# Learning-Based Power Prediction for Data Centre Operations via Deep Neural Networks


Yuanlong Li
Nanyang Technological University
Singapore
liyuanl@ntu.edu.sg

Han Hu
Nanyang Technological University
Singapore
hhu@ntu.edu.sg

Yonggang Wen
Nanyang Technological University
Singapore
ygwen@ntu.edu.sg

Jun Zhang
South China University of Technology
China
junzhanghk@gmail.com



## ABSTRACT

Modelling and analyzing power consumption for data centres can diagnose potential energy-hungry components and applications, and facilitate in-time control, benefiting the energy efficiency of data centers. However, solutions to this problem, including static power models and canonical prediction models, either aim to build a static relationship between power consumption and hardware/application configurations without considering the dynamic fluctuation of power; or simply treat it as time series, ignoring the inherit power data characteristics. To tackle these issues, in this paper, we present a systematic power prediction framework based on extensive power dynamic profiling and deep learning models. In particular, we first analyse different power series samples to illustrate their noise patterns; accordingly we propose a power data de-noising method, which lowers noise interference to the modelling. With the pretreated data, we propose two deep learning based prediction models, including a fine-grained model and a coarse-grained model, which are suitable for different time scales. In the fine-grained prediction model, a recursive autoencoder (AE) is employed for short-duration prediction; in the coarse-grained model, an AE is used to encode massive fine-grained historical data as a further data pretreatment for long-duration prediction. Experimental results show that our proposed models outperform canonical prediction methods with higher accuracy, up to 79% error reduction for certain cases.



## CCS Concepts

•**Applied computing** → **Data centers; IT architectures;**





*E2DC, June 21-24, 2016, Waterloo, ON, Canada*





DOI: http://dx.doi.org/10.1145/2940679.2940685

## Keywords

Data centre; power modelling; power prediction; deep learning


## 1. INTRODUCTION

With the ongoing increase in the volume and scale of data centers globally, the resulted high power consumption problem has attracted great attention from both industry and academy. Google, a leading search service provider, owns at least 12 data centers in US, and a single fully-operated data centre is estimated to consume up to 90 MW power in Oregon. This huge energy consumption brings along a growing number of problems, such as electricity capitalize investment, carbon emission, and serious system performance un-reliability and degradation. Thus, various of power control technologies have been studied, including DVFS [25], power napping [12], and consolidation [3], to save energy from different aspects. However, all these technologies rely on power estimation, which makes it a essential problem for data centre power management.

Power estimation often includes two scopes, i.e., building the relationship between the power consumption and the hardware/software configuration, and tracking the power dynamic within different time scales (e.g., short or long). These two tasks are not straightforward as: 1) there are distinctive hardware/software factors (e.g., CPU, cooling, request workload, etc.), which may be correlated with each other and have different effects on power consumption. Furthermore, several potential confounding factors may also impact the power consumption, e.g., the local temperature, component aging. A static explicit relationship, if existing, is not sufficient; 2) power consumption is a dynamic process, rather than a quantitative value. For instance, the workload fluctuates during the execution period, leading to different levels of power consumption. An adaptive model is needed to capture such dynamics and make continuous accurate predictions within the execution period.

Many researchers have dedicated their efforts on the power estimation problem of data centres. Power modelling has been widely explored by utilizing different functions [5] [19]. These models are derived on specific system settings and fixed workloads, and not easy to be extended for different

configurations. For dynamic power analysis, Wang *et al.* investigated the temporal characteristic of the power series and analyzed the power peaks [26]. However, this work only focuses on analyzing the peak patterns of the power series and cannot provide a complete prediction of the future power consumption. A straightforward idea for power prediction is utilizing the canonical machine learning algorithms to achieve such a goal. Their methods ignore the inherent characteristic of power data, such as noise patterns and fluctuation patterns, and are unsuitable for prediction in different time scales.

In this paper, we present a systematic power prediction framework based on extensive power dynamic profiling and deep learning models, with a complete roadmap which covers from data acquisition, data pretreatment, to prediction models. In data acquisition, we collect power series and some power-related system counters, which are used to build a black-box model without considering the explicit relationship between power and other factors. In data pretreatment, we analyse different power series samples to illustrate their noise patterns and propose a power data de-noising method. With the pretreated data, we propose two deep learning [10] based prediction models for different purposes: a fine-grained prediction model for short-duration prediction and a coarse-grained prediction model for long-duration prediction.

Our contributions of this paper are as follows:

- We use detrended fluctuation analysis (DFA) to analyse the noise patterns of the power series and present a smoothing method that can reduce the white noise and retain as much workload-related fluctuation as possible.
- For the fine-grained power prediction, we propose to use recursive auto-encoder (AE) [10], which handles the historical data in a recursive way. For the coarse-grained prediction model, we leverage AE to encode massive small time-scale historical data, which is used to reduce the dimensionality of the input data while preserving more useful information than common re-sampling based methods.
- Experimental results show that, our fine-grained prediction model can reduce the prediction error by up to 79% on certain cases, and the coarse-grained prediction model can lower the prediction error by at most 50%, in comparison with the canonical methods.

The remainder of this paper is organized as follows. In Section 2, the power-dynamic modelling problem is formulated. In Section 3 we introduce data collection details. Section 4 presents the data pretreatment process. Section 5 present fine-grained and coarse-grained prediction models and the evaluation results. In Section 7, we provide the related works and in Section 8 we conclude the whole paper.

# 2. DATA CENTER OPERATIONS WITH POWER PREDICTION

In this section, we first present the overall architecture and workflow of our emulation-based data center operation system, demanding an accurate prediction engine to improve the business continuity. Then we lay out the roadmap of our proposed learning-based prediction engine including two

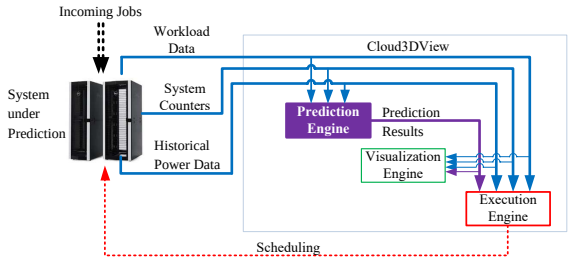



**Figure 1: The overall architecture and workflow of our emulation-based data centre operation system. The prediction engine receives historical data and output the prediction results to the Cloud3DView management platform.**

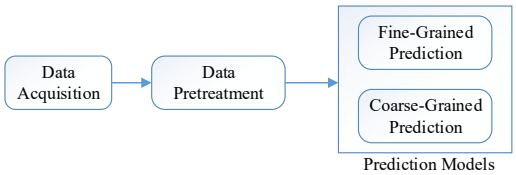



**Figure 2: Learning based prediction engine work flow.**

prediction models, a fine-grained model and a coarse-grained model.

## 2.1 System Flow for Emulation-Based Data Centre Operation

The overall architecture and workflow of our emulation-based data centre operation system is shown in Fig. 1. The system under prediction is a data centre with a monitoring system which can provide various runtime status, such as the workload level, various system counters like CPU usage, memory utility, etc; and also the current power consumption. These data are recorded as several time series and used as the input data of the prediction engine. The prediction engine learns from the historical data to predict the future power consumption. The prediction results along with various other system information are further processed by the Cloud3DView [27] platform to optimize the energy performance via resource/workload scheduling.

The prediction engine is the key module for the above emulation-based data centre operation system. In the following we will detail how to build the prediction engine.

## 2.2 Learning-based Prediction Roadmap

The roadmap of our learning-based prediction engine consists of three steps (as shown in Fig. 2), including data acquisition, data pretreatment, and prediction models. In data acquisition, we collect the historical power consumption data, workload data and various system counters, which are needed for the prediction model. In data pretreatment, as the raw data may contain noises which can bias the prediction, we analyse different power series samples to illustrate their noise patterns and propose a power data de-noising method. For the prediction models, our observation shows that: on the one hand, for a short duration, power series fluctuates severely with high randomness and the prediction model should be fast enough to track changes; on the other

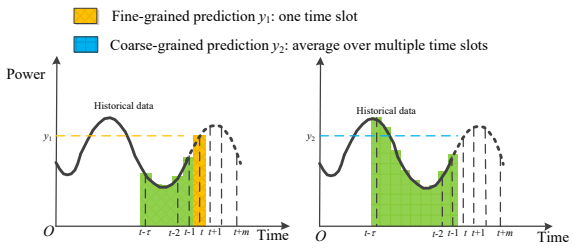



**Figure 3: The two prediction models. The historical data from $t-\tau$ to $t-1$ (green) are used in prediction. For fine-grained prediction, we predict the power consumption at time slot $t$; for coarse-grained prediction, we predict the average power consumption over multiple time slots (from $t$ to $t+m$).**

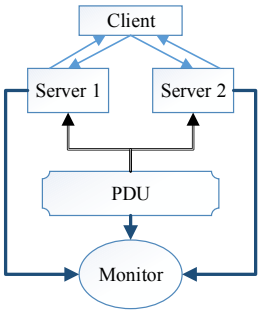



**Figure 4: System setup: two servers with different configurations are used to process http requests sent by the client. The monitor collects data from the server and the PDU.**

hand, for a long duration, the power series shows more stable fluctuation pattern (such as the daily pattern) that can be predicted easily. Also in this case, as the power series is measured in a small time granularity, a large amount of fine-grained data is available, which should be properly utilized in prediction. Thus, we propose a fine-grained prediction model and a coarse-grained prediction model respectively, as shown in Fig. 3:

- *Fine-grained prediction:* The objective is to predict the power consumption of a single time slot (the yellow area in Fig. 3) in the future by using nearby past historical information (the green area in Fig. 3). In this case, the predicted time slot is small (e.g., 30 seconds in this work). The prediction results can benefit online workload scheduling with higher real time requirements.
- *Coarse-grained prediction:* Similarly, we use the historical information for the power consumption prediction; however, we aim to predict the average power consumption in multiple time slots in the future (the blue area in Fig. 3). In this case, the predicted time interval is much longer than the time unit of power series, saying one hour in this work. The prediction result can illustrate the power fluctuation in a larger time scale and is useful for hardware-level scheduling requiring a long-time preparation.

In the following sections we will introduce the prediction engine step by step with more details.

# 3. DATA ACQUISITION

In this section, we build an experimental system to emulate the system running and power consumption of data centers. Using two benchmark workloads as the input to the emulation system, we collect power data and other system status.

## 3.1 Experimental System Architecture

The experimental system architecture is shown in Fig. 4. It contains two servers with a power distribution unit (PDU), a client machine, and a monitor machine. These two servers are with the same hardware configuration (Dell PowerEdge R620), but different software installations. The PDU can monitor the power consumption of each server. The client machine is responsible for sending HTTP requests to two web servers. Workload information, server status, and power consumption from PDU are recorded by the monitor machine.

In our experiments, we configure the client machine with distinctive benchmark workloads to emulate the real user requests, and therefore drive the servers running to generate different power consumptions. The request sending rate is scaled to make both servers running in different states (e.g., idle, normal, heavy loaded).

Note that the experimental system, compared with a real data centre, is simplified in many ways. For instance, we only emulate the web service, without consideration of other complicated services, e.g. big data applications. However, our prediction engine is based on a learning based black-box model and not specified for certain applications. We can easily extend our model to other applications by replacing the corresponding training data.

## 3.2 Benchmark Workload and Server Configuration

In general, the power consumption depends on workload level and server configuration. In our simulation, we adopt two widely-used web request traces and two representative web server configurations.

The web traces used by the client machine are the world cup 98 (WC98) [1] and Clark net (Clark) [2]. These two web traces have distinctive dynamic patterns, as shown in Fig. 5(a) and (d): the WC98 trace has more irregular peaks while the Clark trace has a stable daily fluctuation pattern. In addition, to check the system noises, we consider the situation that the server is empty-loaded. For the web server configuration, we install one server with Nginx and the other with Apache to examine the impact of software setting on power consumption.

## 3.3 Data Collection

We collect three types of data for power consumption prediction:

- *Workload Profile:* The workload level can be approximately measured by the number of requests received by web servers. We record the number of requests every three seconds in each server.
- *System Status:* The OS of each server can monitor the system running status in terms of system counters. We

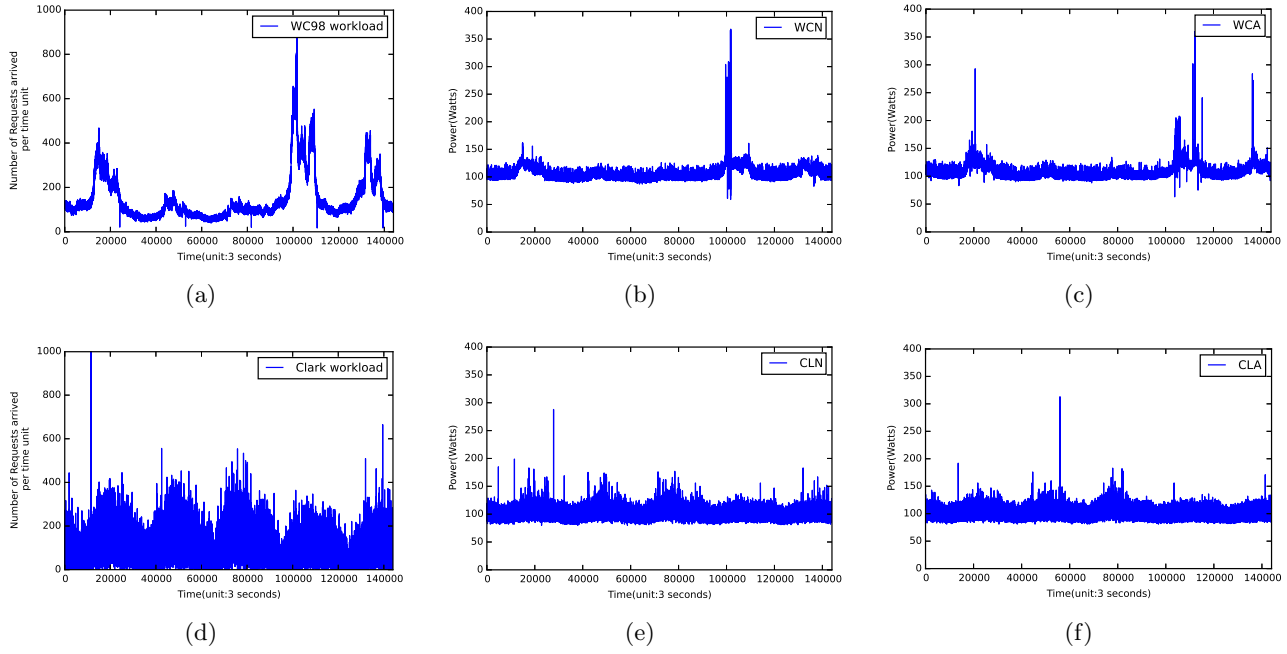



**Figure 5: Workload traces and collected power series under different configurations: (a) WC98 workload trace; (b) Power series WCN; (c) Power series WCA; (d) Clark workload trace; (e) Power series CLN; (f) Power series CLA.**

**Table 1: System counters collected by "collectd".**

| System counters | Dimension | Details |
|---|---|---|
| CPU_usage | 1 | 100-(CPU idle percentage) |
| if_packets | 2 | Packets per second, in and out |
| load | 3 | Number of runnable tasks in the run queue as a 1, 5 and 15 minute average |
| memory usage | 1 | Physical memory used (in bytes) |
| disk_ merged | 2 | Number of disk read/write operations can be merged |
| df | 2 | File system used/available (in bytes) |

**Table 2: Power series with different configurations.**

| Name for short of a power series | | Type of workload trace | | |
|---|---|---|---|---|
| | | WC98 | Clark | Empty |
| Type of web server | Nginx | WCN | CLN | E |
| | Apache | WCA | CLA | |

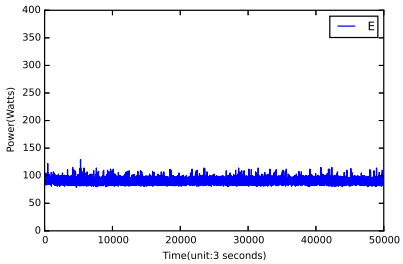



**Figure 6: Power series E when workload is empty.**

collect several power related counters as shown in Table 1. Note that some counters are multidimensional data. These data are also collected every three seconds.

- *Power Profile:* The power consumption (in Watts) from PDU is recorded every three seconds.

The detailed methods used to collect data are as follows: the workload profile of each server is extracted from log information; system status, such as CPU utilization, memory usage and network traffic, are measured by an open-source software "collectd"; Simple Network Management Protocol (SNMP) is used to collect the power consumption (Watts) from PDU. Note that the power consumptions of two web servers are measured independently.

With different workloads and server configurations, we collect five power series as shown in Table 2. Their abbreviations indicate the combinations of workloads and server configurations. Taking "WCN" as an example, the first two letters "WC" denote this power series is driven by workload

**Table 3: Values of $\alpha$ in different cases by DFA. $\alpha$ value near 0.5 indicates white noise, near 1 indicates pink noise, near 1.5 indicates red noise respectively.**

| Power series | WCN | WCA | CLN | CLA | E |
|---|---|---|---|---|---|
| $\alpha$ | 1.30 | 1.33 | 0.92 | 0.91 | 0.62 |

"WC98" and the last letter "N" means that the server configuration is Nginx. Power series WCN, WCA, CLN and CLA are shown in Fig. 5 (b)(c)(e)(f) respectively. The trace "E" denotes the empty workload power series, which is shown in Fig. 6.

# 4. DATA PRETREATMENT

By comparing the empty power series (Fig. 6) with the non-empty power series (such as WCN, in Fig. 5(b)), it is obvious that the amplitude of the power profile without workload, i.e., system noise, is comparable to that of the non-empty power series. It is possible that the system noise may affect the subsequent prediction. Therefore, we utilize detrended fluctuation analysis (DFA) to characteristic the noise pattern, and propose a proper smoothing algorithm to de-noise the power data. The smoothing results are further verified by the correlation test between the power series and workload series.

## 4.1 Power Series Predictability Characterization

We adopt DFA to analyse the noise pattern of the power series. DFA is a time series analysis method that compares the fluctuation degree of a time series in different time scales, which can be used to determine the noise type of a signal, measured by a factor $\alpha$. The value of $\alpha$ indicates the noise type as:

- when $\alpha \approx 1/2$, the signal is composed of white noise;
- when $\alpha \approx 1$, the signal is composed of pink noise, which indicates regular fluctuation pattern;
- when $\alpha \approx 3/2$, the signal is composed of red noise, which indicates random walk.

The white noise is caused by the random fluctuation of the system or measurement error, which is unpredictable and should be removed as much as possible. Pink and red noise indicate that the fluctuation of the signal is following certain hidden trends, which should be preserved as much as possible.

We measure the $\alpha$ value via DFA on the power series, and the results are shown in Table 3. We can observe that: 1) the $\alpha$ value of the empty workload power series "E" is close to 0.5, which confirms that the system noise is almost random white noise; 2) $\alpha$ values for WC98 power series (WCN and WCA) are close to 1.3, which indicates that the noise is between pink and red; 3) $\alpha$ values of the Clark power series (CLN and CLA) are close to 1, which indicates that the noise is pink.

## 4.2 Smoothing for Noise Reduction

Based on the above analysis, there exists three types of noises, including white noise, pink noise and red noise, in our collected raw data. To alleviate the impact from noise, our objective is to remove the white noise, while keep the pink and red noise, which are related to the workload. To this end, we leverage an average value based downsampling method detailed as below.

**Table 4: Correlation coefficient of power series and workload series before and after smoothing.**

| | Spearman's rank correlation coefficient | | | |
|---|---|---|---|---|
| Before smoothing | WCN | WCA | CLN | CLA |
| | 0.8209 | 0.8342 | 0.7490 | 0.6123 |
| After smoothing | WCN | WCA | CLN | CLA |
| | 0.9712 | 0.9686 | 0.9547 | 0.9364 |

Given the original power series with a time unit $\rho$, for each consecutive non-overlapping window consisting of $L$ time slots, we downsample the power series to the average power consumption in this window. This new power series then has a time unit $\rho \cdot L$. Since the white noise has zero mean value, when the window length $L$ is large enough, the white noise can be eliminated using this method.

In practice, $L$ cannot approach to infinity. A key problem is how to choose an appropriate window size $L$. Through extensive experiments, we find that, when $L = 10$, the downsampled series to the 'E' trace approximates a straight horizontal line with variance reduced by 95%. Although a larger $L$ can achieve better smoothing performance, it leads to higher pink/red noise loss, which is undesirable. Therefore, we set $L = 10$.

We apply this smoothing method to all the raw data we collected. The time unit of the downsampled data is changed to 30 seconds.

## 4.3 Noise Reduction Result Verification

To verify the smoothing performance to the power series, we calculate the correlation between the workload series and power series before/after smoothing. The reason we choose the workload series as the reference is that the workload series is measured by the number of requests without system noise. In particular, we compute Spearman's rank correlation coefficient [4] between two time series. The correlation coefficient falls into the range [-1, 1]. A coefficient close to 1 or -1 indicates there exists strong correlation; while the coefficient close to 0 means there is weak correlation.

The correlation coefficients are shown in Table 4. When we compare the coefficients before and after smoothing for four traces, we observe that the coefficients after smoothing are increased significantly for all the cases. Especially, the coefficient for the CLA trace is increased from 0.6123 to 0.9364. We can conclude that our proposed smoothing method can reduce the system noise and preserve the workload correlated trends.

# 5. FINE-GRAINED AND COARSE-GRAINED PREDICTION MODELS

In this section, we first briefly introduce the preliminaries of deep neural network model, which is the basis of our proposed two prediction models. Following that, we present the fine-grained and coarse-grained prediction models. For fine-grained prediction, we propose to use the RAE model to track fast power fluctuation; for coarse-grained prediction, we propose to utilize AE to handle the large amount of fine-grained data.

## 5.1 Deep Neural Network Model

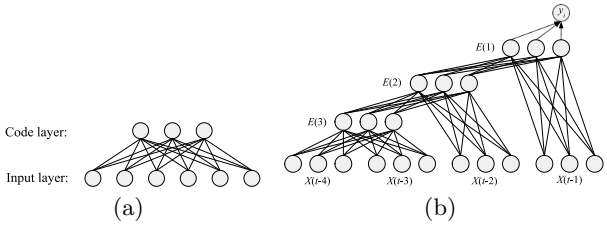



**Figure 7: Illustration of (a) auto-encoder (AE) and (b) recursive auto-encoder (RAE). An AE consisting of two layers can encode the input data into codes with a different length. RAE recursively encodes the data in the series into a short code. In the illustration example, $X(t-4)$, $X(t-3)$, $X(t-2)$ and $X(t-1)$ are encoded into the code $E(1)$ with the same length 3 of any of the given input data, which can be used for further prediction.**

In this subsection, we introduce the key idea of auto encoder and recursive auto encoder.

### 5.1.1 Auto Encoder

AE is a two-layer neural network as shown in Fig. 7 (a) which has an input layer and an output layer. Nodes in an AE are connected with full connections between the two layers. For any input data $X$, where $X$ is a vector, AE can encode $X$ into a code $X'$ of a different dimension (equal to the number of the nodes in the output layer of the AE). There can be different coding methods for AE, here we consider the following method: for an AE with $N$ nodes in input layer and $M$ nodes in output layer, with an activation function $g$, for example, the sigmoid function, a $M \times N$ transformation matrix $W$ and a $M \times 1$ bias vector $b$, given input vector $X$, the code $X'$ is computed as $X' = g(WX + b)$; with a new transformation matrix $W'$ of size $N \times M$ and bias vector $b'$ of size $N \times 1$, $X$ can then be reconstructed as $g(W'X' + b')$. An AE can be pre-trained by adjusting $W$,$W'$, $b$ and $b'$ to minimize its reconstruction error, i.e. $|X - g(W'X' + b')|$.

### 5.1.2 Recursive Auto Encoder

The recursive auto encoder (RAE) [21] built on AE is originally proposed for language modelling, which can recursively encode a whole sentence into a simple code to predict the following phrase. We are particularly interested in RAE here as it can recursively encode the data in a sequence.

We introduce the structure of the RAE in the context of our power prediction model as shown in Fig. 7 (b). Denote $X(t)$ as the input data at time slot $t$, which means $X(t)$ contains all the data including power consumption, workload and all the systems counters aligned as a vector. Then the input data from time slot $t - \tau$ to $t - 1$ can be encoded recursively in the following way: firstly $X(t - \tau)$ and $X(t - \tau + 1)$ are encoded into $E(\tau)$, then $E(\tau)$ and $X(t - \tau + 2)$ are encoded into $E(\tau - 1)$,..., finally $E(2)$ and $X(t - 1)$ are encoded into $E(1)$. $E(1)$ is a code of the same length as $X(t)$, which can be taken as a weighted summation of $X(t - \tau, ..., t - 1)$. $E(1)$ can then be used to predict power consumption $y_t$, for example, with a transformation matrix $W$ and bias vector $b$, the predicted power can be computed as $y'_t = WE(1) + b$.

## 5.2 Fine-Grained Prediction Model

The fine-grained prediction model is designed to predict the power consumption at time slot $t$ given the related historical data at time slots $t - 1, t - 2, .....$ For such purpose, NARX, which is a non-linear neural network (NN) and one of the most widely used prediction models for time series, fits our problem well in the sense that exogenous factors (i.e., workload and system counters) are available in our case. However, we find that NARX cannot track rapid power fluctuations in certain cases, especially for the fine grained prediction. To tackle this problem, we propose a linear recursive auto encoder (RAE) based prediction model.

The RAE based prediction model works as follows: the input of the RAE is all the historical data from $t - \tau$ to $t - 1$ aligned as a vector; A same linear AE is used in the encoding process of the RAE. With the output $E(1)$ of the RAE, the predicted power consumption $y'_t$ at $t$ is computed as $WE(1) + b$, with a transformation matrix $W$ and bias vector $b$.

To train the RAE network, the optimization objective is to minimize the following loss function:

$$\epsilon_{RAE} = \epsilon_{PRD} * 0.95 + \epsilon_{AE} * 0.05, \quad (1)$$

where $\epsilon_{PRD}$ is the prediction error and $\epsilon_{AE}$ is the reconstruction error of the RAE. In other words, $\epsilon_{RAE}$ is the weighted sum of prediction error and reconstruction error with the corresponding weighted factor 0.95 and 0.05 respectively. $\epsilon_{PRD}$ and $\epsilon_{AE}$ are given by:

$$\epsilon_{PRD} = \frac{\sum_{t=1}^{N_{train}} \|y(t) - y'(t)\|^2}{N_{train}} + 0.0001 \cdot \|W\|^2,$$

$$\epsilon_{AE} = \frac{\sum_{t=1}^{N_{train}} Err_{rec}(t)}{N_{train}},$$

where $\epsilon_{PRD}$ is the mean square error (MSE) of prediction with $L_2$-norm parameter regularization, and $N_{train}$ is the number of training samples. $\epsilon_{AE}$ is the mean value of all the reconstruction error $Err_{rec}(t)$ over $N_{train}$ encoding steps.

The control variables of the RAE prediction model include the RAE parameters ($W_{AE}$, $W'_{AE}$,$b_{AE}$, and $b'_{AE}$) and the prediction layer parameters ($W$ and $b$), which are put together as a parameter set denoted by $\theta$. Then the training process is to adjust $\theta$ to optimize the error function. To avoid the over-fitting problem, we follow the general training process of an NN with a validation process, which are shown in Algorithm 1.

The general NN training framework Algorithm 1 works in the following way. In the data preparation (line 1), we choose a piece of power series of $L$ time slots as the test series, and each time slot $t = 1, ..., L$ corresponds to a data pair ($\mathcal{X}_t$, $y_t$), where $y_t$ is the target power consumption, $\mathcal{X}_t$ is the recent historical data aligned as a vector used to predict $y_t$. Then for this collection of data, we choose the first $L_1$ time slots as the training data, the following $L_1 + 1$ to $L_2$ as the validation data, and the left time slots as the test data. During the training process (line 2-14), the training error function Eq. (1) is minimized by adjusting the parameters ($W_{AE}$, $W'_{AE}$, $b_{AE}$, $b'_{AE}$, $W$, and $b$) of the prediction model; at the same time, the validation error is checked to mark the parameter settings that achieve the minimum validation error, which is used as the final setting of the model (line 15). Note that Algorithm 1 is a general NN training framework, which can also be used to train NARX model by setting the corresponding error function in line 5.

**Algorithm 1** General NN Training Framework

1: *Data preparation*: Choose a piece of power series of $L$ time slots, for which we prepare a pair of data $(X^T, y)$, in which $X^T(t)$ is all the data aligned as a vector used to predict $y(t)$ (the power series), $t = 1, ..., L$. The input data with totally $L$ time slots are divided into three parts: time slots from $\tau$ to $L_1$ are used for training; time slots from $L_1 + 1$ to $L_2$ are used for validation; the left time slots are used for test.
2: *Initialization*: set $Epoch = 0$, best validation error $V_{best} = \infty$; denoting $\theta$ as the set of all control parameters of a NN (e.g., for RAE, $\theta$=$\{W_{AE}, W'_{AE}, b_{AE}, b'_{AE}, W, b\}$), randomly initialize $\theta$; set best NN parameter setting $\theta_{best} = \emptyset$; initialize $MaxEpoch$.
3: *//Training process*:
4: **while** $Epoch < MaxEpoch$ **do**
5: Compute the training error $\epsilon$ of the model (e.g., for RAE, the training error is computed by Eq. (1)).
6: Update $\theta$ to reduce $\epsilon$.
7: For $k \in \{L_1 + 1, ..., L_2\}$, compute prediction square error $e(k) = (y_k - y'_k)^2$.
8: $V_c = \sum_{k \in \{L_1+1,...,L_2\}} e(k)/(L_2 - L_1)$.
9: **if** $V_c < V_{best}$ **then**
10: $V_{best} \leftarrow V_c$.
11: $\theta_{best} \leftarrow \theta$.
12: **end if**
13: $Epoch = Epoch + 1$.
14: **end while**
15: *Output*: The best parameter settings $\theta_{best}$.

With the output of Algorithm 1, the best parameter settings are used to set the parameters $W_{AE}$, $W'_{AE}$, $b_{AE}$, $b'_{AE}$, $W$, and $b$ for the RAE. Then this well-trained RAE can be used for prediction.

## 5.3 Coarse-Grained Prediction Model

In this subsection, we present the coarse grained prediction model, aiming to predict the average power consumption in multiple time slots in the future, e.g., the subsequent one hour. As the power series we collected has a small time unit, i.e., 30 seconds, there exists a mismatch between the data collection scale and prediction scale. To tackle this issue, we first review existing methods, and then present our AE based method.

### 5.3.1 Different Methods for Coarse-Grained Prediction

Different methods can be used to utilize the historical data for the coarse-grained prediction. We use our case to illustrate these different methods. In our experiment, the objective is to predict the average power consumption of one hour in the future, which is 120 time slots of the power series we collected (after pretreatment). There are three ways to deal with the historical data (including all power, workload and system counters) as shown in Fig. 8 and their differences are summarized in Table. 5:

- Prediction with the original series: The original power series can be directly used for prediction. In this case, all the small time unit data from the last $\tau$ hours are used to predict the average power consumption of one hour in future. The input data contain $\tau \cdot 120$ time slots in our experiment, as there are 120 time slots in each hour. When these data are directly used in a simple NN for prediction, the input data will have a large dimension, i.e. $\tau \cdot 120 \cdot N$, where $N$ is the number of variables. Such a NN with high dimensional input is hard to train, making this method impractical.

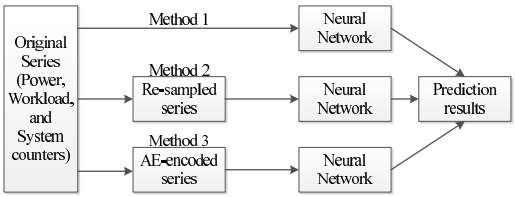



**Figure 8: Three methods to handle the input data for coarse-grained prediction. Method 1 directly utilizes the original series in an NN for prediction. Method 2 first re-samples the time series to the large time unit and then utilizes the re-sampled series for prediction. Method 3 is our proposed method, which first encodes the fine-grained series into lower dimension with an AE and then utilizes the encoded series for prediction.**

**Table 5: Comparison of different methods to handle the input data for coarse-grained prediction.**

| Prediction methods | Information retained | Input dimension |
|---|---|---|
| With the original series | Maximum(best) | Maximum |
| With the re-sampled series | Minimum | Minimum(best) |
| With encoded series | Medium | Medium |

- Prediction with re-sampled series: We can re-sample the power series to change the time unit to an hour by computing the average value in each hour for each variable. The prediction can then be conducted with the new series with NARX or RAE.

- Prediction with encoded series: We propose to use an AE to encode the small time unit data into lower dimension codes and then use these lower dimensional codes (codes of different input variables are combined into a vector) for prediction. Further details are shown below.

### 5.3.2 AE Based Coarse-Grained Prediction

The proposed AE based coarse-grained prediction method works as follows. We use an AE to encode all the data in each hourly time interval. Training process of the AE follows the framework of Algorithm 1, in which the objective is changed from prediction error to reconstruction error. After the training of the AE, the well-trained AE is used to encode all the data (training, validation and test). The codes are then used as the input data of the NARX or RAE model for prediction. Note that the AE code works similar to the average value used in the re-sampling method, but it can be a vector which can store more useful information.

# 6. EXPERIMENTAL RESULTS

In this section we evaluate the proposed fine-grained and coarse-grained prediction models with other baseline algorithms.

## 6.1 Experimental Settings

The experimental settings are as follows. For the fine-grained prediction model, we compare the prediction results of NARX and RAE. Training and test processes of both NARX and RAE follow Algorithm 1. We select 12 test series from WCN and CLN power series of length $L = 1000$, in which the first $L_1 = 700$ time slots are used for training; time slots from $L_1$ to $L_2 = 850$ are used for validation the left are used for test. The maximum number of training epochs is set to $MaxEpoch = 100$. The input data are normalized into zero mean and standard deviation before training. Note that these settings are exactly the same for NARX and RAE for fair comparison. The number of hidden nodes of the NARX is set to 10. The time delay $\tau$ is set to 2.

For the coarse-grained model, we compare different methods on four power series: WCN, WCA, CLN and CLA. Each time series has a total duration of 5 days, in which the first 70% are used for training, 70%-85% are used for validation and the last 15% are used for testing. $\tau$ is set to 5 for all these approaches, which means that the data of the past 5 hours are used to predict the data of the future one hour. For the data encoding with AE, we set the code length to 15 for WCN, 10 for WCA, 3 for CLN and 1 for CLA. Different code lengths can work differently and we manually tuned these parameter settings.

## 6.2 Comparison on Fine-Grained Prediction Models

In this subsection, we evaluate the proposed RAE model by comparing it with the NARX model.

In general, the accuracy of all the prediction methods relies on the similarity of the given training data and test data. The more similar of the two types of data, the better performance we will get. However the power consumption fluctuates pattern can change unexpectedly, for example, as illustrated in Figure 9(a), the training data (the first 70% time slots) and the test data (the last 15%) are distributed in different range. To demonstrate the fast tracking ability of RAE, we present the prediction results on three cases categorized by the change pattern difference of training data and test data in Fig. 9. Fig. 9(a) shows an example that the test data increase out of the range of the training data. This situation can be common in practice as the power demand may increase higher than the recent historical peak. We test the NARX and RAE on this example respectively, with the prediction results shown in Fig. 9(a). Clearly RAE achieves much better prediction results than NARX in this example. Fig. 9(b) shows another example that the test data decrease out of the range of the training data. For this case RAE also performs much better than NARX. Fig. 9(c) shows another example without such clear range difference between the training data and test data, for which RAE and NARX achieve similar prediction accuracy. The above results prove that RAE performs better than NARX on series when there is unexpected fast change.

Second, to make a more detailed comparison of NARX and RAE, we present the prediction error of NARX and RAE on the 12 test series in Table 6. For each test series, the prediction error is defined as normalized root mean square error (RMSE) $\epsilon_{nms}$,

$$\epsilon_{nms} = \frac{\sqrt{MSE}}{\sigma_y}. \quad (2)$$

where $\sigma_y$ denotes the standard deviation of the whole test power series. The experimental results are shown in Table 6, in which we present the "mean±standard deviation" of the prediction errors of 20 independent runs for each test series.

**Table 6: Prediction error (mean±std) comparison of NARX and RAE (smaller is better). Test results are based on 20 independent runs for each test sample. Better results are shown in bold face.**

| | WC98 | | Clark | |
|---|---|---|---|---|
| | NARX | RAE | NARX | RAE |
| 1 (500-1500) | 1.1±0.04 | **0.24±0.06** | 0.87±0.02 | **0.80±0.01** |
| 2 (1000-2000) | 0.22±0.02 | **0.17±0.01** | 0.88±0.02 | **0.87±0.01** |
| 3(1500-2500) | 0.64±0.03 | **0.21±0.02** | 1.14±0.03 | **1.02±0.07** |
| 4 (2000-3000) | 0.22±0.01 | **0.14±0.01** | 0.74±0.01 | **0.72±0.01** |
| 5 (8000-9000) | 0.79±0.09 | **0.76±0.02** | **0.82±0.02** | 1.12±0.18 |
| 6 (9000-10000) | 1.7±0.03 | **1.03±0.13** | 0.90±0.04 | **0.87±0.01** |

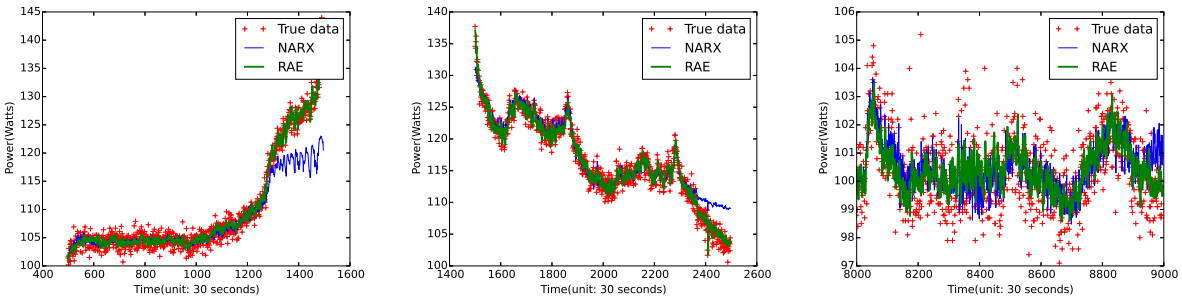
From Table 6 we can see that RAE outperforms NARX on most cases with smaller mean prediction errors. RAE can outperform NARX in the fine-grained prediction is due to that the fast fluctuation pattern of the power series in this small time unit. RAE, as a linear model, is more suitable than NARX. Note that in this case actually a linear version of NARX, namely ARX, can also perform better than NARX. But in our experiments, RAE can still outperform ARX (e.g., ARX can achieve error 0.32, compared to the error 0.24 of RAE for the WCN 500-1500 test case). The recursive structure of RAE shows its advantage in this case.

## 6.3 Comparison on Coarse-Grained Prediction Methods

In this subsection we compare the prediction results of the proposed AE encoded series based method and the re-sampled series based method. The results of prediction directly with the original series are omitted here, at the method failed in our experiments due to large input dimension.

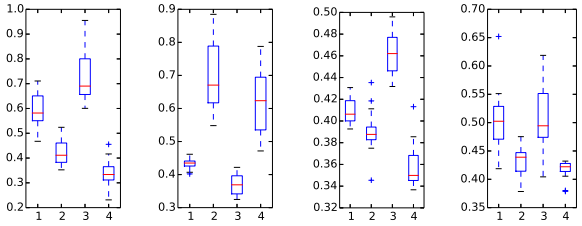
We compare the two methods on the four different power series WCN, WCA, CLN, and CLA. The prediction error distributions of 20 independent runs are shown in Fig. 10. We observe that with the RAE prediction model, the AE encoded series based method always has smaller prediction error (reduced up to 40%) for all four cases; when we use the NARX prediction model, the AE encoded series based method can be worse than NARX with the re-sampled data based method. However, overall, the best prediction results are always generated by the AE encoded series based method (with either NARX or RAE). It should be noticed that for the CLA case, the code length of the AE is set to 1, which means that the code has the same length to the average value used in re-sampling; however, with encoded data the prediction algorithm presents better prediction results, which shows that the code "learned" from the historical data can be more suitable for prediction than the average value directly "computed" from the data.

In summary, the AE encoded series based method can utilize massive historical data to give better prediction results. The AE can properly reduce the dimension of the




(a) (b) (c)

**Figure 9: Prediction results of NARX and RAE on three different cases: (a) increasing out-of-range: WCN (time slots: 500-1500), (b) decreasing out-of-range: CLN (time slots: 500-1500), (c) no out-of-range: WCN (time slots: 9000-10000)**



(a) (b) (c) (d)

**Figure 10: Prediction error distribution of NARX, RAE with re-sampled series (1 and 2), and NARX, RAE with AE encoded series (3 and 4) on different power series: (a) WCN, (b) WCA, (c) CLN, (d) CLA. Results are based on 20 independent runs. (3) or (4) is best in all four cases.**

input data, while retains more useful information than the simple average value used in re-sampling.

## 7. RELATED WORKS

In this section, we review related studies on power modelling of data centres, including static power modelling, dynamic power modelling. Also we review some common methods used in time series prediction and deep learning.

*Static power modelling*: Most of the power models of data centre proposed recently are static. In [7], Beloglazov et al. presented a thorough survey on power consumption of data centres, most of which are static models. According to their study, there are many different levels of power models, such as system level [28], virtualization level [22] [13] [11] [15], and data centre level [18], considering different aspects of data centres such as network [8], storage [29]. Static power models can be used to save energy [6] [16] [12] and achieve "green IT" as shown in [14] [17] [19]. Static models are useful to evaluate the power needed for a certain data centre in certain cases, however, it cannot reflect how the actual power consumption fluctuates in the runtime.

*Dynamic power modelling*: Other than static power modelling, there are also a few works on dynamic power modelling of data centres. In [26] the authors studied the temporal characteristics of power series collected from real data centres. They used hurst exponent to measure the self-similarity of the series. In this paper, we have extended their work and performed more power dynamics analysis. Also there are a number of works that utilizes the temporal continuity of the power consumption and use historical data to do power prediction and controlling. Especially, many are using historical data to do direct controlling and adjustment, such as the reinforcement learning method [24] [23], look ahead controlling [11]. Others are using historical data to build simple prediction models like [7] [3]. In these studies, the prediction is done by using simple mathematical formulas implicitly or explicitly. Different with these works, in this paper we propose a systematic prediction engine.

*Time series prediction methods and deep learning*: On time series prediction, AR and NARX are two popular prediction models, in this paper we compare the NARX with a deep NN model. On deep learning, recent deep learning studies [20] show that a deep NN can be constructed by stacking multiple building blocks, such as auto-encoder [10] and restricted Boltzmann machine (RBM) [9]. The effectiveness of these deep NN have been demonstrated in image recognition [10], language modelling [21] and various other applications.

## 8. CONCLUSION

In this paper, we proposed a deep learning based systematic prediction engine for data centre power consumption. The proposed prediction engine includes three modules, namely data acquisition, data pretreatment, and prediction models. For the prediction models, we propose to utilize RAE for short-duration fine-grained prediction, which can track the fast changes of the power consumption of a data centre. We also proposed to use AE to encode large amount of fine-grained data for long-duration coarse-grained prediction, which achieves smaller prediction error and better stability than common re-sampling based method.

## 9. REFERENCES

[1] M. Arlitt and T. Jin. 1998 world cup web site access logs, 1998.

[2] M. F. Arlitt and C. L. Williamson. Web server workload characterization: The search for invariants. In *ACM SIGMETRICS Performance Evaluation Review*, volume 24, pages 126–137. ACM, 1996.

[3] J. Choi, S. Govindan, B. Urgaonkar, and A. Sivasubramaniam. Profiling, prediction, and capping of power consumption in consolidated environments. In *IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*, pages 1–10. IEEE, 2008.

[4] G. W. Corder and D. I. Foreman. *Nonparametric statistics for non-statisticians: a step-by-step approach*. John Wiley & Sons, 2009.

[5] J. D. Davis, S. Rivoire, M. Goldszmidt, and E. K. Ardestani. Chaos: Composable highly accurate os-based power models. In *IEEE International Symposium on Workload Characterization (IISWC)*, pages 153–163. IEEE, 2012.

[6] E. M. Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. In *Power-Aware Computer Systems*, pages 179–197. Springer, 2003.

[7] A. Gandhi, Y. Chen, D. Gmach, M. Arlitt, and M. Marwah. Minimizing data center sla violations and power consumption via hybrid resource provisioning. In *2011 International Green Computing Conference and Workshops (IGCC)*, pages 1–8. IEEE, 2011.

[8] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: Saving energy in data center networks. In *NSDI*, volume 10, pages 249–264, 2010.

[9] G. E. Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.

[10] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[11] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. *Cluster computing*, 12(1):1–15, 2009.

[12] D. Meisner, B. T. Gold, and T. F. Wenisch. Powernap: eliminating server idle power. *ACM SIGARCH Computer Architecture News*, 37(1):205–216, 2009.

[13] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[14] S. Murugesan and G. Gangadharan. *Harnessing green IT: Principles and practices*. John Wiley & Sons, 2012.

[15] R. Nathuji and K. Schwan. Virtualpower: coordinated power management in virtualized enterprise systems. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 265–278. ACM, 2007.

[16] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load balancing and unbalancing for power and performance in cluster-based systems. In *Workshop on compilers and operating systems for low power*, volume 180, pages 182–195. Barcelona, Spain, 2001.

[17] E. N. Power. Energy logic: Reducing data center energy consumption by creating savings that cascade across systems. *A White Paper from the Experts in Business-Critical Continuity*, 2008.

[18] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. Ensemble-level power management for dense blade servers. In *ACM SIGARCH Computer Architecture News*, volume 34, pages 66–77. IEEE Computer Society, 2006.

[19] R. Sawyer. Calculating total power requirements for data centers. *American Power Conversion, Tech. Rep*, 70:80–90, 2004.

[20] J. Schmidhuber. Deep learning in neural networks: An overview. *arXiv preprint arXiv:1404.7828*, 2014.

[21] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011.

[22] Y. Song, H. Wang, Y. Li, B. Feng, and Y. Sun. Multi-tiered on-demand resource scheduling for vm-based data center. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 148–155. IEEE Computer Society, 2009.

[23] G. Tesauro, R. Das, H. Chan, J. Kephart, D. Levine, F. Rawson, and C. Lefurgy. Managing power consumption and performance of computing systems using reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1497–1504, 2007.

[24] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani. A hybrid reinforcement learning approach to autonomic resource allocation. In *IEEE International Conference on Autonomic Computing (ICAC)*, pages 65–73. IEEE, 2006.

[25] G. Von Laszewski, L. Wang, A. J. Younge, and X. He. Power-aware scheduling of virtual machines in dvfs-enabled clusters. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–10. IEEE, 2009.

[26] D. Wang, C. Ren, S. Govindan, A. Sivasubramaniam, B. Urgaonkar, A. Kansal, and K. Vaid. Ace: Abstracting, characterizing and exploiting datacenter power demands. In *IEEE International Symposium on Workload Characterization (IISWC)*, pages 44–55. IEEE, 2013.

[27] J. Yin, P. Sun, Y. Wen, H. Gong, M. Liu, X. Li, H. You, J. Gao, and C. Lin. Cloud3dview: an interactive tool for cloud data center operations. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 499–500. ACM, 2013.

[28] F. Zanini, D. Atienza, L. Benini, and G. De Micheli. Multicore thermal management with model predictive control. In *European Conference on Circuit Theory and Design (ECCTD)*, pages 711–714. IEEE, 2009.

[29] Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, Y. Zhou, and P. Cao. Reducing energy consumption of disk storage using power-aware cache management. In *IEE Proceedings-Software*, pages 118–118. IEEE, 2004.