Modeling Power Consumption of Applications Software Running on Servers

by

Fadwa Abdulhalim

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2015

Authors Declaration

Abstract

Reducing power consumption in computational processes is important to software developers. Ideally, a tremendous amount of software design efforts goes into considerations that are critical to power efficiencies of computer systems. Sometimes, software is designed by a high-level developer not aware of underlying physical components of the system architecture, which can be exploited. Furthermore, even if a developer is aware, they design software geared towards mass end-user adoption and thus go for cross-compatibility. The challenge for the software designer is to utilize dynamic hardware adaptations. Dynamic hardware adaptations make it possible to reduce power consumption and overall chip temperature by reducing the amount of available performance. However these adaptations generally rely on input from temperature sensors, and due to thermal inertia in microprocessor packaging, the detection of temperature changes significantly lag the power events that caused them.

This work provides energy performance evaluation and power consumption estimation of applications running on a server using performance counters. Counter data of various performance indicators are collected using the CollectD tool. Simultaneously, during the test, a Power Meter (TED5000) is used to monitor the actual power drawn by the computer server. Furthermore, stress tests are performed to examine power fluctuations in response to the performance counts of four hardware subsystems: CPU, memory, disk, and network interface. A neural network model (NNM) and a linear polynomial model (LPM) have been developed based on process count information gathered by CollectD. These two models have been validated by four different scenarios running on three different platforms (three real servers.) Our experimental results show that system power consumption can be estimated with an average mean absolute error (MAE) between 11% to 15% on new system servers. While on old system servers, the average MAE is between 1% to 4%. Also, we find that NNM has better estimation results than the LPM, resulting in 1.5% reduction in MAE of energy estimation when compared to the LPM.

The detailed contributions of the thesis are as follows: (i) develop a non-exclusive test bench to measure the power consumption of an application running on a server; (ii) provide a practical approach to extracting system performance counters and simplifying them to get the model parameters; (iii) a modeling procedure is proposed and implemented for predicting the power cost of application software using performance counters. All of our contributions and the proposed procedure have been validated with numerous measurements on a real test bench. The results of this work can be used by application developers to make implementation-level decisions that affect the energy efficiency of software applications.

Acknowledgements

First and foremost, I am grateful to Almighty Allah for His countless blessings and for giving me the knowledge and strength to accomplish my research work.

I would like to express my thanks and sincere gratitude to my supervisor, Professor Kshirasagar Naik, for the continuous support of my Master study, for his motivation, patience, and knowledge. This work would have not been done without his advice and valuable comments.

Besides my advisor, I would like to thank my committee members, Professor Pin Han Ho and Professor Igor Ivkovic, for their constructive feedback and very valuable comments and suggestions.

My deepest appreciation and my grateful thanks go to my father, Abdulkarim Abdulhalim, and my mother, Gamar Albokhari, who were the permanent source of love, encouragements, and support. Words cannot express how grateful I am for all of the sacrifices that they have made on my behalf. Their prayers are the only reason behind my success in my study and my life in general.

Foremost amongst the individuals to whom I am thankful is my husband, Abdullah Alturkestani, for his love, patience, and continuous motivation. He was always my support in the moments when there was no one to answer my queries. My child, Ameen, who make my life colorful, enjoyable, and full of fun. I can not imagine my life without him.

A special thanks to my brothers and sisters, my father-in-law, my mother-in-law, and my aunt Samia for their support, help, and love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Electrical energy is fundamental to all computer systems [1], [2]. However, it is not ubiquitous, and it is expensive. Those in the business of operating high-volume data centres, servers, and bitcoin mining farms all collectively have an interest in understanding how their systems' resources are utilized. Often times however, fitting vast numbers of components with thermal detection instrumentation is not economically feasible.

Reducing power consumption in computational processes is also important to software developers [3], [4], [5]. Ideally, a tremendous amount of software design efforts goes into considerations that are critical to power efficiencies of computer systems. Sometimes, software is designed by a high-level developer not aware of underlying physical components of the system architecture, which can be exploited. Furthermore, even if a developer is aware, they design software geared towards mass end-user adoption and thus go for cross-compatibility. The challenge for the software designer is to utilize dynamic hardware adaptations. Dynamic hardware adaptations make it possible to reduce power consumption and overall chip temperature by reducing the amount of available performance. However these adaptations generally rely on input from temperature sensors, and due to thermal inertia in microprocessor packaging [6], [7], [8], [9], the detection of temperature changes significantly lag the power events that caused them.

A work-around to dynamically gauge a system's performance is to use performance counters that have been demonstrated to effectively proxy power consumption [10], [11]. Performance counters count the instances of local processor events and calls, and thus eliminate the need for sensors distributed about various parts of the system. However, considerable modelling challenges exist in relating the statistical count information to what is truly happening internally with respect to power consumption. Consequently, there is considerable financial interest in developing accurate models that use the performance counts to cost-effectively provide up-to-date power consumption information. With customization, engineers can target and customize specific aspects of system responses in response to stress tests.

This work aims to provide energy performance evaluation and power consumption estimation of an application running on a server using performance counters. Counter data of various performance indicators will be collected using the *CollectD* tool. Simultaneously during the test, a Power Meter (*TED5000*) will be used to monitor the actual power drawn by the computer server. Furthermore, stress tests are performed to examine power fluctuations in response to the performance counts of four hardware subsystems: CPU, memory, disk, and network interface. A neural network model (NNM) and linear polynomial model (LPM) have been trained based on process count information gathered by CollectD. These two models have been validated by four different scenarios running on three different platforms (three real servers). We provide a modelling procedure and tools that help to estimate system power consumption without the need of using a power meter device.

1.1 Problem Description

The power consumption is majorly affected by the design of the software application [3], [12]. In order to reduce the power consumption, many new techniques have been introduced lately for the mobile phones [5] and software systems in particular [3] [4]. The major cost involved in maintaining a data centre is the power bill [13], and hence analysis should be carried out in order to understand how the energy consumption be minimized while running applications on servers. While the application software is in the design phase, there are ways to minimize the energy cost but they take much of time and effort. Therefore, designers do not pay much attention on the cost aspect of their applications software as they have to make various decisions pertaining to the design of the application software. Thus they do not know how much energy would be consumed by the application software when they are tested on real servers in the data centres.

In this thesis, a number of problems are addressed which share a common objective of achieving energy efficiency in data centre. We cover two primary challenges regarding software application development process. The first challenge is to have high-level energy cost information so that software developer can take advantage of that during the design phase of energy efficiency applications. The second challenge is to have a consistent energy evaluation test bench for comparing test results of system power consumption. We propose a non-exclusive measurement test bench that considers the values of different subsystem parameters during a test, and propose a modelling procedure that can be used with an existing application development process to achieve better energy efficiency.

1.2 Solution Strategy and Contributions

We develop a non-exclusive test bench to measure the power consumption of an application running on a server. The test bench comprises a power meter and a monitoring computer that acquires power readings from the power meter and system performance counters from the server. Our test bench can be used to measure the total power cost of a server. By using this test bench, we proposed a modeling procedure that would help in predicting how much energy would be consumed by a particular application software.

Through the use of this modeling procedure, a developer can estimate the system power consumption without the need of using an actual power meter device. Also, it can help the developers to measure the power consumption of their applications software which can help them to come up with an energy efficient design. Most of this work has been published in 2015 [14].

This study makes three important contributions:

1. It shows how system performance counters can be used to model power consumption profiles for applications software running on server.
2. It lays out a modelling methodology to estimate the power profile of application software, which avoids repeated power modelling for the same server but different applications software.
3. An experiment is performed to contrast the different energy consumption behaviours in terms of different models, platforms and different load scenarios.

1.3 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we present a comprehensive literature review of power efficiency of data centre. In addition, various methodologies of energy efficiency improvement are discussed. We also compare our test bench and modeling procedure with the existing work. In Chapter 3, System model of test bench, implementation details and collecting and simplifying data for determining variables for power consumption model have been explained. We also present our modeling procedure to estimate the power consumption of application software running on servers. Chapter 4 lays out two power consumption models (Neural Network and Polynomial) that have been implemented and validated using three platforms and four different scenarios. Some concluding remarks are provided in Chapter 5.

Chapter 2

Literature Review

2.1 Power Efficiency of Data Centres

Data centre are design to deliver a wide range of services such as cloud computing and storage services to vast variety of organizations and enterprises. During the last two decades, the term data centre started to gain a lot of recognition as the best replacement for old computer rooms. Many markets drivers are related to economic and financial applications such as the growth of online business and banking services; while, some other drivers focus on educational aspects including distance learning and online degrees. Regardless which market driver is motivating this trend towards using data centres, tremendous and real benefits can be earned from using these cloud computing facilities properly. At the beginning, data centres were designed in the absence of established standard and policies. As a result, service providers were focusing mainly on performance, reliability and quality of service delivered to consumers. This focus to raise the cost associated with powering and cooling those huge structures and creating the problem of high power expenditure. Consequently, this exaggerated energy amount consumed by data centres made energy management a critical aspect in designing and developing these huge facilities. The rising energy consumption of such facilities is great concern due to its adverse financial and environmental costs.

Generally, energy consumption can be described as the average power in watt consumed per second:

$$Energy(joule) = AveragePower(watt) \times Time(sec) \tag{2.1}$$

If the energy amount spent by any data centre is relatively small, then that data centre can be considered as energy efficient, but this is not always the case in real time applications. Due to the increasing demand on cloud computing and web services such as search, music downloads, video-on-demand, and social networking, most currently used servers are far away from being green data centres. Data centres are consuming large amounts of power on regular basis, and this issue is becoming more and more serious as demands on computing service are increasing. Due to its adverse impacts on the environment and the global economy, excessive power consumption in data centres needed to be regulated, managed and reduced to lower and acceptable levels or what is known as energy efficiency. According to Irrek and Stefan from Wuppertal institute in Germany [15], the term efficiency can be defined as the ratio of benefits to expenses. Energy efficiency, therefore, describes the ratio between the benefit gained and the energy used. By applying physics laws and using equation 2.1, energy efficiency can be found as shown in equation 2.2.

$$EnergyEfficiency = \frac{workload}{energy} = \frac{workload}{power \times time} \tag{2.2}$$

or simply:

$$EnergyEfficiency = \frac{performance}{power} \tag{2.3}$$

Energy Efficient data centres are the perfect solution to the growing energy loss, but what makes any data centre energy efficient or green? There are some green metrics that can be applied to determine whether a given data centre is energy friendly or not. Table 2.1 presents some suggested green metrics for data centres [16].

The power usage effectiveness (PUE) metric is the most important factor and standard metric for measuring data centres' energy efficiency. It represents the ratio between the total power going into a data centre and the amount of power used to run IT equipment (servers, storage, and network). The power usage effectiveness (PUE) metric was introduced by the green grid, an association of IT professionals focused on increasing the energy efficiency of data centres [17], [18], [19]. If the value of PUE is equal to 1, then efficiency represents the optimal achievable

METRIC	DESCRIPTION	FORMULATION
PUE	Power usage effectiveness	PUE = $\frac{\text{total data centre source energy}}{\text{IT source energy}}$
CUE	Carbon usage effectiveness	CUE = $\frac{\text{total } CO_2 \text{ emission from total data centre energy}}{\text{IT source energy}}$
ERF	Energy reuse factor	ERF = $\frac{\text{Reuse energy outside of the data centre}}{\text{total data centre source energy}}$
DCeP	Data centre energy productivity	DCeP = $\frac{\text{useful work produced}}{\text{total data centre source energy consumed producing this work}}$

Table 2.1: Green Metrics for Data Centres

efficiency, which means that all power entering the facility is consumed at IT equipment and nothing is wasted. In real time applications, PUE value of 1 has not been achieved yet, and 1.12 is the closest number to it, which was announced by Google Company [19]. Power must be transferred to power support infrastructure. In addition, some of the power is consumed due to losses in the power system. The remaining power then goes to run the IT load. We can improve data centre energy efficiency and reduce our PUE by minimize either the power going to the support infrastructure or losses in the power system [19]. When we apply green metrics in the designing process of future data centres, the construction expenses will be higher if compared with energy inefficient structures, but this is only a one-time expense and cost-related savings over time are much higher.

2.2 Energy Efficiency Improvement Methodologies

Before we introduce the main two categories to manage energy usage in data centres, we will present some considerations to be taken into account in order to design functional energy management techniques.

- Examine data centre's components to find out what components are spending more power when compared to other components.
- Think of some possible ways to manage power consumption in these components individually.
- Try to apply these solutions and test their efficiency on each components energy expenditure.

- Examine the whole computing system with one component being adjusted at a time, so that you will know the effect of each part of the data centre on the total energy consumption and performance.
- Using trial and error. You can figure out the best way to manage energy under different situations and loads.
- Different data centres require different tuning procedures to reduce total energy usage, which means what may apply for one data centre may not work on other data centres.
- Hardware based techniques and using power efficient components require software modification in order to achieve the best results.

The public perception of data centres' large power budget is encouraging and motivating owners of such facilities and operators to take an action to reduce the costs. Based on the related work, we can categorize energy management strategies into hardware and software based efforts.

2.2.1 Hardware Based Efforts

Hardware equipment is becoming better suited to control energy in data centres; as a result, a lot of research has been done to develop functional solutions to produce energy conscious hardware components. Based on the existing work that we have surveyed, we divide the hardware based efforts into two main categories: circuits-based (low-level hardware component optimizations) and architecture-based (hardware organization, also known as physical design).

1. Circuit-based: Most recently developed models involve low-level (circuit-based) techniques which manage energy loss in data centres such as utilizing energy efficient circuits, clock gating, or dynamic voltage-frequency scaling [20]. These techniques focus on improving the internal circuits' functionality of a data centre by using less power consuming circuits, preventing clock signal from entering inactive sections and lowering operation voltage/frequency of these circuits. All these approaches share a common principle of achieving more/same performance with less power, but some differences are applied. Zeydel and Oklobdzija claim that designing energy efficient circuits such as Limited Switch Dynamic Logic (LSDL) ensures minimal power consumption with a desirable level of system performance at a constant delay, and possible power savings would be 30-50% [21]. However, when using this approach to reduce energy, the system speed/frequency is slightly decreased, and high performance standards cannot be accomplished.

Bol shows that utilizing energy efficient circuits' technique is more valuable for applications that require low or medium performance requirements [22].

Another example of circuit-based approaches is clock gating. Clock gating techniques are simply based on cutting off the clock signal from inactive regions of digital circuits. When the clock signal does not reach a specific part of a circuit, power savings can be made because the unnecessary switching activities of flip flops and clock buffering tree can be avoided. Yeo and Lee [23] recently proposed two techniques for clock gating known as Latch-free clock gating and Latch-based clock gating. In Latch-free clock gating method, two input AND gate is employed to allow or cut off the clock signal while in the Latch-based clock gating method, the clock signals enabled or disabled using level sensitive latch. The main drawback of using clock gating approaches is due to additional propagation delays/time, which may reduce the circuit frequency/speed.

In addition to energy efficient circuits and clock gating, we can increase power savings of a data centre by the dynamic voltage-frequency scaling (DVFS) technique. The basic principle of this technique is to lower a microprocessor's operating voltage and/or frequency (number of instructions a processor may handle in a specific time interval) and thus increasing energy savings because the active power of a CMOS circuit decreases linearly with frequency and quadratically with voltage as shown in equation 3 [23]. Mechanisms like PowerNow [24], Cool'nQuiet [25], are good examples of the DYFS approach. Generally, these measures can slow down CPU clock frequency or power off parts of the chips, if they are idle [26] [27].

PowerNow technique is a built in mechanism in AMD's processors, and it is designed to decrease both the CPU's frequency/clock speed and operating voltage in an automatic and independent style when the data centre is idle or when its load is low. This voltage/frequency scaling method has many benefits such as reduce power consumption, generated heat and noise because it supports several power modes known as high-performance, power-saver and automatic mode [24].

In *high-Performance mode*, the highest possible performance is achieved at maximum voltage and frequency; whereas, the Power-Saver mode ensures that the microprocessor's operation is under the minimum rated voltage and speed limits with the most possible power efficiency. Finally, the automatic mode both frequency and voltage are adjusted automatically depending on the application's performance requirements and only the right amount of power will be dissipated [24] [28].

Cool'nQuiet technique is similar to PowerNow because they share the same principle of reducing the operating voltage and clock speed when the system is idle to lower the dissipated heat, lost power and generated noise, which can be understood by its name [25]. The main difference between the two methods is that Cool'nQaiet is more suitable with server chips, while PouerNow is used for mobile chips. In general, DVFS technique can increase power savings to as high as 36% for web-services workloads [28]. However, working at lower voltage and/or frequency may affect the system's overall performance [23].

2. Architecture-based: these approaches reduce data centres' energy waste based on their physical design. One of these approaches is using more power-efficient alternatives [29]. This approach focuses on replacing power-consuming hardware components with more power-efficient alternatives which achieve the same task with less energy [29] [30]. For example, more energy conscious blade servers can replace rack-mount servers because Blade servers consume about 10% less power than equivalent rack mount servers [31]. Another example, use energy-efficient power supplies and fans instead of server power supplies that are in use today and are operating at lower efficiencies [29] [31]. Larger power supplies and fine cost more, but is a one-time expense and the power savings are worth it [31].

Felter et al [32] introduced another approach that is called power shifting. Power shifting dynamically re-allocates the system's power among the most active components based on each component's activity. In other words, this technique prevents any data centre from working with its maximum energy which in turn reduces energy consumption and increases savings. The process of re-distributing the power can be accomplished through many mechanisms such as *threshold-based throttling* which dynamically divides power among active components using workload-sensitive policies. By employing power shifting in any data centre, power expenditure decreases with almost no/little impact on the system's performance and quality of service. However, if for any reason the sum of active components' power budgets exceeded the system's budget, a significant drop in performance or even a power supply failure may be resulted.

Grrumurthi et al [33] recently proposed another approach to reduce power consumption by using multi-speed disks. This approach is called dynamic rotations per minute or DRPM, and it modulates disk speed (rpm) dynamically. Standard disks spin at a constant speed rate of 10,000 or 15,000 rpm; while multi-speed disks can spin at various speed rates depending on the existing workloads and thus avoiding working with full speed all the time. For instance, lower spin speed will be assigned to lower workloads, which increases power savings when the data centre is idle or when it is running low loads. Unlike other techniques that spin down the disks, DRPM can decrease the speed rate and restore to normal level in a shorter time, which makes this techniques very effective in servicing requests that with a short idleness interval in between. Although DRPM method ensures serving requests at lower speed rates and with considerable power savings, multi-speeds disks are not used vastly in today's data centres.

Although most of power-reduction mechanisms focus on microprocessors' architects, some active research is concerned in developing on-chip memories or caches. Zhu et al [34] presented a new technique that includes storage cache replacement. This approach selects specific blocks from certain disks in the main memory cache and preserve then, which enables these disks to operate in lower power modes for longer time intervals.

Also, Yeo and Lee [23] introduced an Architecture-based approach called Selective Cache Ways which reduces the power dissipation in processor's caches. This technique switches off selective subset of cache ways to allow associative cache at run-time. These cache ways are created by portioning the on-chip cache into multiple subarrays to minimize latencies, and these cache ways can be turned off by a certain application when the cache activities are low. The selective cache ways techniques have a little impact on performance, and it can improve the overall cache energy dissipation by 40%.

2.2.2 Software Based Efforts

In order to minimize power consumed by hardware equipment which runs the variety of applications and services within data centres, power-management software efforts have been created and implemented. With this power-management software efforts equipment in a data centre can modify how they operate in order to optimize energy savings. Depending on the surveyed work, we can categorize four software based approaches as follows:

1. ***Optimizing software:*** Because most of the existing software codes were written at a time when energy expenditure was not a key factor in designing data centres, it is more efficient to optimize these codes in a way they achieve high performance in fewer clock cycles and make better use of underlying hardware capabilities. By optimizing software codes to consider energy as well as performance, extra benefits can be earned such as monitoring and dynamically controlling power consumption in an effective way. Furthermore, if researchers kept on optimizing existing software, only required software components would be running at any particular time with minimal Power wasted [35].

2. *Server consolidation:* consolidation allows different tasks to share available resources, so they use less equipment and less power. Whenever system resources are not fully utilized, the system may allow other tasks to share the resources or allow the unshared resources to enter a suspended or reduced power mode to save energy. System consolidation includes two mechanisms hardware location and Virtualization.

Hardware location mechanism enables some hardware components to be grouped together depending on some similarities. For example, when hardware components with similar heat load densities and operating temperature are grouped, minimal fan power and more cooling system performance can be accomplished. Similarly, virtualization has a similar principle to hardware location, which is running several independent virtual operating systems on only one physical processor. By using virtualization, we can run different operating systems without lowering the CPU's utilization, and we can also reduce both the number of servers in a data centre and the power consumed by each server without compromising on the system's performance [35] [36].

2.3 System Power Consumption Models

Diverse models, which are created with performance counter information, have been used as a predictor of temperature [37], total system power consumption [9], [11], [38], and subsystem power [10], [39], [40], [41]. Even though those models are simple and fast, they are sensitive to benchmarks chosen for model calibration. Furthermore, those models do not consider full system power consumption. For overall power consumption, each of those models requires one or more measurement phases to determine the contribution of each subsystem. Although all the components vary in their power demands with respect to certain process calls, on average, microprocessors are the largest consumers of power, while other subsystems constituted 40-60 percent of the total power [11].

Must of those models were built using performance counters, and linear regression has been used to train those models [10], [11]. The feasibility of using performance events to model real-time use of power in a computer system has been effectively demonstrated. Performance events, which are readily visible and accessible from the CPU, are highly correlated to power consumption in all the subsystems, including memory, I/O, disc and processor. The authors of the paper [10] produced effective performance counter models that independently measured power for the six main components within a computer: microprocessor, graphics processing unit (GPU), chipset, memory, I/O, and disk. The authors at [42], [43] developed an automated test bench that developers can use to measure the energy cost of the their applications for their various design choices.

Our work is different from those works as follows: (i) our power model of a system is developed through analyzing the various count information from CPU, memory, disk and network interactions; (ii) our test bench estimates the power cost of an application running on server without the need for power sensing hardware; (iii) neural network model and linear polynomial model have been used to train the power consumption model. In order to properly model system performance via counter-based models, it is necessary to have a methodology to completely represent the power consumption of a system, which is what we do in this work.

2.4 Review of Existing Research

Luo et al. [44] have highlighted several key aspects of the power consumption, methods of controlling and reducing it in the cloud data centers. They mention the importance of virtualized systems for conserving energy which would empower the users to turn off sources that are not in use, thereby saving power. Through virtual systems, the cloud data centers can save a large amount of power by turning off machines that are not in use, also helping the data centers to maintain a temperature that is suitable for its performance. Liang Lao has also touched upon the need to understand the users, their profiles and their usage patterns to analyze the actual extent of power that would be required and adopt the most suitable strategy of allocating data and energy resources.

Zhang et al. [45] presented their arguments for having a benchmark of assessing the consumption of power based on the applications used and the pattern of usage of the users. So, they have emphasized on a more user-centric approach for the assessment of the consumption of energy. They also highlighted the importance of associating the user-centric approach to assess the energy consumption patterns and arrive at a more concrete model. A computing device can have several running applications, from web browsers, text-editor applications to complex professional applications all of which have specific energy utilization. Zhang and Hindle have based their approach on three situations and scenarios and have applied the most widely used standardized bench-markings. It was observed that applications can a varied pattern of energy consumption when undergoing the same operation. It was also observed that the platform of the application had a significant bearing on the amount of energy it consumes. Among the other significant observations, Zhang and Hindles scenarios presented facts that web based programs consume a lot more energy than others and also the mote-worthy point that idle applications also consume energy.

Kothiyal et al. [46] have analyzed and thrown light on several ways by which the power consumption of devices can be lowered. The paper discusses various methods of controlling the power consumption used in devices to allow users a better experience. They throw light on several methods of file compression algorithms that can be implemented on several types of different data files and assess their results and energy usage performance. They elaborately expresses the mechanism that has been used to achieve the purpose of the paper. The paper talks about compressing the files at the processing stage and writing them to the disk. Once this is done, the files could be read and accessed from the disk and decompressed, therefore, helping in reducing energy consumption. They talk about the effectiveness of the compression algorithms as an important tool that determines the performance and effective power consumption in a computing device.

Sabharwal et al. [3] discussed about a computing system that encourages environment friendly standards by trying to reduce energy consumption. The emphasis on creating a process for maintaining , controlling and monitoring energy consumption levels has brought to light the need for software and hardware solutions that can monitor the energy consumption levels and focus on greener methods. They talked extensively about optimizing the energy usage at the software and the usage of specific energy efficient techniques like Green Scheduling which assesses the energy usage. So, if a computing system is able to implement green and energy efficient techniques in its processing stage, it can conserve a lot of energy and contribute to the green initiative. The article also stresses on energy conservation through energy efficient software which incorporate several energy conservation methods that can be implemented in the design phase of the software. Although, incorporating energy efficient techniques right at the designing stage can be a difficult process, the concept of virtual machines provide a likely solution. This concept allows developers and programmers to design their compilations on different platforms single or multiple nodes. Their paper has extensively focused on providing hardware and software support in providing an energy efficient computing system.

Sun et al. [47] developed a solution of achieving a low cost mechanism for assessing the power levels at the sub-system which has been named as DiPART or Disaggregated Power Analysis in Real Time. This tool would help estimate and analyze the power levels at the sub-system based on several events and a power sensor that encompasses the system.

McCullough et al. [48] discussed about simpler models and their compatibility with more advanced platforms and has discussed if they can be compatible to the individual subsystems that are operating within a system. They also discussed about having a concrete management and monitoring solution of the power consumption levels than following the present practice of assumptions based on specific benchmarks and standards. They throw light upon the fact how the platforms are capable of adding its own power monitoring and assessment system to its performance. These power assessment systems used in the platforms by the manufacturers can be more accurate and precise in its measurement and response

Chapter 3

Modelling the Energy Cost of Application Software for Developers

This chapter presents a non-exclusive test bench to measure the power consumption of an application running on a server. The test bench consists of power meter and development computer that extracts the power readings and the performance counters of system during running a workload on the server.

Also, this chapter provides modelling procedure that help software developers to evaluate energy performance of their applications. Different models can be trained based on process count information gathered by CollectD and actual real-time power consumption monitored by a TED5000 power meter. By using measurement of actual system running different workloads, power models for four subsystems (CPU, memory, disk and network interface) on three platforms (three real servers) are developed and validated. Through the use of this modeling procedure, a developer can estimate the system power consumption without the need of using an actual power meter device.

3.1 System Model of Test Bench

The test bench in our power modeling process has been shown in Figure 3.1. The definitions of all the terms are as follows:

Server: A system that runs the software application that the developer is interested in evaluating the energy cost of the application.
Load: It is the software application of which we want to evaluate the energy performance.
Performance Counter: We used CollectD which is an open source daemon that collects system performance counters periodically.
Power Meter: An external device that is used for measuring the power drawn by the server. We used TED5000.
Wall Power: This supplies the AC power.
Development Environment: This is a computer equipped with the software tool (MATLAB) that are used to analyze and model the data coming from the Server and the Meter.

Our test bench is used to measure the total power cost of a server. To set up the test bench for power measurement, we connected the power meter to the server via a Current Transformers (CRT) and connected the server and the development environment to the same LAN (Local Area Network).

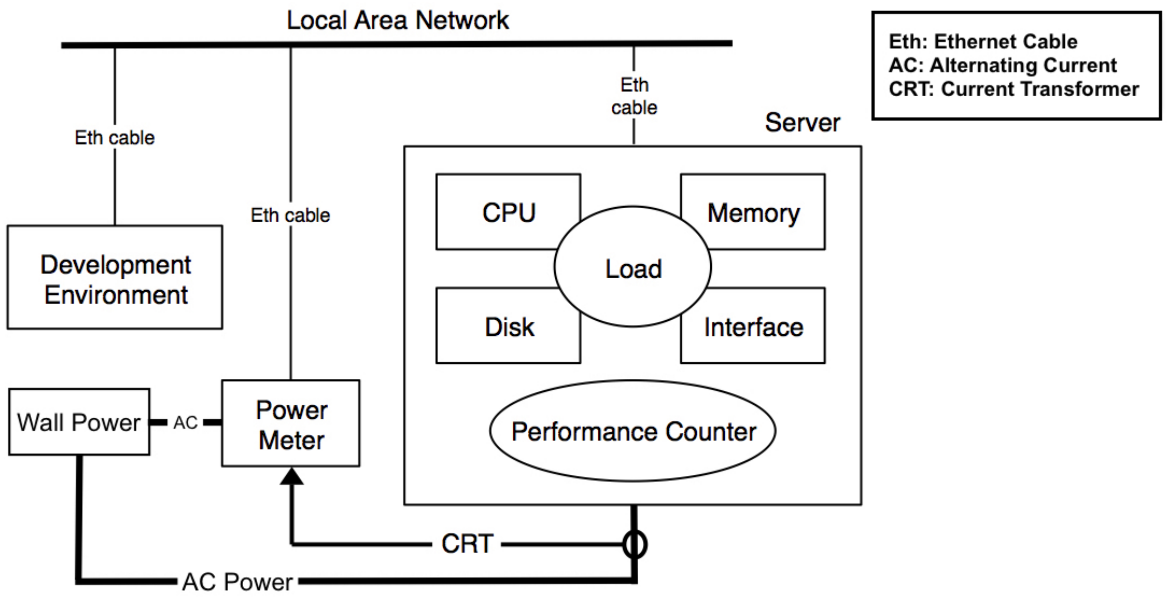

Figure 3.1: Test Bench Framework

3.2 Modeling Process

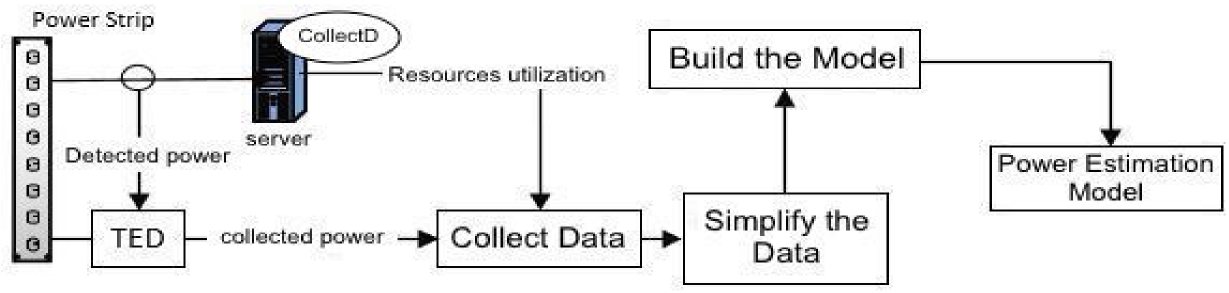

Figure 3.2: Modelling Process

Our modelling process involves several steps that have been illustrated in Figure 3.2. The first step is to run the workload and the performance counters (collectD) on the server that is connected to the power meter (TED5000). For the workload, we used a stress tool called *stress*, which is a simple tool used to impose varying amounts of load on the CPU, memory, I/O, and disk. We used it to produce heavy load on selected subsystems (CPU, memory, and disk) in order to drive the basic correlation between their utilization and power consumption. We will explain more about the workload in section 3.3. To collect the performance information for each subsystem, we implemented *CollectD*. We used *CollectD* to collect performance metrics from four subsystems (CPU, memory, disk and interface). More explanation about *CollectD* will be in section 3.4.

The second step is to analyze the collected data to perform modelling. In the development machine, we wrote a Matlab code that helps to simplify the subsystem performance data files. Each subsystem has many metrics collected for it. We reduced the metrics to have only four main metrics, one for each subsystem: CPU, memory, disk, and interface. We explained the reduction process of the metrics in section 3.4.

The final step is to formulate the model based on the performance metrics and the power meter data that have been collected during the load process to predict the power. A Neural Network technique and Polynomial technique are used to fit the data relating performance metrics to the power variation for the system. By using the four subsystem metrics as input variables and power meter data as a target variable, we performed power estimation model that predicts the power consumption of application running on server. In section 3.5, we will give further details about formulating the power estimation model.

3.3 Workload Design

The workload is important for developing and tuning the power models. Our workload is chosen based on its apparent utilization of a subsystem. In order to meet the requirement of subsystem utilization, we employ the *stress* tool to produce very high utilization.

The *stress* tool is a simple tool used to impose varying amounts of stress on operating systems. We use it to produce heavy loads on selected subsystems (CPU, memory and disk) in order to drive the basic correlation between their utilization and power consumption.

For a server with eight core CPUs, we fully loaded all of them gradually one by one for one minute per core until we reached 100% CPU load. Also, the memory was loaded gradually to reach 100%. Disk also was loaded with the *stress* tool. For the interface, downloading a large file (about 7 Gigabyte) was the load. Loading CPU, memory and disk took 24 minutes and the interface load took about 40 minute. The workload for each platform is showing in table 3.1.

Table 3.1: Workload Design

Utility	Workload
CPU	loading each core to 100% for 1 minute
Main Memory	Gradually loading memory until reach 100 %
Hard Disk	Gradually loading disk to 100 % for 8 minute
Interface	loading the interface by downloading large files (7 Gigabytes)

In Figure 3.3, we plot the CPU utilization behavior during loading the four resources (CPU, memory, disk, and interface) using the stress test. Fig. 3.3(a) shows the CPU Utilization during stressing of the four subsystems: CPU, memory, disk and interface. From 0-800 seconds, the load was in the CPU. Loading the memory was from 800-1200 seconds. However, the load was in the disk from 1200-2000 seconds, while the interface load was from 2000-2900 seconds. It is clear that the CPU utilization is affected by loading each subsystem. Fig. 3.3(b) shows how memory utilization has been affected by loading the four subsystems. Fig. 3.3(c) shows the disk utilization, and Fig. 3.3(d) shows the interface utilization.

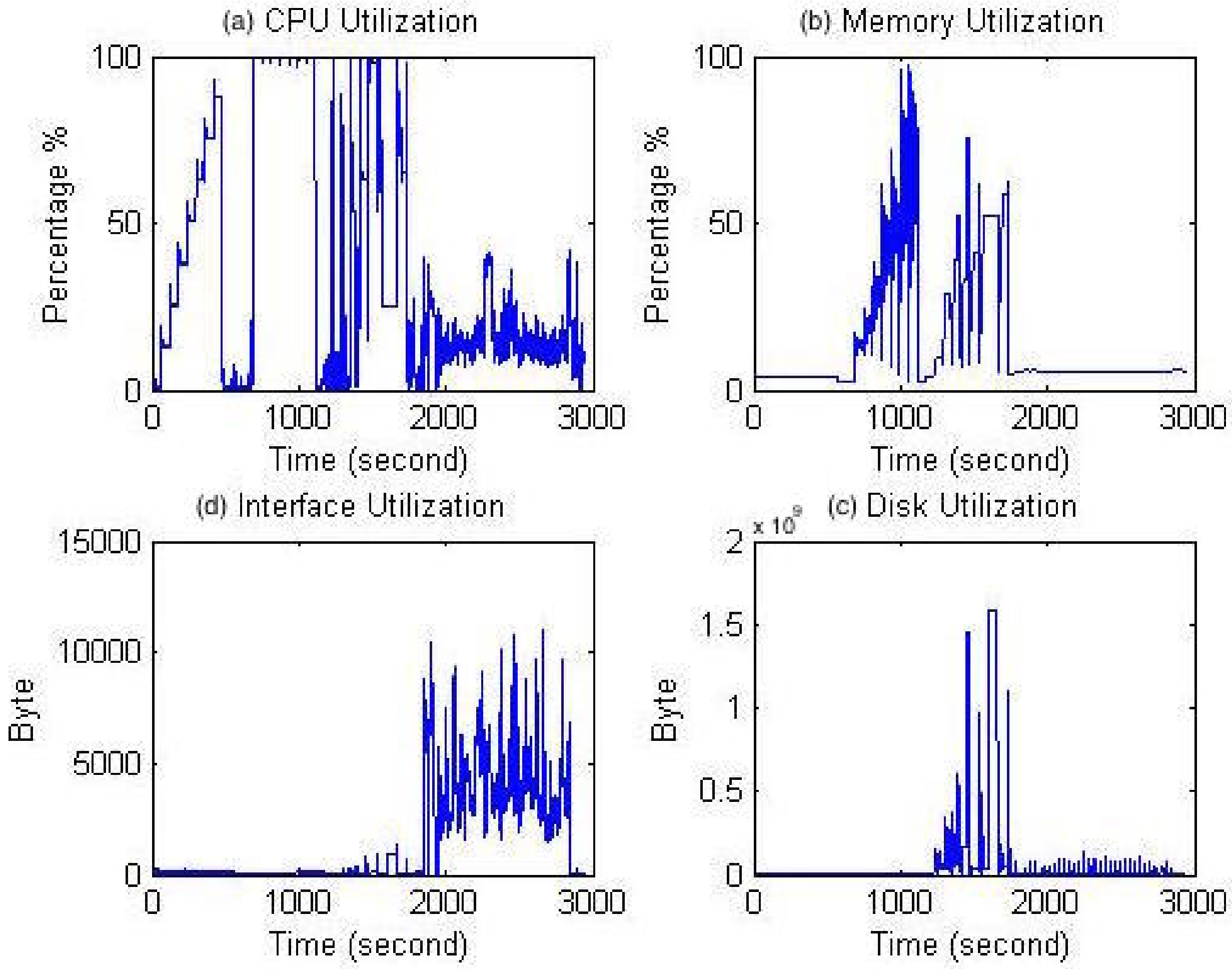

Figure 3.3: Resources Utilization During Applying High Load

3.4 Collecting and Simplifying the Data

To collect the performance information for each subsystem, we implemented *CollectD*. *CollectD* [49] is an open source UNIX daemon that collects resource performance metrics periodically. It has several plugins to collect data from a specific application or service, and/or to write data to a particular storage medium [50]. We used *cpu, memory, disk, interface* and *TED* plugins, and stored the data as CSV (Comma Separated Values) files on the monitoring station. On the monitoring station (disk top computer with Windows OS), we used MATLAB to simplify the data and determine the variables that will used to build the power prediction model. More information about each plugin are given below.

3.4.1 Actual Power

A measurement of the actual power is collected by using the energy detective device (TED 5000). *TED5000* has been used because it is simple to install and its readings are very accurate [51]. The installation instructions can be found on the TED website URL located in References, ”TED Support” [52]. Figure 3.4 shows the metering components that we used. The definitions of all components are as follows:

1. **CTs:** The Current Transformers were installed by pressing on the handles and clipping them over the server’s power line, and connecting the other side of the CTs to the MTU.
2. **MTU:** The Measuring Transmitting Unit is responsible to collect the power data.
3. **Power Cable:** It is a power cable for MTU which connects to the MTU. On the other end, there are three leads the white lead, the black lead and the red lead.
4. **Gateway:** It was plugged into the wall power port (120 VAC outlet).
5. **Ethernet Cable:** It is connected from the gateway to the router that is hosting the LAN.

In our test bench, the CRT will send the power reading to a Gateway tool and from this tool to the router over an Ethernet cable. The machine that has *CollectD* running on it is connected to the same router. In this way, *CollectD* can take the power reading from TED because they are all on the same local area network (LAN). On of the *CollectD* plugins called *TED plugin*. TED plugin collects the power measurements from the energy detective device *TED5000*. The plugin code is as follows:

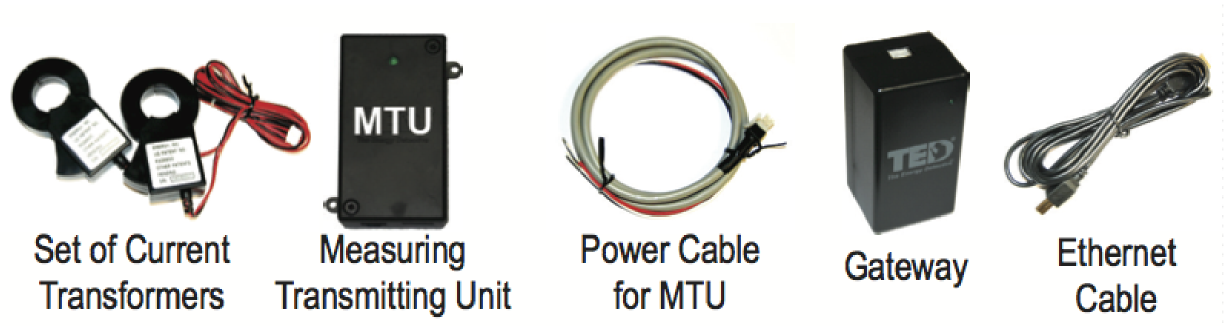

Figure 3.4: TED Components

TED plugin code:

```
<Plugin python>
    Module path "/usr/lib/collectd"
    Import "ted5000"
            <Module ted5000>
                    Host "129.97.10.85"
                    DkipZeroes "true"
                    Verbose "false"
            </Module>
</Plugin>
```

The *Host Ip address* (e.g, 129.97.10.85) is the TED's getaway address that collects the actual power read and is connected to the router.

3.4.2 CPU

Regarding CPU (Central Processing Unit), the measurement was CPU usage as a percentage. This measurement was collected via the *CPU* plugin, which measures the fraction of the time spent by the CPU in several states that described in table 3.2 [53].

However, CollectD gives the CPU utilization core by core. For our system that has 8 cores, CollectD will return the utilities for each core separately as CPU1,CPU2,...CPU8.

Type	Instance	Matrix	Description
CPU	core#	idle	Percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.
		interrupt	Percentage of time spent by the CPU or CPUs to service hardware interrupts.
		nice	Percentage of CPU utilization that occurred while executing at the user level with nice priority.
		softirq	Percentage of time spent by the CPU or CPUs to service software interrupts.
		steal	Percentage of time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor.
		system	Percentage of CPU utilization that occurred while executing at the system level (kernel). Note that this does not include time spent servicing hardware and software interrupts.
		user	Percentage of CPU utilization that occurred while executing at the user level (application).
		wait	Percentage of time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request.

Table 3.2: CPU Collected Performance Data

By applying different loads on the server to observe the CPU states, we found that System and User states have the most performance count. Thus, we calculate the CPU usage as the addition of System and User parameters. We will have $CPU\ Usage = (system + user)$ and $CPU\ Idle = (idle + interrupt + nice + softer + steal + wait)$ for each CPU core. Then, we add all the eight *CPU usage* files together to get one *CPU Utilization* file as follow:

$$CPU\ Utilization = CPU_1\ Usage + CPU_2\ Usage + ... + CPU_c\ Usage$$

where c is the core numbers.

It is important to note that *CollectD* collects statistics measured in *jiffies* (units of scheduling), instead of percentage. On most Linux kernels there are 100 *jiffies* to a second, but depending on both internal and external variables, this might not be always true. However, it is a reasonable approximation.

3.4.3 Memory

Type	Instance	Matrix	Description
Memory		buffered	The amount of memory used as buffers.
		cached	The amount of memory used for caching.
		free	The amount of idle memory.
		used	The amount of memory used.

Table 3.3: Memory Collected Performance Data

For memory resources, we defined the percentage used as the core measurement. This percentage used was calculated using the measurement that was collected via the CollectD *memory* plugin. *Memory* plugin has the four measurement files (Table3.3 [53]): Used (used by applications), Buffered (block devices' caches), Cached (used to park file data), and Free (not being used).

The Linux kernel minimizes the *free* memory by growing the *cache*, but when an application requires extra memory, *cached* and *buffered* memory are released to give extra memory to applications. Therefore, we considered in the calculation of the percentage used only the memory *Used* measurement file.

3.4.4 Disk

For hard-disk, we measured the performance statistics using the CollectD *disk* plugin. This plugin gives different measurement files. *Disk-Merged read/write* is for number of read/write operations. *Disk-Octets read/write* is for bytes read/write from/to disk per second. *Disk-Ops read/write* is for read/write operation from/to disk per second. *Disk-time read/write* is for average time an I/O read/write operation took to complete. See Table 3.4 [53] for more description.

Type	Instance	Matrix	Description
Disk	sda/sda1/sda2/sdb	disk_merged read	The number of read operations, that could be merged into other, already queued operations, i. e. one physical disk access served two or more logical operations.
		disk_merged write	The number of write operations, that could be merged into other, already queued operations, i. e. one physical disk access served two or more logical operations.
		disk_octets read	Bytes read from disk per second.
		disk_octets write	Bytes written to disk per second.
		disk_ops read	Read operation from disk per seconds.
		disk_ops write	Write operation to disk per seconds.
		disk_time read	Average time an I/O- read operation took to complete, equivalent to svctime of vmstat,
		disk_time write	Average time an I/O-write operation took to complete, equivalent to svctime of vmstat.

Table 3.4: Disk Collected Performance Data

However, we took the data from *Disk-Octets read/write* files and combined them together in one file called the *disk activities* file. If the server platform has more than one disk, CollectD will return measurement files for each disk. We add all disk activities' files coming from each disk to have one total *disk activities* file.

3.4.5 Interface

For network interfaces, we used CollectD *interface* plugin, and we chose bytes per second as the unit. The *interface* plugin in *CollectD* collects different measurements as ”transfer” or ”receive”. Table 3.5 [53] has description for each measurements file. It provides the rate of error, rate of bytes, and rate of packets transferred or received. Here we are considering the rate of bytes ”transfer/receive” because it is the actual bytes received/transmitted by the network interface. We combined these data files together to have the *interface activities* file. Moreover, utilization is provided by *CollectD* for each ethernet port in the machine.

Type	Instance	Matrix	Description
Interface	eth0	if_errors rx	Rate of Error in receiving data by network interface.
		if_errors tx	The number of write operations, that could be merged into other, already queued operations, i. e. one physical disk access served two or more logical operations.
		if_octets rx	Rate of Bytes received by network interface.
		if_octets tx	Rate of Bytes transferred by network interface.
		if_packets rx	Rate of packets received by network interface.
		if_packets tx	Rate of packets transferred by network interface.

Table 3.5: Interface Collected Performance Data

3.5 Power Estimation Model

In Figure 3.5, we present a generic measurement procedure that can create instance of information model to estimate the power consumption of software application. Developers can follow this simple procedure to get an idea about the power consumption of their software applications during the development stage. Our measurement procedure involves two main phases:

First phase: The Initialization Phase

This phase needs to be executed only one time for an execution platform, and it includes four steps:

1. Set up the power meter device with the platform for which you need to estimate software application power consumption.
2. Run the workload and the performance counters on the platform which is connected to the power meter.

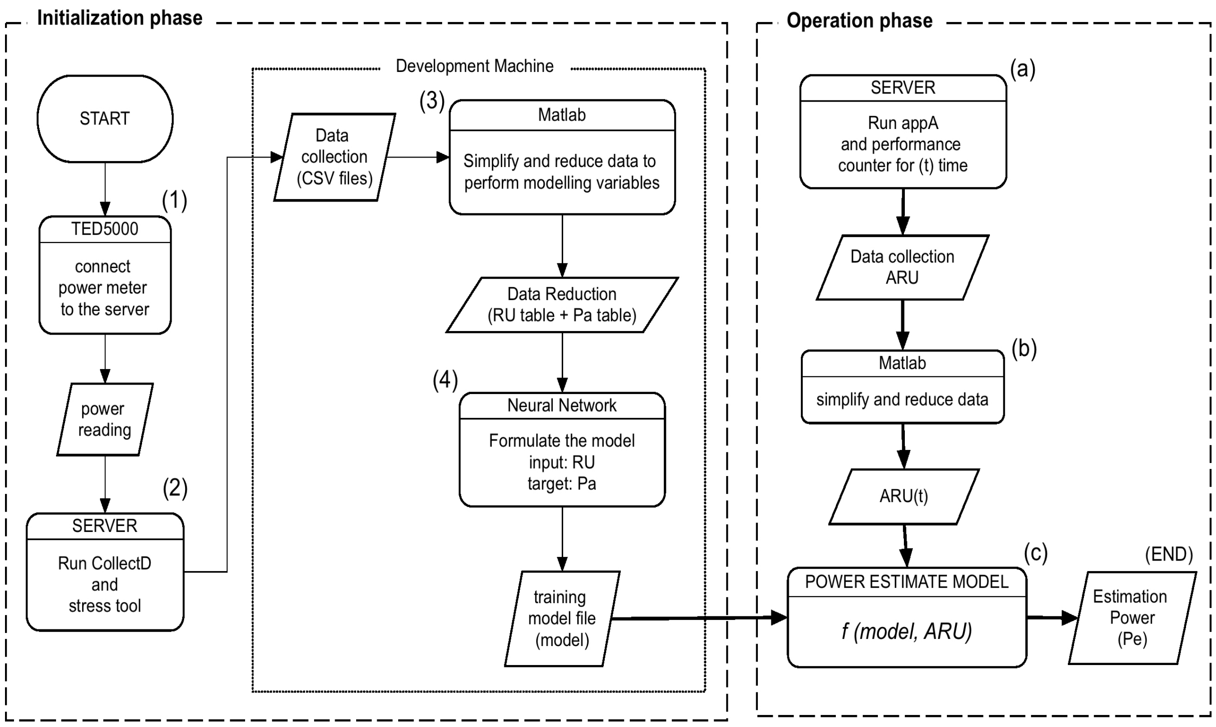
Initialization phase
Development Machine
START
(1)
TED5000
connect power meter to the server
power reading
(2)
SERVER
Run CollectD and stress tool
Data collection (CSV files)
(3)
Matlab
Simplify and reduce data to perform modelling variables
Data Reduction (RU table + Pa table)
(4)
Neural Network
Formulate the model input: RU target: Pa
training model file (model)
Operation phase
(a)
SERVER
Run appA and performance counter for (t) time
Data collection ARU
(b)
Matlab
simplify and reduce data
ARU(t)
(c)
POWER ESTIMATE MODEL
f (model, ARU)
(END)
Estimation Power (Pe)

Figure 3.5: The Initialization and Operation Phases for The Modelling Process.

3. Simplify and reduce the collected performance data to create the modelling variables:

   - Calculate and reduce data to have only four values in resource utilization vectors (*RU*): CPU, Memory, Disk, Interface (Equation 3.1):

   $$RU =< cpu, memory, disk, interface > \tag{3.1}$$

   - Create Resource Utilization matrix RU_t (Equation 3.2) and Actual Power matrix Pa_t (Equation 3.3).

   $$RU_t = [cpu_t, memory_t, disk_t, interface_t] \tag{3.2}$$

   $$Pa_t = [watt_t] \tag{3.3}$$

   - Fix the missing data (NuN) by the average value of the previous cell and the next cell:

   $$missing(RU_t) = \frac{(RU_{t-1}) + (RU_{t+1})}{2} \tag{3.4}$$

4. Formulate the model based on the resource utilization data (RU_t) and the actual power data (Pa_t) to predict the power consumption (Equation 3.5 and 3.6).

$$model = f(input, target) \tag{3.5}$$

$$model = f(RU_t, Pa_t) \tag{3.6}$$

Second phase: The Operation Phase

In this phase, we predict the power consumption of any application running on the same platform that is used in the first phase. This phase is executed once for each case of an application, and it includes three steps:

(a) Run the software application (*app A*) and *CollectD* simultaneously with the intended scenario for *t* seconds.

(b) Simplify and reduce application resource utilization data *ARU* during the same *t* time.

$$ARU_t = [< CPU_t, Mem_t, Disk_t, Inter_t >] \tag{3.7}$$

(c) Input *ARU* and training model file (*model*) to power estimate model function (f_{pem}) to get the estimation power *Pe*.

$$Pe_t = f_{pem}(model, ARU_t) \tag{3.8}$$

Thus, developers need to create the application resource utilization table (ARU) for the software application that is running on the server by running the application with the performance counter. Then, they will use the training model file (*model*) and ARU table as an input to estimate the application power consumption. Thus, the operation phase is ended by giving the estimation power (*Pe*) data of the application running on the sever.

After a subsystems power consumption model has been trained, it does not require retraining when applied to the same server. So, the developer will train the model once for a platform and then apply their applications to predict their power consumption. However, if applying the model to another server platform, retraining is required.

To validate our measurement and modelling procedure, three different platforms, four different scenarios and two different modeling techniques have been used. An explanation and discussion about the validation will be in Cahpter 4.

Chapter 4

Accurate Power Consumption Model

4.1 Introduction

The ever growing software engineering challenge is to increase energy efficiency of the software systems [54]. The energy demands are ever increasing because of the increasing consumption of energy in data centres, networks and mobile phones. In the current scenario, mobile phones are used by almost all individuals and hence the energy demands are quite high [55]. Thus it is very important to find an energy saving software system [56]. The hardware which is currently available in the market has high capabilities and hence they are inevitably high on consumption of energy as well. The providers of operating systems are using the energy saving mode in order to make the systems consume lesser amount of energy [57].

However the truth is that neither the operating system optimization nor the hardware optimization is good enough to meet the ever rising energy demands. In the recent investigations it has been found that it is the application software which increases the consumption of energy at the OS and the hardware level [58].

The most contradicting optimization goals are the performance of the software system and the energy efficiency [59]. This can be explained through an example of optimizing the response time of an enterprising application by duplicating the number of replicas of the system. When more and more replicas are introduced, it automatically increases the energy consumption. However when the component resource demand decreases, it automatically decreases the consumption of energy. Thus it can be pointed out that the performance metric and energy consumption work on the same parameters of resource demand and the capabilities of the hardware. This takes us to evaluate performance counters and understand the power demand and consumption patterns of the software applications.

4.2 Neural Network Model

The human brain processes are different than the working of a digital computer and this has given rise to the Neural Networks (NN), or artificial neural networks [60]. Human brain has very complex way to process information, and neural network is a simplified model of this complex process. It simulates interconnected processing units in a way can be said that resemble the neurons. Neural network models (NNMs) are multivariate statistical models that used to relate predictor variables $x_1, ..., x_p$ to response variables $y_1, ..., y_q$.

This particular model is divided into multiple layers where each layer either consists of some original or constructed variables. Most common structure consists of three layers namely the input layer, which has the predictor variables, the hidden layer, which is comprised of constructed variables, and the outer layer, which comprised of responses variables.

NNM is a flexible model that contains number of parameters and resembles a neural network with nearly universal approximation property. In order to estimate the parameters, the minimum of the overall residual sum of squares is taken over all the responses and all the observations. This is a nonlinear least-squares problem.

Based on that, we used this modeling technique to model the power consumption of system and validate its accuracy in predicting the power consumption of application running on the server. We used the Neural Network Toolbox software in MATLAB for training the data. A Neural Network Fitting tool (*nftool*) is used to the data relating performance metrics to the power variation for the system. Typically, neural network is trained so that a specific input leads to a particular target output. By using the four subsystem metrics as input variables (*CPU, memory, disk, interface network*) and power meter readings as target variable, we performed neural network model (NNM) that predicts the power consumption of application running on data centre.

4.2.1 Modeling Process

To formulate the power consumption model, we used *Neural Network Fitting* tool. Typically, neural network is trained so that a specific input leads to a particular target output. Here, we used the *resource utilization matrices* as an input and the *actual power watts* as a target output. Both resource utilization metrics and actual power watts are measured and collected per second during the workload.

The resource utilization measurements are compiled into one matrix *RU* with one column for each metric (time, cpu, memory, disk, interface) and a row for each time sample (equation 4.1). The actual power measurements are saved in another matrix P_a with one column for each metric (time, watt) and a row for each time sample (equation 4.2).

$$RU = [time, cpu, memory, disk, interface] \tag{4.1}$$

$$P_a = [time, watt] \tag{4.2}$$

While we are collecting data from different measurements per second, there are probabilities for missing some data. To observe any missing data, we matched and unified the time of the resource utilizations matrix with the time of the actual power matrix to have one matrix *M* as follows:

$$M = [time_t, power_t, CPU_t, mem_t, disk_t, interf_t] \tag{4.3}$$

If there is a cell missing data, we will fill it by the average value of the previous cell and the next cell. For example, if we have a missing value (NuN) in CPU column for *t* second (cpu_t), we fill the NuN with the average of the previous value (cpu_{t-1}) and the next value (cpu_{t+1}) as follows:

$$(cpu_t) = \frac{(cpu_{t-1}) + (cpu_{t+1})}{2} \tag{4.4}$$

After fixing the missing data, we used RU matrix as the input variable and Pa matrix as the target variable based on the following equation:

$$net = f(input, target) \tag{4.5}$$

where *net* is the model network file, and f is the neural network fitting function.

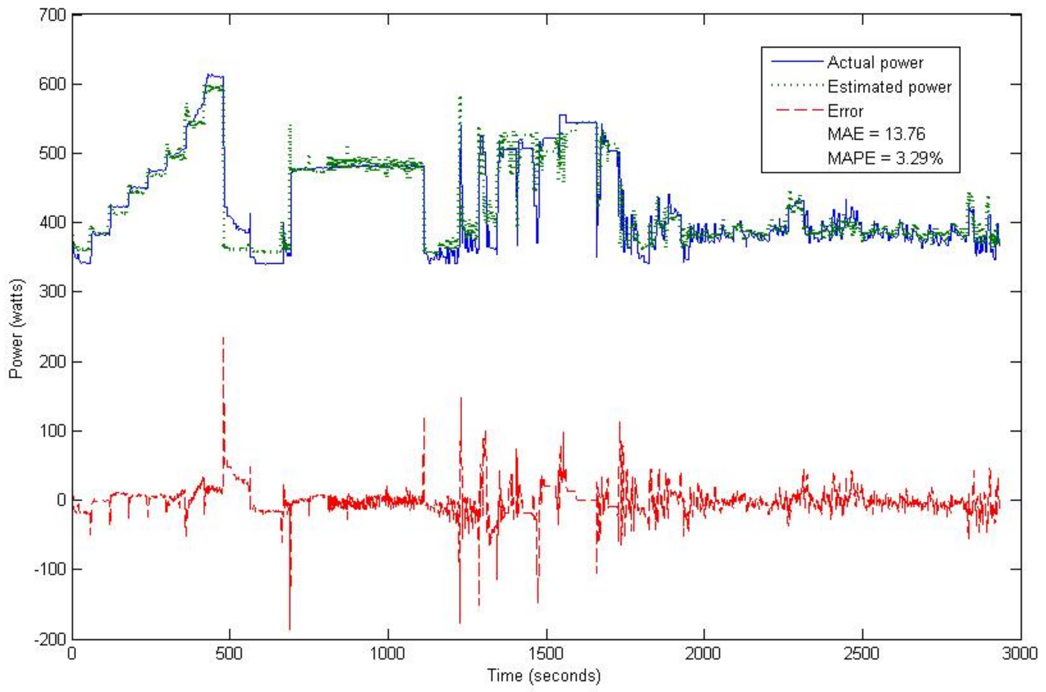

Figure 4.1: Estimated power consumption (model power) and measured power consumption (actual power) during the loading of each subsystem to the maximum load.

After we got the training model file (*net*), we can use it to predict the power consumption of a software application running on the same server platform. The performance of the neural network prediction model is shown in Figure 4.1. The figure presents the actual power, model power (estimated power) and the average error. It is clear that the estimated power is very close to the actual power with Mean Absolute Error (MAE) equal to 13.76 watts.

4.2.2 Validation

For validation, we used three real servers, server 1, server 2 and server 3, as described in Tables (4.1, 4.2 and 4.3), respectively.

Table 4.1: Description of server 1

Parameter	Real Server (Dell PowerEdge 2950)
Processor	7x Intel Xeon, 3 GHz, 8 cores
Hard Disk	1.7 Tera Bytes SAS
Main Memory	32 GB DIMM
Operating System	Linux (Ubuntu 13.10)
Observable Subsystems	CPU, Memory, Disk and Interface

Table 4.2: Description of server 2

Parameter	Real Server (HP ProLiant DL385G5)
Processor	Quad-Core AMD Operon^{TM}, 2.3 GHz, 4 core
Hard Disk	55.2 Giga Bytes
Main Memory	15.7 GB DIMM
Operating System	Linux (Ubuntu 14.04 LTS)
Observable Subsystems	CPU, Memory, Disk and Interface

We applied four different workload scenarios in the server to validate the model. In each scenario, we observed the mean absolute error (*MAE*) of the model. Specifically, we evaluated the Mean Absolute Error of power estimation (MAE_p), the Minimum Error (*MinError*), and the Maximum Error (*MaxError*) which calculated as Equation (4.6, 4.7).

$$MaxError = \frac{MAE_p}{MaxPower} * 100 \tag{4.6}$$

Table 4.3: Description of server 3

Parameter	Real Server (Dell PowerEdge 2950)
Processor	Intel Xeon CPU 5160 @ 3.00 GHz x 4
Hard Disk	277.8 Giga Bytes
Main Memory	7.8GiB
Interface	ethernet cable, 100 Mb/s
Operating System	Linux (Ubuntu 14.04 LTS)

$$MinError = \frac{MAE_p}{MinPower} * 100 \tag{4.7}$$

The *MaxPower* is the highest point the power reached during the workload, and *MinPower* is the lowest point. Moreover, the *Error* was calculated as the difference between the actual power and the model output (the estimation power) as in Equation (4.8).

$$Error = ActualPower - ModelPower \tag{4.8}$$

Also, we calculated the Mean Absolute Error of energy estimation (MAE_e) as Equation 4.9.

$$MAE_e = \frac{Error}{ActualPower} * 100 \tag{4.9}$$

The four scenarios that we applied to validate our modeling procedure are described bellow:

- *The First Scenario is Downloading:* In this scenario we tried to load the system with a real load. That has been done by downloading large fils from the internet. We used Firefox browser to access a web-page that has many movies in high quality. We downloaded four movies from the same web-page at the same time, that has in total around 7 Gigabytes. Then we applied our models to estimate the power consumption of this experiment after collecting the data during the downloading.

- *The Second Scenario is Gaming:* For this scenario we installed a video game in the server and collected the system performance during plying the game. We played the game for about 30 minute.

- *The Third Scenario is Local Movie:* In this scenario we played a high quality movie file that has been already downloaded to the server. We collected the performance matrices during playing the movie as a full screen view for 30 minutes.

- *The Fourth Scenario is Youtube:* A high definition YouTube video is used for this scenario. During playing the YouTube video for 20 minutes, we collected the performance data of the system.

We used Neural Network modelling technique to build the power estimation model. Then, we applied this model to predict the power consumption for the four scenarios.

Figure 4.2 shows the actual power and the estimation power during the downloading scenario in server 1. The MAE_p was 56.45 watts with minimum rate 13.97% and maximum rate 11.38%, and the MAE_e was 12.64%.

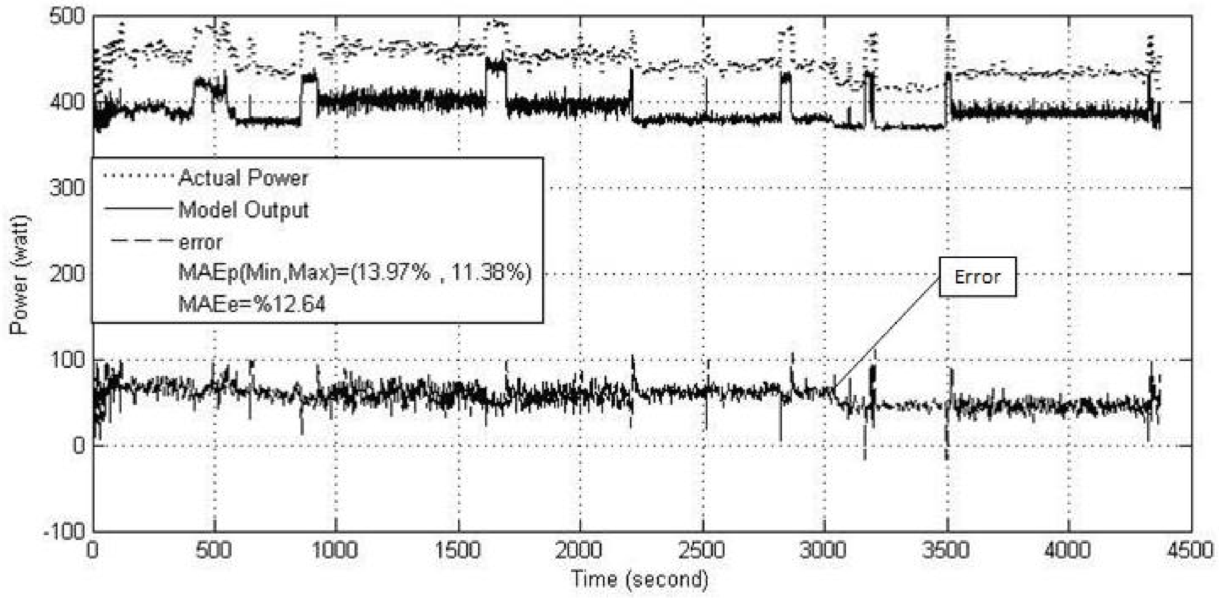

Figure 4.2: Estimated power consumption (model power) and measured power consumption (actual power) during downloading 7GB movie files in server1.

In each scenario, we observed the mean absolute error of power estimation (MAEp) and the Mean Absolute Error of energy estimation (MAEe). The results are presented in Table (4.4), Table (4.8) and Table (4.9 present the estimated errors for those various scenarios in server1, server 2 and server 3 respectively.

Table 4.4: MAE_p and MAE_e for Various workloads in server 1

Downloading	MAEp (MinError, MaxError)	(13.97% , 11.38%)
	MAEe	12.64%
Gaming	MAEp (MinError, MaxError)	(17.94% , 14.82%)
	MAEe	15.30%
Local Movie	MAEp (MinError, MaxError)	(14.77% , 11.21%)
	MAEe	11.8%
Youtube	MAEp (MinError, MaxError)	(15.59% , 12.52%)
	MAEe	13.15%

Table 4.5: MAE_p and MAE_e for Various workloads in server 2

Downloading	MAEp (MinError, MaxError)	(3.66% , 2.78%)
	MAEe	1.52%
Gaming	MAEp (MinError, MaxError)	(5.07% , 4.00%)
	MAEe	4.17%
Local Movie	MAEp (MinError, MaxError)	(3.99% , 3.14%)
	MAEe	3.02%
Youtube	MAEp (MinError, MaxError)	(5.69% , 4.33%)
	MAEe	4.13%

Table 4.6: MAE_p and MAE_e for Various workloads in server 3

Downloading	MAEp (MinError, MaxError)	(2.84% , 3.61%)
	MAEe	0.09%
Gaming	MAEp (MinError, MaxError)	(0.71% , 0.87%)
	MAEe	0.34%
Local Movie	MAEp (MinError, MaxError)	(1.98% , 2.54%)
	MAEe	0.85%
Youtube	MAEp (MinError, MaxError)	(2.47% , 3.07%)
	MAEe	2.10%

4.3 Polynomial Model

We use polynomial regression to create model that helps to estimate and predict the power consumption of application over the range of the platform's subsystem utilization. Regression analysis can be defined as a statistical technique which is used for investigating and modeling the relationship among the variables as an *rth* degree polynomial.

In this model, we have n observations $y = y_1, ..., y_n$ called the response or dependent variables and $x_i = x_{i,1}, ..., x_{i,p}$ for i=1...n that are predictor or regressor variables. The simplest linear regression is of the form $y = c_0 + c_1x + e$. In this formula *c* represents the coefficients used in describing the response as a linear function of predictors plus a random error *e*. The polynomial regression of order *k* can be written as:

$$y = C_0 + C_1x + C_2x^2 + C_3x^3 + C_4x^4 + ... + C_nx^k \tag{4.10}$$

Multiple predictor variables are used in the input data set which cause the response y to be related to *p* which is the regressor/predictor variable. The model then becomes:

$$y = c_0 + c_1x_1 + c_2x_2 + ... + c_px_p + e \tag{4.11}$$

Here y and x are vectors of n numbers (observations). The fitting of a regression model to the observations is done by solving the (*p+1 c*) coefficients.

4.3.1 Multivariable Polynomial Model

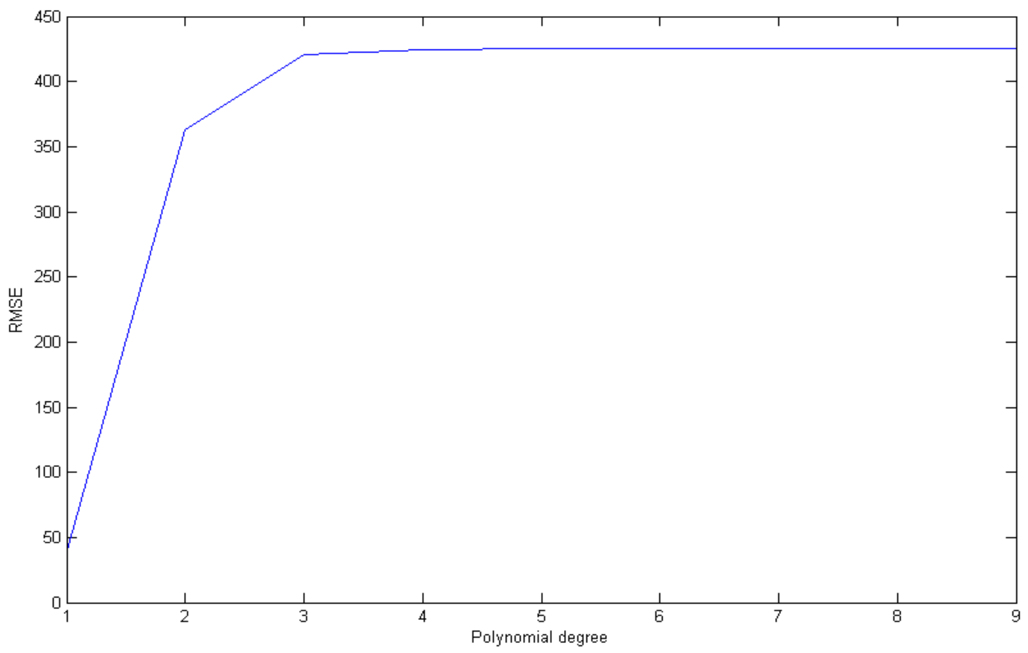

Figure 4.3: RMSE for different polynomial-degree models for server 1

In our case, we have four predictor variables and one dependent variable. The predictor variables are the subsystems' utilization (*CPU, memory, Disk, and Interface*) and the dependent variable is the actual power. Thus, we use multiple polynomial regression technique to calculate the polynomial equation and find the best fit for our data. The multiple polynomial regression can be written as:

$$y = C_{m1}x_1^m + ... + C_{mn}x_n^m + ... + C_{11}x_1 + ... + C_{1n}x_n + C_0 \tag{4.12}$$

By using this equation (4.12), we can write the multiple polynomial regression equation for any degrees. For instance, the second degree of multiple polynomial regression will be as:

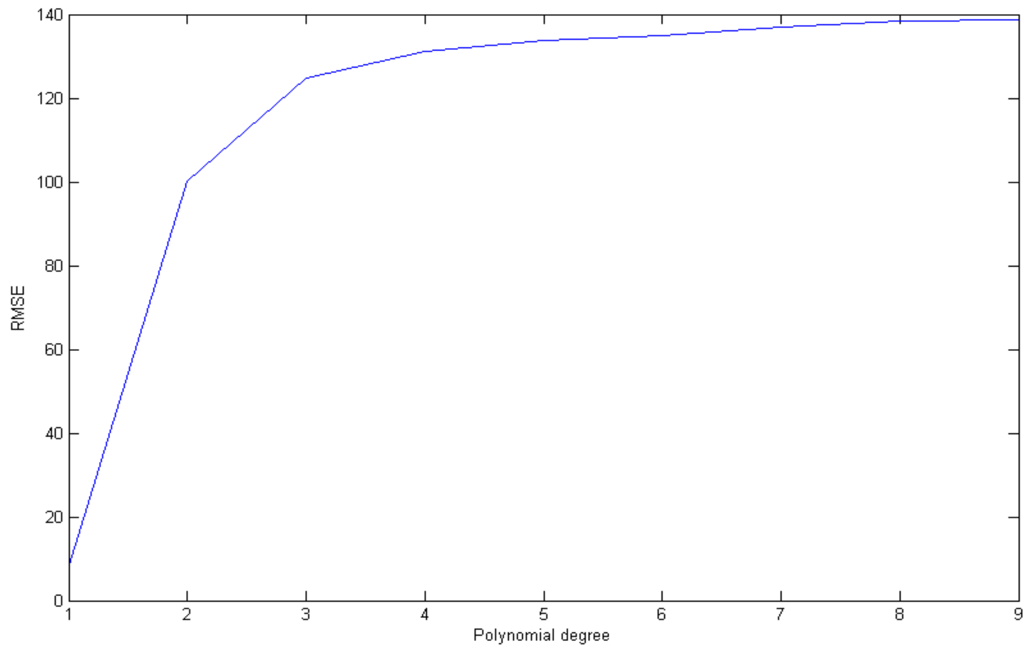

Figure 4.4: RMSE for different polynomial-degree models for server 2

$$\begin{aligned} y = c_0 + c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_1x_2 + c_6x_1x_3 + c_7x_1x_4 \\ + c_8x_2x_3 + c_9x_2x_4 + c_{10}x_3x_4 + c_{11}{x_1}^2 + c_{12}{x_2}^2 + c_{13}{x_3}^2 + c_{14}{x_4}^2 \end{aligned} \tag{4.13}$$

The following regression is the third degree of multiple polynomial regression:

$$\begin{aligned} y = c_0 + c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_1x_2 + c_6x_1x_3 + c_7x_1x_4 + c_8x_2x_3 + c_9x_2x_4 + c_{10}x_3x_4 \\ + c_{11}x_1x_2x_3 + c_{12}x_1x_2x_4 + c_{13}x_1x_3x_4 + c_{14}x_2x_3x_4 + c_{15}{x_1}^2 + c_{16}{x_2}^2 + c_{17}{x_3}^2 + c_{18}{x_4}^2 \\ + c_{19}{x_1}^2x_2 + c_{20}{x_1}^2x_3 + c_{21}{x_1}^2x_4 + c_{22}{x_2}^2x_1 + c_{23}{x_2}^2x_3 + c_{24}{x_2}^2x_4 \\ + c_{25}{x_3}^2x_1 + c_{26}{x_3}^2x_2 + c_{27}{x_3}^2x_4 + c_{28}{x_4}^2x_1 + c_{29}{x_4}^2x_2 + c_{30}{x_4}^2x_3 \\ + c_{31}{x_1}^3 + c_{32}{x_2}^3 + c_{33}{x_3}^3 + c_{34}{x_4}^3 \end{aligned} \tag{4.14}$$

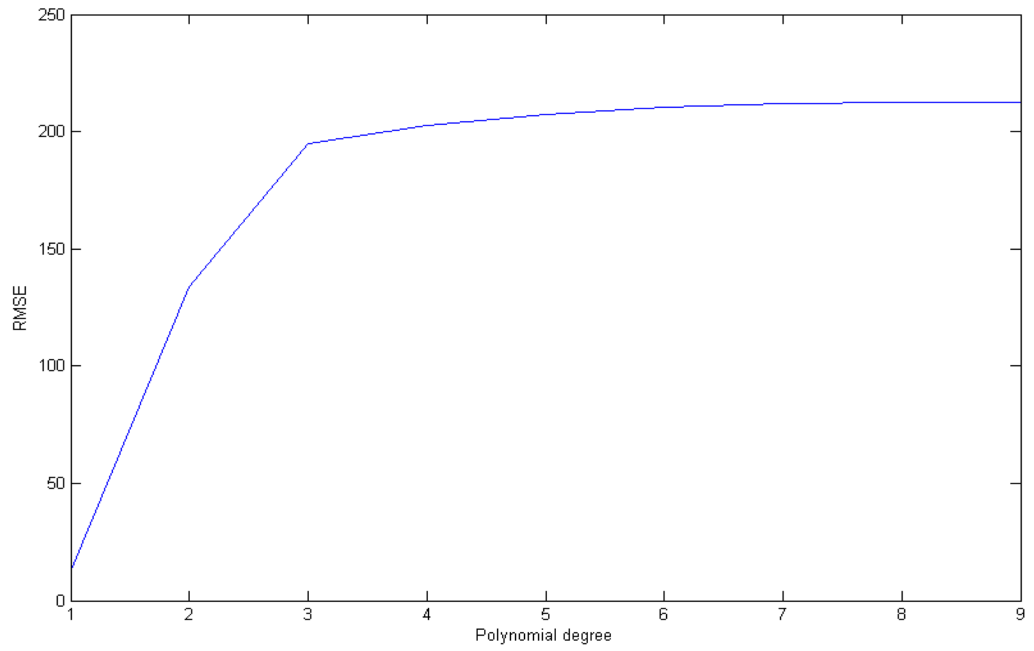

Figure 4.5: RMSE for different polynomial-degree models for server 3

We tried different degrees of of multiple polynomial model, and we observe the root main square error (RMSE) for each model. The following figures (4.3, 4.4 and 4.5) show the RMSE for each polynomial model degree for server 1, server 2 and server 3, respectively.

After we observed all different degrees models, we found that the first degree polynomial model is the model whose RMSE is the minimum. Thus, our model equation can be as follow:

$$Y = C_0 + C_1 X_{cpu} + C_2 X_{mem} + C_3 X_{disk} + C_4 X_{inter} \tag{4.15}$$

where C_n are the coefficients that we can use to predict the power of application.

Then power profile can be written as:

$$EnergyProfile = (C_0 \quad C_1 \quad C_2 \quad C_3 \quad C_4) \tag{4.16}$$

To formulate the power consumption model, we followed the modeling procedure steps. Both resource utilization metrics and actual power watts are measured and collected per second during the workload. We used the *resource utilization matrices* (UR) as predictor variables and the *actual power watts* (P_a) as a dependent variable based on the following equation:

$$poly = f(UR, P_a) \tag{4.17}$$

where *poly* is the model file, and *f* is the polynomial fitting function.

After we got the training model file (*poly*), we can use it to predict the power consumption of a software application running on the same server platform.

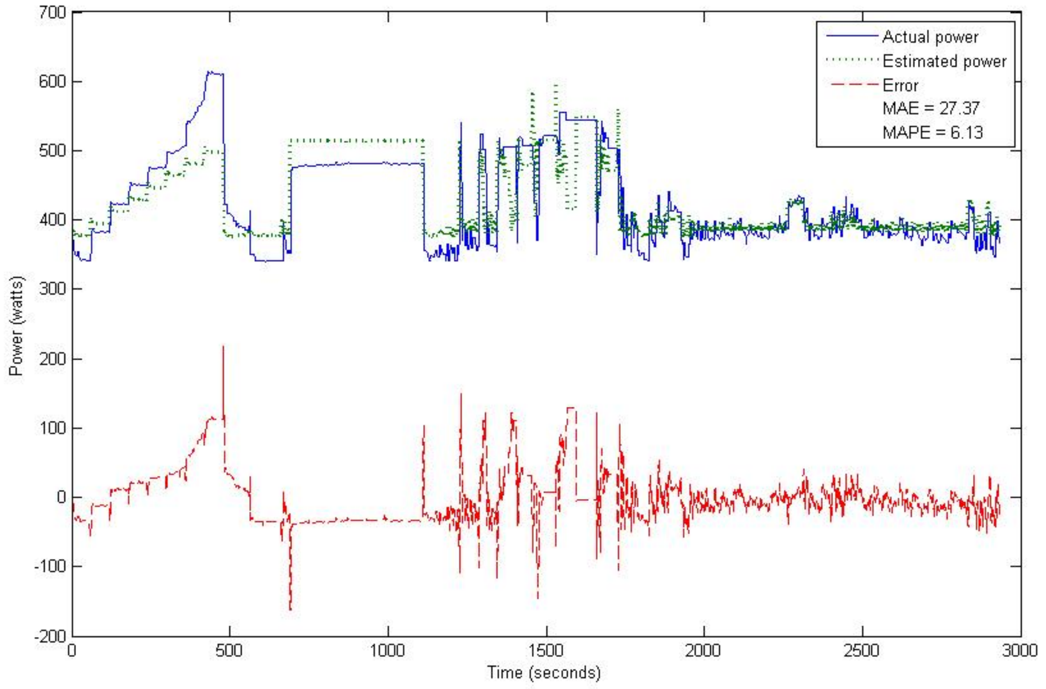

Figure 4.6: The performance of the polynomial prediction model for server 1.

The performance of the polynomial prediction model for server 1 is shown in Figure 4.6. The figure presents the actual power, the estimated power (model power) and the average error. It is clear that the estimated power is very close to the actual power with Mean Absolute Error (MAE) equal to 27.37 watts. Figure 4.7 and 4.8 present the performance of the polynomial prediction model for server 2 and server 3, respectively.

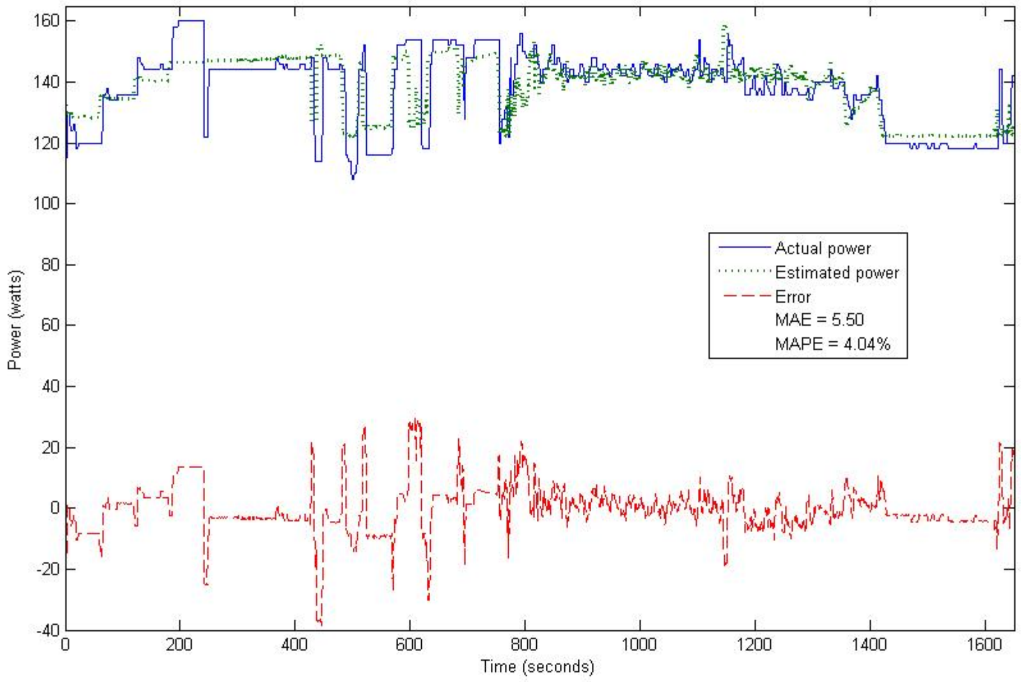

Figure 4.7: The performance of the polynomial prediction model for server 2.

4.3.2 Validation

We used the same three servers that we used before to validate the Neural Network model, server 1, server 2 and server 3, as described in Table 4.1, Table 4.2 and Table 4.3, respectively. Also, we used the same four scenarios that described in Section (4.2.2). Downloading (first scenario) is stressing the most the interface. Gaming (second scenario) is stressing the memory and the CPU for processing. Also, playing a local movie (third scenario) is loading the most the disk and the memory. Finally, playing a Youtube video (fourth scenario) is loading the most the interface and the memory.

After we created the linear polynomial models (LPM) for each server, we predicted the power consumption for the four scenarios in each server. Then, we calculated MAEp and MAEe which are shown on Table (4.7, 4.8 and 4.9).

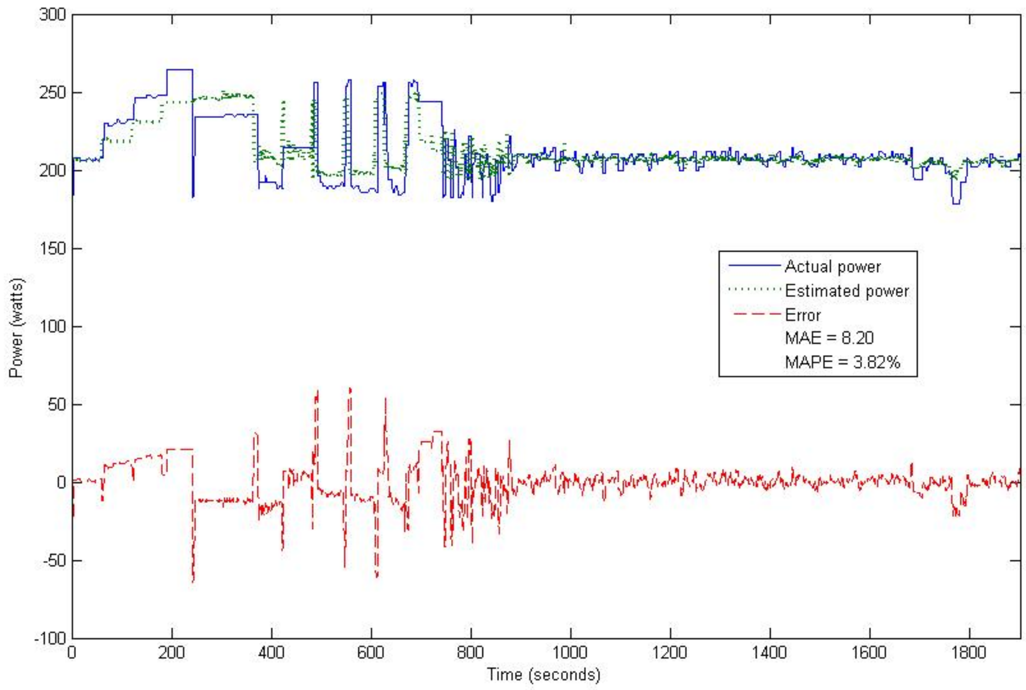

Figure 4.8: The performance of the polynomial prediction model for server 3.

4.4 The Accurate Model

After we created the neural network model (NNM) and linear polynomial models (LPM) of heavy workload for the three servers, we predicted the power consumption for the four scenarios in each server. Then, we calculated MAEp and MAEe. Tables (4.10, 4.11 and 4.12) compare the errors of the two different prediction mechanisms: NNM and LPM in each server. Charts (4.9, 4.10 and 4.11) present the MAE in percentage for NNm and LPM in each server.

For neural network model, we found that in Server 1 the MAEe of NNM was between 11.80% and 15.30%. However, the MAEe in Server 2 was between 1.51% and 4.17%, and in Server 3 the MAEe was between -1.63% and 3.25%. For polynomial model, we found that MARe in server 1 is between 10% to 15%. In server 2, MARe is between 1% to 4% while it is between -4% to 1% in server 3. We noticed that when the estimated power is higher than the actual power, the error value will give a negative number. After comparing the errors for both models, we found that

Table 4.7: MAE_p and MAE_e for Various workload scenarios in server 1

Downloading	MAEp (MinError, MaxError)	(11.72% , 9.55%)
	MAEe	10.60%
Gaming	MAEp (MinError, MaxError)	(18.19% , 15.03%)
	MAEe	15.51%
Local Movie	MAEp (MinError, MaxError)	(17.96% , 13.63%)
	MAEe	14.35%
Youtube	MAEp (MinError, MaxError)	(17.52% , 14.04%)
	MAEe	14.78%

Table 4.8: MAE_p and MAE_e for Various workload scenarios in server 2

Downloading	MAEp (MinError, MaxError)	(3.56% , 2.70%)
	MAEe	1.38%
Gaming	MAEp (MinError, MaxError)	(4.74% , 3.74%)
	MAEe	3.90%
Local Movie	MAEp (MinError, MaxError)	(4.74% , 3.73%)
	MAEe	3.78%
Youtube	MAEp (MinError, MaxError)	(5.15% , 3.91%)
	MAEe	3.67%

NNM gives lesser error than the LPM on the three servers with 3% difference. We also noticed that the old system server, such as server 2 and server 3, give better result in estimation power consumption with average MAE between 1% to 4%. While in the new system server, such as server 1, the average MAE is between 10% to 15%.

Table 4.9: MAE_p and MAE_e for Various workload scenarios in server 3

Downloading	MAEp (MinError, MaxError)	(4.38% , 3.45%)
	MAEe	-3.29%
Gaming	MAEp (MinError, MaxError)	(4.01% , 3.28%)
	MAEe	-3.61%
Local Movie	MAEp (MinError, MaxError)	(2.46% , 1.88%)
	MAEe	0.24%
Youtube	MAEp (MinError, MaxError)	(2.65% , 2.13%)
	MAEe	-2.12%

Table 4.10: MAEp and MAEe errors of models in Server 1.

		Neural Network	Polynomial
Download	MAEp(Min,Max)	13.97%,11.38%	11.72%, 9.55%
	MAEe	12.64%	10.60%
Gaming	MAEp(Min,Max)	17.94%,14.82%	18.19%, 15.03%
	MAEe	15.30%	15.51%
Local movie	MAEp(Min,Max)	14.77%,11.21%	17.96%, 13.63%
	MAEe	11.80%	14.35%
Youtube	MAEp(Min,Max)	15.59%,12.52%	17.52%, 14.04%
	MAEe	13.15%	14.78%

4.5 Making models available to the other developers

After the power estimation model for specific server configuration has been prepared by model developer, the model can be used by application developers to estimate the power consumption of their application software. The application developer need to decide on which server the application software will be run. Then, after the model has been prepared by the model developers, developers can use it to estimate the energy cost of the application software instead of wasting time on direct measurement.

Table 4.11: MAEp and MAEe errors of models in Server 2.

		Neural Network	Polynomial
Download	MAEp(Min,Max) MAEe	3.66%,2.78% 1.51%	3.56%, 2.70% 1.38%
Gaming	MAEp(Min,Max) MAEe	5.06%,3.99% 4.17%	4.74%, 3.74% 3.90%
Local movie	MAEp(Min,Max) MAEe	3.98%,3.13% 3.01%	4.74%, 3.73% 3.78%
Youtube	MAEp(Min,Max) MAEe	5.69%,4.33% 4.13%	5.15%, 3.91% 3.67%

Table 4.12: MAEp and MAEe errors of models in Server 3.

		Neural Network	Polynomial
Download	MAEp(Min,Max) MAEe	2.84%,3.61% 0.09%	4.38%, 3.45% -3.29%
Gaming	MAEp(Min,Max) MAEe	0.71%,0.87% 0.34%	4.01%, 3.28% -3.61%
Local movie	MAEp(Min,Max) MAEe	1.98%,2.54% 0.85%	2.46%, 1.88% 0.24%
Youtube	MAEp(Min,Max) MAEe	2.47%,3.07% 2.10%	2.65%, 2.13% -2.12%

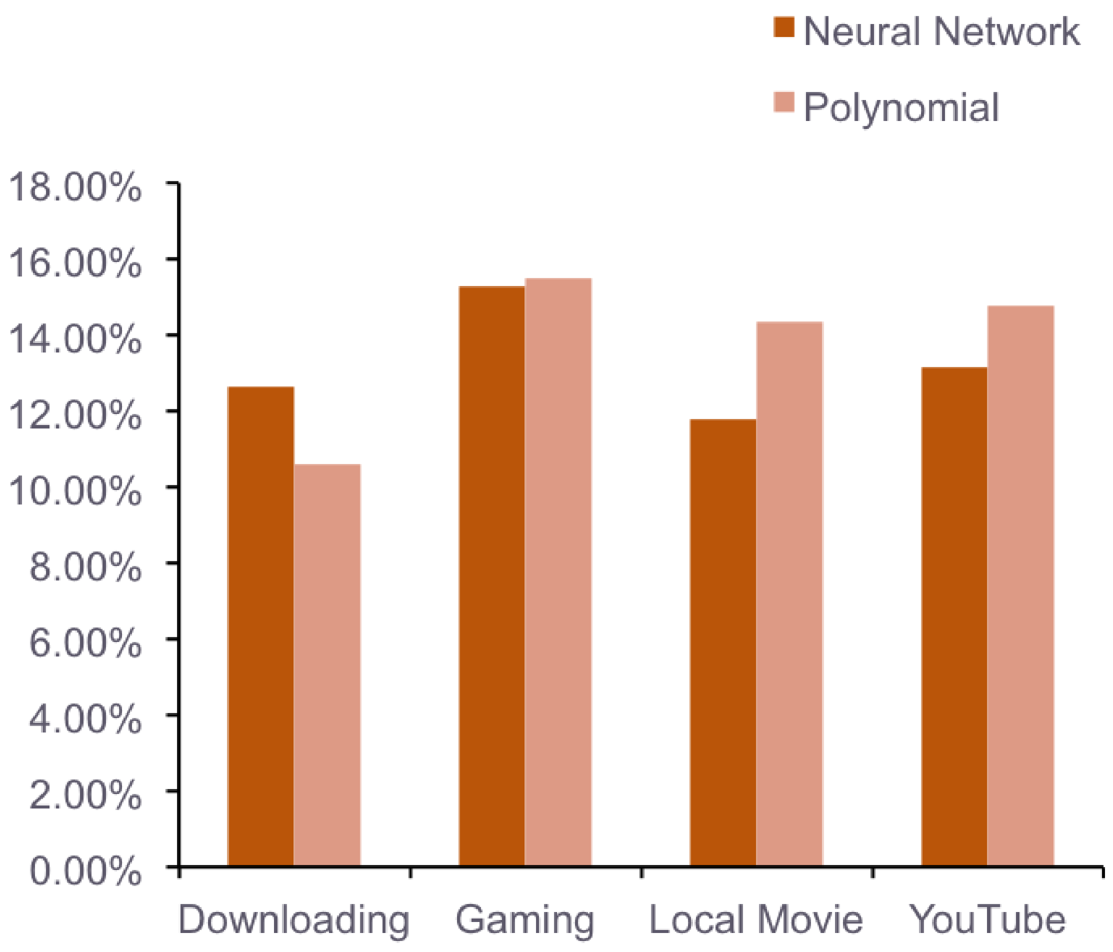

Figure 4.9: MAE of NNM and LPM for each scenario in server 1.

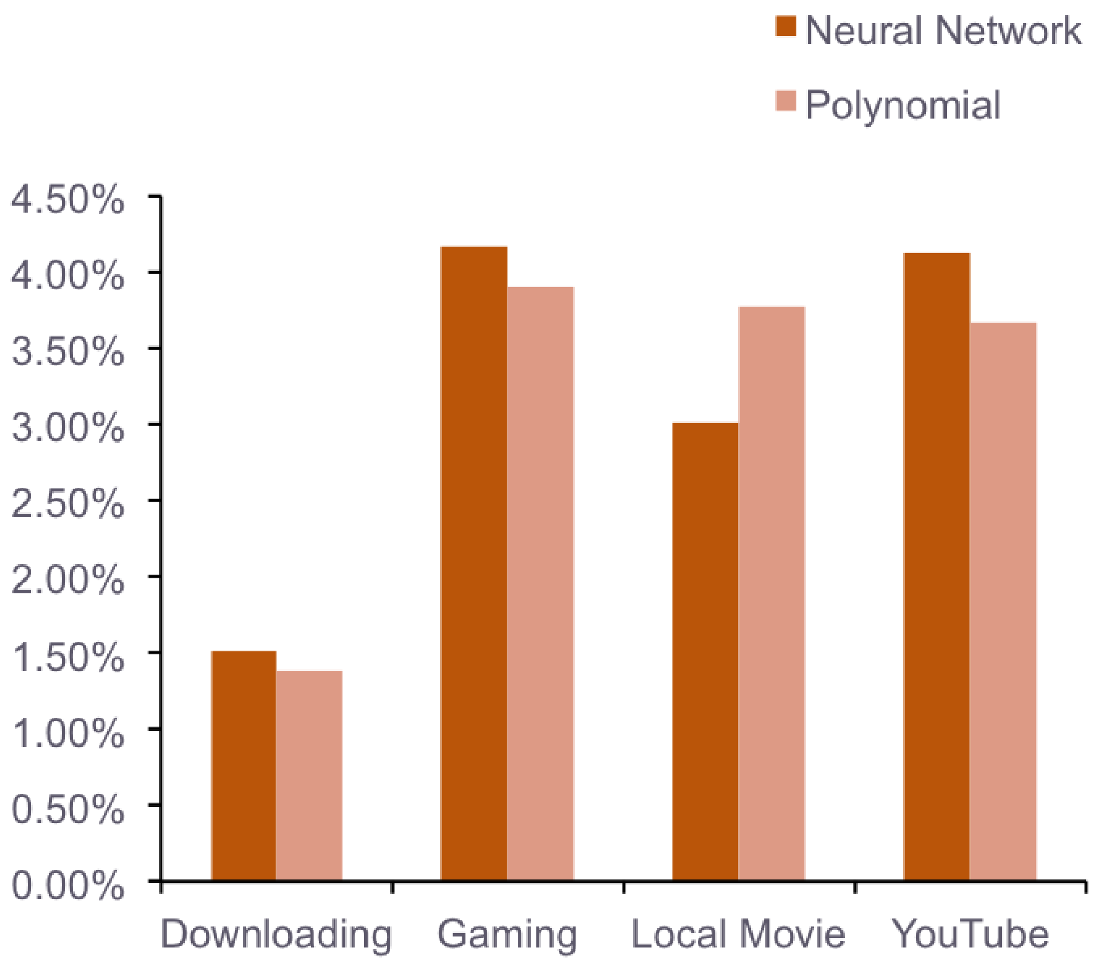

Figure 4.10: MAE of NNM and LPM for each scenario in server 2.

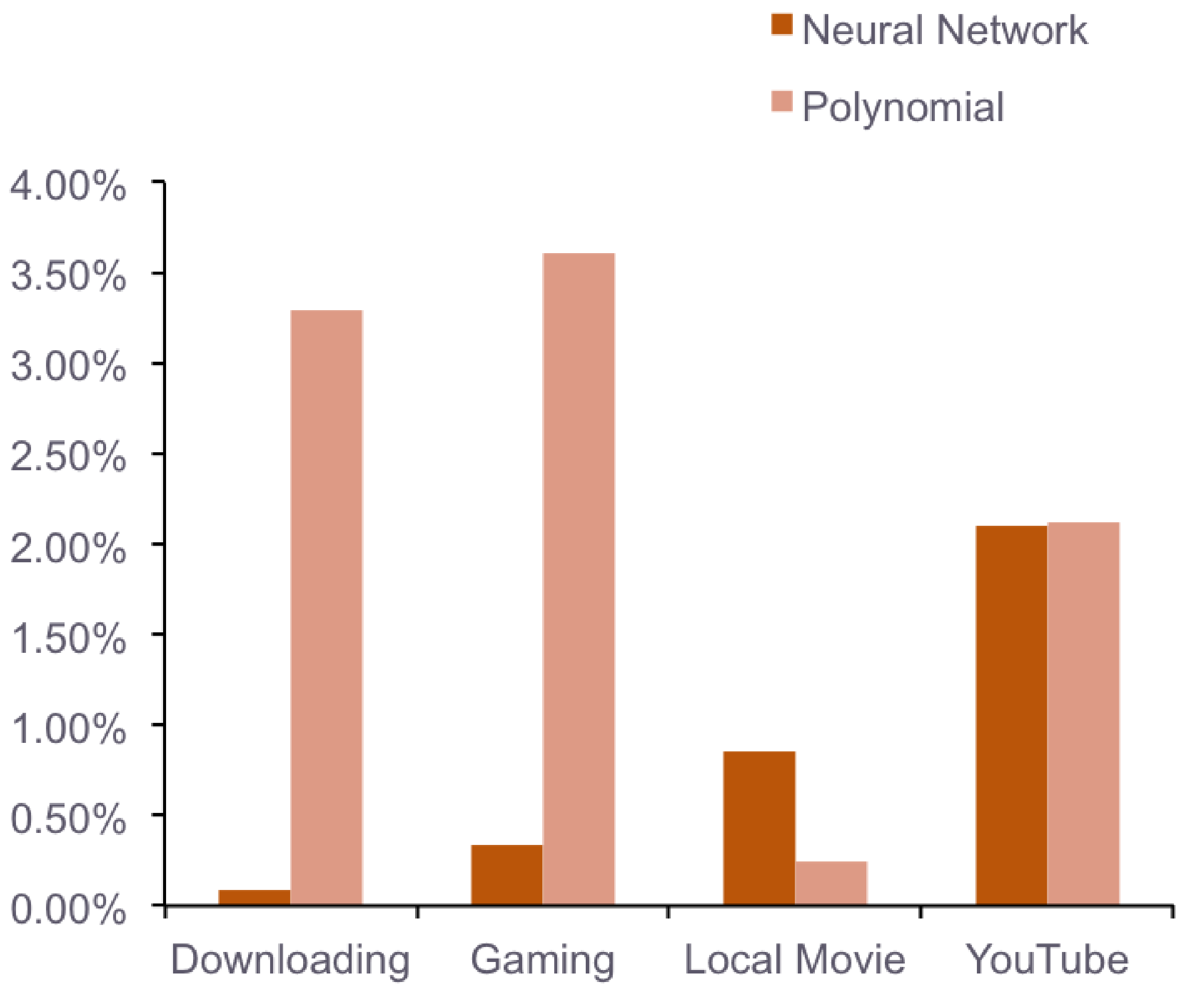

Figure 4.11: MAE of NNM and LPM for each scenario in server 3.

Chapter 5

Conclusion

Given the significant interest in the dynamic use of processes to count power consumption, and the high correlation of subsystems involved in this process, including CPU utilization, memory access and I/O with different power commands, process counts can enable developers to cost effectively measure by real-time proxy estimate of power usage and performance. In Chapter 2, we provided a literature survey of power efficiency of data centre. We discussed various methodologies of energy efficiency improvement. We also compared our test bench and modeling procedure with existing works, and presented some related research.

In Chapter 3, We explained system model of test bench, implementation details and collecting and simplifying data for determining variables for power consumption model. We also presented our modeling procedure to estimate the power consumption of application software running on servers. Our test bench is able to capture the characteristics of a system by correlating a number of user level utilization matrices that conclusively predict hardware performance counters with power consumption during high loads.

Chapter 4 presented two power consumption models (Neural Network and Polynomial) that have been implemented and validated using three platforms and four different scenarios. we demonstrated the feasibility of using a neural network model and a linear polynomial model to predict power consumption using the subsystem performance demonstrated as a means of measuring power meter, server and process count tool. A neural network model and linear polynomial model have been trained based on the process count information gathered by CollectD and the actual real-time power consumption monitored by a TED5000 power meter during applying the workload on three real servers. Our research reveals that a complete system power can be estimated via a new system server with an average MAE of energy estimation between 10% to 15%. While in the old system server, the average MAEe of energy estimation is between 1% to 4%.

Thus, We were able to introduce a cost effective way to create a model for predicting the power consumption of any application. With this ability developers can test their software and be able to understand how its behaviour from a power consumption point of view, can be tested in terms of performance and scalability. After the model has been prepared by the model developers, developers can use it to estimate the energy cost of the application software instead of wasting time on direct measurement. This methodology may offer a variety of solutions to developers involved in sophisticated model development. In the future, we are aiming to observe if the accuracy of the predicted power can be affected by the design of the workload. We also aiming to build a software tool that can does the whole modeling process by pressing few buttons. This tool will help the software developers to create power models that can predict the power consumption of their application software during the developing stage.

References

[1] Trevor Mudge. Power: A first-class architectural design constraint. *Computer*, 34(4):52–58, 2001.

[2] Kshirasagar Naik and David SL Wei. Software implementation strategies for power-conscious systems. *Mobile Networks and Applications*, 6(3):291–305, 2001.

[3] Manuj Sabharwal, Abhishek Agrawal, and Grace Metri. Enabling green it through energy-aware software. *IT Professional*, (1):19–27, 2013.

[4] David J Brown and Charles Reams. Toward energy-efficient computing. *Communications of the ACM*, 53(3):50–58, 2010.

[5] Kshirasagar Naik. *A survey of software based energy saving methodologies for handheld wireless communication devices*. Department of Electrical and Computer Engineering, University of Waterloo, 2010.

[6] Frank Bellosa. The benefits of event: driven energy accounting in power-sensitive systems. In *Proc. of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, pages 37–42. ACM, 2000.

[7] W Lloyd Bircher, Madhavi Valluri, Jason Law, and Lizy K John. Runtime identification of microprocessor energy saving opportunities. In *Low Power Electronics and Design, 2005. ISLPED'05. Proceedings of the 2005 International Symposium on*, pages 275–280. IEEE, 2005.

[8] Canturk Isci and Margaret Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *Proc. of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, pages 93–104. IEEE Computer Society, 2003.

[9] Tao Li and Lizy Kurian John. Run-time modeling and estimation of operating system power consumption. *ACM SIGMETRICS Performance Evaluation Review*, 31(1):160–171, 2003.

[10] William Lloyd Bircher and Lizy K John. Complete system power estimation using processor performance events. *Computers, IEEE Transactions on*, 61(4):563–577, 2012.

[11] Karan Singh, Major Bhadauria, and Sally A McKee. Real time power estimation and thread scheduling via performance counters. *ACM SIGARCH Computer Architecture News*, 37(2):46–55, 2009.

[12] Abdulhakim Abogharaf and Kshirasagar Naik. Client-centric data streaming on smartphones: An energy perspective. In *Mobile and Wireless Networking (MoWNeT), 2013 International Conference on Selected Topics in*, pages 36–41. IEEE, 2013.

[13] John Rath. Data center strategies. Technical report, Vantage Data Centers, 2011.

[14] Fadwa Abdulhalim, Omar Alghamdi, and Kshirasagar Naik. Modelling the energy cost of application software for developers. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, page 57. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015.

[15] Wolfgang Irrek and Stefan Thomas. Defining energy efficiency. 2008.

[16] GMHT Force. Harmonizing global metrics for data center energy efficiency. 2011.

[17] Richard Brown et al. Report to congress on server and data center energy efficiency: Public law 109-431. *Lawrence Berkeley National Laboratory*, 2008.

[18] Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM computer communication review*, 39(1):68–73, 2008.

[19] David Cole. Data center energy efficiency–looking beyond pue. *No Limits Software, White Paper*, 4, 2011.

[20] Jun Wang, Ling Feng, Wenwei Xue, and Zhanjiang Song. A survey on energy-efficient data management. *ACM SIGMOD Record*, 40(2):17–23, 2011.

[21] Bart R Zeydel and Vojin G Oklobdzija. Design of energy efficient digital circuits. In *High-Performance Energy-Efficient Microprocessor Design*, pages 31–55. Springer, 2006.

[22] David Bol et al. Robust and energy-efficient ultra-low-voltage circuit design under timing constraints in 65/45 nm cmos. *J. Low Power Electron. Appl*, 1(1):1–19, 2011.

[23] Sungkap Yeo and Hsien-Hsin S Lee. Peeling the power onion of data centers. In *Energy Efficient Thermal Management of Data Centers*, pages 137–168. Springer, 2012.

[24] AMD Staff. Amd powernow! technology brief. *Advanced Micro Devices, Inc*, 2002.

[25] AMD Staff. Amd cool'n'quiet technology. *Advanced Micro Devices, Inc*, 2002.

[26] STAR ENERGY. Version 5.0 system implementation, 2009.

[27] C Windeck. Energy star 4.0. *Ct German magazine for computer techniques*, 14:52–53, 2007.

[28] Sudhanva Gurumurthi, Jianyong Zhang, Anand Sivasubramaniam, Mahmut Kandemir, Hubertus Franke, Narayanan Vijaykrishnan, and Mary Jane Irwin. Interplay of energy and performance for disk arrays running transaction processing workloads. In *Performance Analysis of Systems and Software, 2003. ISPASS. 2003 IEEE International Symposium on*, pages 123–132. IEEE, 2003.

[29] David Andrew Roberts. *Efficient Data Center Architectures Using Non-Volatile Memory and Reliability Techniques*. PhD thesis, The University of Michigan, 2011.

[30] Otto VanGeet. Best practices guide for energy-efficient data center design. *US Department of Energy Federal Energy Management Program*, 2011.

[31] Emerson Network Power. Energy logic: Reducing data center energy consumption by creating savings that cascade across systems. *White paper, Emerson Electric Co*, 2009.

[32] Wes Felter, Karthick Rajamani, Tom Keller, and Cosmin Rusu. A performance-conserving approach for reducing peak power consumption in server systems. In *Proceedings of the 19th annual international conference on Supercomputing*, pages 293–302. ACM, 2005.

[33] Sudhanva Gurumurthi, Anand Sivasubramaniam, Mahmut Kandemir, and Hubertus Franke. Drpm: dynamic speed control for power management in server class disks. In *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, pages 169–179. IEEE, 2003.

[34] Qingbo Zhu, Francis M David, Christo F Devaraj, Zhenmin Li, Yuanyuan Zhou, and Pei Cao. Reducing energy consumption of disk storage using power-aware cache management. In *Software, IEE Proceedings-*, pages 118–118. IEEE, 2004.

[35] John Judge, Jack Pouchet, Anand Ekbote, and Sachin Dixit. Reducing data center energy consumption. *ASHRAE J., Nov*, pages 14–26, 2008.

[36] Michael R Marty and Mark D Hill. Virtual hierarchies to support server consolidation. In *ACM SIGARCH Computer Architecture News*, volume 35, pages 46–56. ACM, 2007.

[37] Frank Bellosa, Andreas Weissel, Martin Waitz, and Simon Kellner. Event-driven energy accounting for dynamic thermal management. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP03)*, volume 22, 2003.

[38] Russ Joseph and Margaret Martonosi. Run-time power estimation in high performance microprocessors. In *Proc. of the 2001 international symposium on Low power electronics and design*, pages 135–140. ACM, 2001.

[39] Youngjin Cho, Younghyun Kim, Sangyoung Park, and Naehyuck Chang. System-level power estimation using an on-chip bus performance monitoring unit. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 149–154. IEEE Press, 2008.

[40] Gilberto Contreras and Margaret Martonosi. Power prediction for intel xscale® processors using performance monitoring unit events. In *Low Power Electronics and Design, 2005. ISLPED'05. Proc. of the 2005 International Symposium on*, pages 221–226. IEEE, 2005.

[41] Ramon Bertran, Marc Gonzalez, Xavier Martorell, Nacho Navarro, and Eduard Ayguade. Decomposable and responsive power models for multicore processors using performance counters. In *Proceedings of the 24th ACM International Conference on Supercomputing*, pages 147–158. ACM, 2010.

[42] Jasmeet Singh, Veluppillai Mahinthan, and Kshirasagar Naik. Automation of energy performance evaluation of software applications on servers. In *Proc. of SERP*, volume 14, pages 7–13, 2014.

[43] Jasmeet Singh, Kshirasagar Naik, and Veluppillai Mahinthan. Impact of developer choices on energy consumption of software on servers. In *Proc. of SCSE'15*, 2015.

[44] Liang Luo, Wenjun Wu, WT Tsai, Dichen Di, and Fei Zhang. Simulation of power consumption of cloud data centers. *Simulation Modelling Practice and Theory*, 39:152–171, 2013.

[45] Chenghui Zhang, Adrian Hindle, and Daniel M German. The impact of user choice on energy consumption. *Software, IEEE*, 31(3):69–75, 2014.

[46] Rachita Kothiyal, Vasily Tarasov, Priya Sehgal, and Erez Zadok. Energy and performance evaluation of lossless file data compression on server systems. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*, page 4. ACM, 2009.

[47] Yuwen Sun, Lucas Wanner, and Mani Srivastava. Low-cost estimation of sub-system power. In *Green Computing Conference (IGCC), 2012 International*, pages 1–10. IEEE, 2012.

[48] John C McCullough, Yuvraj Agarwal, Jaideep Chandrashekar, Sathyanarayan Kuppuswamy, Alex C Snoeren, and Rajesh K Gupta. Evaluating the effectiveness of model-based power characterization. In *USENIX Annual Technical Conf*, volume 20, 2011.

[49] Aparna Datt, Anita Goel, and Suresh Chand Gupta. Comparing infrastructure monitoring with cloudstack compute services for cloud computing systems. In *Databases in Networked Information Systems*, pages 195–212. Springer, 2015.

[50] Georges Da Costa and Helmut Hlavacs. Methodology of measurement for energy consumption of applications. In *GRID*, pages 290–297, 2010.

[51] Z Cihan Taysi, M Amac Guvensan, and Tommaso Melodia. Tinyears: spying on house appliances with audio sensor nodes. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, pages 31–36. ACM, 2010.

[52] TED Support. Ted5000 series installation guide. http://www.theenergydetective.com/5000docs. accessed april 28, 2014.

[53] Cisco webex social troubleshooting guide, release 3.3 and 3.3 sr1. 2013.

[54] Andreas Brunnert, Christian Vgele, Alexandru Danciu, Matthias Pfaff, Manuel Mayer, and Helmut Krcmar. Performance management work. *Business and Information Systems Engineering*, 6(3):177–179, 2014.

[55] Felix Willnecker, Andreas Brunnert, and Helmut Krcmar. Model-based energy consumption prediction for mobile applications. In *Proceedings of Workshop on Energy Aware Software Development (EASED)@ EnviroInfo*, pages 747–752, 2014.

[56] Jan Jelschen, Marion Gottschalk, Mirco Josefiok, Cosmin Pitu, and Andreas Winter. Towards applying reengineering services to energy-efficient applications. In *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pages 353–358. IEEE, 2012.

[57] Marion Gottschalk, Mirco Josefiok, Jan Jelschen, and Andreas Winter. Removing energy code smells with reengineering services. In *GI-Jahrestagung*, pages 441–455, 2012.

[58] Giuseppe Procaccianti, Patricia Lago, Grace Lewis, et al. A catalogue of green architectural tactics for the cloud. In *Maintenance and Evolution of Service-Oriented and Cloud-Based*

Systems (MESOCA), 2014 IEEE 8th International Symposium on the, pages 29–36. IEEE, 2014.

[59] Hagen Höpfner, Maximilian Schirmer, and Christian Bunse. On measuring smartphones' software energy requirements. In *ICSOFT*, pages 165–171, 2012.

[60] Raghu Ramakrishnan and Johannes Gehrke. *Database management systems*. McGraw-Hill, Inc., New York, San Francisco, Washington, DC, 2000.

[61] Brad Cowie. Building a better network monitoring system. 2012.

[62] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*, volume 821. John Wiley & Sons, 2012.