Accounting for Variability in Large-Scale Cluster 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]
[john.d, moises]@microsoft.com suzanne.rivoire@sonoma.edu eka@soe.ucsc.edu

Abstract

Studying the energy efficiency of large-scale computer systems requires models of the relationship between resource utilization and power consumption. Prior work on power modeling assumes that models built for a single node will scale to larger groups of machines. However, we find that inter-node variability in homogeneous clusters leads to substantially different single-node models. Furthermore, these models have much higher error when scaled to the cluster level than models built using multiple nodes. We report on inter-node variation for model feature selection and model training for four homogeneous five-node clusters using embedded, laptop, desktop, and server processors. These results demonstrate the need to sample multiple machines in order to produce accurate cluster models. Furthermore, we determine the necessary sample size for the machines and applications in this study by applying a theoretical worst-case error bound based on the mean power interval across the cluster.

1 Introduction

Power consumption is a major concern in the design and operation of large-scale computing facilities [2]. It also presents a modeling and instrumentation challenge to researchers and infrastructure providers.

Physical instrumentation alone is not sufficient for challenges such as attributing power consumption to virtual machines, predicting how power consumption scales with the number of machines, and predicting how changes in utilization affect power consumption. These tasks require accurate models of the relationship between resource usage and power consumption.

Therefore, a substantial body of literature models system-level power consumption by sampling various metrics available in software (CPU utilization, memory bandwidth, disk utilization, etc.) and fitting them to the measured power consumption [3,4,6,8,10,12,14-18,20-23,25]. However, most of this previous work has built and validated models for individual nodes, with the implicit or explicit assumption that these models would extrapolate to the cluster level and beyond. Throughout this paper, we use node and machine interchangeably.

In this paper, we test that assumption by building node-level and cluster-level power models for four homogeneous clusters running MapReduce-style applications. The clusters include components from the embedded, mobile (laptop), desktop, and server processor spaces, reflecting energy-efficient server recommendations from recent research [1,9,13,24] as well as traditional servers prevalent today.

Our results clearly demonstrate that single-node power models do not scale to the cluster level:

- We show that the parameters chosen for single-node models by a standard feature selection process vary across individual nodes in a homogeneous cluster.
- We further show that, for a given set of features, the coefficients of a fitted single-node model are highly sensitive to the particular node.
- We apply a formal error bound to real cluster data to compute the number of nodes that must be sampled to accurately represent the cluster.
- We observe that node-to-node variation is distinct from, and an order of magnitude higher than, run-to-run variation on these four clusters.

The rest of this paper is organized as follows. Section 2 discusses the explicit and implicit assumptions of scaling power models in related work. Section 3 describes our hardware and software experimental infrastructure. Section 4 presents the model variability both in feature selection and cluster power model prediction. We conclude with Section 5.

2 Related Work

Previous studies model the power consumption of single nodes using different predictors and modeling techniques [3,4,6,12,14,16-18,20-23]. Some studies predict power consumption based on CPU utilization alone [6,18], while others use board-level measurements [16]. The modeling techniques also vary in complexity, from simple lookup-based models [20] to chaotic attraction predictors [16]. However, all of these studies build and validate models on a single node, assuming that these models can then be applied to other identically configured nodes without requiring refitting. We challenge that assumption in this work.

Other studies use different validation techniques. Li and John validate their routine-specific models on a full-system simulator [17], which again assumes no inter-node variability. Vasan et al. present power measurements from a medium-scale datacenter but only build single-node models [25]. Heath et al. model the total power of a four-node cluster; however, the cluster has little dynamic power variation, and they do

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

System Class	CPU	Memory	Disk(s)	OS, FS
Embedded	Intel Atom, dual-core, 1.6 GHz, 8W TDP [24]	4 GB DDR2-800*	1 Micron SSD	Windows Server 2008 R2, NTFS
Mobile	Intel Core 2 Duo, dual-core, 2.26 GHz, 25W TDP [13]	4 GB DDR3-1066*	1 Micron SSD	
Desktop	AMD Athlon, dual-core, 2.8 GHz, 65W TDP [9]	8 GB DDR2-800	1 Micron SSD	
Server	AMD Opteron, quad-core, 2.0 GHz, 50W TDP	32 GB DDR2-800	2 10K RPM SATA	

not try to scale their model to additional nodes [10]. Lang and Patel model the energy, rather than the instantaneous power, of a 24-node cluster [15]; it is unclear whether they do so by scaling the measured power consumption of a single node.

Finally, Fan et al. scale a single-node, CPU utilization-based power model to a few hundred servers [8]. However, they must add a large constant offset to the predicted power, which compensates for the constant power consumption of networking equipment as well as inter-node variations in idle power. They do not separate these two components of the added offset.

3 System Overview

We build models for four homogeneous five-node clusters running data-intensive, MapReduce-style applications. In this section, we describe the hardware platforms, the software infrastructure, and the workloads we use to build large-scale power models.

3.1 Hardware Infrastructure

Our systems have different CPU dynamic voltage and frequency scaling (DVFS) capabilities, which affects the resulting power models. Table 1 lists the features of these systems. Starting at the low end, the Atom N330 does not provide DVFS at all. This cluster also has the smallest dynamic power range, on the order of 15W over the entire cluster. On the other hand, the mobile- and desktop processor-based systems both use DVFS. For these two systems, the two cores on a single node report the same operating frequency 99.8% of the time for our workloads. Finally, the server-class system has the ability to have the cores operate in different p-states (frequency), and can transition the system into the C1 idle state when all processors are idle. For our workloads, the frequencies of the cores on a single server node differed up to 12% of the time.

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.

3.2 Software Infrastructure

Each system runs Windows Server 2008 R2, which provides a standardized OS-level performance counter interface. We measure a wide range of Event Tracing for Windows (ETW) performance counters provided by the OS. For each machine, we collect metrics, at 1 Hz, relating to the processor, memory, physical disk, process, job object, file system cache, and network interfaces [19]. Overall, we collect approximately 250 counters per node. Statistically redundant counters are removed through a systematic feature selection process, described in Section 4.1. We also verified that the data collection process does not interfere with program behavior or power consumption. Table 2 lists the final subset of performance counters used by the various cluster models (6-8 counters per model).

We ran an assortment of distributed workloads using the Dryad and DryadLINQ application framework [11]. These workloads are diverse; some are CPU-intensive, while others are dominated by disk and network. We run a single instance of each application at a time. One machine acts as the job manager, and the other four machines compute the tasks from the task graph. All workloads are run five times per cluster to allow each node to act as the job scheduler, which provides diversity in the work done even for the same application. The workloads used are described below:

- **Sort:** sorts 4GB of data with 100-byte records. The data is separated into 20 partitions, distributed randomly across the cluster. All of the data must first be read from disk and ultimately transferred back to disk on a single machine, so this workload has high disk and network utilization.
- **Staticrank:** runs a graph-based page ranking algorithm over the billion-page ClueWeb09 dataset [5], spread over 80 partitions on a cluster. It is a 3-step job in which output partitions from one step are fed as inputs to the next step. Thus, Staticrank has high network utilization.
- **Prime:** checks primeness of approximately

Table 2. The set of significant ETW performance counters used across all the cluster models.

Category	Performance counter	Ctr. ID
Memory (Mem)	Page Faults/sec	18
	Cache Faults/sec	24
	Pages/sec	26
	Pool Nonpaged Allocs	34
Physical Disk (PD)	Disk Total Disk Time %	54
	Disk Total Disk Bytes/sec	66
Process (Proc)	Total IO Data Bytes/sec	99
Processor (uP)	Total Processor Time %	102
File System Cache (FSC)	Data Map Pins/sec	121
	Pin Reads/sec	122
	Copy Reads/sec	126
	Fast Reads Not Possible/sec	139
	Lazy Write Flushes/sec	140
Job Object Details (JOD)	Total Page File Bytes Peak	167
Proc. Performance (MHz)	Processor_0 Frequency	209

1,000,000 numbers over 5 partitions in a cluster. It has high CPU usage but little network traffic.

- **WordCount:** reads through 50 MB text files on 5 partitions in a cluster and tallies the occurrences of each word. It has little network traffic.

4 Model Variability

We evaluated four classes of power models: linear, piecewise linear, interactive, and switching. For brevity's sake, we present only the best linear models for each cluster. The overall predicted cluster power is the sum of the single-machine models built using the metrics from each machine, as shown in Equation 1.

$$Power_{cluster} = \sum_{i=1}^{n} Power_{machine\,i} \quad (1)$$

The challenge was to produce a single-node model that provides the lowest root-mean-squared error across all workloads on the cluster. We report this error as a percentage of the cluster's *dynamic* power range; we refer to this metric as dynamic range error (DRE). Equation 2 gives the formula for DRE.

$$Error\ (DRE) = \frac{\sqrt{Mean\ Square\ error}}{Max\ Power_{Cluster} - Min\ Power_{Cluster}} \quad (2)$$

The linear power models for each machine in each cluster are as follows, with the subscripts referring to the counter IDs in Table 2:

Server (Opteron):

$$\hat{f} = \beta_0 + \beta_{26}a_{26} + \beta_{54}a_{54} + \beta_{66}a_{66} + \beta_{102}a_{102} + \beta_{121}a_{121} + \beta_{122}a_{122} + \beta_{167}a_{167} + \beta_{209}a_{209} \quad (3)$$

Desktop (Athlon):

$$\hat{f} = \beta_0 + \beta_{18}a_{18} + \beta_{26}a_{26} + \beta_{99}a_{99} + \beta_{102}a_{102} + \beta_{167}a_{167} + \beta_{209}a_{209} \quad (4)$$

Mobile (Intel Core2Duo):

$$\hat{f} = \beta_0 + \beta_{24}a_{24} + \beta_{34}a_{34} + \beta_{54}a_{54} + \beta_{102}a_{102} + \beta_{122}a_{122} + \beta_{167}a_{167} + \beta_{209}a_{209} \quad (5)$$

Embedded (Atom):

$$\hat{f} = \beta_0 + \beta_{24}a_{24} + \beta_{34}a_{34} + \beta_{66}a_{66} + \beta_{102}a_{102} + \beta_{121}a_{121} + \beta_{126}a_{126} + \beta_{139}a_{139} + \beta_{140}a_{140} + \beta_{167}a_{167} \quad (6)$$

In these equations, a_i is the value of counter *i*, and β_i is the regression coefficient for that counter; β_0 is the static power consumption. The next subsection describes how we selected these particular counters.

4.1 Feature Selection

We initially collected about 250 performance counters per machine and then reduced this set using a pairwise correlation matrix. Features with a correlation of |0.95| or greater to another feature were removed; reducing this threshold produced diminishing returns. We also used performance counter definitions to remove obviously co-dependent features. These two steps reduced the number of counters to about 45. Finally, we used linear regression to select the final set of performance counters on a per-node basis. Since we seek to understand the relationship between resource utilization and power, we chose a modeling process that preserves the initial feature set. Principal component analysis (PCA) is an alternative approach, but it sacrifices the ability to attribute changes in power directly to changes in resource utilization.

Figure 1 shows which performance counters are significant for each individual machine and each workload. The x-axis entries correspond to the performance counters or categories listed in Table 2. A counter is assigned a weight of 1 if it is identified as significant at the end of the stepwise regression on a particular workload and node. If it does not survive the stepwise regression, it is assigned a weight of 0.1.

As Figure 1 demonstrates, some features, like CPU utilization (102), are significant across all workloads and all machines, while others are not. Unfortunately, even for the same workload, this process selected different features for different machines, as the different bar heights in Figure 1 show. In order to select the best features for an overall cluster power model (Equations 3-6), we selected a deliberately low threshold, which started at 5, (the horizontal line in Figure 1) for the feature's significance in the individual datasets. We performed another stepwise regression on this superset of features to reduce it to a core set of significant features. For all clusters except the Athlon, this final stepwise regression eliminated features from the superset, features at or above the threshold line in Figure 1.

This process provided a set of cluster-specific features for each cluster power model. Defining a unified set of features that can be used across all clusters without degrading fidelity is future work.

4.2 Machine Variability

With the model features selected, we built single-node power models for each cluster and used two different methods to scale these models to predict cluster power. We estimate the cluster-level power models' error using five-fold cross validation.

The first method we used to predict cluster power was to build a model to predict the power of a single node, and then simply multiply this predicted power by the number of nodes in the cluster. Unsurprisingly, this method was highly inaccurate; yielding worst-case dynamic range errors of up to 150%.

The second method collects performance counter data from all nodes and applies the single-node model to each node in turn, summing the predictions. Figure 2 shows the results of this method. For each cluster, columns n1 through n5 show the dynamic range error when the cluster models are trained using data from only one node, each of nodes 1-5 and then applied to all nodes. The remaining columns show models trained

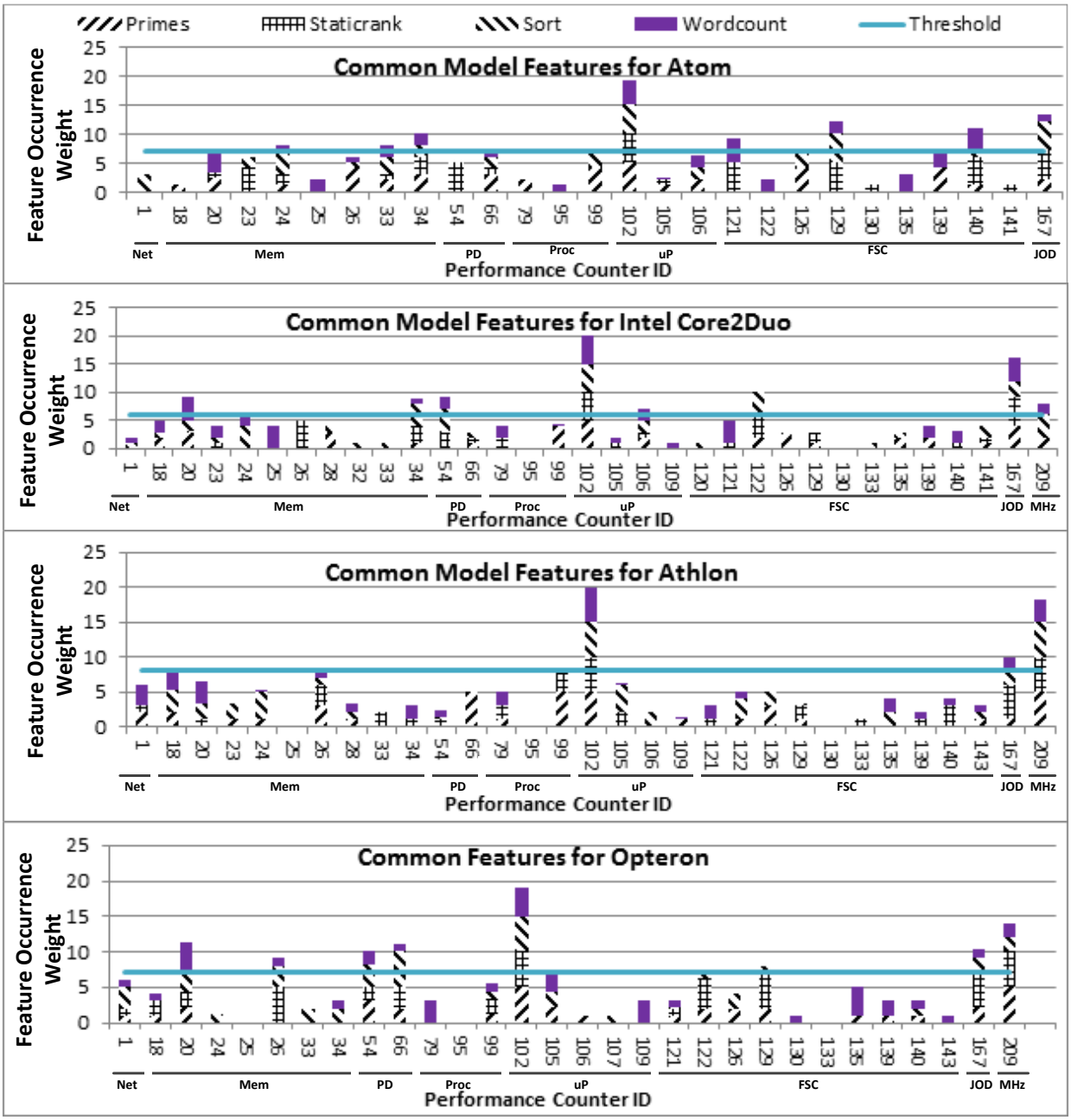

Figure 1. Feature significance across all the clusters and all the workloads. ETW categories appear under the range of counter IDs. Network (Net) is the only group not used in the models. Refer to Table 2 for the other categories. We use a heuristic to assign weights to significant features identified for each machine and for each workload and then select a common set of features across all workloads and machines in the cluster, signified by the threshold line. We then remove redundant features from this set.

using data from subsets of the five nodes (i.e. n12 is a model trained on nodes 1 and 2 and applied to the entire cluster). Using data from multiple machines is far superior to simply scaling a single node's power, decreasing the worst-case error to only ~50% for the Atom cluster compared to ~150% when multiplying a single nodes predicted power by N.

As Figure 2 shows, the machine power model trained using a particular node was sometimes a good proxy for cluster power model coefficients, while in other cases it was not. In general, as we added more machine data from different nodes of the cluster to train the model and determine the feature coefficients, the accuracy of the model improved, reducing worst-case error from ~50% down to less than 20% for the Atom cluster and 10% for the other clusters. However, for large-scale data centers, it is impractical to train the model with all the machines in the data center; Section 4.3 examines the question of the number of machines that must be sampled to meet a given error bound.

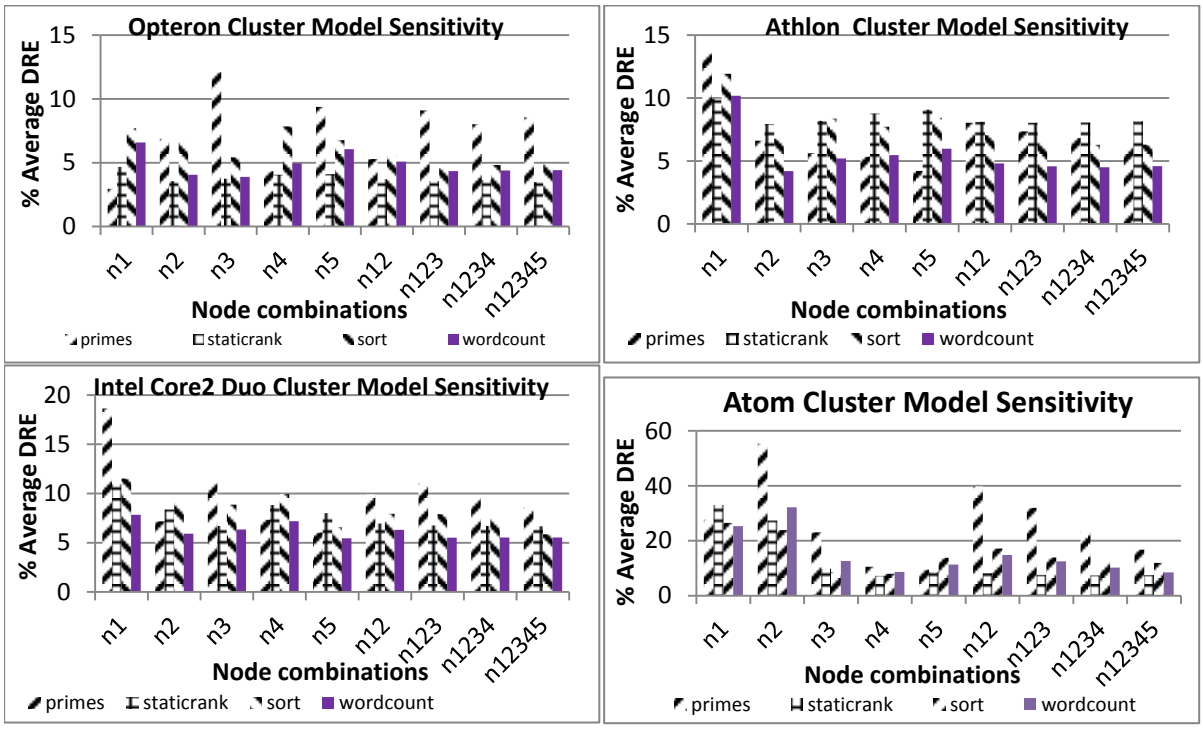

Figure 2. Sensitivity to the machine(s) used to train cluster models. Column labels identify the node(s) used in training.

Table 3. The first column group is the inter-node difference in idle power and average power for different workloads. The next two show the number of machines that must be sampled to obtain a sampling error (δ) no higher than the meter error with 99% and 90% probability, respectively.
* For these cases, the inter-node difference (I) is less than δ, which makes it unnecessary to sample multiple machines.

Clusters	Average power range					Sampled machines (@99%)					Sampled machines (@90%)				
	@idle (I)	primes (P)	staticrank (SR)	sort (S)	wordcount (WC)	I	P	SR	S	WC	I	P	SR	S	WC
Opteron	3.0	3.1	0.2	0.8	2.6	6.0	6.4	1*	1*	4.5	2.5	2.7	1*	1*	1.9
Athlon	2.9	7.7	6.5	3.8	2.2	89.1	628.3	443.6	154.6	51.3	3.1	22.2	15.7	5.5	1.8
Intel Core2 Duo	3.1	3.8	0.8	0.9	0.5	101.8	153.0	6.8	8.6	2.6	14.4	21.6	1.0	1.2	1*
Atom	2.0	0.1	0.2	0.2	0.2	42.4	1*	1*	1*	1*	6.0	1*	1*	1*	1*

4.2.1 *Application inter-run variation*

We also compared the run-to-run variation in idle power on the individual nodes to the machine-to-machine variation in idle power in the cluster. The inter-run idle power range for a single node was as much as an order of magnitude smaller than, and never larger than, the results presented in Table 3 for the cluster idle power range. These results demonstrate that multiple application run measurements on the same node are not sufficient to capture the inter-node variability that we have observed on the server cluster.

4.2.2 *Meter error vs. measured power ranges*

The Watts-Up Pro meter error is reported as 1.5%. When looking at the idle, average, and maximum power ranges across all the clusters and benchmarks, only the measured ranges for the Opteron cluster is less than ±1.5% of the possible meter error. All other clusters report measured power ranges greater than the meter error for at least one application on the cluster, as reported in Table 3. The error ranges have been omitted for brevity. Simply using measurement error does not capture machine variability for all the clusters.

4.3 *Sampling bound*

As Figure 2 shows, the prediction error decreases as we add more machines to the cluster model's training set. To determine the number of machines we must sample to achieve a given level of accuracy, we can use a classic Chernoff-Hoeffding bound [7]. This bound can be stated as $2e^{-\frac{2*\delta^2}{I^2}q} \leq P$. The variables in this equation are:

- δ: the allowable per-node difference between the sample nodes' average power consumption and the true cluster average
- P: the probability that the actual per-node error is greater than or equal to the desired value δ
- I: the inter-node difference in mean power
- q: the number of machines that must be sampled to obtain the desired values of δ and P, given I

We thus choose values for δ and P and solve for q,

the number of machines to sample. An advantage of this technique is that q is independent of the total number of machines in the cluster. We choose values of δ corresponding to the meter error for each system type. Note that δ is a sampling error due to inter-node variation; it is a separate topic from the prediction errors of the models in the previous sections.

Table 3 provides the range of the average power for each workload and cluster as well as the number of machines required to sample to build the model for P = 0.01 and 0.1. For the very tight bound of 0.01 (99% probability that the error is less than δ), Table 3 shows that the number of machines that must be sampled generally exceeds our cluster size of 5 nodes. Relaxing P to 0.1 (90%) yields more tractable sample sizes: less than a 40-node rack in all cases.

We can reduce the requirement on the sample size q, by making more benign assumptions on the variability of the power consumption of the nodes. The Chernoff-Hoeffding bound provides guarantees under the worst case scenario that there is significant likelihood that node power consumption will fall on the extremes of the interval I. If we have evidence that the variability of power consumption is concentrated on the variance (which can be verified empirically), we can then rely on tighter bounds.

We could also relax the value of δ depending on our application needs. Finally, we should note that the workloads on these clusters are not homogeneous, even when considering five runs, and this affects the interval I used to calculate the sample size q.

5 Conclusions

Previous work on power modeling assumes that it is sufficient to build and then scale a single-node power model for each system class of interest. For high-fidelity cluster power models, our results show that the choice of model predictors will vary from node to node. Furthermore, even for a given set of predictors, inter-node variability will result in different model coefficients when models are fit using data from a different single node. These variations in single-node models result in larger errors than using multiple nodes to train the models for predicting cluster-level power consumption.

We also observed greater inter-node measured power variation than run-to-run variation on a single node. Instead of using a single node, it is often necessary to build models based on a sample from the population of machines. We also examined the required number of sampled machines to achieve a given Chernoff-Hoeffding error bound for the data in this study. The number of machines to sample is independent of the machine population size and given reasonable parameters is on the order of a single rack of machines or less. We believe this to be a reasonable number in the context of planning and provisioning at the data center scale.

Finally, the combination of the portable (across different machine types) ETW framework, feature selection heuristic, sampling bounds, and standard statistical methods provides a methodology that can be applied to new clusters composed of different systems and/or new workloads to generate high-fidelity full system cluster power models.

6 References

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

[2] L.A. Barroso and U. Hölzle, "The case for energy-proportional computing," in *Computer*, 40 (12), 2007.

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

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

[5] ClueWeb09 dataset, available at: http://boston.lti.cs.cmu.edu/Data/clueweb09/

[6] G. Dhiman et al., "A system for online power prediction in virtualized environments using Gaussian mixture models," in *DAC*, 2010.

[7] D.P. Dubhashi and A. Panconesi, *Concentration of Measure for the Analysis of Randomized Algorithms*, Cambridge Press, 2009.

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

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

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

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

[12] A. Kansal et al., "Virtual machine power monitoring and provisioning," in *SoCC*, 2010.

[13] L. Keys, S. Rivoire, and J.D. Davis, "The search for energy-efficient building blocks for the data center," in Wkshp. on Energy-Efficient Design (WEED), June 2010.

[14] R. Koller et al., "WattApp: An application aware power meter for shared data centers," in *ICAC*, 2010.

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

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

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

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

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

[20] P. Ranganathan and P. Leech, "Simulating complex enterprise workloads using utilization traces," in *CAECW*, 2007.

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

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

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

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

[25] A. Vasan et al., "Worth their watts? - An empirical study of datacenter servers," in *HPCA,* 2010.