CHAOS: Composable Highly Accurate OS-based Power Models

John D. Davis†, Suzanne Rivoire‡, Moises Goldszmidt†, Ehsan K. Ardestani§
† *Microsoft Research, Silicon Valley*
‡ *Dept. of Computer Science, Sonoma State University*
§ *Dept. of Computer Engineering, University of California Santa Cruz*
{*joda, moises*}*@microsoft.com, rivoire@sonoma.edu, eka@soe.ucsc.edu*

***Abstract*—Models of computers' power consumption enable a variety of energy-efficiency optimizations and reduce data center instrumentation costs. In this paper, we present Composable, Highly Accurate, OS-based (CHAOS) full-system power models for machines and clusters. CHAOS models, which use high-level OS performance counters, yield highly accurate predictions without the intrusiveness and portability problems of hardware counters or board-level instrumentation. Furthermore, they are automatically generated by a low-overhead software framework (less than 1% CPU utilization on a mobile-class processor).**

We evaluate CHAOS models using MapReduce-style workloads, executed on server-class systems as well as energy-efficient low-power desktops, laptops, and embedded systems. We also generate and validate a generic, cross-platform feature set for cluster power models. To facilitate comparisons across different models and platforms, we define a metric called Dynamic Range Error (DRE) to describe how well the model characterizes the dynamic system behavior. Using this metric, we quantify the tradeoffs between model complexity and accuracy for different workloads. Our results show that the generic cross-platform feature set degrades prediction accuracy by at most 1% DRE compared to power models using the best cluster-specific feature set. 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.

I. 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 [1], thus motivating techniques for power provisioning and planning, online power capping, and power-aware software tuning. All of these optimizations benefit from accurate models of computer systems' power consumption. Depending on the application, power models can exist as a complement to physical power measurement instrumentation or as a cost-saving replacement for it. Full-system power models are particularly useful for legacy systems and for new low-cost server designs, like the Open Compute Project, that cannot monitor server-level power consumption [2]. Furthermore, data centers are well suited for application-targeted power models because they repeatedly run large-scale batch applications, such as search index updating and analytics. To be suitable for large-scale use, power models require generic, portable predictors that are (1) highly accurate for emerging server designs and workloads and (2) can scale beyond a single machine.

In this paper, we build composable, highly accurate OS-based (CHAOS) power models for online use. At the core of CHAOS is a set of automatically generated full-system power models. These models accurately predict power based only on portable OS-level performance counters, requiring no hardware-based instrumentation. 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. Furthermore, our modeling techniques account for the significant power variability among identical components and systems [3, 4, 5], which allows our single-machine models to be accurately expanded to the cluster level.

To validate our approach, we build machine-level power models and then use them to compose cluster-level power models for six different clusters of homogeneous machines and one heterogeneous cluster. We also use our technique to identify a set of model features across all clusters that also yields high fidelity, pushing the model's validity beyond a single application to a group of applications with high CPU, network, and/or disk utilization. In order to conduct this model exploration, we build and evaluate over 1200 full-system power models per cluster using different combinations of predictors and modeling techniques.

The machines that make up the clusters span the embedded, mobile, desktop, and server processor spaces, reflecting energy-efficient server recommendations from recent research [6, 7, 8, 9, 10] as well as more traditional servers used in current practice. Our workloads are a variety of MapReduce-style applications running the same software stack. Figure 1 shows the total cluster-level AC power consumption of these workloads over five runs on our mobile-class cluster. Although the power signatures of these workloads differ greatly due to differing application characteristics, we seek to accurately predict their power using a single cluster power model.

To demonstrate the portability and generality of CHAOS's full-system power modeling framework, we make the following contributions:

- We demonstrate an automatic, generic framework that builds composable, high-fidelity cluster power models

978-1-4673-4532-3/12/$31.00 ©2012 IEEE

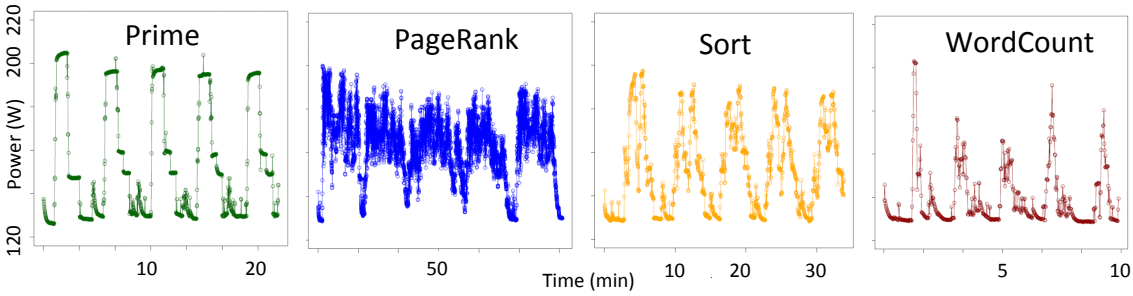

Figure 1. Full-system cluster power measured from 5 Intel Core 2 Duo machines for five runs of each workload. Each workload has different run times and dramatically different power signatures, with dynamic power ranges between 120W and 220W.

using portable OS-level performance counters. This approach has the following advantages:

  - Accuracy: Our models' error is less than 12% of the dynamic power range of our machines, a stricter error measure than prior work has used. The median absolute error ranges from 0.5-2.5%, which is on par with the most accurate prior work.
  - Portability: We use the same OS-level performance counter infrastructure to build our power models across six different platforms composed of different processor types from different vendors. The overhead of the power modeling is less than 1% CPU utilization and requires no specialized hardware.
  - Scalability: Our models are able to account for server-level power variability in the feature selection process and in the number of machines sampled to achieve a given error bound.

- 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 show the relationship between model complexity and accuracy for different workloads and present a general set of predictors that yield accurate models across the different clusters.

The rest of this paper is organized as follows. Section II provides a description of related work. Section III describes our clusters, workloads, and measurement infrastructure. Section IV explains our feature selection process and modeling techniques. Section V evaluates the power models and discusses insights and intuitions for high-level power modeling, and Section VI concludes.

II. Background

In this section, we examine previously proposed approaches to modeling full-system power and energy consumption. First, we categorize their results with respect to accuracy, portability, and scalability. Then, we categorize the models themselves with respect to the predictors used, the types of models, and the frequency of sampling. However, we do not discuss power modeling of individual components, such as CPUs, for several reasons. First, the goals of those models tend to be different: the threshold for accuracy is higher, the sampling frequency often must be higher, and generality and portability are sacrificed. Second, those models are typically much more fine-grained and detailed than is possible at the full-system level. Furthermore, even if it were possible to accurately model the power of each individual component, the full-system power goes beyond the superposition of components. It also includes "glue" such as power regulators, power supply inefficiency, chipsets, and a variety of other elements that contribute to both static and dynamic power. This makes techniques for modeling full-system power more high-level and less exact than individual component power.

Accuracy. Comparing model accuracy against previous work, which was evaluated on different platforms, is difficult because of the metrics that are typically used. In some papers, the only metric of accuracy is that the model was sufficient for some energy-saving technique [11, 12]. Other papers use metrics such as absolute error in watts [13] or mean or median relative error [4, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]. However, these errors are reported relative to total power rather than the dynamic power range, which means that they will appear more accurate for systems with relatively large idle power and/or small dynamic ranges. This makes it difficult to understand how well the model captures variation in power. Based on our results with a tighter error metric, our modeling approach yields models at least as accurate as any previously proposed generalizable approach.

Portability and scalability. Some models require application-specific profiling [14], while others were developed solely for processor-intensive workloads [16]. Rivoire et al. compared simple models for a breadth of machines and workloads but did not go beyond a single machine [17]. In fact, very few papers have examined the power consumption of a group of machines. Those

Table I
WE DEVELOP FULL-SYSTEM POWER MODELS FOR 5-MACHINE CLUSTERS OF THE PLATFORMS BELOW. *SYSTEM MAXIMUM MEMORY CAPACITY.

System Class	CPU, TDP	Power Range	Memory	Disk(s)	OS, FS
Embedded	Intel Atom, 2-core, 1.6 GHz, 8W	22-26 W	4 GB DDR2-800*	1 Micron SSD	
Mobile	Intel Core 2 Duo, 2-core, 2.26 GHz, 25W	25-46 W	4 GB DDR3-1066*	1 Micron SSD	Windows Server
Desktop	AMD Athlon, 2-core, 2.8 GHz, 65W	54-104 W	8 GB DDR2-800	1 Micron SSD	2008 R2,
Server	AMD Opteron, 4-core, dual socket, 2.0 GHz, 50W	135-190 W	32 GB DDR2-800	2 10K RPM SATA	NTFS
Server	Intel Xeon, 4-core, dual socket, 2.33 GHz, 80W	250-375 W	16 GB DDR2-667	4 7.2K RPM SATA	
Server	Intel Xeon, 4-core, dual socket, 2.67 GHz, 80W	260-380 W	16 GB DDR2-667	6 15K RPM SAS	

papers have focused on a particular machine or application and have not formally analyzed the process of scaling up a model developed on a single machine like this work has [4, 14, 25].

Choice of predictors. Early power models were based solely on CPU utilization [25, 15, 18, 11] or even CPU frequency state [12]. Subsequent work uses board-level measurements [13, 21] or hardware performance counters to capture the behavior of CPU, memory, and I/O devices [12, 19, 22, 23, 20]. 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 [26, 27].

Choice of power modeling techniques. Most previous work has used linear models [25, 15, 28, 13, 17, 24], piecewise linear models [16], or models that do not capture interaction between predictors [12]. Our work shows that more complex modeling techniques and robust feature selection are needed to capture the behavior and power variability of data center workloads on more recent machines using only portable OS-level metrics. However, Lewis et al.'s recent work found that even Multivariate Adaptive Regression Splines (MARS) models were inadequate to capture the behavior of the systems they studied, so they used chaotic attraction predictors [21]. We found MARS to be adequate, but the difference between our studies may be the choice of predictors; it is not clear whether they explicitly use processor frequency as a predictor or whether the model must infer it.

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 [18]. 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 [25] or modeled total energy over a workload [29, 23, 20], which misses application-level behavior patterns [30].

III. INFRASTRUCTURE

We evaluate the cluster power models across six homogeneous clusters with embedded-, mobile-, desktop-, and server-class processors *and* a heterogeneous cluster of mobile- and server-class processors, using a diverse set of data-intensive workloads; these clusters are further described in Table I. Although this infrastructure is specific to a Windows environment, it is publicly available [31, 32]. Furthermore, our approach is general and applicable to other operating systems with similar software tracing mechanisms.

A. Hardware and workload details

Our systems have different CPU voltage and frequency scaling capabilities, which affects the resulting power models. Starting at the low end, the Atom N330 does not provide dynamic voltage and frequency scaling (DVFS) and has the smallest dynamic power range. On the other hand, the mobile- and desktop processor-based systems both use DVFS, with the two cores in a system reporting the same operating frequency 99.8% of the time for our workloads. Finally, the server-class systems have the ability to have the cores operate in different P-states (frequency), and can transition the system into state C1 (processor frequency is 0 MHz), when all processors are idle. During our workload runs, Core 0 had a different operating frequency than at least one of the other cores up to 12%, and 20% of the time for the Opteron- and Xeon-based servers, respectively. This allows us to use one core's frequency as a proxy for all cores. Finally, we constructed a heterogeneous cluster comprised of Intel Core 2 Duo and Opteron servers as a final test of cluster power model composability.

We run an assortment of distributed MapReduce-style workloads using the Dryad and DryadLINQ application framework [33, 30]. All of the workloads are multithreaded, which fully utilizes all the cores in the system at some point during the workload run. Some of the workloads are CPU-intensive, while others are dominated by disk or network. We use different training and test data sets. Furthermore, even for the same data set, different machines may operate on different data partitions depending on the non-deterministic task scheduler. Training and model building requires up to 2 hours, depending on the cluster. This process can be incorporated into the normal system evaluation and characterization phase and thus can be done using a small collection of machines [3], removing or augmenting instrumentation from the install base in a data center. The workloads used are:

- **Sort.** This workload sorts 4 GB per machine 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 [34], 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 500 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 or disk activity.

Each workload has a dramatically different cluster-level AC power consumption profile. Our automatic power model generation framework builds a single machine power model that can be used in a cluster context despite machine-to-machine power variation and multiple different workloads.

B. 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 logged by ETW (Event Tracing for Windows) [31]. We collect data every second. This frequency is capable of measuring power spikes in the data center, which can happen on the order of a minute [35].

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 and compared different power meter readings across different machines. Machine-to-machine power variation can be as high as 10% at idle or under load.

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 log software counters once per second, including the WattsUp? Pro power meter readings [32].

IV. Modeling Methods

The OS provides a high-level view of system resource utilization and exposes performance counters, which can be observed and collected with a low-overhead tracing framework like ETW [31]. We explore various modeling techniques to capture the correlation between full-system power and these OS-level performance counters. Because these models are intended for online deployment, we favor simplicity in both the number of features and in the modeling techniques in order to minimize computational overhead whenever possible.

The first step in the process of building our models is feature selection, or identifying a subset of the many performance counters to use for modeling power. Windows Server 2008 exposes approximately 10,000 different system performance counters, and our objective is to reduce the number of counters to 10-20. Reducing the number of features reduces the overhead of collecting counters and computing predictions, and it also increases the robustness of the model to outliers and overfitting. To this end, we combined a set of well-known feature selection techniques and algorithms, resulting in the final algorithm and feature sets described in Section IV-A.

The second step is to fit a model to the selected features. In Section IV-B, we discuss the techniques we use to build power models for each cluster based on the chosen feature set. We use four different modeling techniques of varying complexity.

A factor we took into account in our methodology is variation in power consumption from machine to machine, which can be significant [3, 4, 5]. Most previous work assumes that a model built for a single machine can be generalized to a cluster of homogeneous systems [14, 25, 17, 24]. However, both the initial choice of features and the model coefficients associated with those features may significantly differ depending on the individual machine [3]. Our techniques for both feature selection and model fitting include steps to account for this machine-to-machine variation and build a model that can be accurately applied across the entire cluster. In both techniques, we incorporate variability by pooling information from individual machines in the cluster. In the case of feature selection, described in Section IV-A, our algorithm is based on the union of all the significant features from all the machines in the cluster. In the case of model fitting, described in Section IV-B, we pool performance counters and power measurements from all the machines in the cluster. An alternative approach is to build hierarchical Bayesian or mixed models [36]. This alternative adds an extra level of complexity in the modeling, the statistical fitting algorithms, and in the inference/prediction process. Fortunately, according to the results of the recommended statistical tests in [36], comparing the variances in the different models, pooling is a suitable approach with no significant loss of accuracy.

A. Feature selection

Current measurement infrastructure enables the collection of a large number of features – in our case over 10,000. This is a trend that seems only to be increasing. Our first task is therefore to explore methods for automatically extracting those features that are most relevant to the modeling of power. To this end, we will rely on Algorithm 1 and on well established regression techniques that are geared towards extracting the most relevant covariates (regressors/features) from regression problems with a large number of variables.

Algorithm 1 Feature reduction used to produce the cluster-specific model feature set.

Input: Correlation matrix and feature definitions

1: **if (Step 1)** Correlated features: $(a = ab) > 0.95$ **then**
2: Remove feature b
3: **end if**
4: **if (Step 2)** Co-dependent feature: $(a = b + c)$ **then**
5: Remove features *a and b*
6: **end if**
7: **while** Insignificant model features (f) **do**
8: **(Step 3)** Remove f_i where

$$f_i = \sum \alpha_j f_j, i \neq j, \ for\ colinearity\ test$$

9: **(Step 4)** Build new machine power model
10: **end while**
11: **for (Step 5)** $Each$ significant model feature across all machines (m) and applications **do**
12:

$$f_i = \sum m_j f_i$$

13: **end for**
14: **while (Step 6)** Insignificant **cluster** model features (cf) **do**
15: Remove f_i where

$$cf_i = \sum \alpha_j cf_j, i \neq j, \ for\ colinearity\ test$$

16: Build new cluster power model
17: **end while**

These techniques are L1 regularization and stepwise regression. The benefits of reducing the number of model features needed are: economy on data collection, the models induced are more robust to noise and outliers, and easier to interpret.

We began by identifying a subset of OS performance counters related to the activity of various hardware and OS components. The result was a set of approximately 250 counters taken from the following categories: processor, memory, physical disk, process, job object, file system cache, and network interfaces [37]. Different combinations of these features will capture the behavior of different platforms and workloads; there is no single optimal feature set that holds across all possible systems. Therefore, our strategy consists of developing an automated framework that can rapidly and easily build new models for applications, thus adapting to new characteristics and workloads.

Our initial, naïve strategy was to combine the counters from all machines in the cluster into a single set and then use regression techniques to identify the most significant features across the combined data. The problem with this approach is that, for our MapReduce-style workloads, the behavior of the different machines in the cluster is highly correlated. In this situation, statistical regression algorithms will, in an attempt to be parsimonious, end up eliminating one or more machines from the feature set entirely. This resulted in fragile workload-specific and even run-specific models that failed in the presence of small variations.

Therefore, we instead chose the strategy of separately building models for each machine in the cluster, and then iteratively selecting from the union of relevant features across the models to create an abstract machine-level model that captures all of the machines in the cluster. In Section IV-A1, we explain this feature selection algorithm, and in Section IV-A2, we present the features that were selected for different platforms and workloads.

1) Feature selection algorithm: The final procedure consists of six steps, outlined in Algorithm 1.

In steps 1 and 2 of the algorithm (lines 1-6 in Algorithm 1), we eliminate correlated counters from the feature set, since the presence of correlated counters tends to artificially inflate coefficients in the resulting models [38]. In step 1, we compute the features' pairwise correlation matrix across all workloads and reduce groups of features that have pairwise correlations greater than $|0.95|$, removing about 80 features. We performed a sensitivity analysis on this threshold value and found that reducing it below 0.95 provided diminishing returns. In step 2, we manually eliminate co-linear features based on the performance counter definitions [37]. After applying these two steps, we reduced the 250 features down to 50.

In step 3 (line 8 in Algorithm 1), we use a linear regression fitting with L1 regularization, which bounds the sum of the coefficients in order to eliminate irrelevant features in high-dimensional spaces [39]. In step 4 (lines 7-10 in Algorithm 1), we apply stepwise regression; that is, we iteratively eliminate features for which the Wald significance test [40] shows a low confidence that their coefficient value differs from 0. These two steps reduce the number of features to on the order of 10. Due to machine-to-machine variation, these steps did not yield the same 10 features for the different cluster types or even across all machines in the same cluster and workload.

The final two steps enable feature selection across multiple machines and workloads in a particular cluster by dealing with the differences between the machines. In step 5 (lines 11-13 in Algorithm 1), we begin with the union of the feature sets identified in step 3 for each machine and workload. We apply a weighted function to each feature to distinguish between the insignificant features that were eliminated during the stepwise regression in step 4 and those that were not eliminated (a weight of 1 for significant features and 0.1 for insignificant features).

At the end of step 5, we have a histogram of the features for each cluster, where each bucket contains the feature's weighted occurrence count from the different machines in the cluster. Based on these weighted occurrences, we use a low threshold to select the feature set for the final cluster-specific, machine-level model. Step 6 (lines 14-17 in Algorithm 1) is similar to step 4, except that step 6 uses the features selected in step 5 and the full cluster data set. We repeat these steps until no more features can be eliminated, and then we have a feature set that can be used for each

Table II
SIGNIFICANT PERFORMANCE COUNTERS USED IN CLUSTER POWER MODELS AND A GENERAL FEATURE SET THAT CAN USED ON ALL PLATFORMS.

Category	Performance counter	Atom	Core2	Athlon	Opteron	Xeon SATA	Xeon SAS	General
Network	Datagram/sec					X	X	
Memory	Page Faults/sec			X		X	X	
	Committed Bytes					X	X	
	Cache Faults/sec	X	X		X			X
	Pages/sec			X		X	X	X
	Page Reads/sec					X	X	
	Pool Nonpaged Allocs	X	X					X
Physical Disk	Disk Total Disk Time %		X		X	X	X	
	Disk Total Disk Bytes/sec	X			X		X	X
Process	Total Page Faults/sec					X	X	
	Total IO Data Bytes/sec			X				
Processor	Total Processor Time % (Utilization)	X	X	X	X	X	X	X
	Total Processor Interrupts/sec					X		
	Total Processor % DPC Time					X		
File System Cache	Data Map Pins/sec	X			X	X	X	
	Pin Reads/sec		X		X	X	X	X
	Pin Read Hits %						X	
	Copy Reads/sec	X						
	Fast Reads not Possible/sec	X					X	
	Lazy Write Flushes/sec	X				X	X	
Job Object Details	Total Page File Bytes Peak	X	X	X	X	X	X	X
Processor Performance	Processor_0 Processor Frequency		X	X	X	X	X	X

machine in the cluster.

2) Feature selection results: The final, cluster-specific model features for each platform are shown in Table II. This process removes feature selection variability across workloads and machines. It also guarantees that the features are independent, which is needed to extract stable models using the techniques described in the next section. Finally, it reduces the data collection and storage overhead as well as the computational complexity of the models, making them less expensive and appropriate for online use.

As Table II shows, different features are selected on different systems. In particular, the two Xeon systems, which have storage subsystems that consume significant dynamic power, use many more features associated with paging and the filesystem than the other systems do. None of these features are individually as significant as processor frequency or utilization, but together they capture the behavior of the I/O subsystem.

We also investigated simplifying feature selection by aggregating all the features from all the clusters and applying our algorithm again for each cluster across this superset of features to identify a new set of significant model features. This led to the "general" column in Table II. After the first three clusters, we found that adding new cluster feature sets did not dramatically change the general feature set. This may remove or simplify the feature selection process for similar platforms. However, different hardware may necessitate different features; the idea behind the feature selection technique we present here is to provide an automatic framework for quickly selecting features and generating robust models for new platforms.

Finally, Figure 2 illustrates the process of aggregating individual machines' features (steps 5 and 6 of Algorithm 1) for the Opteron cluster. In Figure 2, step 5 of the algorithm creates the stacked bars per feature across all machines and workloads for a cluster, where higher bars show counters that were identified as significant across more combinations of machines and workloads. As expected, processor utilization was the most commonly identified feature, and the importance of different features varied across workloads.

The horizontal line in Figure 2 shows the final threshold for the Opteron cluster. Features whose weighted occurrence count was above this line were selected for the model, and others were discarded. We initially started the threshold at a weighted occurrence count of 5, and the stepwise regression (step 6 of the algorithm) moved that threshold up to 7 for all platforms in our study. Conversely, if no insignificant features are found, the threshold can be reduced until that happens, providing a complelety automated feature selection heuristic.

B. Modeling techniques

We evaluated four different modeling techniques, listed below in Equations 1- 4, of varying degrees of conceptual and implementation complexity. In each equation, the full-system power is represented as a function $\hat{f}()$ of high-level OS counters represented by $(x_1, \ldots, x_n)$. For each technique, we also varied the number of model features, ranging from CPU utilization alone to the full cluster-specific and general feature sets shown in Table II.

We start with a basic linear regression model (Equation 1), where the parameters $(a_i)_0^n$ are fitted by minimizing the

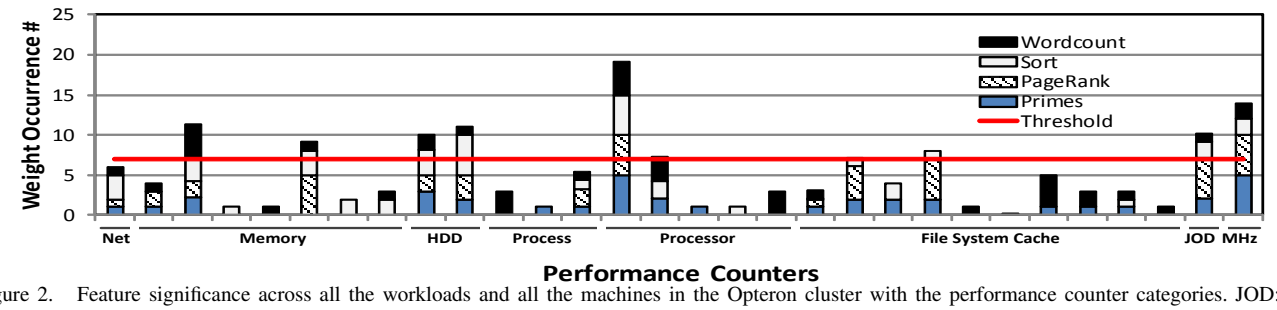

Figure 2. Feature significance across all the workloads and all the machines in the Opteron cluster with the performance counter categories. JOD: Job Object Detail, see Table II.

squared error. This is the form used by most previous work [25, 15, 13, 17, 24, 22]. It is a useful baseline against which we can compare all other proposals for $\hat{f}(x_1, \ldots, x_n)$ and evaluate the increase in accuracy of more complex models.

Baseline linear power model:

$$\hat{f}() = a_0 + \sum_i a_i \times x_i \tag{1}$$

Piecewise linear power model:

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

Quadratic power model:

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

Switching power model:

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

where I(f) = 1 iff x_i < threshold; otherwise I(f) = 0.

The piecewise linear power model (Equation 2) provides an extra degree of freedom. In this model, the parameter s can be positive (+) or negative (-), and the basis functions $B^s_{i,j}$ are hinge functions. $B^+_{i,j}(x,t)$ takes a value of 0 if $x = t$ and a value of $x - t$ otherwise. Similarly, $B^-_{i,j}(x,t)$ takes a value of 0 if $x > t$ and $t - x$ otherwise. The t thresholds are called knots and the j indices permit a feature to be responsible for multiple knots. 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 [41]. Intuitively, these models can express that a feature, such as CPU utilization, may consume different amounts of full-system power based on different regions of operation. These models are continuous, like the systems they model. They can model nonlinearity over the entire operational range, but be linear within each region.

The quadratic model (Equation 3) is an extension of the piecewise linear model that introduces nonlinearity within each segment 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, $I(f)$; each p-state/frequency can have its own linear model. The result is a set of (possibly) different linear models depending on the clock frequency. Unlike the piecewise model, where the knots only partition the space for a particular feature, the switching model's indicator function partitions the space for all the features, creating completely separate models for each frequency state. The switching model is more rigid, even though it may require more parameters (since we must fit coefficients for every feature at every frequency state) and it may have discontinuities at the knots, i.e., the frequency transitions.

Taking into account power and feature selection variability in the machine model makes composing cluster power models trivial, which we validate in the next section. We model the cluster power by summing the power predictions for each machine, $\hat{f}(x_1, \ldots, x_n)$, as shown in Equation 5:

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

V. Evaluation

All models are evaluated by using 5-fold cross validation with a training set about ten times smaller than the test data set. The training and test sets are taken from separate application runs, and the models must handle the fact that the job scheduler partitions the work differently among machines for different runs.

Table III
AVERAGE MACHINE DYNAMIC RANGE ERROR (DRE) COMPARED TO TWO OTHER COMMON METRICS: AVERAGE RMSE AND THE PERCENT ERROR (% ERR = AVERAGE RMSE/ AVERAGE POWER).

	Intel Core 2 Duo - Mobile			Atom - Embedded		
Workloads	rMSE	% Err	DRE	rMSE	% Err	DRE
Primes	2.69	8.7%	14.7%	0.57	2.4%	30.8%
Pagerank	2.74	8.1%	14.7%	0.64	2.6%	19.4%
Sort	2.19	6.7%	12.8%	0.69	2.8%	11.5%
Wordcount	2.22	6.8%	12.5%	0.64	2.6%	22.7%

A. The Dynamic Range Error Metric

Comparing model accuracy across platforms can be difficult. In prior power modeling work, error metrics have been confined to standard statistical metrics like mean squared error or absolute error. However, 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 is common in the literature for this reason, but it 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 and stringent error metric is the *dynamic range error (DRE)*, which we define as the root-mean-squared error divided by the dynamic range, as shown in Equation 6. As shown in Table III, a small rMSE, on the order of 2% of total power, can translate into a large DRE of 30% for systems using the Atom-based processor, the system with the smallest dynamic range. Furthermore, the Core 2 Duo-based systems has a large dynamic range, yet the other error metrics still overestimate the model accuracy. Thus, using DRE provides a platform-independent view of a model's explanatory power, demonstrating how well the power models actually model the dynamic range of the system. Finally, the use of DRE also enables the high-level results summary shown in figures 3 and 4, which apply both quantitatively and qualitatively to all workloads and clusters.

$$\textbf{DRE} = \frac{\sqrt{MSE}}{Power_{max} - Power_{idle}} \tag{6}$$

B. Cluster-specific Model Accuracy

The full-system power is non-linear in CPU utilization. Therefore a linear model will have to make compromises and as a result is not able to model the entire dynamic range of the system, especially at the high end of the range. As a consequence, the linear model loses accuracy. On the other hand, the piecewise linear and more complex models cover the entire dynamic range of the system using knots or other methods to provide basis functions to enable modeling of the different power regions, resulting in lower error. Figure 4 quantifies the increase in accuracy moving from simple linear models to more complex models and their ability to represent the full dynamic range.

Table IV shows the best average DRE for each platform and workload. Each entry is labeled with the modeling technique and feature set that yielded the best result. Overall, the quadratic model ('Q') and cluster-specific features ('C') yielded the best results, with some exceptions. The simplest models and feature sets only worked well for the simplest benchmark (WordCount) and the simplest system (the Atom-based system, which lacked DVFS). Overall, the models are highly accurate, with DRE less than 12% of the dynamic power range of the system for all models. Recall that this is a more stringent error metric than mean squared error or absolute percentage error, which is less than 2.5% for our models. These results show that accurate power modeling does not require low-level counters or instrumentation.

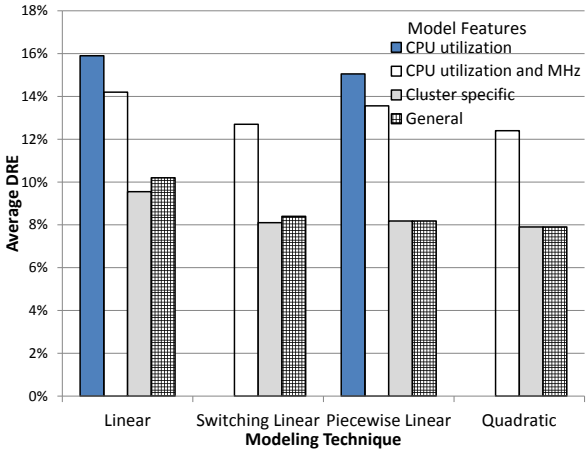

Figure 3. Opteron average DRE for representative workload (Pagerank), demonstrating that feature selection is required. Note: The quadratic and switching models do not use the CPU-utilization-only feature set because they require multiple features.

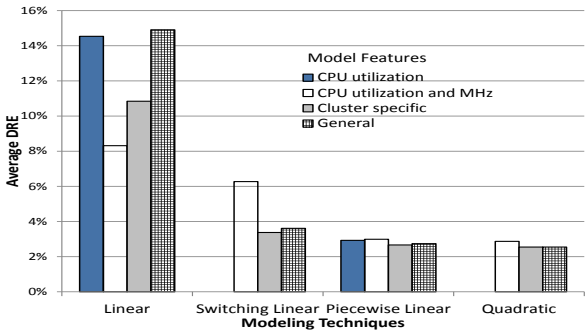

Figure 4. Opteron average DRE for representative workload (Prime), demonstrating that more complex models are required. Note: The quadratic and switching models do not use the CPU-utilization-only feature set because they require multiple features.

We next examine the question of whether the models' accuracy is due to feature selection process or the modeling techniques; the answer turns out to be different for different workloads. Figures 3 and 4 address this question by showing the DRE of all combinations of model types and feature sets on the Opteron for two different workloads. In both figures, the bars are clustered by modeling technique, and the different feature sets used to build the models are ordered from left to right in each cluster. Note that the quadratic and switching models require multiple features, which means that the CPU-utilization-only feature set does not make sense for these models.

Impact of feature selection. PageRank has high network

Table IV
BEST AVERAGE DRE FOR EACH WORKLOAD AND CLUSTER (DRE, MODELING TECHNIQUE AND FEATURE SET). PU: PIECEWISE LINEAR (CPU UTILIZATION), LC: LINEAR (CLUSTER FEATURES), SC: SWITCHING (CLUSTER FEATURES), QC:QUADRATIC (CLUSTER FEATURES), AND QCP: QUADRATIC (CLUSTER FEATURES + MHZ(T-1)), QG:QUADRATIC (GENERAL FEATURES).

Workload	Atom	Core 2	Athlon	Opteron	Xeon SATA	Xeon SAS
PageRank	9.2%, PU	7.4%, QC	8.9%, QC	7.7%, QCP	9.6%, QCP	8.1%, QCP
Prime	10.7%, QC	4.9%, QC	3.6%, QC	2.5%, QC	8.6%, QC	9.9%, QC
Sort	10.2%, QC	7.4%, QC	6.1%, QC	7.9%, QC	11.0%, QG	10.5%, QC
WordCount	11.4%, LC	9.8%, SC	6.0%, QG	7.6%, QC	9.8%, QC	9.2%, QC

utilization and over 800 tasks to run. It has the most power variation of all the workloads in this study and the longest running time. Figure 3 shows the dynamic range errors of various models and model features on the Opteron cluster running the PageRank workload. As Figure 3 shows, the DRE of the cluster-specific and general feature sets is up to 5 percentage points lower than that of the CPU-based models, demonstrating that for this workload, feature selection is more important than modeling techniques with respect to accuracy. WordCount exhibited similar behavior.

Impact of modeling techniques. Figure 4 shows that, for the CPU-intensive Prime workload, the choice of modeling technique is more important than the choice of feature set. Using piecewise linear models with one feature dramatically improves accuracy compared to a linear model. Sort also exhibited the same modeling behavior.

Heterogeneous clusters. An added benefit of building cluster power models based on a generalized machine power model is the ability to compose cluster power models for heterogeneous clusters, essentially for free. To demonstrate this capability, we constructed a 10-machine, heterogeneous cluster composed of Intel Core 2 Duo machines and Opteron machines. We scaled up the test data sets to maintain constant amounts of data and work per machine in the cluster. Then, we applied the appropriate machine power model (Core 2 Duo or Opteron) to each machine, resulting in the same worst-case 12% DRE as the homogeneous clusters. Thus, CHAOS power models could be used in a heterogeneous cluster environment for power capping and power-aware resource scheduling.

C. Cross-platform Model Accuracy

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 many features. By selecting the features that were common across all the models and adding the most common features from the categories that were not represented in this set (*i.e.* repeating steps 5 and 6 in Algorithm 1) for all clusters and cluster-specific features, we defined a general set of features that can be used across all platforms. This is the general model shown in Table II.

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 in the worst case, and no more than 0.25% DRE excluding the worst-case outlier. As Figures 3 and 4 demonstrate, the general-model DRE is on par with the other models. Likewise, as Figure 5 shows, the general model can predict the entire dynamic range of the cluster. In contrast, a strawman cluster model based on what prior work has suggested for cluster models — a scaled single-machine linear model that only uses CPU utilization — does not predict the upper ∼20% of the cluster power.

In general, although we selected workloads with a wide variety of characteristics: compute-, memory-, disk-, and network-bound, we do not claim that these general models are applicable for any and all workloads that run on this hardware. This is the main motivation for the automated model generation framework that can generate new workload-specific or multi-workload power models.

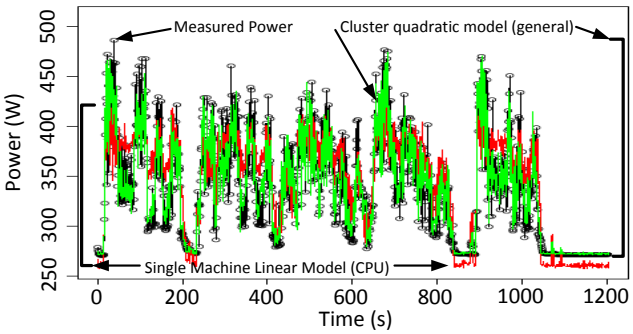

Figure 5. Worst-case full-system power prediction for the desktop (Athlon) cluster using a single machine linear power model compared to the cluster quadratic model using the general feature set. Note the upper region of the graph where the linear model cannot predict power.

D. Discussion

Together, figures 3 and 4 demonstrate that more complex models are required *and* that feature selection matters to produce high-fidelity models that can be applied across the collection of workloads. We have pushed this process further by also using the same technique to generate a general feature set for these workloads across all the platforms. In the future, as more power management options

become available, feature selection will continue to be important. Likewise, power management and the quest for more power/energy-efficient systems will mandate the use of non-linear models, as our results have demonstrated. However, linear models continue to be relevant as they are (a) very robust to noise and outliers, (b) valuable for implementing iterative feature selection search algorithms, and (c) easier to understand and interpret than more complex modeling techniques.

Feature selection. The CHAOS framework builds a single-machine power model across multiple applications. Feature selection is a critical component in building robust models. Using our algorithm has led to some interesting observations that can help with feature selection for future systems. Prior to Windows Server 2008 R2, core frequency was not available in the default set of OS-level performance counters and led to earlier models with much higher error because of the system's "hidden" frequency states. Because the processor frequencies were so highly correlated, we were able to use a single core's frequency as a proxy for the full system. Although one core's frequency was the dominant feature for the non-traditional servers, we did start to see a second core frequency start to emerge as a significiant feature for the traditional servers. Future systems with the ability to operate cores fully independently will have less-correlated core frequencies (less than 80%) and will require individual core frequencies as features. Furthermore, as Table II shows, more storage-related features were significant when systems had more disks. Moving forward, non-volatile memory, like high-performance SSDs, and other features and components, like per-core DVFS, core parking and GPU and accelerator activity with hidden system state will require performance counters that can capture this activity, areas of future work.

Modeling Technique. In general, our results demonstrate that all systems with DVFS require non-linear models to minimize the error. The Atom platform operates at a single clock frequency and does not require more features or more complex modeling techniques, similar to servers 10 years ago. Our evaluation indicates the necessity of more complex modeling techniques and judicious feature selection, like we have presented, given the use of more advanced power management techniques in the future.

Modeling Accuracy. Actual measured power is the gold standard of accuracy for all power modeling work. The more inaccurate a model is, the larger the necessary guard band for any mechanism that relies on its predictions. For example, in model-based power capping, inaccurate models would result in more conservative power caps and therefore would strand power. In resource allocation, inaccurate power models would require conservative provisioning with too few machines deployed in a fixed area, requiring more capital expenditures to meet performance demands.

Comparisons between the accuracy of our approach and prior approaches are difficult, since the results in prior work are system- and platform-dependent with respect to static and dynamic power ranges. However, our models match the reported accuracy using their metric for our systems. We have also used our platforms to evaluate some previously proposed modeling techniques (e.g. linear) and feature sets (e.g. CPU only); Figures 3 and 4 illustrate the limitations of these simpler methods. Finally, we have demonstrated a systematic approach in which we incorporated more modeling techniques and features until we reached diminishing returns with respect to accuracy.

VI. Conclusion

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 [6, 7, 8, 9, 10] and used in current practice, from embedded- and mobile-processor-based systems to desktop- and server-processor-based systems, and from homogeneous to heterogeneous cluster configurations. 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 relative error and rMSE divided by average power, and under 12% using our error metric, DRE. As future systems become more energy-proportional with larger dynamic power ranges and less static power, accurately capturing the dynamic range will be increasingly important.

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 the solid-state disks used in this study 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 significantly improve model accuracy. We only used the previous processor frequency and not a window as described in [21]. 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 or if a new model should be generated. Finally, the general feature set 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.

REFERENCES

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

[2] J. Park, "Open Compute Project: Data center v.1.0," Online. Available: http://opencompute.org/wp/wp-content/uploads/2011/07/DataCenter-Electrical-Specifications.pdf, April 2011.

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

[4] J. C. McCullough *et al.*, "Evaluating the effectiveness of model-based power characterization," in *Proc. USENIX Annu. Technical Conf.*, 2011.

[5] K. Rajamani *et al.*, "Power management solutions for computer systems and datacenters," in *Proc. ISLPED*, 2008.

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

[7] J. Hamilton, "Cooperative expendable micro-slice servers (CEMS): Low cost, low power servers for internet-scale services," in *Proc. Biennial Conf. Innovative Data Systems (CIDR)*, 2009.

[8] L. Keys *et al.*, "The search for energy efficient building blocks for the data center," in *Proc. Wkshp. on Energy-Efficient Design (WEED)*, 2010.

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

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

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

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

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

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

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

[16] K. Singh *et al.*, "Real time power estimation and thread scheduling via performance counters," in *Proc. dasCMP*, 2008.

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

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

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

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

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

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

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

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

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

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

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

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

[29] W. Lang and J. M. Patel, "Energy management for MapReduce clusters," in *Proc. VLDB*, vol. 3, no. 1, 2010.

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

[31] I. Park and R. Buch, "Improve debugging and performance tuning with ETW," *MSDN Magazine*, April 2007.

[32] Microsoft, "Joulemeter," Online. Available: http://research.microsoft.com/en-us/downloads/fe9e10c5-5c5b-450c-a674-daf55565f794/, 2011.

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

[34] "ClueWeb09 dataset," Online. Available: http://lemurproject.org/clueweb09/.

[35] L. A. Barroso, "Warehouse-scale computing: Entering the teenage decade," plenary talk, Federated Computing Research Conf. (FCRC), 2011.

[36] A. Gelman *et al.*, *Bayesian Data Analysis*, 2nd ed. Chapman and Hall/CRC, 2003.

[37] Microsoft, "Windows 2000 Resource Kit performance counters: Counters by object," Online. Available: http://msdn.microsoft.com/en-us/library/ms803998.aspx.

[38] T. Hastie *et al.*, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.

[39] M. Y. Park and T. Hastie, "L_1-regularization path algorithm for generalized linear models," *Journal of the Royal Statistical Society: Series B*, vol. 69, 2007.

[40] L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference*, 2nd ed. Springer, 2010.

[41] J. H. Friedman, "Multivariate adaptive regression splines," *Annals of Statistics*, vol. 19, no. 1, pp. 1–67, 1991.