Even though this paper applies the modeling concepts to PCM, the chosen modeling abstraction is independent of it. A detailed description of the models and the integration with PCM is available in [20].

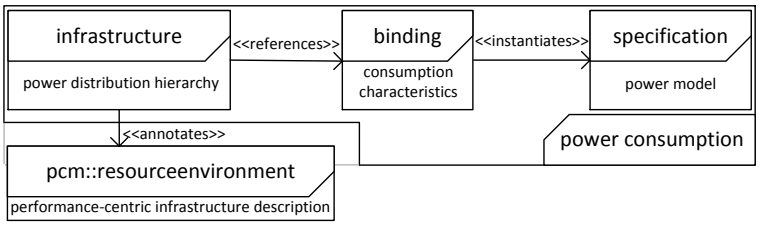
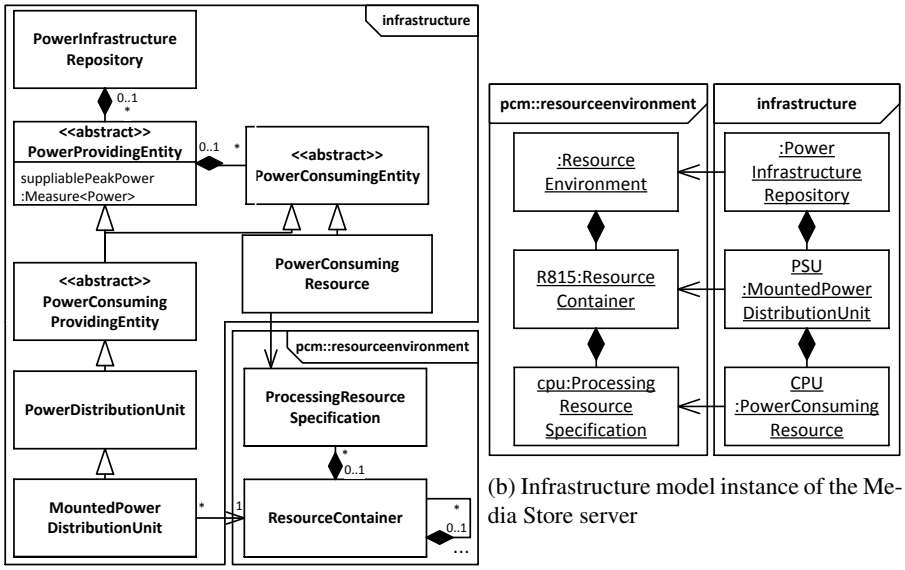


Fig. 2: Overview of the Power Consumption model

Figure 2 provides an overview of our proposed model of power consumption characteristics as an extension to the PCM ADL. The *Power Consumption* model annotates PCM's Resource Environment model with the consumption characteristics of the software system. The Power Consumption model is subdivided into three orthogonal model views. These views are presented in the following sections.

5.1 Infrastructure

The *Infrastructure* view describes the power distribution infrastructure of a software system. Its abstraction is based on the power distribution infrastructure of data centers and enterprise-scale software systems. Power distribution infrastructure in data centers is structured hierarchically [8]. For example, data center-level uninterruptible power supplies (UPSs) distribute power to PDUs mounted to racks. UPSs provide backup



(a) Infrastructure meta-model

(b) Infrastructure model instance of the Media Store server

Fig. 3: Infrastructure meta-model and example instance of the Media Store server

power and correct irregularities such as voltage spikes. The rack-mounted PDUs then supply the power supply units (PSUs) of individual servers with power.

Figure 3a depicts the Infrastructure's meta-model. *PowerInfrastructureRepository* hosts a set of power distribution infrastructure definitions. The power in each distribution infrastructure originates from a common *PowerProvidingEntity*, e.g. the PDU of a group of racks. The PowerProvidingEntity supplies power to a set of connected *PowerConsumingEntities*. PowerConsumingEntities are distinguished into entities that consume power, e.g. *PowerConsumingResource*, and entities which both consume and provide power (*PowerConsumingProvidingEntity*). This modeling allows to capture the conversion losses that may incur for UPSs. PowerConsumingResource annotates the processing resources in PCM's Resource Environment and puts them into context with the power distribution infrastructure.

Figure 3b shows the Infrastructure instance of our running Media Store example. It enhances PCM's performance-centric Resource Environment model with power consumption and distribution characteristics. The server onto which the Media Store application is deployed hosts a set of CPUs. The CPUs all contribute to the power consumption of the server. Thus, they are modeled as a PowerConsumingResource. All CPUs draw their power from the same PSU.

5.2 Specification

The *Specification* view allows to capture power models as introduced in Section 2. Power models are defined in terms of their input parameters. We opted against including the evaluation semantics of power models in the Specification meta-model in order

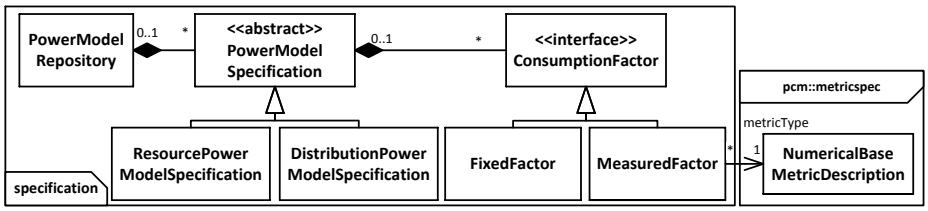


Fig. 4: Specification meta-model used for defining power models

to support the inclusion of power models of arbitrary computational complexity. The evaluation semantics of each power model are specified as part of extensible plugins to our analysis tool.

Figure 4 depicts the Specification meta-model. Software architects can use a common *PowerModelRepository* instance to manage recurring types of power models. Every PowerModelSpecification represents a power model. The model distinguishes between power models of infrastructure elements that distribute power (*DistributionPowerModelSpecification*) and models of resources (*ResourcePowerModelSpecification*). Power consumption of resources results from their utilization as part of a task. For example, a CPU draws power when it performs mathematical operations. A PDU's power consumption depends solely on the power draw of connected resources.

ConsumptionFactors form the input parameters of each power model. An instance of ConsumptionFactor specifies an input parameter type and not its concrete value. Type and value definition are separated to enable reuse of the power model specifications. Section 5.3 explains how the types are instantiated. A *FixedFactor* represents a fixed consumption characteristic that does not change with the load of a PDU or resource. The power consumption of a CPU under idle load (P_{idle}) falls into this category. A *MeasuredFactor* expresses a dependency of the power consumption to a measured system metric such as CPU utilization. All MeasuredFactors come with semantic information on their type and the unit in which they are measured (*NumericalBaseMetricDescription*). In the context of this paper, MeasuredFactors are gauged on the basis of metrics extracted from simulation. However, real measurements could also be a source of these measurements if the Power Consumption model were to be applied to runtime power consumption evaluations. The right hand side of Figure 5 provides an example on how a linear power model can be defined using the Specification model.

5.3 Binding

PowerBinding links the power consumption characteristics of elements in the Infrastructure with Specification's abstract, type-level definition of power models. A PowerBinding instance specifies how a specific resource type consumes power. Figure 5 shows an example Binding for the Media Store server. The PowerBinding ties the server to the linear power model. The server's consumption is characterized to follow a linear power model with $P_{Idle} = 332$ W and $P_{Busy} = 477$ W. Every FixedFactor in the Specification is matched with a *FixedFactorValue*. The PowerBinding of the server binds the values 332 W and 477 W to P_{idle} and P_{Busy}. The utilization parameter u of the linear power model does not need to explicitly get instantiated in the Binding view. It can

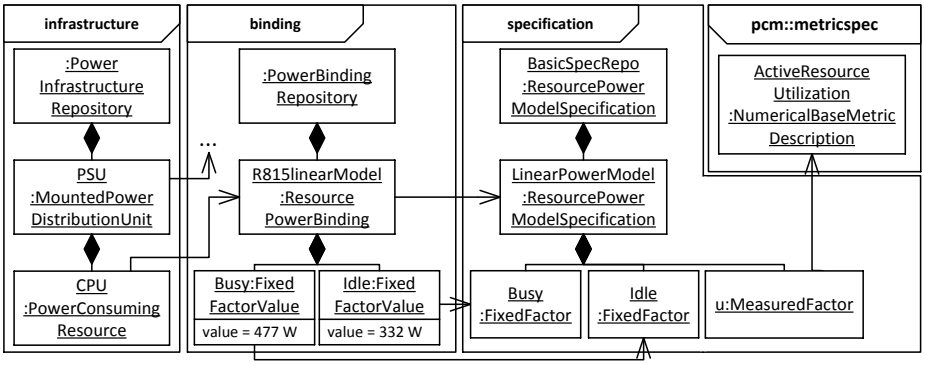


Fig. 5: Excerpt from Specification and Binding instance of the Media Store server

be implicitly resolved for all servers in the Infrastructure since it does not depend on server-specific consumption characteristics.

6 Evaluating Energy and Power Consumption

In the following, we present our approach for evaluating power and energy consumption of a software system. Section 2.1 outlined that the power consumption of individual servers can be estimated using power models. In order to apply power models to estimate the consumption of a software system, a source for system metrics it depends upon is needed.

System metrics can be extracted from software architectures described in an ADL using analytical or simulation-based approaches. Analytical approaches offer fast evaluation times with a good average-case accuracy. They do, however, make simplifying assumptions to achieve acceptable computational complexity. Analytical approaches commonly disregard temporal effects in user load, such as strong bursts during a specific interval. As we are interested in an analysis that accounts for these effects, we opted to employ Discrete Event Simulation (DES) as the basis of the power consumption predictions presented in this paper. Palladio's DES provides utilization metrics for a specified software system and usage profile. We use these utilization metrics as input for our post-simulation analysis. The implementation is available on our website[1].

Our analysis evaluates the power consumption of the resources in the power distribution infrastructure. It calculates power consumption using the consumption characteristics specified in the Binding view. Subsequently, the analysis aggregates power consumption for the elements provisioning power to the resources. As part of the aggregation, conversion losses can be factored in.

The energy consumption E_s of a software system s between two points in time a and b is calculated by integrating its power consumption P_s over the investigated interval $[a,b]$. Once power consumption predictions are available for a software system, we evaluate energy consumption by means of numerical integration.

[1] https://sdqweb.ipd.kit.edu/wiki/Power_Consumption_Analyzer

7 Evaluation

We evaluated our approach for architecture-level power consumption predictions using the Media Store system described in Section 4. The evaluation investigated the following evaluation questions:

- **Q1:** Is our approach suited to accurately predict energy consumption on an architectural level?
- **Q2:** Does the approach support the identification of power consumption trends and peaks under varying workloads?
- **Q3:** Is the approach applicable to evaluate the energy efficiency (EE) of architectural design decisions?
- **Q4:** Is the approach applicable to evaluate the impact of deployment decisions on EE?

7.1 Deployment Environment

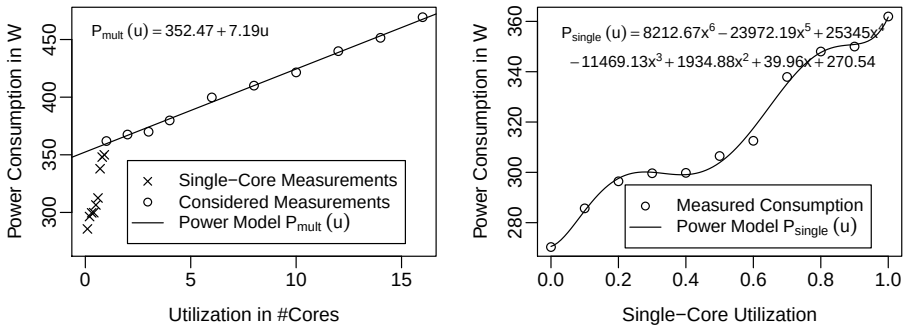
We conducted our evaluation on a Dell PowerEdge R815 server equipped with four Opteron 6174 CPUs. The Media Store implementation used as part of this case study was realized in Java EE. We deployed Media Store on a Glassfish 3.1 server running on an Ubuntu 12.04 VM. The VM was running atop a Xen hypervisor controlled by XenServer 6.2. 16 out of the server's 48 available cores were assigned to the VM. No other VMs besides the Ubuntu VM and the XenServer instance were deployed onto the server. Media Store's *DB* component was realized as a MySQL 5.5 server instance, *Encoder* used LAME 3.99.3 for encoding MP3s. The Vorbis *Encoder* variant introduced in Section 7.4 used *libvorbis* 1.3.2 bundled within the *ffmpeg* framework.

7.2 Power Model Extraction

In order to reason on power consumption we measured the server's power consumption in relation to its load. We derived a power model from the measurements by correlating power consumption and load. The Linux microbenchmarks *stress* and *lookbusy* were used to put varying degrees of load on the server. For each load level a monitoring utility measured power consumption using the server's built-in power meter. The utility collected the measurements through Intelligent Platform Management Interface (IPMI)[1]. While the power meter's accuracy and resolution is limited when compared to a dedicated external meter, it is sufficient to derive a full-system power model. The server was operated in a maximum performance mode for all the experiments to exclude side-effects induced by switching server components into low power states.

First, we investigated how the number of busy cores affected the power consumption of the server. The dots in Figure 6a represent the average-case power consumption when 0 to 16 cores were stressed. The linear function $P_{mult}(u)$ models the relation between power consumption and the number of used cores with an R-squared error of 0.996 for 1 to 16 busy cores (c.f. Figure 6a).

[1] http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-home.html, retrieved 19.12.2014



(a) Power measurements with model built on the basis of multi-core measurements

(b) Single-core measurements with polynomial power model

Fig. 6: Power measurements and models based on CPU utilization of the R815 server

As the idle power consumption significantly deviated from the overall trend we further investigated the power consumption for utilization increments of 10% between zero and one utilized core. Figure 6b shows the power model $P_{single}(u)$ for utilization in this range. Putting both power models together, we derived a piecewise-defined function for the whole utilization domain:

$$P_{full}(u) = \begin{cases} P_{single}(u) & \text{if } u \leq 1 \\ P_{mult}(u) & \text{if } 1 < u \leq 16 \end{cases}, \; u \in [0,16].$$

Finally, we modeled this function using our Power Consumption model.

7.3 Accuracy of Architecture-Level Consumption Predictions

A power model can only be as accurate as the system metrics with which it is parametrized. In order to warrant sufficient accuracy of the CPU utilization, we calibrated the Media Store performance model using end-to-end service measurements extracted via *perf4j*. We employed a single-user workload to calibrate the performance model.

As expected, performance predictions for the calibration scenario match the performance measurements from the real system. The average response time (RT) prediction error of the calibrated model was 0.05%.

Our analysis derives power consumption predictions from the system metrics that are produced by the simulation, as is discussed in section 6. Due to the low RT prediction error of 0.05% the calibration scenario is well-suited as a benchmark for evaluating the accuracy of the energy consumption predictions. We measured the power consumption of our Media Store server using the built-in power meter while it was processing W_1. Applying the power model to our calibration scenario, we achieved an error of 0.17% for the total energy consumption in the single-user scenario (workload W_1).

Subsequently, we evaluated the accuracy of energy consumption predictions under workloads different from the calibration scenario. In the evaluation workload W_2 16 users repeatedly downloaded a random song from Media Store. The model we had calibrated with a single-user workload managed to predict the RT under the increased

load with an error of 2.31%. Energy consumption predictions under the increased load reached an error of less than 5.5% when compared to actual measurements.

Finally, we investigated the accuracy gained from using a piecewise-defined power model P_{full} over strictly linear models as applied by Brunnert et al. [6]. If we were to use a linear power model for the full domain instead of the piecewise-defined P_{mult} the energy consumption prediction error would go up to 1.41% for W_1, and 7.65% for W_2.

In summary, our approach produced accurate energy consumption predictions with an error of less than 5.5%. This indicates that the approach has suitable accuracy (Q1).

7.4 Consumption Peaks and Trends

Besides an average-case energy consumption analysis our approach also supports the evaluation of power consumption under changing user load. This enables software architects to assess the suitability and efficiency of the software system's power distribution infrastructure under changing load.

Instead of the previously examined closed workloads W_1 and W_2 we now compared power consumption measurements with the predictions for a gradually increasing open workload. In W_3 initially no users submitted download requests onto Media Store. For every 160 seconds that had passed, the users' interarrival rate was increased by an additional user request per 16 seconds. Since Palladio's baseline simulative analysis [4] does not support the evaluation of dynamic workloads we used the alternative SimuLizar [3].

The energy consumption prediction error for W_3 amounted to 3.68%. Figure 7 depicts both measured and predicted power consumption for W_3. It can be seen that measured and predicted consumption values differ from each other. However, our prediction managed to identify the power consumption trend with reasonable accuracy (Q2).

7.5 Impact of Design Decisions on Energy Consumption

Encoding is the most resource-intensive service in the Media Store architecture. We thus investigated whether we could save by exchanging the LAME MP3 encoder with a Vorbis encoder (*libvorbis*) using comparable audio quality settings. First, we estimated the resource demand caused by encoding a music file based on measurements for the

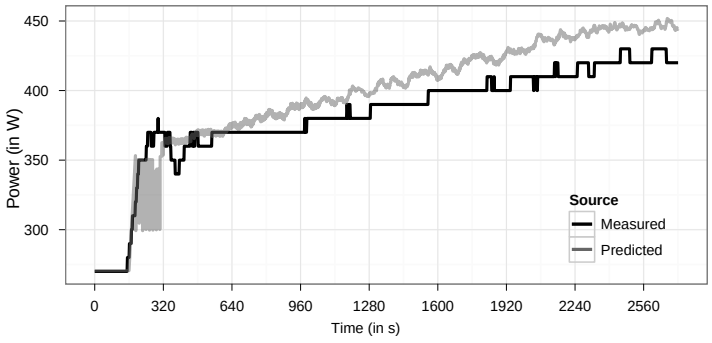


Fig. 7: Power consumption under incrementally increasing load

Table 1: Predicted and measured power consumption for LAME and libvorbis encoder

(a) Workload W_4 with interarrival time of 16s

	Energy Consumption		
Encoder	Measured	Predicted	Error
LAME	173.77 Wh	171.00 Wh	-1.60%
libvorbis	129.11 Wh	133.10 Wh	+2.78%
Saved Energy	44.67 Wh	37.91 Wh	-15.14%

(b) Workload W_5 with interarrival time of 1s

	Energy Consumption		
Encoder	Measured	Predicted	Error
LAME	215.30 Wh	223.06 Wh	+3.60%
libvorbis	195.05 Wh	198.97 Wh	+2.01%
Saved Energy	20.25 Wh	24.09 Wh	+18.94%

Vorbis encoder. These measurements were conducted in isolation of the original calibration. Second, we modeled the Vorbis encoder in Palladio and integrated it into Media Store's architectural model. We then predicted the effect of using the Vorbis instead of the MP3 encoder using Palladio's simulation in combination with our consumption analysis. Finally, we compared our predictions to actual measurements.

First, we investigated energy consumption under the open workload W_4 with an interarrival time of 16 seconds that was run for 30 minutes. Table 1a compares the energy consumption of the MP3 Encoder component implementation with the Vorbis Encoder. The total predicted energy consumption error was less than 2.8%. To estimate the effect of using an alternative Encoder component on the EE of the Media Store architecture we determined the saved energy as the difference between the predicted consumption for both architecture variants. Our approach was able to predict the saved energy consumption with an error of 15.14% (c.f. Table 1a, Saved Energy row).

Table 1b depicts predicted and measured energy consumption under workload W_5. W_5 is an open workload with an interarrival time of 1 second. Prediction errors of both encoder variants amounted to less than 3.6%. The gained EE was predicted with an error of 18.94% (c.f. Table 1b, Saved Energy row). Even though the error of the predicted saved energy was noticeable, the saved energy could still be assessed both qualitatively and quantitatively. The results thus positively answer Q3.

7.6 Architectural Sizing Decisions

Multi-user load put onto Media Store so far could be handled while maintaining QoS close to a single-user scenario by hosting all components on a single server. When load rises, the simple single-node deployment of Media Store becomes infeasible.

Workload W_6 varies between 5 and 16 users per second. When they are put onto the Media Store system, the system is overloaded. In order to achieve performance close to the single-user scenario under W_6, the software architect considers to scale the architecture horizontally and use storage with higher performance. He or she realizes the horizontal scaling by adding a load balancer to Media Store's architecture that distributes the encoding tasks across multiple *Encoder* instances. Now, a possible solution is to balance load between ten *Encoder* components deployed on ten servers. However, the predicted worst-case RT of this solution is more than twice as high as the worst-case RT in the single user scenario W_1. Thus, the software architect might consider to increase

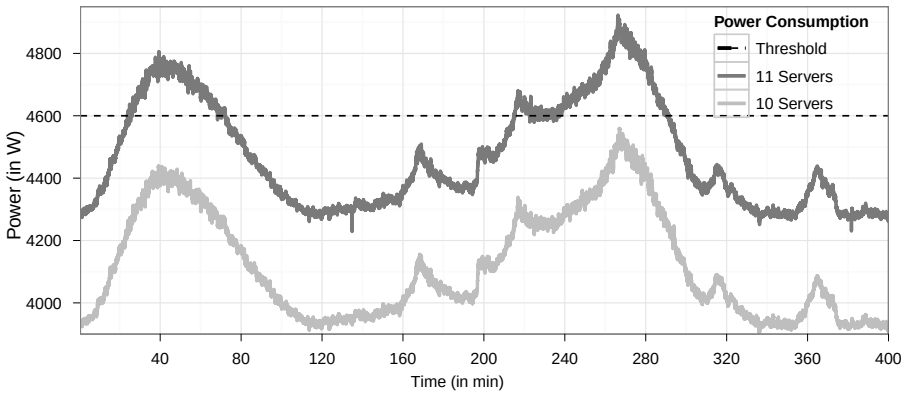


Fig. 8: Aggregate consumption of scaled-out Media Store

the number of *Encoder* instances and servers to eleven. This improves the worst-case RTs to only 23.3% more than in the single user case.

Software architects may consider the cost of adding a new server as a flat fee when evaluating the trade-off between gained QoS and cost. This does not, however, account for the step fixed cost induced by surpassing certain power consumption thresholds. Power distribution infrastructure such as PDUs and UPSs can supply power up to a certain peak power threshold. Once the peak power threshold is surpassed the infrastructure either breaks down or the connected servers are forced into lower power states. Both scenarios result in inferior QoS and should therefore be avoided.

Figure 8 shows that the critical peak power of a UPS supplying 4600 W would be surpassed if eleven *Encoder* instances were deployed. Consequently, the software architect would have to request a larger-scale power infrastructure from the infrastructure operator. An infrastructure upgrade might be warranted if the architect expects a further increase in future load. Otherwise, the architect may favor balancing load between ten *Encoder* instances to avoid a significant additional investment. In summary, the scale-out scenario illustrates effects of power distribution infrastructure sizing decisions on deployment decisions (Q4).

7.7 Threats to Validity

Our evaluation focused on a specific application deployed on one specific server. It does not address whether the proposed energy efficiency analysis methodology applies to different software systems. The use of an internal power meter instead of a dedicated certified power meter potentially results in inaccuracies. Furthermore, the consumption trend analysis experiment was only conducted once. All other experiments were conducted in steady states in which we monitored the system for more than 25 minutes.

8 Conclusions

This paper presents a model-based approach that supports the analysis of energy efficiency (EE) of software architectures. The proposed hierarchical Power Consumption

model extends existing architecture models by central power consumption and distribution characteristics of targeted deployment environments. We extend an architecture-level performance analysis to predict the energy consumption of a modeled software system. The evaluation illustrates that our approach accurately predicts the power consumption of software systems in different usage contexts (Q1, Q2). The average energy consumption predictions from our approach reached errors below 5.5% for all experiments. The consumption predictions were accurate enough to assess how individual architectural design decisions affect EE for a specific system and usage context (Q3). We validated this by predicting the effect that the use of an alternative music encoder would have on EE. Our approach predicted the increase in efficiency with an error of less than 18.94%. A horizontal scaling scenario in which we replicated the encoder component illustrated how the explicit consideration of power consumption affects deployment decisions (Q4).

The approach proposed in this paper enables software architects to make systematic trade-offs between EE and other quality dimensions. Software architects do not need to rely solely on best practices and design patterns that have been shown to improve EE. Our approach allows to quantify the effect of design decisions on EE. It uses power models to evaluate the power consumption of software architectures. The power models are extracted via microbenchmarks, and characterize the consumption characteristics of resources independent of specific software architectures. This allows software architects to evaluate the EE of arbitrary software architectures without relying on architecture-specific consumption measurements.

In future work, we plan to extend our approach in three main areas. First, we will integrate our approach with the analysis of self-adaptive software systems to evaluate the effect of energy-conscious self-adaptation tactics (c.f. [16]) on the EE of a designed system. We further intend to include transient performance and energy cost with the analysis. Examples for such costs are the energy consumption and performance degradation caused by VM migration. Second, we aim to reduce the required effort for applying our approach by automating the extraction of power models. Finally, we plan to reduce the effort to identify optimal trade-offs between EE and other quality dimensions using approaches for automated design space exploration.

Acknowledgments

This work is funded by the European Union's Seventh Framework Programme under grant agreement 610711.

References

1. Barroso, L.A., Clidaras, J., Hölzle, U.: The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. Synthesis Lectures on Computer Architecture, Morgan & Claypool Publishers, 2 edn. (2013)
2. Basmadjian, R., Ali, N., Niedermeier, F., de Meer, H., Giuliani, G.: A Methodology to Predict the Power Consumption of Servers in Data Centres. In: e-Energy '11: Proc. of the 2nd International Conf. on Energy-Efficient Computing and Networking. pp. 1–10. ACM, New York, NY, USA (2011)

3. Becker, M., Becker, S., Meyer, J.: SimuLizar: Design-Time Modelling and Performance Analysis of Self-Adaptive Systems. In: Proc. of the Software Engineering Conf. (SE 2013) (Feb 2013)
4. Becker, S., Koziolek, H., Reussner, R.: The Palladio component model for model-driven performance prediction. Journal of Systems and Software 82(1), 3 – 22 (2009)
5. Bircher, W., John, L.: Complete System Power Estimation Using Processor Performance Events. IEEE Transactions on Computers 61(4), 563–577 (April 2012)
6. Brunnert, A., Wischer, K., Krcmar, H.: Using Architecture-level Performance Models As Resource Profiles for Enterprise Applications. In: Proc. of the 10th International ACM SIGSOFT Conf. on Quality of Software Architectures (QoSA 2014). pp. 53–62. ACM, New York, NY, USA (2014)
7. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Softw. Pract. Exper. 41(1), 23–50 (Jan 2011)
8. Fan, X., Weber, W.D., Barroso, L.A.: Power Provisioning for a Warehouse-sized Computer. SIGARCH Computer Architecture News 35(2), 13–23 (Jun 2007)
9. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The Cost of a Cloud: Research Problems in Data Center Networks. SIGCOMM Comput. Commun. Rev. 39(1), 68–73 (Dec 2008)
10. Isci, C., Martonosi, M.: Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data. In: Proc. of the 36th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, Washington, DC, USA (2003)
11. Kansal, A., Zhao, F., Liu, J., Kothari, N., Bhattacharya, A.A.: Virtual machine power metering and provisioning. In: Proc. of the 1st ACM Symposium on Cloud Computing. pp. 39–50. ACM, New York, NY, USA (2010)
12. Kurowski, K., Oleksiak, A., Piątek, W., Piontek, T., Przybyszewski, A., Węglarz, J.: DCworms – A tool for simulation of energy efficiency in distributed computing infrastructures. Simulation Modelling Practice and Theory 39, 135 – 151 (2013)
13. Martens, A., Koziolek, H., Prechelt, L., Reussner, R.: From monolithic to component-based performance evaluation of software architectures. Empirical Software Engineering 16(5), 587–622 (2011)
14. Meedeniya, I., Buhnova, B., Aleti, A., Grunske, L.: Architecture-Driven Reliability and Energy Optimization for Complex Embedded Systems. In: Research into Practice - Reality and Gaps (Proc. of QoSA 2010). LNCS, vol. 6093, pp. 52–67. Springer Berlin Heidelberg (2010)
15. Memari, A., Vornberger, J., Marx Gómez, J., Nebel, W.: A Data Center Simulation Framework Based on an Ontological Foundation. In: EnviroInfo 2014 - ICT for Energy Efficiency. pp. 461–468. BIS-Verlag (2014)
16. Procaccianti, G., Lago, P., Lewis, G.A.: Green Architectural Tactics for the Cloud. In: Working IEEE/IFIP Conf. on Software Architecture (WICSA 2014). pp. 41–44 (April 2014)
17. Raghavendra, R., Ranganathan, P., Talwar, V., Wang, Z., Zhu, X.: No "Power" Struggles: Coordinated Multi-level Power Management for the Data Center. SIGARCH Comput. Archit. News 36(1), 48–59 (Mar 2008)
18. Rivoire, S., Ranganathan, P., Kozyrakis, C.: A Comparison of High-level Full-System Power Models. In: Proc. of the 2008 Conf. on Power Aware Computing and Systems. HotPower'08, USENIX Association, Berkeley, CA, USA (2008)
19. Seo, C., Edwards, G., Malek, S., Medvidovic, N.: A Framework for Estimating the Impact of a Distributed Software System's Architectural Style on its Energy Consumption. In: Working IEEE/IFIP Conf. on Software Architecture (WICSA 2008). pp. 277–280 (February 2008)
20. Stier, C., Groenda, H., Koziolek, A.: Towards Modeling and Analysis of Power Consumption of Self-Adaptive Software Systems in Palladio. Tech. rep., University of Stuttgart, Faculty of CS, EE, and IT (Nov 2014), ftp://ftp.informatik.uni-stuttgart.de/pub/library/ncstrl.ustuttgart_fi/TR2014-05/TR-2014-05.pdf