



UvA-DARE (Digital Academic Repository)

Evaluation of non-linear power estimation models in a computing cluster

Zhu, H.; Liao, X.; de Laat, C.; Grosso, P.

DOI

[10.1016/j.suscom.2016.02.002](https://doi.org/10.1016/j.suscom.2016.02.002)

Publication date

2016

Document Version

Final published version

Published in

Sustainable Computing: Informatics and Systems

License

Article 25fa Dutch Copyright Act (<https://www.openaccess.nl/en/policies/open-access-in-dutch-copyright-law-taverne-amendment>)

[Link to publication](#)

Citation for published version (APA):

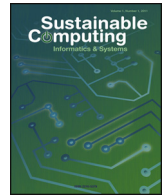
Zhu, H., Liao, X., de Laat, C., & Grosso, P. (2016). Evaluation of non-linear power estimation models in a computing cluster. *Sustainable Computing: Informatics and Systems*, 11, 26-37. <https://doi.org/10.1016/j.suscom.2016.02.002>

General rights

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.



Evaluation of non-linear power estimation models in a computing cluster



Hao Zhu^{a,b,*}, Xiangke Liao^b, Cees de Laat^a, Paola Grosso^a

^a System and Network Engineering, University of Amsterdam, The Netherlands

^b Department of Computer, National University of Defense Technology, Changsha, China

ARTICLE INFO

Article history:

Received 15 May 2015

Accepted 17 February 2016

Available online 3 March 2016

Keywords:

Power model

Workload characterization

Energy monitoring

Machine learning

Gaussian Mixture Model

ABSTRACT

The data center industry is responsible for 1.5–2% of the world energy consumption. Energy management technologies have been proposed for energy-efficient scheduling of computing workloads and for allocating resources in such computing infrastructures. One of the important factors for this energy management is the estimation of power consumption as a result of the workload schedule to be carried out. The commonly used power models are a linear function of resource features. Based on measurement data sets from our cluster, we extended power model study to multiple non-linear dependencies. We provide several novel contributions: we apply neural network models and unsupervised classification models with basic OS-reported resource features for power estimation; we build and test the power estimation models in a cluster environment from a large size of measurement data; we evaluate the power estimation models in terms of not only accuracy but also portability and usability.

We prove that a multiple-variable linear regression approach is more precise than a CPU-only linear approach. The neural network approaches have a slight advantage – their mean root mean square error is at most 15% less than that of the multiple-variable linear model. The neural network models have worse portability when the models generated on a node are applied on other homogeneous nodes. Gaussian Mixture Model has the highest accuracy but requires the longest training time. In the end, we prove that models trained using the system-level full features have the highest accuracy comparing to only use part of features.

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

Recent energy statistics indicate that the data center industry is responsible for 1.5–2% of the world consumption [1] and it would rank as high as the 5th among the countries of the world if considered as unique entity [2]. The predictions for annual increases in data center power demand are as high as 15–20% [3]. Moreover, energy use of data centers is starting to prompt environmental attention in terms of greenhouse gas emissions. Data centers worldwide are estimated to account for over 2% of global greenhouse gas emissions [2].

The data center owners (such as Google, Apple and Microsoft [4]) have started monitoring their energy consumption and employing various energy-saving technologies to lower their operating cost as well as reduce their environmental impacts.

Workload consolidation is one of the most effective solutions for modern data centers to improve the energy efficiency of the infrastructure [5–7]. Workload consolidation migrates workloads or virtual machines (VMs) from a set of current physical machine (PM) servers to a smaller set of servers. Data centers are usually under used, and rarely work at the maximum performance [8]. Workload consolidation can achieve high energy efficiency by switching off unused nodes and increasing overall usage of resources.

In order to minimize energy consumption while meeting performance objective when dynamically scheduling workloads, workload consolidation requires understanding of various cost factors. The most important factors are prediction of the incoming workloads on a server as well as estimation of the energy consumption of VMs or the server as a result of the workload schedule to be carried out.¹ Considering that energy consumption equals to power

* Corresponding author.

E-mail addresses: h.zhu@uva.nl, haozhu@nudt.edu.cn (H. Zhu).

¹ We do not estimate the power consumption of VMs in this paper. There is no VM migration between PMs. In this case, workload consolidation can be performed

Table 1

Correlation of the resource features and power consumption obtained by measurement data of one typical node in our cluster. Part of feature types include 2 features due to two disks and two Network Interface Cards (NICs) equipped in one node.

Type of resource features	Corr (feature, power consumption)
CPU usage	0.89
Memory usage	0.79
Disk I/O time	0.48 (sda), 0.49 (sdb)
Disk read speed	0.46 (sda), 0.46 (sdb)
Disk write speed	0.38 (sda), 0.38 (sdb)
NIC incoming speed	0.37 (eth0), 0.36 (eth1)
NIC outgoing speed	0.38 (eth0), 0.18 (eth1)
ALL page fault	0.28
Major page fault	0.27

consumption multiplied by time, it is necessary to estimate power consumption. *Power consumption estimation* computes the power according to current resource states from Operating System (OS) or Performance Monitoring Counters (PMCs). The majority of power estimation models used before are linear models [10,11], namely a linear function of resource features such as CPU, memory or disk load.

From the measurement data of our cluster, we found that power consumption was a not fully linear relationship with resource features, in particular, I/O load. Table 1 shows a few types of resource features measured in our cluster in the left column and the correlation coefficient of each feature and power consumption in the right column. The correlation coefficient of CPU usage and power consumption is close to 1, while the coefficient for other resource features such as disk load and network traffic load is small. In fact, I/O resources show non-linear behaviour, e.g. disk I/O operations have startup or stop delay of disk plates and seek time of disk heads. Previous work also proved that power consumption was not fully linear with CPU or I/O load [12,17,18]. Therefore, non-linear approaches exist an opportunity to achieve better accuracy than linear, only using basic resource features of CPU, memory, disk and NIC.

Polynomial models and classification models, which are two common non-linear approaches in the machine learning area, are suitable for power estimation. Polynomial models can capture the distinct change rate of power consumption to resources features while classification models can describe that power consumption is a function of resource features at different levels. We propose multiple non-linear approaches to train these models, and evaluate these models by comparing with highly used linear models. The power models provide the basis of controlling resources state for the schedule in future. To pursue high usability in the schedule, our approaches only select the basic controllable resource state in OS as their features.

The accuracy of trained power models depend on the executed workloads. We trained power models for servers on the data collected from long term previous runs rather than subjective benchmarking in a cluster. We consider three main reasons: First, there is no specialized benchmark for stressing all main server components. Existing benchmarks are mainly CPU or memory intensive but there is no standardized benchmarks for evaluating hard disk or network resources. Second, benchmarks are designed to stress some of the server components but they could not represent the actual workloads in a cluster. It is reasonable to train models using the actual workloads in the cluster. Third, the effect of machine learning methods depend on the size of a training set. Therefore, we collect a large size of measurement data for over 3-month from our cluster to train models.

The power models are evaluated on each selected node with actual workload and some benchmarks. We measure the accuracy of the models on Hadoop nodes and on nodes with GPUs, evaluate the portability of the models among homogeneous nodes and analyse the usability of the models by comparing execution time, CPU usage and the size of a training set needed.

The following are the main contributions of our work:

1. We apply neural network models and unsupervised classification models with basic OS-reported resource features for power estimation.
2. We build and test the power estimation models in a cluster environment from a large size of measurement data.
3. We evaluate the power estimation models in terms of not only accuracy but also portability and usability.

The structure of this paper is as follows: Section 2 presents related work on power estimation. Section 3 proposes the architecture of our monitoring and scheduling system; Section 4 focuses on characterization of cluster workloads; Section 5 shows our various approaches of building power models. Section 6 provides the result of evaluation on the models. Finally, Section 7 shows our conclusions and future work.

2. Related work

We differentiate between the concepts of *power consumption estimation* and *power consumption prediction*. Power prediction computes power consumption of nodes in future periods according to historical power consumption. Power consumption estimation contains the models for CPU, VM and server [19]. The models for VM and server is similar in terms of training approaches and features. The power consumption of a PM server can be calculated directly or as the sum of the power consumption of hosted VMs. It is therefore appropriate to discuss the state of the art about the power estimation of VMs and PM servers.

Many previous approaches have used linear models for estimating power consumption of a whole server. Fan and Weber [20] implemented a linear model based on CPU usage only. As Rivoire et al. [21] proposed, the CPU usage only linear model is not an accurate reflection of the CPU power, only suitable for CPU-intensive workloads.

The models that use both OS-reported component utilization and CPU performance counters are increasingly necessary for accurate power estimation. Heath et al. [22] proposed a linear model for a heterogeneous server cluster. The power consumption of individual servers is estimated with a linear model that solely employs utilization metrics. The key factor is to determine the utilization level of each resource for a single server. The total consumption of a resource is the sum of the fraction of requests served locally, the cost of sending requests to other nodes and the cost of serving requests on behalf of other nodes. The utilization of a resource is derived by the ratio of resource consumption to the capacity. Zamani and Afsahi [23] created linear models using different combination of features such as OS-layer resource feature, PMCs or various tasks execution cycles. Arbor et al. [24] analysed the correlation coefficient of PMCs and power consumption. They select 5 events with the highest correlation coefficients in their system-level power model, which is derived by multi-variable linear regression. Cache contention is considered as well. Economou et al. [25] also presented a full-system linear model, which considered the OS-reported features of main resource components such as CPU usage, memory usage, disk I/O rate and network I/O rate. The multiple-variable linear approach evaluated in this paper uses the similar features. All approaches above only generate a single

through task migration – moving independent tasks between PMs, or killing tasks in current PM and setting up them on a different machine [9].

Table 2
Summary of power estimation models for VMs and PMs; The default accuracy is on average if not specified.

Literature	Platform	Features	Approach	Benchmarks	Accuracy
Fan and Weber [20]	PM	CPU utilization	Linear	GMail; Google Search	<1%
Rivoire et al. [21]	PM	CPU utilization; PMCs	Linear	SPECcpu; SPECjbb; stream	8% (SPECfp); 9.5% (SPECint); 1.5% (stream)
Heath et al. [22]	PM	CPU utilization; network bandwidth; disk bandwidth; maximum number of concurrently open sockets	Linear	Micro benchmark	1.3% avg; 2.7% max
Arbor et al. [24]	PM	PMCs; Cache contention	Linear	SPEC CPU2000	2.54% (avg); 4.14% (max)
Economou et al. [25]	PM	CPU utilization; memory accesses; disk I/O rate; network I/O rate	Linear	SPEC CPU2000; stream	[0%,15%]
Kansal et al. [7]	VM	CPU utilization; LLCM	A set of linear	SPEC CPU2006	[1.6, 1.8] W
Koller [29]	PM	Throughput; virtualization ratio	A set of linear	TPCW; SPEC Power; HPL	<5%; <5 W (mixed)
Lent [18]	PM	The utilization of CPU, disk, NIC; memory access	Logistic	Web request	<5%
Dhiman [12]	VM	CPU utilization; instructions; memory accesses; cache transactions	Gaussian Mixture Model	SPEC CPU2000	[8%, 11%]; [1%, 17%] abs
Wabmann et al. [31]	VM	The utilization of CPU, disk, NIC	Polynomial	Synthetic CPU workload	<4%

linear model for one server. The accuracy of a single linear power model is limited for a variety of workloads.

The researchers address this challenge by characterizing workloads and generating a set of linear models for a server according to the workloads. A model is selected for power consumption estimation according to running workloads. Rajamani et al. [26] created linear models for CPU differentiated by value of parameters according to CPU operating frequencies and voltages. Li et al. [27] focused on the application of linear power model in a Cloud environment. They observed that low accuracy at peaks and valleys of power consumption. The improved linear model is divided into 3 subclass models according to the high, medium and low level of CPU utilization. Kansal et al. [7] learned a set of parameters for a model of each VM separately, based on the initial observation. The parameters for the VM can be shared if the hosting server has the similar configuration. But the methods of characterizing workloads in these literatures are not systemic, and are based on observations of limited number of works. These methods are only feasible for known or observable workloads with obvious patterns.

Besides the common features from OS and PMCs mentioned before, few studies employed novel features in their models. Dhiman and Rosing [28] proposed a feature, which is ratio of the instruction execution cycles to the overall cycles, to characterize the workload. They created an Application Power Table of each application for each server type hosting the application. Parameters of VM power model can be obtained from the table. Koller [29] introduced a throughput-based power model according to observations from a set of experiments on a mix of diverse applications.

A new trend is to estimate power consumption through non-linear approaches. Lien and Ying-Wen [30] suggested that a linear power model was only acceptable up to the medium utilization level. They built an exponential model for a web server, which is more accurate for different utilization. Lent [18] added a logistic function to model non-linear disk and OS behavior in a linear model for web servers. Wabmann et al. [31] computed power consumption of each VM in the server using multiple-variable polynomial approach. Their features are the utilization of CPU, hard disk and NIC in the VM Piga et al. [32] proposed a *k*-means model using a correlation-based feature selection. Zamani and Afsahi [23] considered a time-series factor in their model. Dhiman [12] used Gaussian Mixture Model (GMM) to estimate power consumption of each VM. The features they consider are IPC, MPC and CPU usage. They manually divide the CPU usage into many groups, and then cluster and fit each measurement vector using quantizer mismatch (QM)

distortion in each group. After that, multiple Gauss clusters are generated inside each group. When estimating the corresponding power consumption for a given feature vector, the closest cluster to this vector can be calculated using QM distortion. The estimated power consumption is the power value in the center of the cluster. We implement a different GMM regression approach. We employ the Expectation-Maximization (EM) [33] algorithm to find clusters, and use conditional probability to estimate power consumption. Their GMM obtains the same approximate estimation value for all feature vectors fit in the same cluster while our approach is more fine-grained. Our approach calculates the estimation value for different feature vectors, even which are in the same cluster, using Bayesian theory.

Table 2 summarizes the power consumption models for VMs and PMs we describe above.

3. System architecture

In order to further explain the role of power estimation models during the workload schedule in clusters, we describe our monitoring and scheduling system in a cluster environment.

Fig. 1 shows the system architecture. The computing subsystem executes users' jobs and monitors state of each node. The manager subsystem generates an energy efficient workload schedule plan the computing nodes should execute. The manager subsystem is deployed on the manager nodes. The system aims to perform automatic and dynamic workload schedule every fixed time period. We call each schedule period a time window. We explain each component in what follows:

Monitor module. The monitor module is deployed on each computing node. The monitor modules manage hardware-based sensors such as power distribution units (PDUs) and software-based information sources such as MDS [34] and Ganglia [35]. The monitor modules customize drivers for various information sources to produce measurement data.

Information module. The information module collects and organizes the measurement data from the distributed monitor modules. All the data is stored in a database system. The module provides interfaces for data access.

Models. Multiple models are built to support the workload prediction and power estimation, similar to the models in Yuan et al. [36]. For whatever approaches, parameters of models are trained based on historical data from the information module. But the metadata of historical data for approaches should be specified

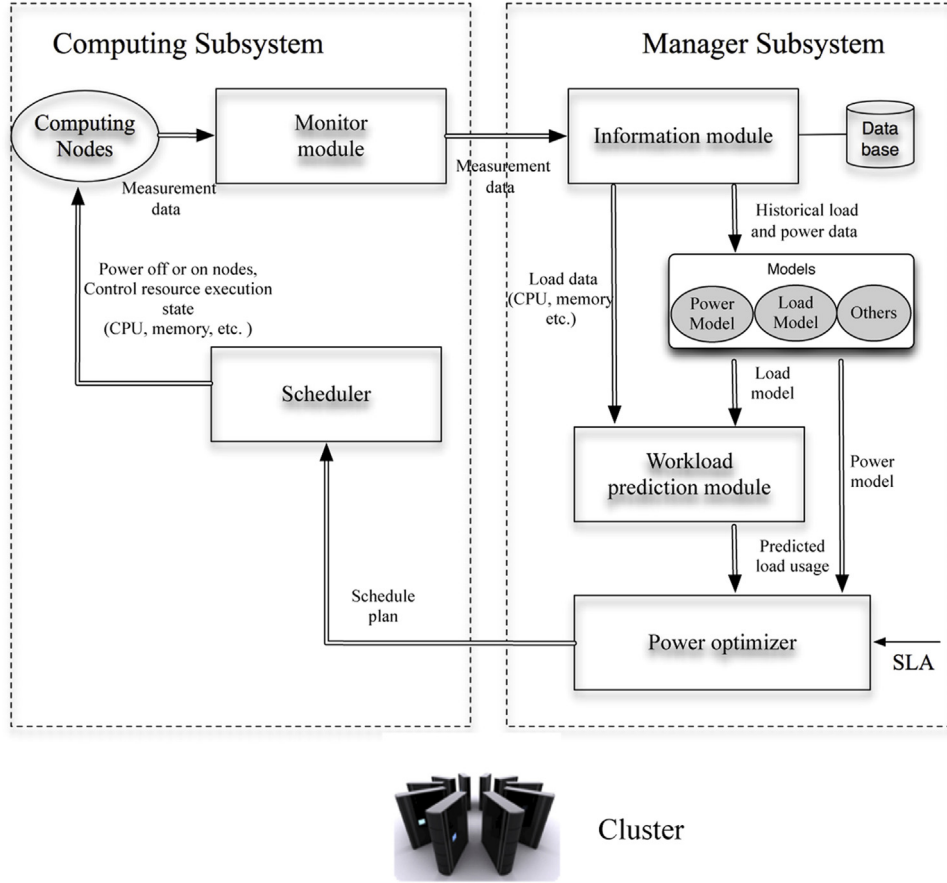


Fig. 1. The proposed monitoring and scheduling system architecture. A cluster consists of computing subsystem and manager subsystem.

Table 3
Parameters, models and decision variables.

Input parameters	α_i	Power weight of node i
	β	Penalty weight
	r_{SLA}	Average task response time in SLA
	T_k	Length of control time window k
	x_{k-1}^i	Resource loads of node i at window $k-1$
	N	The number of nodes in the cluster
Intermediate parameters	p_k^i	Power consumption of node i at window k
	r_k	Average task response time of cluster at k
	x_k	Resource loads of cluster at time window k
Models	F_p^i	Power estimation model of node i
	F_l^i	Load prediction model of node i
	F_r	Task response time model
Decision variables	x_k^i	Resource loads of node i at window k
	n_k^i	1 if node i at k is running, 0 otherwise

before training according to the resource features in models, as some approaches only consider the CPU usage data while some approaches concern more features. Other models are necessary to calculate execution performance, which is used to compare with the Service Level Agreement (SLA).

Workload prediction module. The workload prediction module leverages the load model to predict resource loads in the next schedule period. The module uses the resource loads in the previous and current time window to predict incoming load in the next time window k in Formula (1). The notations adopted in our system is summarized in Table 3.

$$x_k^i = F_l^i(x_1^i, \dots, x_{k-j}^i, \dots, x_{k-1}^i) \quad 1 \leq j \leq k-1 \quad (1)$$

Power optimizer. The power optimizer calculates the schedule plan for minimizing the power consumption of the cluster without heavily violating the SLA. In our system, the SLA is the average task response time of the cluster and the expected value should be predefined. The optimizer needs to estimate the power consumption and the response time of the cluster in the incoming window k using power estimation model F_p and response time model F_r .

The decision variables in the plan can be calculated by solving Formula (2) and (3) online. Decision variables are about how many nodes should be powered off or on in a cluster and how much resources of each node should be used. The objective function is the sum of energy cost and SLA violation's penalty cost. The average task delay can be calculated by resource loads. The response time model F_r can be obtained from queue theoretic models as Ardagna et al. [10] and Lent [18] did, or directly from historical data statistical analysis as Zhang et al. [11] observed.

$$\min \sum_i \alpha_i n_k^i p_k^i T_k + \beta(r_k - r_{SLA}) \quad (2)$$

subject to

$$\begin{aligned} p_k^i &= F_p^i(x_k^i) \\ x_k &= \sum_i x_k^i \frac{\sum_i n_{k-1}^i}{\sum_i n_k^i} \\ r_k &= F_r(x_k) \\ n_k^i &= 0 \text{ or } 1 \\ 0 &< \sum_i n_k^i \leq N \end{aligned} \quad (3)$$

Table 4
The configuration of target nodes (two groups) in our cluster.

Group	Node with GPUs	Hadoop nodes
# of nodes	7	8
Processor	E5-2620 (2.0 GHz), 12 cores	E5-2620 (2.0 GHz), 12 cores, hyperthreading enabled
Memory	64 GB	64 GB
Storage	2*1TB	2*1TB
NIC	IB and GbE	IB and GbE
GPU	7 K20m, 1 Phi	None

Scheduler. The scheduler receives the plan from the optimizer module and starts to allocate resources. It switches off and on nodes and migrates the tasks between nodes to control the usage or the rate of resources.

In our architecture, the role of power models when workload consolidation is to estimate the power consumption of a server at a given performance state, and finally to provide energy-saving solutions to controlling state of resources without heavily violating the SLA for the schedule in future.

4. Load measurement and workload characterization

Our power estimation study relies on measurement data, so we briefly introduce the collected data sets from DAS-4 cluster.

4.1. Load and power measurement

The DAS-4 cluster system [37] is mainly used for research purposes, such as to process and test students' homework, benchmark mathematical models and validate research ideas. We only targeted the single nodes with independent PDUs. We categorized 15 target nodes into two groups according to their use as shown in Table 4. One group is 7 nodes with GPUs which mainly run CPU intensive high performance computing; another is 8 Hadoop nodes which run short tasks that are assumed to be both CPU intensive and I/O intensive. We also run different types of benchmarks on the two groups for evaluation. The first group includes 7 nodes with K20m "Kepler GPUs" and one of nodes with an extra Intel Xeon Phi accelerator. They are equipped with dual Intel "Sandy Bridge" E5-2620 (2.0 GHz) processors. All of the Hadoop nodes are homogeneous, equipped with the same processors as the nodes with GPUs, but Hadoop nodes are hyperthreading enabled. All the nodes in the cluster are homogeneous in memory, disks and NICs. The OS is CentOS 6.2 x86_64. The Dynamic Voltage Frequency Scale (DVFS) capability of all the nodes is enabled, using default governor – "on demand".

The measurement data is stored in one previous developed semantic information system – Energy Knowledge Base (EKB) [38] for DAS-4 cluster. Semantic-web technologies improve data interoperability between different clusters. The EKB system collects measurement data from PDUs and Ganglia [35]. The type of resource features the information system can measure are listed in Table 1.

We generated one data set for each node from the EKB information system. Each data set includes the measurement data for 100 days of 2013. The data is sampled every 3 minutes. Each data vector in a data set, which is an observation from different features and power consumption at the same time stamp, is called one example in machine learning. Each data set contains about $100 * 24 * 60 / 3 = 48,000$ examples.

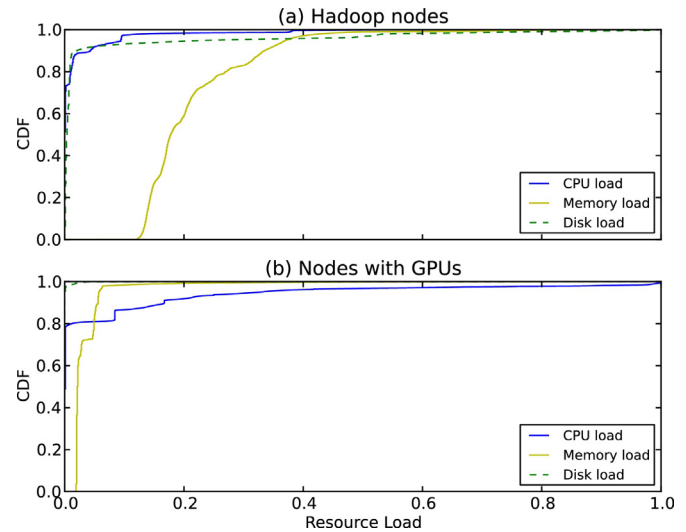


Fig. 2. The CDF distribution of CPU, memory and disk load in our cluster: nodes with GPUs vs. Hadoop nodes.

4.2. Workload characterization

The accuracy of power estimation approaches in a cluster depends on its workloads according to previous studies [18,39]. The approaches exhibit different effects in an I/O or CPU intensive environment. Some approaches are effective in an environment with mild workload pattern fluctuation, and are not suitable in a drastic fluctuation environments. For this reason, we characterize the workloads to understand our cluster environment before building estimation models.

According to the measurements data, the nodes with GPUs have a wide power range from around 130 to 330 W while Hadoop nodes only change at most 100 W between 110 and 210 W.

In order to clearly describe the distribution of nodes in terms of main resource features, we compute separate normalizations for each of resource fields. The normalization is a scaling relative to the largest capacity of resources on any machine. This normalization method is also used in Google cluster data trace [40]. In Fig. 2 we show the CDF distribution of CPU, memory and disk load for all the nodes. The resource loads are CPU usage, memory usage and disk I/O time respectively. For the most of time the CPU usage is close to 'idle'; this is expected as our cluster is not a production system. Nodes with GPUs run few I/O tasks. Hadoop nodes run more I/O intensive tasks than nodes with GPUs because Hadoop nodes have higher memory and disk load.

To understand the fluctuation of our workload pattern, we visualize load change of each node in our cluster. We compare the load change of our cluster with Google clusters, because they are known to exhibit strong workload variations. We compute the average load change of every 2 hours for each node respectively, and then draw the curves of changes between two consecutive periods in Fig. 3. The Google nodes have indeed more drastic load change than our Hadoop nodes and nodes with GPUs, because Google Cloud runs many short tasks, which are abundant in task types. Our observation to Google cluster is similar to what Reiss et al. [41] did. This shows that our cluster tends to run workloads with mild load fluctuation. The load in the Hadoop nodes exhibits higher noise than that in the nodes with GPUs, because the Hadoop nodes tend to run shorter term tasks compare with the nodes with GPUs. This further proves that the Hadoop nodes is more I/O intensive than the nodes with GPUs.

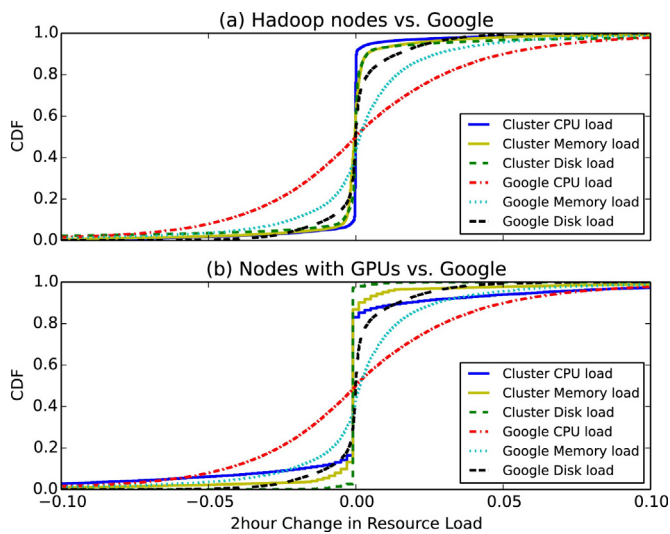


Fig. 3. Changes in average CPU, memory and disk load between consecutive 2 hours sampling periods: nodes with GPUs vs. Hadoop nodes vs. Google.

We conclude that the Hadoop nodes are an environment with CPU and I/O intensive mixed workload while the nodes with GPUs mainly run CPU-intensive workloads.

5. Power estimation approaches

In this section, we describe the linear and non-linear power estimation approaches to be evaluated. All the approaches are based on the common features. We first discuss the selection of the features from all the measured features.

5.1. Feature selection

As seen in Table 1, we can obtain the measurement data with 9 types of resource features. Two rules are followed to select the features for power estimation.

- First, we select at least one feature for each resource component. A server is composed of a few resource components; the main components are CPU, memory, storage and NIC.
- Second, we choose basic controllable features in OS for high usability of models. Power models are the basis of controlling resources state for the schedule in future. So even if the power consumption is estimated using the complicated features like CPU hardware counters, OS could not precisely and easily control the state of these features. The scheduling plan could be impossible to be executed.

According to the Rule 2, we exclude *page fault*. We calculate the correlation coefficient between *Disk IOTime* and *Disk Speed* based on the data. The coefficient is very high (more than 0.88). This means they are the same features for disks. We exclude *Disk IOTime* also. According to the Rule 1 we select 6 types of features: *CPU usage*, *memory usage*, *Disk write/read speed* and *NIC incoming/outgoing speed*. Effectively, we have 10 features due to two disks and two NICs equipped for each node in our cluster.

5.2. Approach description

The analysis of Table 1 and previous work in Section 2 shows power consumption has a non-linear relationship with resource features. We propose and implement two neural network approaches and one unsupervised classification approach to

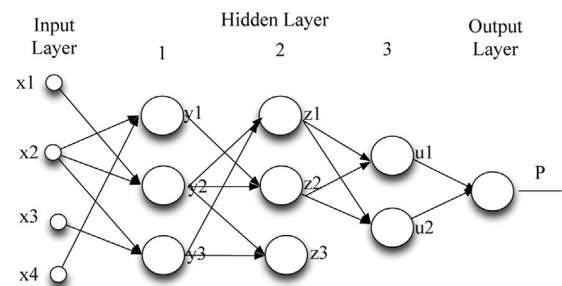


Fig. 4. An example of GMDH network structure.

capture this relationship. The change rate of power consumption is distinct for different ranges of resource loads. For instance, power consumption grows slowly as resource loads increase from idle [17] or resources meet the bandwidth bottleneck for intensive workloads [18]; power consumption could take off when medium resource loads. The polynomial function in the neural networks models can describe the non-uniform change rate of power consumption. A single set of parameters generated by linear or polynomial regression could not fully capture the complicated relationship between resource features and power consumption. This follows an observation from [12] – power consumption is a function of resource features at different levels. The relationship could be better represented using multiple sets of parameters generated by unsupervised classification models.

5.2.1. Linear and multiple-variable linear (M-Linear)

We implement two linear approaches; they both use an gradient descent algorithm to find the local minimum. One approach is the function of CPU only, named by **Linear**, which was used in early power estimation studies. Another is the multiple-variable linear regression with 10 features we selected, named by **M-Linear**, which is a direct optimization of the CPU only approach.

5.2.2. Artificial neural network (ANN)

A feed forward neural network is one of typical artificial neural networks, named by **ANN**. Data flow always moves in one direction to calculate weights for each neuron. ANN adjusts the weights using the gradient descent. Our ANN uses 2-degree polynomial functions in the neuron. The structure of our ANN network is predefined. The ANN network structure has 10 input variables in the input layer and 1 output variable in the output layer, because we input 10 features and only output power consumption. Based on our research experience on neural networks, we define a single hidden layer but with a large number of neurons in the layer.

5.2.3. Group method of data handling (GMDH)

GMDH [13] is also known as polynomial neural network, which can be represented as a set of neurons in which different pairs in each layer are connected through a polynomial function, thus producing new neurons in the next layer. The structure of a GMDH network is not predefined but evolves during the estimation process. Fig. 4 shows an example of GMDH network with 4 inputs and 1 output. The number of input variables in each neuron is 2 and the maximum number of neurons in hidden layers is 3. Fig. 4 shows an example of GMDH network with 4 inputs and 1 output. The number of input variables in each neuron is 2 and the maximum number of neurons in hidden layers is 3.

Similar with ANN, GMDH use all 10 features as input variables and power consumption as a output variable. Our GMDH network training algorithm proceeds as follows:

1. Separating one data set into training set and validation set.

2. Creating the combinations of the m input variables in each layer. In the first hidden layer, we input 10 features, so we create $\frac{10!}{m!(10-m)}$ combinations.
3. Calculating the regression for each combination. Each neuron is a d (2 or 3) degree polynomial that captures relationship between power consumption and the combinations. We estimate the weights of the neuron by the least square method for each combination. After regression, we compute estimation errors on the training set.
4. Selecting the intermediate neurons. Select the maximum n best neurons from all combinations according to the errors. All errors of the selected neurons should be less than the largest error in current layer. These n intermediate neurons are set as input variables of the next layer.
5. Computing the errors of the selected neurons on the validation set. When the least error on the validation set for neurons in next layer stops decreasing if compared with the least error on training set in the current layer, stop training. Otherwise, jump back to step 2 to construct the next layer.

5.2.4. Gaussian Mixture Model (GMM)

GMM [14] is a parametric probability density function represented as a weighted sum of Gaussian cluster densities. Different from the two neural networks we talked above, GMM is an unsupervised method, usually used as information classification. One of input parameter is the number of Gaussian clusters, denoted by c . The classification process is to find the parameters about each cluster, including cluster center, cluster co-variances matrices and mixing coefficients.

We leverage the GMM algorithm for regression because other classic classification methods such as K -means are not accurate for regression [15]. The process of GMM regression is as follows. First, 10 features together with power consumption are all as input variables during training. Second, we use iterative EM to find the model parameters. Third, given the value of one feature vector, this feature vector's conditional probability density function can be obtained based on the model parameters [16]. So the cluster center, cluster co-variances matrices and mixing coefficients of conditional probability density function is known. Finally, the power consumption for this given feature vector can be calculated as the expected value of the conditional probability density function, which is the sum of the mix coefficient for each cluster multiplied by the center of the cluster.

5.3. Method of training models

We train the models for each node using the **Linear, M-Linear, ANN, GMDH and GMM** approaches. Before training, we classify each 100-day data set into three groups. The **Training set (TrS)** is used to fit the models and find the model parameters, for instance, for computing the weights for each neurons in GMDH and ANN, cluster parameters in GMM and weights for each features in the linear models. The **Validation set (VS)** is used to decide the proper input parameters and to prevent overfit of models, and the predefined input parameters for each model is shown in **Table 5**. In cases where training is performed too long or where examples in TrS are rare, overfit may take place. The performance on the TrS increases while the performance on unseen test set becomes worse. For one set of input parameters for an approach, we obtain one model by training. We can get a couple of models for the approach because of different value of input parameters. We choose the model with the optimal input parameters, which causes the least root mean square error on the validation set. The **Test set (TeS)** is used to evaluate the effect of the chosen model. We set the ratio of TrS size, VS size and TeS size as 6:2:2. The size is the number of examples on each set.

Table 5
Predefined input parameters.

ANN	n	Number of neurons in each hidden layer
GMDH	v	Number of input variables for each neuron
	d	Degree of polynomial function
	n	Maximum number of neurons in the hidden layer
GMM	c	Number of clusters
Linear	λ	Regularization coefficient
M-Linear	ϵ	Learning rate

We use a random data set classification method to mitigate the locality of workloads. If the TrS and VS have widely different workload patterns, this could influence the choice of models on the VS and finally impact the evaluation result on the TeS. We shuffle all the examples in each initial data set according to time stamps. All the examples are random in the time sequence. From an overall data set, we fetch the first 60%, the following 20% and the remaining 20% of all the examples into the TrS, VS and TeS. We shuffle and classify the data sets 50 times. For one shuffle, we train, select and evaluate model one time. All the evaluation results we show in the next section are the mean value of 50 times.

6. Evaluation

6.1. Metrics for accuracy

We introduce two metrics to evaluate the accuracy of estimation: Root Mean Square Error (RMSE) and success ratio. For one node, RMSE measures errors between the estimated power values and the measured values on its test set. The size of examples on the estimated data set are M . We denote the m th real power consumption value of node i in this set by $P^i(m)$. The RMSE of one node i can be represented as:

$$RMSE(i) = \sqrt{\frac{1}{M} \sum_{m=1}^M (p^i(m) - P^i(m))^2}$$
 (4)

In addition, we compute success ratio, which is the ratio of the number of accurate estimations to the total number of estimations. An estimation is deemed accurate if it falls within some Δ of the real value. In our case, Δ can be 1%, 5% or 10%. The formula for computing the success ratio of one node i is shown as:

$$SR(i) = \frac{\sum_{m=1}^M g(m)}{M}$$

$$g(m) = \begin{cases} 1 & \frac{p^i(m) - P^i(m)}{P^i(m)} \leq \Delta \\ 0 & \text{otherwise} \end{cases}$$
 (5)

Success ratio and RMSE are two different metrics. RMSE measures the mean performance of multiple estimations, while success ratio depicts the distribution of accurate estimations. If the estimations have high errors very few times and most estimations are precise, they may show high RMSE but high success ratio.

6.2. Accuracy result

We compare the RMSE and the success ratio of all nodes on the originally shuffled test set, and then we analyse the results on nodes with GPUs and on Hadoop nodes respectively for various benchmarks.

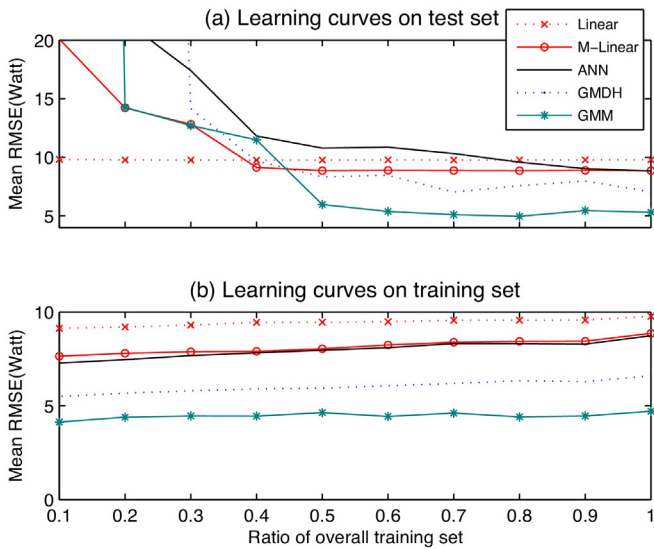


Fig. 5. Learning curves on the test set (a) and on the training set (b). The mean RMSE of all the nodes on the test set (a) and on a portion of training set (b) when training the estimation models on a portion of overall training set.

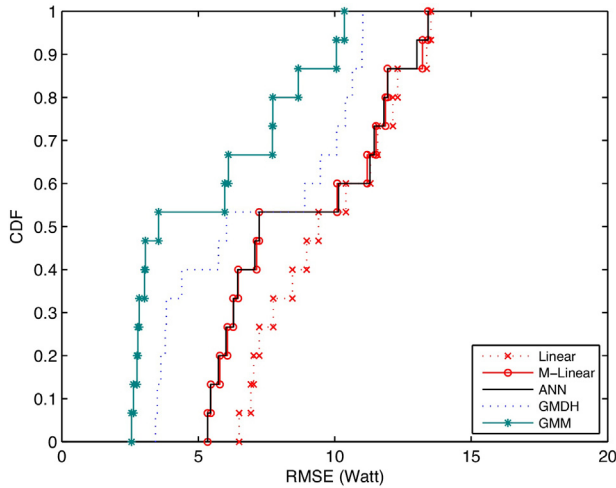


Fig. 6. The CDF distribution of RMSE of all the nodes on the test set.

6.2.1. On the test set

To validate whether the models we train exist overfit problem, we first evaluate the learning curves of the various models, which capture the change of the estimation errors on the TrS or on the TeS when the size of TrS increases slowly. In the process of obtaining these learning curves, the size of TeS is fixed at 20% of the overall data set while the training set increases from 10% to 100% of the original overall training data set.

The final learning curves are shown in Fig. 5. The X-axis is the size of TrS, while the Y-axis is the mean RMSE of all nodes. The errors of any model on the training set and on the test set are very close and nearly converge as the TrS size increases. This tells us that all the models reach the same state without suffering overfit when using 100% of TrS. From Fig. 5(a), we see that both linear models show good performance even on the small size of TrS while non-linear models do not reach their best performance until using 50% of the TrS. This proves that the non-linear machine learning approaches require a larger size of train set than linear approaches. So far, we have shown there is no overfit problems when training our models on the overall training set.

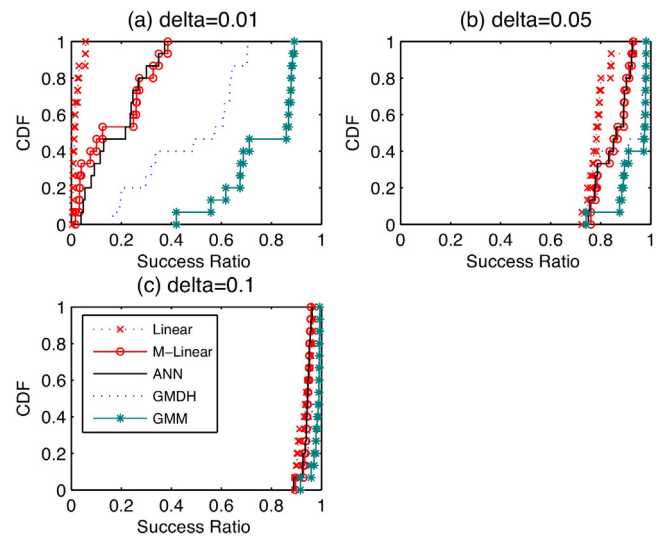


Fig. 7. CDF of success ratio of all the nodes with the Δ 1% (a), 5% (b) and 10% (c) on the original test set.

Fig. 6 depicts the CDF distribution of RMSE of all the nodes. GMM and GMDH show the lowest RMSE for all the nodes, and have good estimation effect. ANN has the similar RMSE error with the M-Linear model. For Hadoop nodes, the Linear model has slightly higher RMSE than the M-Linear model; for nodes with GPU, Linear model has some performance with M-Linear model. For the Hadoop nodes, the CPU is not the only one component that dominates the power consumption of the whole node considering that many I/O intensive tasks run.

Fig. 7 is the CDF distribution of success ratio of all the nodes with the Δ 1%, 5% and 10%. Success ratio goes up with the increase of Δ. Although the Linear model has similar RMSE performance with M-Linear and ANN for some nodes, the success ratio of the Linear model is lower than that of M-Linear and ANN for all the nodes. The success ratio of ANN is nearly identical with that of M-Linear. GMM has the best result when Δ is 1%, and the gap between GMM and GMDH becomes narrow when Δ increases. When we set the Δ is 10% in Fig. 7(c), the success ratio of all the models is more than 90%. This is to say less than 10% of estimations have the ratio of the estimation error to the real power consumption over 10%.

On the test set, we conclude that GMM and GMDH perform better than other approaches, because they capture more complex data dynamics and key feature factors. Among neural networks, GMDH is better than ANN approach. The reason is that GMDH network structure is solved by means of an adaptive synthesis during estimation process while ANN is simply predefined. The structure of ANN depends on experience. Moreover, ANN is difficult to guarantee global convergence. Simple-structure ANN has no advantage over M-Linear. M-Linear is better than the Linear approach because it captures I/O features. Due to shuffled data sets, the TeS has similar workload patterns to the TrS. The 5 approaches exhibit high RMSE and success ratio accuracy on the TeS.

Next, we emulate the approaches for some benchmarks, which probably have different workload patterns.

6.2.2. For the benchmarks

We further study the estimation performance for two types of benchmarks. The benchmark information is shown in Table 6. We run Linpack on the 4 nodes only with a single GPU and we run MapReduce benchmarks on the 8 Hadoop nodes. MapReduce Writer is a low CPU usage but I/O intensive benchmark, which writes random keys and values into a big HDFS file. MapReduce Sort is a CPU-intensive benchmark that sorts the data generated by

Table 6
Benchmarks and target resources.

Benchmark	Parameters	CPU load	Target resources
MapReduce Writer	20 G data per map; 10 maps per node	5–20%	8 Hadoop nodes
MapReduce Sort	200 G per node; 10 maps/reduces per node	75–100%	8 Hadoop nodes
Linpack	N: 40000; P*Q: 12, 24, 36, 48	25%, 50%, 75%, 100%	4 nodes with GPUs
Linpack-GPU	N: 40,000; P*Q: 24	10–20%	4 K20 GPUs

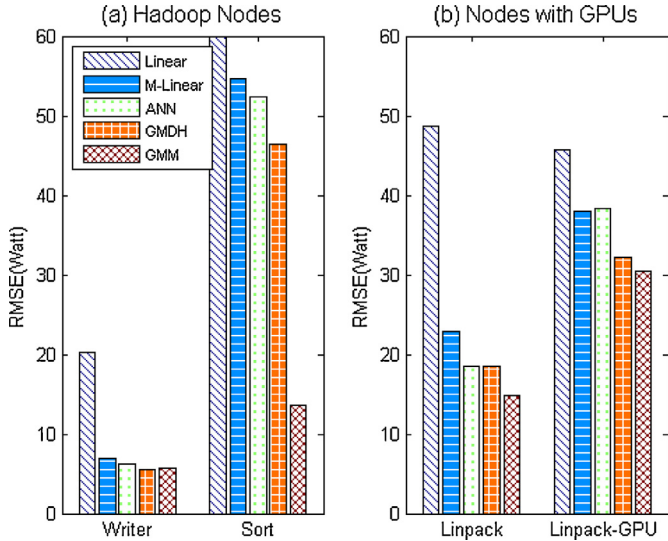


Fig. 8. Mean RMSE of nodes for the benchmarks. (a) The MapReduce benchmarks includes Writer and Sort. (b) The Linpack test and Linpack-GPU test are both with the process grid ratio of 24.

Writer benchmark. The CPU usage changes in a range of 5–20% and 75–100% when running Writer and Sort respectively by the parameters we specify. *Linpack* is a CPU stress benchmark. Our Linpack test set has the problem size of 40,000 and it includes 4 Linpack tests. We adjust the process grid ratio ($P*Q$) to change among 12, 24, 36 and 48 in our cluster, which makes CPU load change among 25%, 50%, 75% and 100% on each node. *Linpack-GPU* is a same benchmark with the Linpack test with the process grid ratio of 24, but this benchmark is executed on GPUs rather than CPUs.

Fig. 8(a) shows mean RMSE of the models on the Hadoop nodes. For Writer, all the models achieve high accuracy except the Linear model. The mean RMSE for Sort is much higher than for Writer for all the models. This means the estimation error for CPU intensive tasks is higher than that for I/O intensive tasks. GMM has a remarkable improvement over the other models, even in high CPU usage situations on the Hadoop nodes. The mean RMSE of GMM is at least half of that of the other models. The ANN model has a small enhance for RMSE, only about 4% less compared with M-Linear. GMDH is 15% better than M-Linear. M-Linear obviously outperforms Linear.

Fig. 8(b) shows the mean RMSE for the Linpack test and for the Linpack-GPU test both with the process grid ratio of 24. Two neural networks have a slight improvement over M-Linear. Compared with the results under Linpack and Linpack-GPU, power consumption is worse estimated when running tasks on the GPUs.

Fig. 9 presents mean success ratio on the Hadoop nodes and on the nodes with GPUs under different Δ . It shows that GMM has the best success ratio in most situations. In some cases models have lower RMSE error but less success ratio. For example, mean success ratio of GMDH is higher than GMM when Δ is 5% for the Writer benchmark. This is because some estimations of GMM and M-Linear have large errors.

We evaluate mean RMSE for the Linpack test set which consumes different CPU usage on the nodes with GPUs. In Fig. 10, the

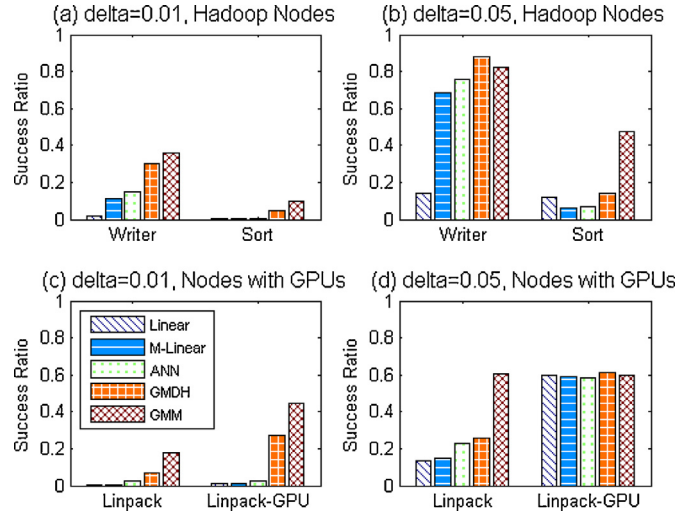


Fig. 9. Mean success ratio of nodes for the benchmarks. (a), (b) The MapReduce benchmark includes Writer and Sort. (c), (d) The Linpack test and Linpack-GPU test are both with the process grid ratio of 24.

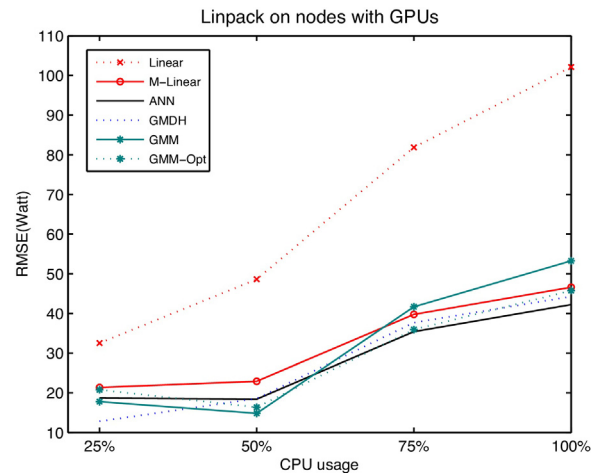


Fig. 10. Mean RMSE of nodes with GPUs for Linpack. The Linpack tests consume CPU usage with 25%, 50%, 75% and 100%.

mean RMSE basically increases when the CPU usage rises for all the models. M-Linear is still much better than Linear. GMDH becomes less accurate than ANN with the growing usage. They are both better than the two linear models. But GMM no longer remains the high accuracy at high CPU usage on the nodes with GPUs. This might be the result of imprecise cluster. We propose a simple optimized GMM model (**GMM-Opt**) with the awareness of CPU usage. We manually classify each data set into 5 groups according to the CPU usage: [0, 0.2], [0.2, 0.4], [0.4, 0.6], [0.6, 0.8] and [0.8, 1]. Then we train models independently in each group. We fit each feature vector in a proper group based on its CPU usage value to determine a model, and finally compute the power estimations. The figure shows GMM-Opt improves the accuracy of GMM and has a similar effect to the neural network models.

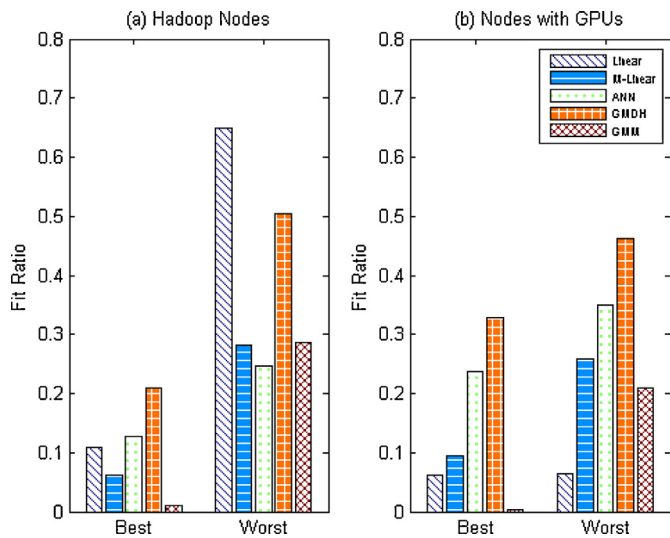


Fig. 11. The best and worst fit ratio using model migration for the benchmarks. (a) The MapReduce benchmarks includes Writer and Sort and (b) the Linpack is the test with the process grid ratio of 24.

In summary, for the Hadoop nodes with the I/O and CPU-intensive mixed computing environment, GMM has the better accuracy than GMDH in particular for CPU-intensive workloads; For nodes with GPUs that are the pure CPU intensive environment, GMM can be optimized with manual cluster to achieve the better performance. In the both environments, ANN shows its improvement over M-Linear under resource intensive workloads, but improvement of the two neural network approaches over M-Linear is not remarkable. M-Linear is much better than the Linear approach.

6.3. Portability and usability result

In a cluster environment, not all the nodes are installed with power meters. How to obtain their power models? Power is consumed by resource components to carry on tasks. Power consumption should be same for computer systems with homogeneous hardware and software configuration. Thus, models can be shared between homogeneous nodes. We have generated 5 different models using 5 approaches for each node. We migrate the models generated on one node onto its homogeneous nodes to evaluate errors on the test set. This is a process of testing the portability.

Fig. 11 shows the fit ratio of each model for the Linpack and MapReduce benchmarks. The MapReduce benchmark includes Writer and Sort. The fit ratio is the absolute error between the estimated power consumption using migrated models and the originally estimated power consumption divided by the originally estimated power consumption. We traverse all the nodes to the fit ratio for its homogeneous nodes. We list the best and worst fit ratio value. The two neural network approaches have the worst effect because the fit error could be enlarged by high degree polynomial function in neurons even with minimal difference from weights. GMM performs well among homogeneous nodes. M-Linear has the better portability than Linear for Linpack.

Table 7
The training and estimation time cost, resource usage and TrS demand of the approaches.

	Linear	M-Linear	ANN	GMDH	GMM
Training time (s)	2–4	4–7	25–73	17–60	132–227
Estimation time of single example (s)	<10 ⁻⁸	<10 ⁻⁷	[10 ⁻⁷ , 10 ⁻⁶]	[10 ⁻⁴ , 10 ⁻³]	[10 ⁻⁴ , 10 ⁻³]
CPU load	<7%	<7%	<8%	<8%	<8%
Large size of TrS demand	No	No	Yes	Yes	Yes

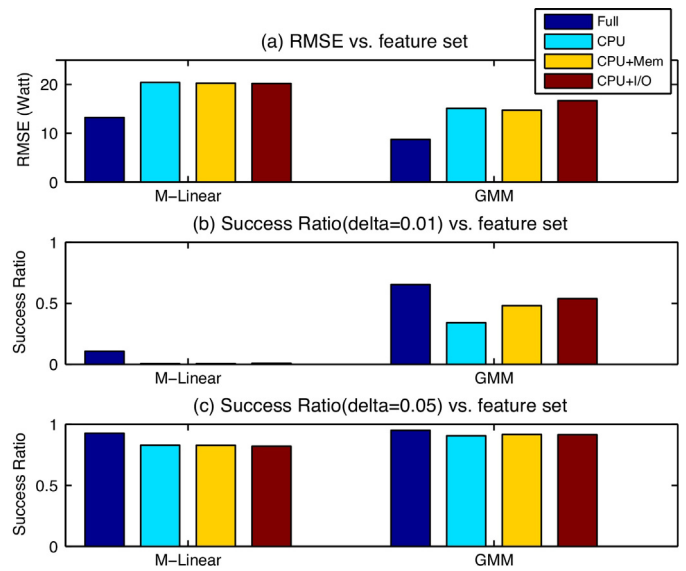


Fig. 12. RMSE (a) and success ratio with the Δ 1% (b) and 5% (c) of a typical node with GPU against different feature sets for M-Linear and GMM on the test set.

In practice, system administrators are not only concerned with the estimation accuracy of models. They may also consider the training time, estimation time and resource usage of each approach. In our last experiment, we study these factors. We train all the models and estimate power on a server node with Intel E5620 processor (2.4GHz) and 24GB memory. The range of results on the test set and for the benchmarks we use above are shown in Table 7.

The training time varies a lot. Both linear models only need less than 10 seconds. The time of GMM and GMM-Opt are the longest, nearly 30–100 times of linear models. ANN and GMDH take about 60 seconds. All the models only require less than 1 ms to make one estimation on average. The CPU usage for the models training is nearly the same, less than 8%. From the learning curves in Fig. 5, we have observed that the two linear models do not require a large size of training data to reach of stable RMSE value, while non-linear models could suffer overfit with inaccurate results if the TrS is not large enough.

The cluster operators can use GMM or optimized GMM approach if they demand precise model in a long schedule time window. They have enough time to train the model, estimate power consumption and perform schedule in the window. If the operators concern computing cost in a short schedule window or in a hard deadline for scheduling, they should choose M-Linear. Machine learning approaches take too much computation time. When they compute the models, the incoming short schedule window could have passed in the worst case. The simple approaches are adaptive in the environment where the configuration of nodes changes or updates frequently. The linear approaches allow operators to update their models in a short time, while the complicated machine learning approaches need quite a long time to collect new measurements for new parameter training.

6.4. Impact of feature selection

Fig. 12(a)–(c) shows the RMSE and success ratio of a node with GPU against different sets of features for M-Linear and GMM on the test set. Fig. 12(b) shows the success ratio result when Δ is 1%; Fig. 12(c) shows when Δ is 5%.

For M-Linear models and GMM models using the full feature set, RMSE is the lowest and success ratio is the highest. This result supports the full feature selection in the approaches.

For the M-Linear models using a set with CPU only, or a set with CPU and memory features, or a set of CPU and I/O features, leads to worse results than in the case of a full feature set. In all cases we have comparable RMSE and success ratios. In essence training a linear model using other features than the full set is not optimal on these nodes.

For the GMM models, the feature set with CPU and I/O has the highest RMSE. When Δ is 1%, success ratio for GMM models using CPU only is the lowest. This tells us the set with CPU and memory features is a suboptimal choice when the complete features cannot be observed.

7. Conclusion and future work

We discussed the importance of power models, which motivated our evaluation of non-linear machine learning approaches including ANN, GMDH and GMM. To our knowledge, this is the first attempt to make use of these approaches based on a large size of measurement data to estimate power consumption in a cluster environment.

We can conclude our finding as follows. The non-linear machine learning approaches can improve the estimation accuracy of the linear approaches using basic resource features. The GMM approach has the best power estimation accuracy compared with the neural network approaches and the linear approaches in the I/O and CPU-intensive mixed environment; GMM is not suitable in the pure CPU-intensive environment, but the optimized GMM approach can improve GMM and achieve the similar performance compared to the two neural networks. The GMM regression consumes the longest time to train the model. The neural network approaches only have a slight accuracy advantage over multiple-variable linear approach, meanwhile these neural network approaches have the worse portability on the homogeneous nodes. A multiple-variable linear approach highly improves the estimation accuracy of CPU-only linear approach. We suggest to use GMM or optimized GMM if anyone only concerns with the estimation accuracy, and to use a multiple variable linear regression in the time constraint environment. In the end, we prove that models trained using the system-level full features have the highest accuracy comparing to only use part of features.

Although our cluster is not a production cluster, our study on designing power estimation approaches and evaluation methods in the paper can be useful for future work in production clusters. We can further improve the accuracy of approaches by considering operating frequency of CPU [32] and time series in the future. The CPU power for the different frequencies is totally distinct. The pattern of resource features about sample time can better characterize workloads. Two factors make power estimation model more fine-grained. It is also expected that features about GPUs e.g. GPU memory usage should be taken into consideration for precisely estimating power consumption of GPU workloads [42].

Acknowledgements

This work was supported by NWO through the GreenClouds project, by RAAK-MKB through the Greening the Cloud project, by the Dutch national program COMMIT and by the European

Community Seventh Framework Programme under grant agreement no. 605243 (GN3plus). We also thank Jos Wezenberg for his support on this work.

References

- [1] G. Meijer, Cooling energy-hungry data centers, *Science* 328 (April) (2010) 318–319.
- [2] G. Cook, J. Van Horn, How Dirty is Your Data? A Look at the Energy Choices that Power Cloud Computing, 2011 May 24, Greenpeace Report.
- [3] T. Brunschwiler, B. Smith, E. Ruetsche, B. Michel, Toward zero-emission data centers through direct reuse of thermal energy, *IBM J. Res. Dev.* 53 (3) (May) (2009) 476–488 <http://dl.acm.org/citation.cfm?id=1850646.1850657>.
- [4] F. Kong, X. Liu, A Survey on Green-Energy-Aware Power Management for Datacenters, *ACM Computing Surveys* 47 (2) (2014).
- [5] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, 2008, pp. 10–18.
- [6] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, F. Zhao, Energy-aware server provisioning and load dispatching for connection-intensive internet services, in: *NSDI'08 Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008, pp. 337–350.
- [7] A. Kansal, F. Zhao, A.A. Bhattacharya, Virtual machine power metering and provisioning, in: *Proceedings of the 1st ACM Symposium on Cloud Computing*, 2010, pp. 39–50.
- [8] L. Barroso, U. Holzle, The case for energy-proportional computing, *Computer* 40 (12) (2007) 33–37.
- [9] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, *Commun. ACM* 51 (1) (January) (2008) 107–113.
- [10] D. Ardagna, B. Panicucci, M. Trubian, L. Zhang, Energy-aware autonomic resource allocation in multitier virtualized environments, *IEEE Trans. Serv. Comput.* 5 (1) (January) (2012) 2–19.
- [11] Q. Zhang, M.F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, J.L. Hellerstein, Dynamic energy-aware capacity provisioning for cloud computing environments, in: *Proceedings of the 9th International Conference on Autonomic Computing*, 2012, pp. 145–154.
- [12] G. Dhiman, A system for online power prediction in virtualized environments using Gaussian mixture models, in: *Proceedings of the DAC'10. No. 3*, 2010, pp. 807–812.
- [13] S.-K. Oh, W. Pedrycz, The design of self-organizing polynomial neural networks, *Information Sciences* 141 (3–4) (April) (2002) 237–258.
- [14] R. Chellappa, Gaussian mixture models, *Encycl. Biom.* 2009 (2) (2009) 659–663.
- [15] R. Stanforth, E. Kolossov, B. Mirkin, Hybrid k-means: combining regression-wise and centroid-based criteria for QSAR, in: *Selected Contributions in Data Analysis and Classification, Part of the series Studies in Classification, Data Analysis, and Knowledge Organization*, Springer Berlin Heidelberg, 2007, pp. 225–233.
- [16] C.M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
- [17] Q. Chen, P. Grosso, K. van der Veldt, C. de Laat, R. Hofman, H.E. Bal, Profiling energy consumption of VMs for green cloud computing, in: *Proceedings of the International Conference on Cloud and Green Computing (CGC2011)*, 2011.
- [18] R. Lent, A model for network server performance and power consumption, *Sustain. Comput.: Inf. Syst.* 3 (2) (2013) 80–93.
- [19] C. Möbius, W. Dargie, S. Member, A. Schill, Power consumption estimation models for processors, virtual machines, and servers, *IEEE Trans. Parallel Distrib. Syst.* (2013) 1–14.
- [20] X. Fan, W.-d. Weber, Power provisioning for a warehouse-sized computer, in: *ISCA'07 Proceedings of the 34th Annual International Symposium on Computer Architecture*, June, San Diego, CA, 2007, pp. 13–23.
- [21] S. Rivoire, P. Ranganathan, C. Kozyrakis, A comparison of high-level full-system power models, in: *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, 2008, pp. 31–38.
- [22] T. Heath, B. Diniz, E.V. Carrera, W.M. Jr., R. Bianchini, Energy conservation in heterogeneous server clusters, in: *Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming – PPOPP 05*, 2005, p. 186.
- [23] R. Zamani, A. Afsahi, A study of hardware performance monitoring counter selection in power modeling of computing systems, in: *Proceeding of the Second Conference on Power Measurement and Profiling (PMP 2012)*, 2012, pp. 1–10.
- [24] A. Arbor, X. Chen, A. Arbor, R.P. Dick, Z.M. Mao, A. Arbor, Performance and power modeling in a multi-programmed multi-core environment, in: *Proceedings of DAC*, 2010, pp. 1–6.
- [25] D. Economou, S. Rivoire, C. Kozyrakis, Full-system power analysis and modeling for server environments, in: *Proceedings of the 2006 Conference on Modeling, Benchmarking, and Simulation (MoBS)*, 2006.
- [26] K. Rajamani, H. Hanson, J. Rubio, S. Ghiasi, F. Rawson, Application-aware power management, in: *Proceedings of the 2006 IEEE International Symposium on Workload Characterization*, 2006 October, pp. 39–48.
- [27] Y. Li, Y. Wang, B. Yin, L. Guan, An online power metering model for cloud environment, in: *Proceedings of the 11th International Symposium on Network Computing and Applications*, 2012 August, pp. 175–180 <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6299092>.

- [28] G. Dhiman, T. Rosing, System-level power management using online learning, *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 28 (5 (May)) (2009) 676–689.
- [29] R. Koller, WattApp: an application aware power meter for shared data centers, in: *Proceedings of the 7th International Conference on Autonomic Computing*, 2010, pp. 31–40.
- [30] C.-H. Lien, B. Ying-Wen, Web server power estimation, modeling and management, in: *14th IEEE International Conference on Network (ICON 06)*, vol. 00, 2006, pp. 1–6.
- [31] I. Wabmann, D. Versick, D. Tavangarian, Energy consumption estimation of virtual machines., in: *Proceedings of the 28th Annual ACM Symposium on Applied Computing – SAC'13*, 2013, p. 1151 <http://dl.acm.org/citation.cfm?doid=2480362.2480579>.
- [32] L. Piga, R. Bergamaschi, S. Rigo, Empirical and analytical approaches for web server power modeling, *Clust. Comput.* (2014) 1–15.
- [33] T.K. Moon, The expectation-maximization algorithm, *IEEE Signal Process. Mag.* 13 (6 (November)) (1996) 47–60.
- [34] J.M. Schopf, I. Raicu, L. Pearlman, N. Miller, C. Kesselman, M.D. Arcy, Monitoring and Discovery in a Web Services Framework: Functionality and Performance of Globus Toolkit MDS4, 2005, Tech. Rep.
- [35] M.L. Massie, B.N. Chun, D.E. Culler, The ganglia distributed monitoring system: design, implementation, and experience, *Parallel Comput.* 30 (7 (July)) (2004) 817–840.
- [36] H. Yuan, C.J. Kuo, I. Ahmad, Energy efficiency in data centers and cloud-based multimedia services: an overview and future directions., in: *Green Computing Conference*, 2012, pp. 375–382.
- [37] DAS4, The Distributed ASCI Supercomputer 4, 2014 <http://www.cs.vu.nl/das4/home.shtml>.
- [38] H. Zhu, K.V.D. Veldt, P. Grosso, X. Liao, C.D. Laa, EKB: semantic information system for energy-aware monitoring in distributed infrastructures, in: *2013 International Conference on Cloud and Green Computing*, 2013 Sep, pp. 60–67.
- [39] S. Di, D. Kondo, W. Cirne, Host load prediction in a Google compute cloud with a Bayesian model, in: *2012 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2012 November, pp. 1–11.
- [40] Google, Google Cluster Trace, 2011 <https://code.google.com/p/googleclusterdata/>.
- [41] C. Reiss, G.R. Ganger, R.H. Katz, M.A. Kozuch, Heterogeneity and dynamicity of clouds at scale: Google trace analysis, in: *Proceedings of the Third ACM Symposium on Cloud Computing*, 2012, pp. 7:1–7:13.
- [42] S. Song, C. Su, B. Rountree, K.W. Cameron, A simplified and accurate model of power-performance efficiency on emergent GPU architectures, in: *Proceedings – IEEE 27th International Parallel and Distributed Processing Symposium, IPDPS 2013*, 2013, pp. 673–768.



Hao Zhu is a researcher in the Department of Computer, National University of Defense Technology. He received his PhD degree in the SNE group at the UvA, in 2015. His research interests are in the area of Green IT, cloud computing and semantic web. To be particular, he studies power estimation models, power management of distributed system and semantic descriptions of energy-aware infrastructures.



Xiangke Liao received his BS degree in the Department of Computer Science and Technology from Tsinghua University, Beijing, China, in 1985, and his MS degree in 1988 from National University of Defense Technology, Changsha, China. He is currently a full professor and the dean of the Department of Computer, National University of Defense Technology. His research interests include parallel and distributed computing, high-performance computer systems, operating systems, cloud computing and networked embedded systems. He is a member of the IEEE and ACM.



Cees de Laa is chair of the System and Network Engineering research section at the University of Amsterdam. Research covers optical and switched networking and workflows for processing of big data in PetaScale e-Science applications, Semantic Web to describe e-infrastructure resources, information complexity, Authorization architectures and Systems Security & privacy of information in distributed environments. He serves as gfs member of Open Grid Forum, is chair of GridForum.nl <http://gridforum.nl> and serves on the Lawrence Berkeley Laboratory Policy Board on matters regarding ESnet, is co-founder and organizer of several past meetings of the Global Lambda Integrated Facility (GLIF) and founding member of CineGrid.org <http://cinegrid.org>, <http://delaat.net/>.



Paola Grosso is assistant professor in the SNE group at the UvA. She is the lead researcher of the group activities in the field of GreenIT, optical networking and distributed infrastructure information modelling. She participates in several national and international projects, such as COMMIT, GreenClouds, Green Software and MOTE. Her research interests are green ICT, provisioning and design of hybrid networks for lambda services; development of information models for hybrid multi-domain multilayer networks <http://www.science.uva.nl/grosso>.