Run-time Prediction of Power Consumption for Component Deployments

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

Maximilian Deffner
Universität Würzburg
maximilian.deffner@stud-mail.uni-wuerzburg.de

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

***Abstract*—The Power consumption of servers in data centers depends greatly on the software running on each server and how it interacts with the hardware. Different deployments of distributed software components on heterogeneous servers can lead to significant differences in power consumption, depending on the server allocation and the current workload. As workloads and load intensity change, components may be re-deployed or exchanged in order to reduce the power consumption for the current load profile. The decision on which component to place on which server during run-time remains difficult as the power consumption that would result from such a placement remains unknown. Existing work on component deployment optimization at run-time focuses on maximizing performance or considers power in the context of static design time decisions.**

In this paper, we introduce a model to predict the power consumption of component placements at run-time based on the load and power profile collected for a running distributed application in a heterogeneous environment. In addition, we present a model that enables the use of our approach without dedicated power measurement devices, predicting power consumption based on load intensity and performance counters. We show that we can predict the power consumption of two different distributed web applications with a mean absolute percentage error of 2.21% and with an error of 1.04% when predicting a previously unobserved load level.

I. Introduction

The energy efficiency of servers and data centers has become an ever more important issue over the last decades. In 2010, the U.S. Environmental Protection Agency (U.S. EPA) estimated that 3% of the entire energy consumption in the U.S. is caused by data center power draw [1].

The power consumed by servers in data centers depends on the specific software running on each server. Differences in the software design, specific implementation, and the hardware on which it runs can result in different power profiles [2]. As a result, the deployment of distributed software components on heterogeneous servers can lead to significant differences in power consumption depending on which component is deployed on which server. For each software component, the power consumption may scale differently as the load increases. Therefore, the most efficient component placement on the servers under consideration may change over time.

As the load changes over time, components (or services) may be re-deployed to different devices in order to reduce the power consumption for the current load profile. They may also be replaced by component implementations that provide better efficiency for the current load level. In addition, components or (micro-)services may be replicated across multiple devices. The power consumption of such deployments is unknown in advance, making informed management decisions difficult.

Existing approaches for prediction of non-functional properties of component deployment focus on performance. Architectural models, such as [3] and [4] support the prediction of performance characteristics at design time and run-time. Models that consider power focus on a static deployment context, using pre-defined system power models.

In this paper, we propose a modeling approach for predicting the power consumption of component placements at run-time. The deployment model is designed to only require basic information about the system, specifically about the structure and deployment infrastructure, such that it can be easily assembled without much modeling effort. The power prediction model makes use of run-time monitoring data about the observed power consumption under varying load levels. It captures the individual components' power profiles and their interactions based on which it can predict the power consumption of a new component placement configuration. We also introduce a separate single server power model leveraging measurements of an industry standard energy rating tool. This single server power model eliminates the need for dedicated power measurement devices at run-time that would otherwise be required to use our general end-to-end prediction model.

The goal of this paper is to enable informed component placement decisions that minimize the system power consumption. Our modeling approach can be used to find the least power consuming deployment option for a target throughput level. It can also be used in conjuction with any existing software performance model to optimize the energy efficiency of distributed systems. Note that our modeling approach is agnostic of the type of distributed architecture, i.e. if it is a component or (micro-)service based architecture. However, we use the term *component* throughout this paper to refer to components or (micro-)services.

The major contributions of this paper are as follows:

1) We propose an end-to-end modeling approach for run-time prediction of the power consumption of component deployments in heterogeneous environments.
2) We present a deployment model that supports the use of replicated components with alternate implementations with the goal of enabling power predictions.

3) We present a separate single server power model that eliminates the need for dedicated power measurement devices for run-time power prediction.

We evaluate our models using two different web applications deployed in a heterogeneous environment with servers of different CPU hardware architectures. We predict the power consumption of different component placements, including placements that make use of component replication and that vary the component implementations. We show that we can predict the power consumption of previously unobserved deployments with a mean absolute percentage error of 2.21%. In addition, we show that we can predict the power consumption of a system at a previously unobserved load level and component distribution with an error of 1.22%.

II. Related Work

We survey related work in three areas: *Architectural performance models*, *system-level power prediction models*, and *energy-aware power management*.

Architectural performance models, such as [3] and [4], model the software architecture and infrastructure of a component-based system for performance prediction, requiring detailed information on the system's software architecture, resource demands, and hardware infrastructure. They can be used for design-time performance prediction [3] or run-time prediction [4]. These models can be combined or extended with power information on power consumption to enable power prediction, as done for example in [5]. An example of an architecture-level model for power prediction, based on fine granular modeling of the software architecture and underlying hardware infrastructure, can be found in [6].

System-level power prediction models predict the power consumption of a single physical machine. These models can utilize a variety of methods, such as interpolation [7], [8], regression [9], [10], or others [11]. For the purpose of our work, we distinguish between hardware-based power models and workload-based power models. Hardware-based models, such as [9], [10] and [11] are based on system-level data. Workload-based models, such as [8] and [12], are trained for specific workloads and use application-level parameters.

Our model goes beyond single system power prediction models, predicting for multiple heterogeneous machines. However, it leverages established regression models as part of its end-to-end modeling approach.

Energy-aware power management techniques may employ power models to predict the power consumption of the system states that might result from management decisions. The typical goal is to minimize power consumption within certain Quality-of-Service (QoS) constraints [13], or to maximize overall efficiency [14], [15]. The impact of the power management decisions is often estimated using established basic power prediction methods. For example, [13] uses a utilization-based linear power model, whereas [16] uses a quadratic power model. Other works construct their own model of the underlying systems, such as [17], which models a server farm using stochastic Petri nets.

III. Power Prediction Model

We introduce two separate power models, an end-to-end model for predicting the power consumption of a workload based on its deployment on physical or virtual machines and based on the load intensity, and another model for predicting the power consumption of a physical server based on performance counters and standard benchmark results. The two models can be combined, as the end-to-end model requires the sum total power consumption of the physical machines on which it runs. It uses this data for training. This required information can be obtained using run-time measurements, which requires power instrumentation. This instrumentation can be omitted when using our server power prediction model.

A. Workload Deployment Power Prediction

The workload deployment power prediction model is designed to predict the power consumption of an end-to-end system for a specific component deployment at a specified load.

The model input, training data, and expected output are illustrated in Fig. 1. The power model requires the current throughput and the current deployment as input. It is trained using the power consumption of the entire system, which depends on the deployment and throughput. The model predicts the end-to-end system power consumption for a specified new deployment.

In the following, we describe our modeling approach in detail.

1) Prediction Method: Power prediction is a prediction problem on a continuous scale and can thus be posed as a regression problem. Consequently, both of our prediction models are designed to pose and solve regression problem statements. These problem statements can be solved by various

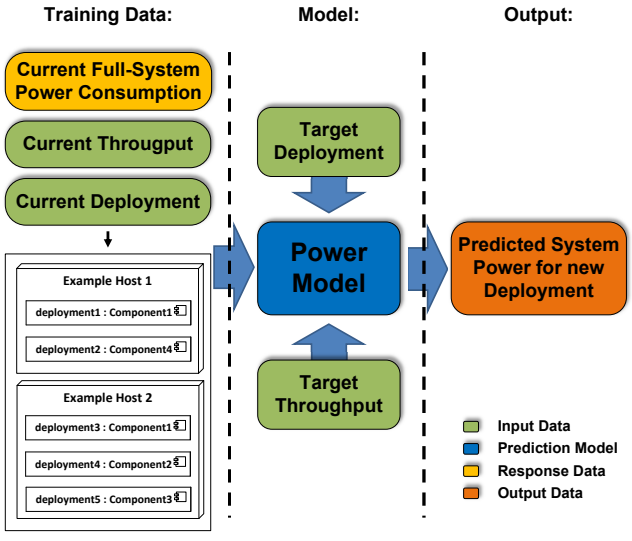

Fig. 1. Workload deployment power prediction data flow with example deployment.

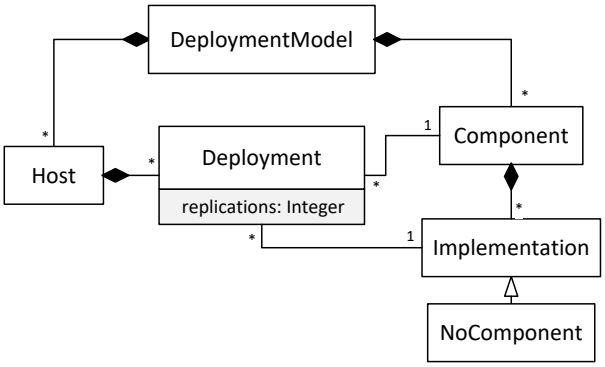

Fig. 2. Deployment meta-model.

regression and/or classification algorithms. In general, regression problems have the following form (Eq. 1):

$$\boldsymbol{Y} \approx f(\boldsymbol{X}, \boldsymbol{\beta}) \tag{1}$$

where $\boldsymbol{Y}$ is the set of *dependent* variables (also called *response* variables), $\boldsymbol{X}$ the set of *independent* variables (also called *regressor* variables), and $\boldsymbol{\beta}$ the set of regression *parameters* to be trained. In this work, we use generic regressor variables and regression parameters, allowing any type of regression models to be used, as opposed to limiting ourselves to a single model, such as e.g., linear regression. We use Gradient Tree Boosting [18] as the primary method in this paper, but other methods are supported.

2) Throughput as Input Variable: We consider throughput on the level of the entire distributed application. At load levels below saturation, throughput equals the load intensity. Once the load intensity exceeds system capacity (the maximum throughput), both the throughput and the power consumption stop scaling and stabilize at a given level. When mapping to regression model regressor variables, throughput can be mapped to a single variable.

3) Deployment Model as Input Variable: The deployment model describes which component (or service) is deployed on which host. It supports component replication (multiple instances of the same component) and multiple competing component implementations. Fig. 2 shows the deployment model's meta-model.

The deployment model maps the *components* to the physical *hosts* on which they are deployed. We refer to each mapping of a component to a host as a *deployment*. Each component may feature multiple deployments, as it may be replicated across multiple hosts. For each host, the components within its deployments must all be unique, however, if a component is deployed multiple times on the same host, the replication parameter of the deployment may be used to indicate this. Each component may have multiple component *implementations*. Each implementation has exactly one (possibly replicated) component that it implements. When deploying a component, the concrete implementation used in the deployment must be specified. We provide a *no component* implementation, which is used to create valid models for regression algorithms that require the specification of a mapping of each component to each host, even though the component may not be deployed on that host. Also note that *hosts* may be either physical or virtual hosts, even if multiple virtual hosts are co-located on the same physical machine. However, our model assumes that virtual hosts are fixed to physical hosts to avoid having to explicitly model mappings between the two. Consequently, a VM migration would be modeled as a component changing its deployment to a new virtual host.

Algorithm 1: Mapping of the deployment model to regressor variables.

```
Data: regressors: ordered list of regressors X,
deploymentModel: instance of DeploymentModel;
Function createRegressors()
    regressors.append(deploymentModel.hosts);
    for host ← deploymentModel.hosts do
        for deployment ← host.deployments do
            regressors.append(deployment.replications);
            if
             |deployment.component.implementations| >
             1 then
                regressors.append(
                mapImplementationToInteger(
                deployment.implementation));
            end
        end
    end
end
```

When mapping a deployment model to regressor variables $\boldsymbol{X}$, every component must feature a deployment for each host. It should use a *no component* implementation in case it is not actually deployed on the specific host. We specify regressor variables as shown in Algorithm 1. First, we create a regressor for the total number of hosts. Then, for each host and each deployment, we create a regressor with the replication count of that deployment. We assign an integer value to each component implementation in ascending order, starting from 0. This implementation tag is used for the final regressor for the specific deployment and may be omitted if only a single known implementation exists for the component in question.

4) System Power Response Variable for Training: Aggregate power consumption of all physical hosts is collected during run-time. Power can be collected using power measurement devices or the single server power prediction model in Section III-B.

5) System Power as Expected Output: The model predicts system power for a given system throughput and deployment model instance. The predicted power in Watts is the sum total consumption of all physical hosts.

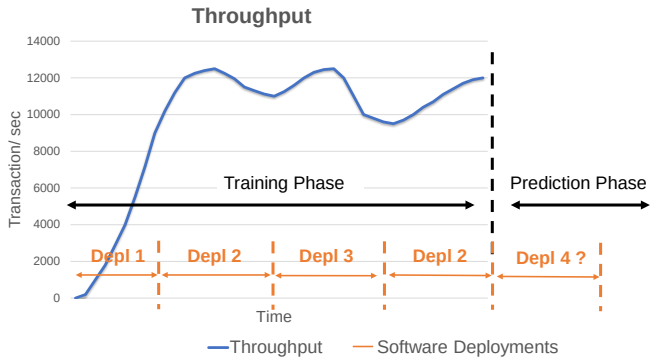

Fig. 3. Example scenario for use of the workload deployment power prediction model.

6) Applying the Model to a Running System: The power prediction model is designed for use in a run-time system where data on current deployments, throughput, and power consumption is continuously being gathered. The model may be retrained at each point in time with the most current data sets. We envision the following application scenario (an example scenario is illustrated in Fig. 3).

The load intensity of production server systems typically varies over time. We train the model by observing the system under varying load intensity. We log the throughput of the system on a continuous basis (e.g., once per second). In addition, we log the power consumption and information on the current deployment. From these observations, we obtain a new training vector. The model may be retrained at any point in time and is expected to be retrained periodically. Once trained, it may be used to predict power consumption of a future configuration.

B. Single Server Power Prediction

The run-time power model described in the previous section requires power consumption data for training. However, we cannot assume that every physical server has a power analyzer attached to it. Consequently, in this section, we propose a server power prediction model that can be used to predict the power of a single server. This model is designed to be used as a substitute for separate power analyzers in case they are not available.

The single server power model requires offline training in order to fully characterize the power behavior of the servers in question. For this, we use the SPEC Server Efficiency Rating Tool (SERT), which is run on all server models as part of the U.S. EPA Energy Star testing process [1]. SERT uses a total of 13 different mini-workloads, called "worklets", which are executed at multiple load levels. Each of these worklets, except for the *idle* worklet, are executed at multiple load levels. For each of the load levels, SERT measures worklet throughput and power consumption on a per second basis. We configure our SERT runs to 25 load levels per worklet. We also modify SERT to measure average CPU performance counters in addition to throughput and system power consumption for each load level. We train our model using those worklets within SERT for which power consumption scales with throughput and which are run at multiple throughput levels. Worklets fitting these criteria are the seven CPU worklets and the hybrid SSJ worklet [2].

Again, regression is used as prediction method. The CPU performance counters are mapped to regressor variables, the measured workload power is used as response variable.

We make use of the ability of modern CPUs to report their own power consumption. We employ this feature for power measurements, considering that CPU power consumption has been shown to correlate significantly with full-system power consumption [20]. In addition, we consider workload memory characteristics. Unfortunately, the *memory power consumption* performance counter is only available on very few platforms, so we use *memory bytes written* counter instead. We expect it to have a greater correlation to the memory power consumption than the CPU counter for *memory bytes read*, as the latter includes cache reads [21], which do not cause any memory power draw.

IV. Evaluation

We evaluate the applicability and accuracy of our prediction models by testing their ability to predict power consumption in a run-time scenario under varying load in a heterogeneous environment. We run two common web applications in a distributed environment consisting of two different servers using CPUs of different architectures.

A. Methodology

The two test applications are deployed on two separate physical servers from different generations and using different processor architectures (see Table I), which are put under load by a load generator running on a separate machine. The load generator collects application level performance data and is controlled by a dedicated experiment controller, which collects power measurements using external measurement devices.

We use two common 2-tier, PHP-based, web applications for testing. The advantage of using PHP-based web applications is that we can emulate the effect of different (functionally identical) implementations by exchanging the web server (and PHP module) on which the applications are run. We use the Dell DVD Store 6.1.1.4 and RUBiS 2.3.2, each executed on Apache HTTP Server 2.4.10, Lighttpd 1.4.35, or Nginx 1.6.2 and a MySQL database. We derive 15 different deployment

TABLE I
System under test specification including power characteristics as measured using SERT.

	SUT 1	SUT 2
Model	HP ProLiant DL20	HP ProLiant DL160
CPU	Intel Xeon E3-1230 v5, 4 cores at 3.4 GHz	Intel Xeon E5-2640 v3, 8 cores a 2.6 GHz
Memory	2 x 8 GB	2 x 16 GB
Storage	1 x 460 GB	1 x 460 GB
Idle Power	27.1 W	36.4 W
Max Power	98.1 W	140.6 W

TABLE II
POWER CONSUMPTION OF THE MOST AND LEAST POWER CONSUMING DEPLOYMENT CONFIGURATION AT FULL LOAD.

Deployment Config.	SUT 1	SUT 2	Total
SUT 1: Apache; SUT 2: MySQL	81.56 W	54.45 W	136.01 W
Both SUTs: Lighttp + MySQL	45.83 W	59.27 W	105.1 W
Difference	35.73 W	-4.82 W	**30.91 W**

options from the three different web tier implementations and the database, with the two SUTs as deployment targets.

We create load with varying arrival rates over time using the TeaStore's HTTP load generator [22]. The load generally increases over time and is intended to cover a range of incoming arrival rates, depending on the conrete experiment. The user-actions (requests) sent to the web applications follow a cyclic pattern, which is based on a user profile that emulates a user browsing the store(s), viewing random items, and adding them to the shopping cart. We report Mean Absolute Error (MAE), Median Absolute Error, and Mean Absolute Percentage Error (MAPE) in Watts for our measurements.

B. Power Saving Potential

Table II shows the power consumption of the most and least consuming deployment at maximum load. These two deployments feature a power difference of 30.91 W (29.4%), which offers a bound for model prediction errors, being the maximum deviation of power consumption that can actually occur in practice in our scenario.

C. Predicting Previously Unobserved Deployments

To evaluate the prediction accuracy for previously unobserved deployments we train a varying number of deployments. Each of these deployments is trained using a linearly increasing load profile over 300 seconds (resulting in 300 training vectors per deployment). We then evaluate the accuracy of the resulting power prediction model for a deployment that is not in the set of training deployments.

Table III shows the power prediction errors using gradient tree boost depending on the number of training deployments as well as the number of different Web UI component implementations that occur in the set of training deployments. The table shows that prediction accuracy generally increases significantly with additional training deployments, with exception of one outlier. The predictions for all training data sets, except for the one with five deployments, have an error of less than 10%, well below our upper bound of 29.4%.

RUBiS shows similar results, with a minimum MAE of 1.25 W, when prediction a previously unobserved deployment with the other 14 deployments in the training set.

TABLE III
PREDICTION ERROR FOR PREVIOUSLY UNOBSERVED DEPLOYMENTS DEPENDING ON # TRAINED DEPLOYMENTS AND # WEB UI COMPONENT IMPLEMENTATIONS.

#Tr. Dep.	#UI	MAE	MAPE	MdAE	Error Interval
3	2	1.98 W	2.18%	1.54 W	[-3.9 W, 4.3 W]
5	2	12.4 W	12.48%	10.47 W	[-17.1 W, 1.2 W]
7	2	6.41 W	6.68%	5.89 W	[-11.1 W, 3.8 W]
11	3	3.1 W	3.17%	2.51 W	[-5.3 W, 4.2 W]
14	3	1.9 W	2.21%	1.62 W	[-4.9 W, 3.2 W]

TABLE IV
PREDICTION ERROR FOR PREVIOUSLY UNOBSERVED THROUGHPUT DEPENDING ON # TRAINED DEPLOYMENTS.

#Train. Depl.	Observed Depl.	Observed Thr.	MAE	MAPE
3	Yes	Yes	0.86 W	0.93%
3	Yes	**No**	**1.29 W**	**1.35%**
14	No	Yes	0.9 W	1.04%
14	No	**No**	**1.1 W**	**1.22%**

D. Power Prediction for Previously Unobserved Throughput

We investigate the prediction accuracy when prediction the prediction of power consumption for a load intensity that has not been observed in the training set. Again, we use 300 training vectors (load levels) per deployment. The training vectors include power and throughput values for all load levels, except the ranges between 3000 and 6000 requests per second and 10000 and 13000 requests per second. When predicting for the target deployments, we predict 35 power consumption samples for throughputs in the omited ranges.

Table IV shows the errors of the prediction. Our regression-based power extrapolates the power consumption for previously unobserved load intensity ranges with high accuracy. The prediction for observed and unobserved throughput levels differ by less than 0.4 Watts. In addition, the prediction error for scenarios where both the throughput level and deployment are unobserved is not significantly higher compared to scenarios where only the deployment is unobserved. In our case, the difference is only 0.2 Watts.

E. Combined Models Prediction Accuracy

The end-to-end scenario for applying our modeling approach includes using both the workload deployment power prediction model and calibrated single server power models for all SUTs. We evaluate the prediction error of the workload deployment power prediction model when using the calibrated single server power models to estimate the power consumption of each SUT. Both SUTs are instrumented with performance counter listeners, based on Intel PCM [23], to collect performance counters at run-time. These performance counters are used to estimate the current power draw of the SUT.

To evaluate the impact of the error added by using the single server power prediction models, we choose a self-prediction scenario for the workload deployment model. The deployment

TABLE V
ERROR OF USING BOTH MODELS VS. DEPLOYMENT ONLY.

System Power	MAE	MAPE
Measured	1.38 W	1.46%
Model	1.8 W	2.01%

model is trained with all of our 15 deployment options and we evaluate its accuracy when predicting the power consumption of one of the training models for various target throughput levels. We compare the relative and absolute errors of the power prediction when using both models together against the power consumption measured by the dedicated power measurement devices for the predicted deployment.

Table V shows the relative and absolute mean self-prediction errors. The MAE when using power measurements of the power analyzer is 1.38 Watts. When substituting the power analyzer with the single server power model, the MAE increases by 0.42 Watts to 1.8 Watts. Generally, we can conclude that using both models together results in acceptably small errors warranting their use in case of the unavailability of hardware power analyzers.

V. Conclusions

This paper presents an approach for predicting the power consumption of distributed applications at run-time The approach uses run-time monitoring data to train a deployment and power model that allows the prediction of power consumption for different deployment options and load levels in heterogeneous environments. It can predict the power consumption of a previously unobserved deployment with an error of 2.21%. It can also predict the power consumption of a previously unobserved throughput level with an error of 1.22%.

The approach in this paper can be used in multiple contexts: It can be used at run-time to find the least power consuming deployment option for the current throughput of a distributed applications, but it could also be used for energy efficiency prediction when applied in conjunction with run-time performance prediction models, such as DML [4].

Acknowledgment

This work was funded by the German Research Foundation (DFG) under grant No. (KO 3445/11-1) and was supported by the Power Working Group of the SPEC Research Group.

References

[1] K.-D. Lange and M. G. Tricker, "The Design and Development of the Server Efficiency Rating Tool (SERT)," in *Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '11. New York, NY, USA: ACM, 2011, pp. 145–150. [Online]. Available: http://doi.acm.org/10.1145/1958746.1958769

[2] 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 *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering (ICPE 2015)*, ser. ICPE '15. New York, NY, USA: ACM, February 2015.

[3] S. Becker, H. Koziolek, and R. Reussner, "The Palladio component model for model-driven performance prediction," *Journal of Systems and Software*, vol. 82, pp. 3–22, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2008.03.066

[4] N. Huber, F. Brosig, S. Spinner, S. Kounev, and M. Bähr, "Model-Based Self-Aware Performance and Resource Management Using the Descartes Modeling Language," *IEEE Transactions on Software Engineering (TSE)*, vol. 43, no. 5, 2017. [Online]. Available: http://dx.doi.org/10.1109/TSE.2016.2613863

[5] C. Stier, A. Koziolek, H. Groenda, and R. Reussner, "Model-Based Energy Efficiency Analysis of Software Architectures," in *Proceedings of the 9th European Conference on Software Architecture (ECSA '15)*, ser. Lecture Notes in Computer Science. Springer, 2015. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-23727-5_18

[6] R. Basmadjian, N. Ali, F. Niedermeier, H. de Meer, and G. Giuliani, "A Methodology to Predict the Power Consumption of Servers in Data Centres," in *Proceedings of the 2Nd International Conference on Energy-Efficient Computing and Networking*, ser. e-Energy '11. New York, NY, USA: ACM, 2011, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/2318716.2318718

[7] X. Fan, W.-D. Weber, and L. A. Barroso, "Power Provisioning for a Warehouse-sized Computer," in *The 34th ACM International Symposium on Computer Architecture*, 2007. [Online]. Available: http://research.google.com/archive/power_provisioning.pdf

[8] J. von Kistowski and S. Kounev, "Univariate Interpolation-based Modeling of Power and Performance," in *Proceedings of the 9th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2015)*, December 2015.

[9] B. C. Lee and D. M. Brooks, "Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction," *SIGPLAN Not.*, vol. 41, no. 11, pp. 185–194, Oct. 2006. [Online]. Available: http://doi.acm.org/10.1145/1168918.1168881

[10] A. Lewis, S. Ghosh, and N.-F. Tzeng, "Run-time Energy Consumption Estimation Based on Workload in Server Systems," in *Proceedings of the 2008 Conference 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

[11] G. Dhiman, K. Mihic, and T. Rosing, "A system for online power prediction in virtualized environments using gaussian mixture models," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, June 2010, pp. 807–812.

[12] D. Economou, S. Rivoire, and C. Kozyrakis, "Full-system power analysis and modeling for server environments," in *In Workshop on Modeling Benchmarking and Simulation (MOBS*, 2006.

[13] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012. [Online]. Available: http://dx.doi.org/10.1016/j.future.2011.04.017

[14] 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 *Proceedings of the IEEE 23nd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2015)*. IEEE, October 2015.

[15] G. Jung, M. Hiltunen, K. Joshi, R. Schlichting, and C. Pu, "Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures," in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, June 2010, pp. 62–73.

[16] R. Urgaonkar, U. Kozat, K. Igarashi, and M. Neely, "Dynamic resource allocation and power management in virtualized data centers," in *Network Operations and Management Symposium (NOMS), 2010 IEEE*, April 2010, pp. 479–486.

[17] Y. Tian, C. Lin, and M. Yao, "Modeling and analyzing power management policies in server farms using stochastic petri nets," in *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, May 2012, pp. 1–9.

[18] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

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

[20] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A Comparison of High-level Full-system Power Models," in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, ser. HotPower'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 3–3. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855610.1855613

[21] N. Schmitt, J. von Kistowski, and S. Kounev, "Emulating the Power Consumption Behavior of Server Workloads using CPU Performance Counters," in *Proceedings of the 25th IEEE International Symposium on the Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, ser. MASCOTS '17, September 2017.

[22] J. von Kistowski, "HTTP Load Generator," https://github.com/joakimkistowski/HTTP-Load-Generator, 2017, last accessed: June 15, 2018.

[23] T. Willhalm, R. Dementiev, and P. Fay, "Intel® Performance Counter Monitor - A better way to measure CPU utilization," https://software.intel.com/en-us/articles/intel-performance-counter-monitor-a-better-way-to-measure-cpu-utilization, August 2012.