# No Hardware Required: Building and Validating Composable Highly Accurate OS-based Power Models

John D. Davis[1], Suzanne Rivoire[2], Moises Goldszmidt[1], and Ehsan K. Ardestani[3]
Microsoft Research - Silicon Valley Lab[1] Sonoma State University[2] University of CA, Santa Cruz[3]

ABSTRACT
In this paper, we present an automatic framework for modeling node- and cluster-level power consumption, using only portable OS-level performance counters. We evaluate these models using an emerging class of MapReduce-style workloads, executed on current server-class systems as well as energy-efficient low-power desktops, high-end laptops, and embedded systems. We also validate generic, cross-platform (with respect to model features) cluster power models for our four workloads running on six types of clusters. Our models yield highly accurate predictions without the intrusiveness and/or the correctness and portability problems of hardware performance counters or board-level measurements.

We define a new metric called Dynamic Range Error (DRE) to describe how well the model characterizes the dynamic system behavior (a tighter bound than MSE or median error) and facilitate inter- and intra-cluster model accuracy comparisons. Using this metric, we quantify the tradeoffs between model complexity and accuracy for different workloads. The generic feature model removes the feature selection process and only degrades prediction accuracy by at most 1% DRE when compared to the best cluster power model for the workloads and clusters we studied. To the best of our knowledge, this is the most complete study of system power modeling covering such a wide variety of platforms, workloads, and models.

## 1. Introduction

Power consumption is a first-order design constraint in the data center (DC). Power infrastructure accounts for approximately 80% of data center facility costs and about 40% of operating costs [12]. Provisioning DCs for peak absolute system power avoids the dire consequences of exceeding the DC's power budget, but it is overly conservative. The goal of this paper is to build software-based high-fidelity cluster power models that can be used for DC power provisioning and planning, online power capping, and power-aware software tuning, eliminating the need for expensive hardware and/or software solutions. In order for these models to be suitable for large-scale use, we require them to use generic, portable predictors; to be highly accurate for emerging server designs and workloads; and to scale beyond a single node.

We use OS-level performance counters to build and validate machine-level and cluster-level power models for six different clusters of homogeneous machines. OS-level performance counters are easy to collect and consistent across multiple platforms (portable), and they can be collected in real time for online power prediction. The machines that make up the clusters span the embedded, mobile, desktop, and server processor spaces, reflecting energy-efficient server recommendations from recent research [1,13,19,31,32] as well as more traditional servers used in current practice. We build and validate these models on MapReduce-style applications that all use the same software stack.

In order to conduct this model exploration, we build and evaluate over 1200 full-system power models per cluster in a design exploration of the modeling techniques and model features. Our power modeling requirements are to simultaneously provide high accuracy, with errors of less than 10% of the dynamic power range (a stricter error measure than used in prior art), minimize the number of model features, use portable features across platforms, and minimize the model complexity, thereby reducing modeling overhead. Accurate and low-cost models are critical when planning new DCs and provisioning existing ones.

In this paper, we make the following contributions:

- We define and evaluate a new model error metric called Dynamic Range Error (DRE), which is based on the mean squared error and the *dynamic* power range of the system. This metric yields a more appropriate basis for cross-platform comparison than currently used error metrics.
- We demonstrate an automatic, generic framework that builds high-fidelity cluster power models using portable OS-level performance counters. We achieve prediction errors of less than 10% DRE, a stricter error measure than prior work has used, or 0.5-2.5% using median error.
- We show the relationship between model complexity and accuracy for different workloads and present a general set of predictors that yields accurate models across the different clusters.

The rest of this paper is organized as follows. Section 2 provides a brief summary of related work. Section 3 describes our hardware, software, and instrumentation infrastructure. In Section 4, we present the model feature space and modeling techniques we used. Section 5 evaluates our models' accuracy and generality, and Section 6 concludes.

## 2. Related Work

In this section, we compare our full-system power models to prior work based on five characteristics: choice of predictors, choice of modeling techniques, sampling frequency, portability/scalability, and overall accuracy. We discuss only system-level power models and omit component-specific models, which have very different goals and characteristics.

*Choice of predictors.* Early power models were based solely on CPU utilization [5,9,24,26] or even CPU frequency state [4]. Subsequent work uses board-level measurements [21,22] or hardware performance counters to capture the behavior of CPU, memory, and I/O devices [2,8,23,29,30]. Our framework uses only OS-level performance counters, avoiding the intrusiveness of board-level measurements and the correctness and portability problems of hardware performance counters [3,33].

*Choice of modeling techniques.* Most previous work uses linear models [5,9,14,21,23,27], piecewise linear models [28], or other models that do not capture interaction between predictors [2]. Other recent work found that even nonlinear interactive models fail to capture the behavior of the systems studied, but chaotic attraction predictors provide the desired accuracy [22]. Our work confirms that linear models are often insufficient, but we find interactive models to be highly accurate for our machines and

**Table 1**. We develop full-system power models for 5-machine clusters of the platforms below. *System maximum memory capacity.

| System Class | CPU | Memory | Disk(s) | OS, FS |
|---|---|---|---|---|
| Embedded | Intel Atom, dual-core, 1.6 GHz | 4 GB DDR2-800* | 1 Micron SSD | Windows Server 2008 R2, NTFS |
| Mobile | Intel Core 2 Duo, dual-core, 2.26 GHz, 25 W TDP | 4 GB DDR3-1066* | 1 Micron SSD | |
| Desktop | AMD Athlon, dual-core, 2.8 GHz, 65 W TDP | 8 GB DDR2-800 | 1 Micron SSD | |
| Server | AMD Opteron, quad-core, 2.0 GHz, 50 W TDP | 32 GB DDR2-800 | 2 10K RPM SATA | |
| Server | Intel Xeon, quad-core, 2.33 GHz, 80 W TDP | 16 GB DDR2-667 | 4 7.2K RPM SATA | |
| Server | Intel Xeon, quad-core, 2.67 GHz, 80 W TDP | 16 GB DDR2-667 | 6 15K RPM SAS | |

workloads. Our choice of predictors may be the reason for this difference from prior work.

*Sampling frequency.* A sampling frequency of 1 Hz is common in the literature, since many power meters and OS event interfaces do not support faster sampling. Recent work shows that this sampling rate does not capture short power-supply-induced spikes; modeling these spikes requires sampling at every invocation of the OS scheduler [24]. The inability to model these spikes is a limitation of less intrusive models, including ours. At the other extreme, some models have used 10-minute intervals [9] or modeled total energy over a workload [20,29,30], which misses application-level behavior patterns [16].

*Portability and scalability.* Some previous models require application-specific profiling [11,23], while others are solely for processor-intensive workloads [14,28]. One study compared simple models for a breadth of machines and workloads but did not go beyond a single node [27]. In fact, only a few studies model the power consumption of multiple machines [7,9,11,14]. This work builds single-node and cluster-level models for a variety of machines and data-intensive workloads.

*Accuracy.* In many papers the only metric of accuracy is that the model was sufficient for some energy-saving technique. Other papers use metrics such as (r)MSE or median error, but they compare these metrics to static power rather than the dynamic power range. This makes it difficult to understand how well the model captures variation in power. Our modeling approach yields models at least as accurate as any previously proposed generalizable approach, with the added advantages of portability and non-intrusive metric collection.

## 3. Infrastructure

This section describes the clusters for which we build power models, the workloads we use to build and validate models, and our measurement infrastructure.

### 3.1 Hardware and software infrastructure

We build power models for multiple small (five-node) homogeneous clusters. Using these testbeds, which are described in Table 1, we demonstrate an automatic, general, and portable framework for building high-fidelity cluster power models, and we evaluate the accuracy of several modeling techniques for a variety of workloads.

We run an assortment of distributed workloads using the Dryad and DryadLINQ application framework [15]. Some of the workloads are CPU-intensive, while others are dominated by disk or network. We run all workloads five times per cluster to allow each node to act as job scheduler, which provides diversity in the work done even within an application. The workloads used are:

- **Sort.** This workload sorts 4GB of data with 100-byte records. This workload has high disk and network utilization.
- **PageRank.** This workload runs a graph-based page ranking algorithm over the ClueWeb09 dataset [6], a corpus of about 1 billion web pages. PageRank has high network utilization.
- **Prime.** This workload checks for primeness of each of approximately 1,000,000 numbers on each of 5 partitions in a cluster. This workload is CPU-intensive and produces little network traffic.
- **WordCount.** This workload reads through 50 MB text files on each of 5 partitions in a cluster and tallies the occurrences of each word that appears. It produces little network traffic.

### 3.2 Measurement infrastructure

The measurement infrastructure consists of a hardware component that physically measures total system power and the software components that collect both the power measurements and OS-level performance counters.

**Hardware:** Every machine in every cluster is individually instrumented with a power meter. We use the WattsUp? Pro digital power meter to capture the wall power once per second. Each machine reads its own power measurements over a USB port. The power meters have an error of 1.5%. We verified the meter calibration, but we leave the explicit extraction of meter error for future work.

**Software:** Each system runs Windows Server 2008 R2, which has a convenient and standardized OS-level performance counter interface and tool suite. We use Windows Perfmon to record measurements once per second for Windows ETW (Event Tracing for Windows) software counters as well as the WattsUp? Pro power meter readings.

## 4. Modeling Framework

This section briefly reviews the feature selection process, which is described in more detail in [7]. Then it presents the four different modeling techniques used to generate full-system and full-cluster power models.

### 4.1 Feature Selection

To fully explore the space of predictors, we sample a wide

**Table 2.** Significant ETW performance counters used in cluster models.

| Category | Performance counter | counter ID |
|---|---|---|
| Network | Datagram/sec | 1 |
| Memory | Page Faults/sec | 18 |
| | Committed Bytes | 20 |
| | Cache Faults/sec | 24 |
| | Pages/sec | 26 |
| | Page Reads/sec | 28 |
| | Pool Nonpaged Allocs | 34 |
| Physical Disk | Disk Total Disk Time % | 54 |
| | Disk Total Disk Bytes/sec | 66 |
| Process | Total Page Faults/sec | 79 |
| | Total IO Data Bytes/sec | 99 |
| Processor | Total Processor Time % (Utilization) | 102 |
| | Total Processor Interrupts/sec | 105 |
| | Total Processor % DPC Time | 106 |
| File System Cache | Data Map Pins/sec | 121 |
| | Pin Reads/sec | 122 |
| | Pin Read Hits % | 125 |
| | Copy Reads/sec | 126 |
| | Fast Reads not Possible/sec | 139 |
| | Lazy Write Flushes/sec | 140 |
| Job Object Details | Total Page File Bytes Peak | 167 |
| Processor Performance | Processor_0 Processor Frequency | 209 |

range of performance counters provided by the OS and tool suite. This set includes counters from the following categories: processor, memory, physical disk, process, job object, file system cache, and network interfaces [25]. In aggregate, approximately 250 performance counters are collected per second per node. We verified that the data collection process does not interfere with program behavior or power consumption.

After removing insignificant or redundant counters, about 50 features remain. For each system type, we further pare this list to 5-15 significant features. The equations below show the final features for each cluster-specific model. Power is denoted by $y$, and $x_i$ denotes the features measured on each machine. We collect data of the form $<y, x_1,\ldots,x_n>$, and we fit functions $\hat{f} = (x_1, \ldots x_n)$ so that $\hat{f}()$ approximates $y$, minimizing some loss function. The numeric subscripts refer to the counter IDs in Table 2.

$$P_{OpteronNode} = \hat{f}(x_{26}, x_{54}, x_{66}, x_{102}, x_{121}, x_{122}, x_{167}, x_{209})$$

$$P_{IntelSATANode} = \hat{f}(x_1, x_{18}, x_{20}, x_{26}, x_{28}, x_{54}, x_{79}, x_{102}, x_{105}, x_{106}, x_{121}, x_{122}, x_{140}, x_{167}, x_{209})$$

$$P_{IntelSASNode} = \hat{f}(x_1, x_{18}, x_{20}, x_{26}, x_{28}, x_{54}, x_{66}, x_{79}, x_{102}, x_{121}, x_{122}, x_{125}, x_{139}, x_{140}, x_{167}, x_{209})$$

$$P_{AthlonNode} = \hat{f}(x_{18}, x_{26}, x_{99}, x_{102}, x_{167}, x_{209})$$

$$P_{Core2Node} = \hat{f}(x_{24}, x_{34}, x_{54}, x_{102}, x_{122}, x_{167}, x_{209})$$

$$P_{AtomNode} = \hat{f}(x_{24}, x_{34}, x_{66}, x_{102}, x_{121}, x_{126}, x_{139}, x_{140}, x_{167})$$

We found that systems with aggressive DVFS required the processor frequency as a model feature. Furthermore, the Intel servers had fewer frequency states and more hard disk drives. For these models, the processor frequency feature was not as strong a signal as other models and hard drive related features were stronger that single disk configurations, both with respect to hardware related counters and software related counters like the file system and related cache.

## 4.2 Modeling techniques

We experiment with four different modeling techniques of varying degrees of conceptual and implementation complexity. We present the simple linear model first, followed by the piecewise linear, interactive, and switching model, the latter based on domain-specific system knowledge.

*Baseline linear power model*:

$$\hat{f}(x_1, \ldots, x_n) = a_0 + \sum_i a_i \times x_i \qquad (1)$$

*Piecewise linear power model*:

$$\hat{f}(x_1, \ldots, x_n) = a_0 + \sum_i \sum_j a_{i,j} \times B^s_{i,j}(x_i, t_{i,j}) \qquad (2)$$

*Interactive power model*:

$$\hat{f}(x_1, \ldots, x_n) = a_0 + \sum_i \sum_j a_{i,j} \times B^s_i(x_i, t_i) \times B^s_j(x_j, t_j) \qquad (3)$$

*Switching power model*:

$$\hat{f}(x_1, \ldots, x_n) = I(f)(a_0 + \sum_i a_i \times x_i) + (1 - I(f))(a'_0 + \sum_i a'_i \times x'_i) \quad (4)$$

where *I(f) = 1 iff the frequency < threshold; otherwise I(f) = 0.*

We start with a basic linear regression model (Equation 1), where the parameters ${a_i}_0^n$ are fitted by minimizing the squared error. This is the form used by most previous work. It is a useful baseline against which we can compare all other proposals for $\hat{f}()$ and evaluate the increase in accuracy of more complex models.

The piecewise linear power model (Equation 2) provides an extra degree of freedom. In this model, *s* can be positive (+) or negative (–), and the basis functions $B_{i,j}^s$ are hinge functions. $B^+_{i,j}(x,t)$ takes a value of 0 if $x \le t$ and $x–t$ otherwise. Similarly, $B^-_{i,j}(x,t)$ takes a value of 0 if $x > t$ and $t–x$ otherwise. The *t* thresholds are usually called *knots*, and the *j* indices permit a feature to be responsible for more than one knot. Fitting these models requires finding the knots $t_{i,j}$ and the parameters $a_{i,j}$. To do so, we use an implementation of the Multivariate Adaptive Regression Splines (MARS) algorithm [10].

**Table 3.** Dynamic range error (DRE) compared to two other common error metrics: rMSE and the percent error (rMSE/ average power).

| | Opteron - Server | | | Atom - Embedded | | |
|---|---|---|---|---|---|---|
| Workloads | rMSE | rMSE/ Avg Power | DRE | rMSE | rMSE/ Avg Power | DRE |
| Primes | 4.35 | 2.9% | 11.3% | 0.57 | 2.4% | 30.8% |
| Staticrank | 4.13 | 2.9% | 7.6% | 0.64 | 2.6% | 19.4% |
| Terasort | 2.76 | 2.0% | 9.4% | 0.69 | 2.8% | 11.5% |
| Wordcount | 3.84 | 2.6% | 9.5% | 0.64 | 2.6% | 22.7% |

Intuitively, these models can express that a feature, such as CPU utilization, may consume full-system power at different rates in different regions. It could thus be nonlinear over the entire space, but linear within each of these regions.

The interactive model (Equation 3) is an extension of the piecewise linear model that introduces nonlinearity by letting the basis functions interact. We restrict this interaction to degree = 2 and use the same algorithm (and implementation) as in the piecewise linear case to select knots and fit parameters and to select which bases would interact. Finally, the switching model (Equation 4) uses CPU frequency as the indicator function, *I(f)*. The result is a set of (possibly) different linear models depending on the clock frequency. Unlike the piecewise model, where the knots partition only the space of a particular feature, the frequency state and the indicator function partition the space for all the features, creating completely separate models for each frequency. This model is more rigid, even though it may require more parameters (since we must fit coefficients for every feature at every frequency state).

## 5. Evaluation

In this section, we discuss the highlights of our findings from building over 1200 power models. We use a scripting language to generate high-level statistical computing code. In some case, the high-level statistical environment uses the scripting language to process intermediate results, automating model building and testing. In the following subsections, we explain these results in detail. First, we define and justify the dynamic range error (DRE), the metric we will use to evaluate the models. Then we evaluate our platform-specific and general model. Finally, we discuss the impact of these modeling techniques.

## 5.1 Dynamic range error

To evaluate the accuracy of power models, absolute error terms like mean squared error (MSE) or median error are difficult to compare across platforms whose operating power may differ by orders of magnitude. Presenting these error terms as a percentage of the total power still obscures the fact that the overall power may contain a large static component, making it trivially predictable. The real question from a research perspective is how well the model captures the dynamic variation in power.

Therefore, a more meaningful number is the dynamic range error (DRE), which we define as the root-mean-squared error divided by the dynamic range. Table 3 compares various error metrics across our platforms. It shows that a small rMSE, on the order of 2% of total power, can translate into a large DRE of 30%. Using DRE provides a platform-independent view of a model's explanatory power.

## 5.2 Cluster Model Discussion

We compare 16 different cluster models from the 4 categories of models described in Section 4.2, using various subsets of the features from the 22 independent features identified in Table 2. We use 5-fold cross validation to generate an average DRE per

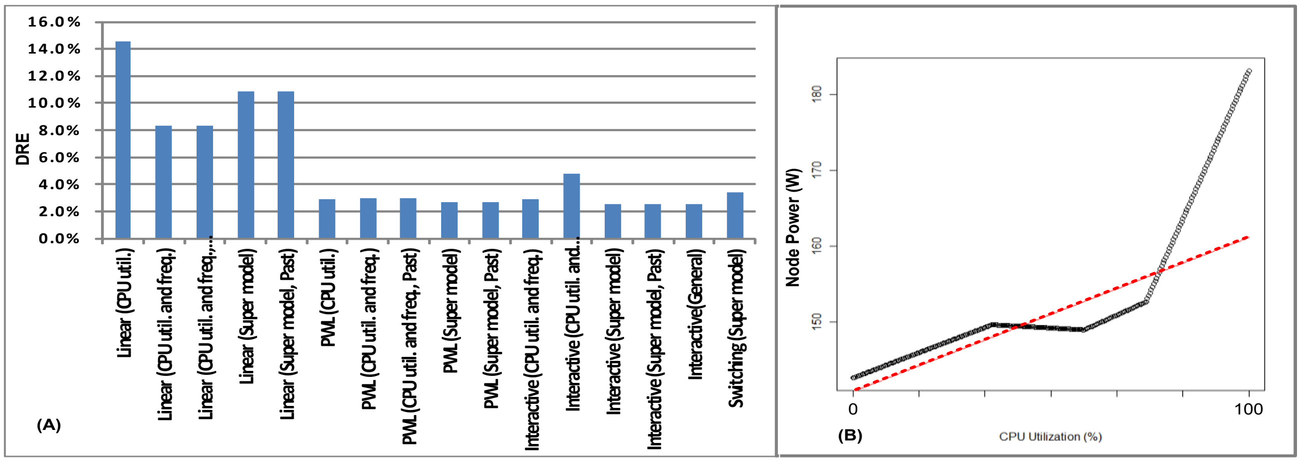

**Figure 1.** (A) Opteron average DRE for one representative workload (Primes), demonstrating that more complex models are required. Other workloads like StaticRank (not shown) demonstrate that feature selection is critical. (B) The piecewise linear model (black circles) can predict the full-system power range, while the trained linear model (red dotted line) cannot.

cluster configuration and workload. Using this methodology, we are able to compare models across the various platforms and quickly understand how good the models are.

### 5.2.1 *Cluster Power Models*

The cluster power model is built using a general machine model, $\hat{f}(x_1, \dots, x_n)$, as described in Section 4.2, that is trained using several machines, as shown in Equation 5. The best model accuracy was achieved using a subset of the features listed in Table 2 for each machine in the cluster, specified in Section 4.1.

$$Power_{cluster} = \sum_i \hat{f}_i(x_1, \dots, x_n) \qquad (5)$$

Some of our initial attempts relied on aggregated features across the cluster to find correlations to the cluster power. In this case, there were too many features that correlated and the generated models were fragile. The features that these models selected excluded some nodes and as a result, there was no clear way to scale the models. The same holds true for models at the rack level when trying to aggregate features from all the machines. Our goal is to reduce the number of features used to model the cluster and use the simplest model with the highest fidelity. Of the models we investigated, the PWL and interactive models use fewer features than the switching models, reducing their overhead. By reducing the model overhead, we are able to also use these models for online power prediction, which can then be used to attribute costs to the appropriate user, screen for hardware and/or software failures, or other data mining studies.

### 5.2.2 *Cluster model accuracy*

Figure 1A lists the modeling techniques, from left to right, in increasing order of the number of features used in the models. It should be noted that the results presented here are typical across all platforms, and so we only present data from one cluster and one of the workloads.

There are several observations that can be made from Figure A related to model complexity and feature selection. This figure shows the CPU-only models are, in general, the least accurate. Even in CPU-bound applications like Primes, using piecewise linear (PWL) models dramatically improves accuracy. Furthermore, regardless of the application, feature selection played a critical role in the modeling accuracy. When using a small number of features, we found a few cases of the piecewise linear and linear modeling techniques producing the same models for a particular workload and cluster. However, when we use the model feature set from Section 4.1, all PWL, interactive, and switch models outperformed all linear models. These results demonstrate that more complex models are required and that feature selection matters to produce high-fidelity models. Finally, as prior art has shown [22], past data may be useful in improving model accuracy. We used the past (t-1) core frequency in the piecewise linear and interactive models, but we found this additional feature did not improve model accuracy significantly. Unlike prior work [22], we only used the previous core frequency and not a window.

Finally, as Figure 1B demonstrates, a linear model using only CPU utilization that minimizes the model error cannot predict the full dynamic range of the node. However, a piecewise linear model can predict the full dynamic node range. In the next section, we demonstrate this using the general cluster model.

## 5.3 General Feature Set

Ideally, we would be able to identify a unified set of model features that can be used for all platforms. After building the various platform-specific models, we noticed that at a high level, the models shared a lot of features. By selecting the features that were common across all the models and adding the most common

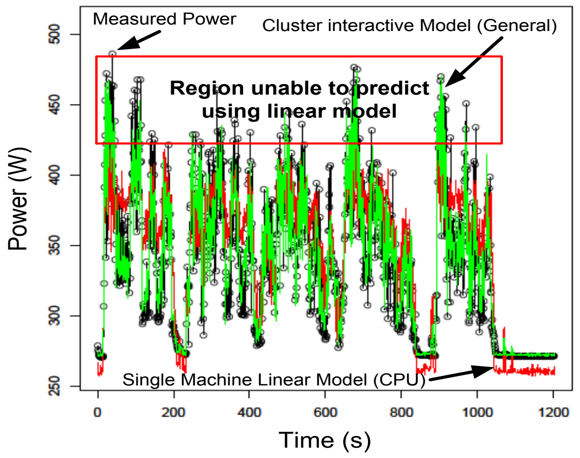

**Figure 2.** Worst-case full-system power prediction for the desktop (Athlon) cluster using a single machine linear power model compared to the cluster interaction model using the general feature set. Note the upper region of the graph where the linear model cannot predict power.

features from the categories that were not represented in this common set, we defined a general set of features that can be used across all platforms. This is the general model shown below:

$$P_{GeneralNode} = \hat{f}(x_{24}, x_{26}, x_{34}, x_{66}, x_{102}, x_{122}, x_{167}, x_{209})$$

This model simplifies model and feature extraction and provides the ultimate portability. Furthermore, the general feature set model reduced accuracy by less than 1% DRE, worst case and no more than 0.25% DRE excluding the worst-case outlier. As Figure 1A demonstrates, the general-model DRE is on par with the other models. Likewise, as Figure 2 shows, the general model can predict the entire dynamic range of the cluster, whereas the scaled single machine linear model that only uses CPU utilization does not predict the upper ~20% of the cluster power.

## 6. Conclusions

In this paper, we developed and validated high-fidelity cluster power models for six server platforms. These platforms covered a range of server designs proposed in recent literature [1,13,19,31,32] and used in current practice, from embedded- and mobile-processor-based systems to desktop- and server-processor-based systems. Our models are based on a statistically sound and automatic framework that is capable of absorbing a very large number of initial features and returning a tractable number of features used in a variety of models: Out of CHAOS comes clarity. All of these features can be collected by the OS to provide online power estimates. To the best of our knowledge, these models are the first to use OS-level counters to predict full-system and cluster power with this high level of accuracy. The use of high-level OS counters makes metric collection convenient and consistent across all the platforms we tested, unlike hardware performance counters and board-level measurements.

In order to evaluate and compare models across systems and workload types, we introduced a new error metric, called dynamic range error (DRE), based on the familiar mean squared error. This metric provides a frame of reference for model accuracy with respect to the application's dynamic power consumption range on a particular platform, making it easier to evaluate tradeoffs between cost and accuracy. This metric can also be used to compare the accuracy of models across platforms. Our cluster models demonstrate accuracy in the 0.5-2.5% range using metrics like median error and rMSE divided by average power, and under 10% using our error metric, DRE.

Our cross-platform results show that models based on CPU metrics alone do not capture the behavior of data-intensive cluster-level applications. Furthermore, disk utilization metrics significantly improve model accuracy even on systems with solid-state disks. This result is surprising since solid-state disks generally have low static and dynamic power consumption. For the data-intensive applications examined, disk utilization may also be a proxy for memory traffic. We also found that historical processor frequency information did not improve model accuracy. We only used the previous processor frequency and not a window as described in [22].

From the modeling perspective, we quantified the loss of using a unified set of features across disparate hardware platforms. This quantification enables informed decision-making about whether the effort of collecting these additional statistics is necessary in a given context. Finally, the general model can be used across multiple platforms and does not impact accuracy compared to platform-specific models, which can be deployed for online or offline cluster power prediction.

## 7. REFERENCES

[1] D. Andersen et al., "FAWN: A fast array of wimpy nodes," in *SOSP 2009.*

[2] W. L. Bircher and L. K. John, "Complete system power estimation: A trickle-down approach based on performance events," in *ISPASS 2007.*

[3] S. Bird et al., "Fixing performance counters: Performance monitoring hardware for the datacenter," in *Wkshp. on Architectural Concerns in Large Datacenters (ACLD)*, 2009.

[4] Y. Chen et al., "Managing server energy and operational costs in hosting centers," in *SIGMETRICS 2005.*

[5] J. Choi et al., "Profiling, prediction, and capping of power consumption in consolidated environments," in *MASCOTS 2008.*

[6] ClueWeb09 dataset. Available at http://boston.lti.cs.cmu.edu/Data/clueweb09/

[7] J. D. Davis et al., "Accounting for variability in large-scale cluster power models," in *Wkshp. on Exascale Evaluation and Research Techniques (EXERT),* 2011.

[8] D. Economou et al., "Full-system power analysis and modeling for server environments," in *Wkshp. on Modeling, Benchmarking, and Simulation (MoBS)*, 2006.

[9] X. Fan et al, "Power provisioning for a warehouse-sized computer," in *ISCA 2007.*

[10] J. Friedman, Multivariate Adaptive Regression Splines, 1991, *The Annals of Statistics*, 19:1-141.

[11] S. Govindan et al., "Statistical profiling-based techniques for effective power provisioning in data centers," in *EuroSys 2009.*

[12] J. Hamilton, "Annual fully burdened cost of power," Dec. 2008. Available at: http://perspectives.mvdirona.com/2008/ 12/06/AnnualFullyBurdenedCostOfPower.aspx

[13] J. Hamilton, "CEMS: low-cost, low-power servers for Internet-scale services," in *CIDR 2009.*

[14] T. Heath et al., "Energy conservation in heterogeneous server clusters," in *PPoPP 2005.*

[15] M. Isard et al., "Dryad: Distributed data-parallel programs from sequential building blocks," in *EuroSys 2007.*

[16] M. Isard et al., "Quincy: Fair scheduling for distributed computing clusters," in *SOSP 2009.*

[17] A. Kansal and F. Zhao, "Fine-grained energy profiling for power-aware application design," in *HotMetrics 2008.*

[18] A. Kansal et al., "Virtual machine power monitoring and provisioning," in *Proc. Symp. on Cloud Computing (SoCC)*, 2010.

[19] L. Keys et al., "The search for energy-efficient building blocks for the data center," *WEED 2010.*

[20] W. Lang and J. Patel, "Energy management for MapReduce clusters," in *VLDB 2010.*

[21] A. Lewis et al., "Run-time energy consumption estimation based on workload in server systems," in *HotPower 2008.*

[22] A. Lewis et al., "Chaotic attractor prediction for server run-time energy consumption," in *HotPower 2010.*

[23] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," in *SIGMETRICS* 2003.

[24] D. Meisner and T.F. Wenisch, "Peak power modeling for data center servers with switched-mode power supplies," in *ISLPED 2010.*

[25] Microsoft, "Windows 2000 Resource Kit Performance Counters, Counters by Object," available at: http://msdn.microsoft.com/en-us/library/ms803998.aspx

[26] P. Ranganathan, et al. "Ensemble-level power management for dense blade servers," in *ISCA 2006.*

[27] S. Rivoire et al., "A comparison of high-level full-system power models," in *HotPower 2008.*

[28] K. Singh et al., "Real-time power estimation and thread scheduling via performance counters," in *Wkshp. on Design, Architecture, and Simulation of Chip Multi-Processors (dasCMP)*, 2008.

[29] D. C. Snowdon et al., "Koala: a platform for OS-level power management," in *EuroSys 2009.*

[30] D. C. Snowdon et al., "Accurate on-line prediction of processor and memory energy usage under voltage scaling," in *EMSOFT 2007.*

[31] A. S. Szalay, et al., "Low-power Amdahl-balanced blades for data-intensive computing," in *HotPower 2009.*

[32] V. Vasudevan, et al., "Energy-efficient cluster computing with FAWN: workloads and implications," in *e-Energy 2010.*

[33] V. M. Weaver and S. A. McKee, "Can hardware performance counters be trusted?" in *IISWC 2008.*