Master's Thesis

Characterization and Modelling of Server Energy Consumption

Bernd Kampl

Institute for Technical Informatics
Graz University of Technology
Head of the Institute: Univ.-Prof. Dipl.-Ing. Dr.techn. Gernot Kubin

Assessor: Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger
Advisor: Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger
Dr. Damian Dalton (University College Dublin)

Graz, November 2013

If I have seen further
it is by standing on the
shoulders of giants.

ISAAC NEWTON

Abstract

The past years have seen a huge increase of energy consumption in the IT (information technology) industry. With more and more people requiring access to information, the server infrastructure is destined to grow larger in the coming years. This demand necessitates a better understanding and managing of IT equipment's energy consumption behavior. An increasing interest in sustainable computing has begun with the key word Green IT. Research to lower the rate with which IT industry's energy consumption increases is being done. This includes the fields of data center infrastructure management (DCIM), cooling, and resource management. However often times data center managers are still in the dark about the actual energy consumption of the servers they should be managing. The irish company Stratergia has developed Papillon, a software to estimate the energy consumption of servers in real-time. To estimate the energy consumption Papillon requires a power model of the server it is monitoring. This work contributes to the field of Green IT in the following manners. We investigate into the field of server energy consumption estimation and compare several methods and competitors. We analyze the existing power model generation and design a new solution with the given requirements. The improved software is subsequently implemented and evaluated. Additionally we devise a method to visualize the power models. The thesis is concluded by presenting insights gained and future research challenges.

Keywords: Stratergia Papillon, Server Power Models, IT Energy Consumption, Java, Matlab, Development

Kurzfassung

Der Energieverbrauch in der IT (Informationstechnologie) Industrie hat sich in den letzten Jahren stark erhöht. Dadurch, dass immer mehr und mehr Menschen Zugang zu Information benötigen wird sich die Server Infrastruktur weiter vergrößern. Diese Nachfrage macht ein besseres Verständnis und Verwalten des Energieverbrauchs von IT Infrastruktur notwendig. Mit dem Stichwort Green IT hat sich das Interesse an ökologisch nachhaltiger und zukunftsfähiger EDV erhöht. Bereits wird daran geforscht den wachsenden Energieverbrauch der IT Industrie einzubremsen. Die Hauptgebiete der Forschung sind dabei Datenzentrumsinfrastrukturverwaltung (DCIM), Kühlung, und Ressourcenverwaltung. Häufig aber ist der tatsächliche Energieverbrauch der Server unklar. Die irische Firma Stratergia hat mit Papillon eine Software entwickelt, die den Energieverbrauch von Servern in Echtzeit abschätzen kann. Dafür sind Power Models für die jeweiligen Server notwendig. Die vorliegende Arbeit leistet folgenden Beitrag im Bereich Green IT. Wir erforschen den Bereich der Energieverbrauchabschätzung bei Servern und vergleichen verschiedene Methoden und Mitbewerber. Wir analysieren den vorhandenen Prozess der Powermodelerzeugung und entwerfen eine neue Lösung mit festgelegten Anforderungen. Diese verbesserte Software ist dann implementiert und evaluiert. Zusätzlich konstruieren wir eine Methode um die Power Models zu visualisieren. Die Arbeit schließt mit dem Präsentieren von gewonnenen Erkenntnissen und zukünftigen Herausforderungen.

Keywords: Stratergia Papillon, Server Power Models, IT Energy Consumption, Java, Matlab, Development

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....
date (signature)

Acknowledgments

This master thesis was conducted in 2013 in cooperation between Graz University of Technology (TUG) and University College Dublin (UCD). I am deeply thankful for the help, advice, and feedback provided by my colleagues at both institutes at the School of Computer Science (CSI) in Dublin and the Institute for Technical Informatics (ITI) in Graz. I am especially indebted to my supervisors Christian Steger, Damian Dalton, and Abhay Vadher. The amount of immediate attention and feedback provided by them made this thesis possible and proved to be invaluable.

I also want to thank my family and friends for their support during my studies in Graz and abroad. My parents Luis and Maria and my sister Tanja supported me far more than could be expected through my long and challenging journey.

Special thanks also goes to Harald Tranninger, who saved me a tremendous amount of time by giving me information about living in Dublin. He is one of my fellow students of Telematics that I had the honor to encounter along the way. Most noticeable among those are Peter Riegler, Bernd Bergler, and Robert Stögbuchner, who have all helped me in one way or another by giving me their valuable time and input.

Last but not least I want to thank all of my friends who are not in any way connected to my studies. I appreciate their patience and companionship as well as the strength I gained through them.

Graz, November 2013 Bernd Kampl

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

While the term cloud computing has been extensively used as a buzz word in the technology industry similar to the proverbial knight in shining armor, it has been often overlooked that behind it there is a huge environmental impact involved, of which the majority of effects is still to be found out. Behind the scenes of the beautiful imagery of computing in the cloud is an extremely complex and intricate set of machines consuming vast amounts of energy. It is estimated that 1.5% of the world's energy consumption [Koo11] is used to maintain the operations carried out by data centers.

To efficiently tackle this problem several solutions have been proposed. Instead of picking a single solution, it is necessary to concurrently employ a number of actions, including efficient use of available resources, consciously selecting the source of power, and collaborations between data center operators. One part of this solution will be to accurately measure and predict the energy consumption of data centers and the servers they are housing.

Measuring the energy consumption of a data center itself is trivial, but most often a finer granularity and insight into the individual server's energy consumption is required to take purposeful action in lowering the energy consumption of the whole data center. One such way to gain information about a single server's energy consumption would be to connect power measuring equipment in between hardware and wall plug. Depending on the size of the data center this task can be very cumbersome and in the case of virtual machines it still does not provide the level of detail necessary to predict the individual server's energy consumption. In order to avoid connecting costly hardware to every server it is possible to evaluate the server's component's current usage and correlate it with the associated energy consumption.

Stratergia Ltd. has developed such a tool that takes on the task of measuring the power consumption of servers in use by correlating various system usage parameters to previously created power models, thusly gaining non-intrusive access to knowledge about a data center's energy consumption. The power models used to predict energy usage constitute an important part of the analysis software provided by Stratergia Ltd., therefore it is of great necessity that the power models themselves represent the system they model as accurately as possible.

Creating power models that accurately represent a system's power usage and reviewing the models to validate their quality are non-trivial tasks. This work concerns itself with those two tasks at hand and provides solutions by developing a unified solution to create power models on a given server using only Java and existing frameworks.

1.2 Goals

While the main focus of this work lies on the design and development of an integrated solution to generate the power models for the Papillon software by Stratergia Ltd., it is also looking to formalize very important aspects that need to be cautiously attended to in order to advance with this methodology. For this reason the following tasks are carried out and detailed in this work.

- An integrated solution is designed and developed with a focus on portability and easy maintainability of the software. Existing frameworks are incorporated and standardized solutions are given high priority.
- A means of visualizing the quality of the generated power models is developed to determine the strengths and weaknesses of the models.
- We explore the parameters used in the method to gain insight into the validity.
- We advance the currently employed prediction algorithm and show that the benchmarking process is of great importance to the accuracy of the power models.

1.3 Outline

Chapter 2 takes a closer look at energy efficiency in connection with data centers. We explore the big scale impact of the energy consumption in data centers. In order to understand the direction this industry is heading we look at the history and the current developments. After this we turn to observations on a smaller scale, namely server and server component level energy consumption observations and their impact.

We then compare several approaches, methods, and commercially available options in Chapter 3. We also give a closer look at the system developed by Stratergia, Papillon.

In Chapter 4 we analyze the currently existing system to generate power models. The requirements for the new system are determined. We also investigate into the methodology. The design for the new system is presented.

The implementation of the new system is detailed in Chapter 5. We present the challenges we faced and how we decided to overcome them. In addition we show the visualization of a sample power model. The changes to the methodology and the development of the figure of merit are also detailed in this chapter.

The experiments we conducted in order to verify the implementation and methodology are detailed in Chapter 6. We specify the hardware the experiments were executed on and our experiment methodology. The results to our all our experiments are presented in this chapter.

We finally conclude the thesis with Chapter 7. We give a summary over the thesis and recommendations for future work.

Chapter 2

Fundamentals

2.1 Data Center Evolution

Cloud computing starts with data centers. To be able to understand the power demands of data centers one has to possess knowledge about the historical developments that lead to the currently existing situation. As given in the timeline by Figure 2.1 the early beginnings of data centers are rooted in the huge computer rooms of the early ages housing the big single mainframes. To satisfy the need for security, stability, and to avoid damage to the expensive components, basic design guidelines for controlling access to the data center were devised. However these guidelines did not give much thought to problems that would arise with the boom of the microcomputer industry during the 1980s, especially power requirements, cooling issues, or controlled access to the system's resources.

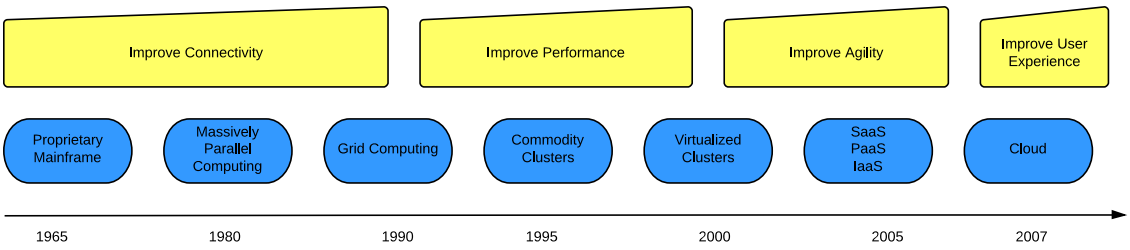

Figure 2.1: Historical development of data centers from the early beginnings during mainframe times up to the introduction of the term Cloud Computing.

Inexpensive network equipment and the success of the client-server model started a trend during the 1990s for companies to dedicate a specific room inside a company for always-on microcomputers, marking the birth of small in-company data centers. Establishing a presence on the Internet brought a flood of data centers during the dot-com bubble in the 2000s. The capabilities of in-house data centers were quickly exhausted and many companies solved that problem by building very large facilities, Internet data centers. New practices and technologies were devised to handle the complexity of such large-scale

operations, eventually turning these data centers into separately managed entities.

2.2 From Data Centers To Cloud Computing

The first and to date most successful company selling a complete set of web services is Amazon, which is expected to generate nearly $3 Billion in Amazon Web Service revenue in 2013 [Vel13]. This business was born of the realization that the company was able to build and sell a set of services based on the experience gained through building and operating the infrastructure of their own website Amazon.com.

Competing with Amazon are a number of companies that have their roots in different corners of information technology [MLB+11]. The established players include IBM, Google, Microsoft, and AT&T. These companies operate their own data centers and provide services on a broad scale ranging from the bottom layer of Infrastructure as a Service to the top layer of Software as a Service. Apache, EMC, and Cisco are seen as key technology providers. Both EMC and Cisco are quickly gaining grounds by smart acquisitions of technologies of growing importance, e.g. in the fields of virtualization and power awareness.

The transition from isolated data centers to the cloud shows a trend of collaboration between data centers of different locations. The rate of growth of servers and data centers that are housing them shows that the amount of energy that is being consumed by them can not simply stay an afterthought anymore.

2.3 Competitive Energy Analysis Techniques

With an estimated number of 30 Million servers connected to the Internet [Koo11] consuming 1.5-2% of all global electricity reported by Greenpeace in 2011 [CH11] it has become obvious that the energy consumption of data centers requires a better effort of monitoring and coordinating resources. The lack of transparency across the industry about IT's own greenhouse gas footprint is in part a result of a lack of insight into where the energy consumed ends up.

Interestingly enough and in line with this described lack of information is the inability of Greenpeace to get detailed information on whether cloud computing surpasses the traditional desktop solution in terms of energy efficiency [Tra13].

2.3.1 Greenpeace Cool IT Challenge

In 2009 Greenpeace launched the Cool IT Challenge to call on IT companies to power technological solutions needed to fight climate change. The three main goals of the campaign are to drive green energy innovation, champion more efficient operations, and seek green, renewable sources of power for the proliferation of data centers [Gre13].

In order to identify which firms are leading efforts to drive change in the energy sector Greenpeace evaluates global IT companies on a yearly basis through its Cool IT Leaderboard. April 2013 marks the 6th edition of the Cool IT Leaderboard, showing a slow but steady improvement in offering energy solutions and with companies demonstrating their willingness to make major investments to drive clean energy deployment. Additionally a growing number of companies are increasing their commitment to power their operations with greater percentages of renewable energy [Gre13].

The numbers in the Greenpeace Cool IT Leaderboard are made up of points received in three categories, based upon the effort of the company in each category. A total of 100 points can be achieved in the three categories *Climate Solutions* (40 points), *IT Energy Impact* (25 points), and *Political Advocacy* (35 points with a possible penalty of -5 to -15 points for *Negative Lobbying*) [Gre13].

The 2013 ranking sees Google and Cisco sharing 1st place with a total of 58 points achieved. These two companies and Ericsson on 3rd place with 51 points are the only companies on the Leaderboard to earn over half of the points possible, which shows the huge potential for improvement.

2.3.2 Google Green Data Centers

Occupying the top positions in the yearly Greenpeace Cool IT Leaderboard is Google with their efforts to have a neutral carbon footprint. Their data centers operate with the best energy efficiency in the industry not only by applying a number of steps in the data center management [Goo13b]. To maintain the top position and improve their Cool IT Leaderboard score, a number of projects exceeding just the scope of the data center operation are employed, including extensive collaborations with local businesses close to the data centers and investment in global development of green energy production.

Google also provides best practice guidelines to improve data center energy efficiency [Goo13a]. Their five outlined major steps consist of:

- Measuring performance following the credo of *you can't manage what you don't measure*. Sampling as often as possible without constraining the overall performance.
- Intelligently managing and treating airflow as a resource with a big impact on energy efficiency.
- Avoiding to unnecessarily cool the data centers below acceptable working temperature.
- Taking advantage of free cooling wherever possible using chillers.
- Eliminating as many power conversion steps as possible to minimize power distribution losses.

2.3.3 The Green Grid

The Green Grid is worth mentioning as it is a non-profit consortium made up of a wide variety of members including companies, facility architects, technology providers, policy-makers, and end-users. Through their website[1] the consortium provides white papers, tools, and definitions to make it easier for data center operators and managers to evaluate their own operational efficiency. This comprehensive set of tools and instructions represent an elaborate collection of data center efficiency guidelines.

[1] `http://www.thegreengrid.org`

2.4 Data Center Energy Efficiency Metrics

Several metrics have been defined to assist in determining the energy effectiveness of data centers. The most important metrics used in the industry are as follows.

2.4.1 Power Usage Effectiveness

Power Usage Effectiveness (PUE, sometimes Power Usage Efficiency) is a measure for how efficiently a building delivers energy to the IT equipment inside. Developed by The Green Grid Association, it has been globally adopted by the industry since its publication in 2007, with the newest document from 2012 superseding prior definitions [AAF12]. The ideal PUE of 1.0 stands for zero facility overhead energy, meaning every watt of power going into the building is going straight to the servers and IT equipment and nowhere else. PUE is calculated according to Equation 2.1.

$$\text{PUE} = \frac{\text{Total Facility Energy}}{\text{IT Equipment Energy}} \tag{2.1}$$

PUE can be computed using either energy (kilowatt-hour) or power (kilowatt) measurements. Energy measurements are more accurate, since power measurements only sample the energy flow at the exact time of the measurement, while energy measurements accumulate power flow over time [AAF12].

For PUE to be a useful indicator it must be measured over a long period of time and considered in combination with other metrics. PUE does not give any information about the amount of energy consumed or the type of energy source and therefore can not be regarded as a standalone, comprehensive efficiency metric.

2.4.2 Data Center Infrastructure Efficiency

Data Center Infrastructure Efficiency (DCiE) is the inverse of PUE: it is IT equipment energy divided by total facility energy. Both PUE and DCiE were introduced by The Green Grid and PUE has since surpassed DCiE in industry adoption. According to The Green Grid PUE is being seen as the industry-preferred metric for measuring infrastructure energy efficiency in data centers [AAF12] and is only being shown in Equation 2.2 for the sake of completeness.

$$\text{DCiE} = \frac{1}{\text{Power Usage Effectiveness}} \tag{2.2}$$

2.4.3 Carbon Usage Effectiveness

Carbon Usage Effectiveness (CUE) is proposed by The Green Grid to address carbon emissions associated with data centers [Bel10]. When used in combination with PUE, data center operators can quickly assess the sustainability of their data centers and determine the improvements need to be made. For data centers obtaining their entire power source from the energy grid without generating local CO_2. Equation 2.3 defines CUE. An alternate approach is to multiply the carbon emission factor (CEF) by the data center's annual PUE as shown in Equation 2.4. The units of the CUE metric are kilograms of

carbon dioxide ($kgCO_2eq$) per kilowatt-hour (kWh). The ideal value is 0.0, indicating that no carbon use is associated with the data center's operations.

$$\text{CUE} = \frac{\text{Total } CO_2 \text{ emissions caused by the Total Data Center Energy}}{\text{IT Equipment Energy}} \tag{2.3}$$

$$\text{CUE} = \text{CEF} \times \text{PUE} \tag{2.4}$$

2.4.4 Water Usage Effectiveness

Analogous to the definitions used to describe CUE, Water Usage Effectiveness is a metric to assess the water used for operations of the data center [Pat11]. It is generally used in combination with PUE and CUE and calculated according to Equation 2.5. The units of WUE are liters/kilowatt-hour (L/kWh).

$$\text{WUE} = \frac{\text{Annual Site Water Usage}}{\text{IT Equipment Energy}} \tag{2.5}$$

2.4.5 Energy Reuse Effectiveness

The extensive use of PUE has caused some confusion in assessing data center facilities that reuse waste energy and claim a PUE of less than 1.0. While the intent is based upon laudable motives, the math and application has been in conflict with the definition of 1.0 as the lowest possible PUE. This deficiency is corrected with the introduction of the Energy Reuse Effectiveness (ERE) metric [Pat10].

The simplest example for energy to be considered for ERE would be a chiller being driven by data center waste heat. The energy amount used for the metric is the waste heat going into the chiller and not the cooling energy delivered by the chiller. The values used in ERE are all units of energy depending on the energy type, e.g. kW-hrs for electricity, gallons for diesel fuel, etc. Equation 2.6 defines the calculation. The lowest theoretical ERE of 0.0 would mean that no energy from outside is being used in powering the data center.

$$\text{ERE} = \frac{\text{Cooling + Power + Lighting + IT Equipment - Reuse}}{\text{IT Equipment Energy}} \tag{2.6}$$

2.4.6 Data Center Productivity

Data Center Productivity (DCP) is a methodology for quantifying the useful work that a data center produces relative to the quantity of any resource that it consumes to produce this work [ACD+08]. DCP therefore defines a family of metrics each with a different quantity in the denominator. Equation 2.7 expresses the concept mathematically.

$$\text{DCP} = \frac{\text{Useful Work Produced by Data Center}}{\text{Resource Consumed Producing the Work}} \tag{2.7}$$

2.4.7 Data Center Compute Efficiency

Data Center Compute Efficiency (DCcE) and it's underlying sub-metric Server Compute Efficiency (ScE) enable data center operators to determine the efficiency of their compute resources and in turn allow them to identify areas of inefficiency [Bla10]. DCcE is not a productivity metric and does not allow for comparison between data centers. It can be used to determine the proportion of measured work providing the primary services of the data center compared to the total amount of energy consumed.

2.4.8 Performance Per Watt

Another widely used but not always very meaningful metric is Performance Per Watt (PPW). It is generally understood that when this metric is presented the system under test is running is at maximum load. The measuring unit is generally defined as FLOPS per watt. The Green500 website keeps an annually updated list[2] of the 500 most power efficient supercomputers. At the top of the list updated on June 2013 is the Eurotech Aurora HPC 10-20 located in the CINECA data center in Italy. In terms of green computing the PPW measurement can be used in situations where it is necessary to determine if newer hardware could do the same job while requiring less energy.

2.5 Physical Limits and Energy Efficiency Observations

Even before the term Green Computing was a much discussed topic, several interesting observations were made in regards of energy consumption. Landauer's Principle defines a lower theoretical limit of energy consumption of a computation, while the Khazzoom-Brookes Postulate details a paradoxic behavior about higher energy consumption with increased energy efficiency called the rebound effect.

2.5.1 Landauer's Principle

The physicist Rolf Landauer stated in 1961 that there is a lower physical limit regarding the energy consumption of a computation. Known as the Landauer limit, this principle asserts that changing one bit of information requires a minimum possible amount of energy [FDOR12]. This Landauer limit is given in Equation 2.8, equalling to the minimum possible amount of energy.

$$k \cdot T \cdot ln(2) \tag{2.8}$$

In Equation 2.8 k stands for the Boltzmann constant, T is the temperature of the circuit in kelvin and *ln(2)* is the natural logarithm of 2.

While currently computation takes a million times more energy than theoretically necessary, the Landauer principle postulates a hard lower limit for power consumed in information processing. Experimental studies have recently been able to provide further evidence that Landauer's principle is correct [BAP+12].

[2] `http://www.green500.org/lists`

2.5.2 Khazzoom-Brookes Postulate

The Khazzoom-Brookes postulate is a hypothesis put forward in the 1980s independently by the economists Daniel Khazzoom and Leonard Brookes. The economist Harry Saunders showed a decade later that the postulate was true over a wide range of assumptions [Sau92].

The postulate states that energy efficiency improvements on a microscopic level, like improving the fuel efficiency of a car, lead to more energy consumed on a macroscopic level, e.g. people traveling further with their cars. The Khazzoom-Brookes postulate is actually a special case of the rebound effect and was first observed in 1865 by William Stanley Jevons.

Jevons described in his book *The Coal Question* how England's consumption of coal increased after James Watt introduced his coal-fired steam engine, which greatly improved the efficiency of Thomas Newcomen's earlier design. This phenomenon called Jevons paradox appeared because increasing the cost efficiency of coal as a power source led to the increased usage of the steam engine in a wider range of industries.

As previously mentioned, the Khazzoom-Brookes postulate is only one of three possible outcomes of the increase in efficiency of a power source. The three possible outcomes regarding the size of the rebound effect are:

1. The actual resource savings are higher than expected, resulting in a negative rebound effect. This is usually due to a mandated switch to a more resource efficient technology, that is also more costly to use.

2. The actual savings are less than the expected savings, with the rebound effect being in between 0% and 100%.

3. The resource savings are negative, the rebound effect is higher than 100%. This situation is the Jevons paradox or Khazzoom-Brookes postulate in practice.

Given the knowledge about the rebound effect it is not enough to just increase the energy efficiency and continue using fossil energy. It is necessary to switch to sustainable energy sources. Therefore the planning of an energy efficient data center already begins with selecting the site where the data center will be located.

2.6 Data Center Power Provisioning

While the cost of hardware and building data centers is continuing to decrease, the recurring energy consumption costs have already risen to rival the costs of building the data centers able to deliver a certain power capacity. For this reason facilities should operate as close to possible to maximum capacity, so that the non-recurring facility costs can be best amortized [FWB07]. Fan et al. estimate for a facility operating at 85% of its peak capacity on average, the cost of building the facility will still be higher than all electricity expenses for ten years of operation.

The risks involved in operating a data center very close to maximum capacity exists in exceeding its maximum capacity, resulting in outages or costly violations of service agreements. It is therefore critical to determine the right deployment and power management strategy. Estimating and observating the power usage characteristics of thousands of servers in a data center is complicated by three important factors:

- The nameplate value (rated maximum power value) of servers is generally extremely conservative, and in most cases is never even come close to at all. This limits its usefulness severly.
- Energy consumption per server is heavily dependent on actual hardware utilization, resulting in a very hard to predict dynamic power range.
- On a data enter level the application using the servers might have a big impact and applications might vary to a great extent in energy consumption.

Of these three factors the main dynamic source of inefficiency in power deployments is load variation, meaning the variable levels of energy consumption of a server based on utilization. Power deployment decisions in a data center are generally made at the rack level, power distributions unit (PDU) level, and data center top level. These decisions are complicated by the fact that most facilities lack on-line power monitoring and data collection systems that are needed. Power deployment at the rack level using server nameplate values is guaranteed to under-utilize the provided power, as Fan et al. found that using their most power intensive benchmark reached a maximum of less than 60% of the nameplate value [FWB07].

Several methods to help achieve this goal are proposed and a number of them are already successfully in use.

2.6.1 Power Capping

Given that very seldom servers actually reach their peak capacity and then only for a short time, the power capacity of the data center still has to be calculated to account for these peak times. This results in wasted resources during most of the times when the data center is not running at full capacity. If somehow it could be guaranteed that a server will not exceed a given power limit, a data center might be able to increase the number of machines hosted within a given power budget.

Power Capping essentially is a solution to this problem. Through some sort of control loop it is ensured that no server draws more than a certain specified amount of power. Numerous ways to implement this already exist. They basically consist of a power monitoring system and a power throttling mechanism. When the a certain power threshold is about to be crossed and the monitoring system notices this, the power draw might be throttled by descheduling tasks or other component-level power management methods, such as CPU voltage/frequency scaling.

The biggest advantage to dynamic power capping is that it can relax the requirement to accurately characterize workloads prior to deployment, and provide a safety valve for cases where workload behavior changes unexpectedly [FWB07].

2.7 Server Power Consumption Observations

Analyzing the current behavior of server power consumption is essential in order to be able to understand and provide a good solution. Therefore analysis has to be conducted on various levels, from the top level of data center power consumption including numerous servers down to the bottom level of a server's single component's power consumption behavior.

2.7.1 Server Utilization Observations

Barroso and Hölzle published an article [BH07] in which they call on data center operators and hardware manufacturers to give more attention to the energy usage profiles of system components. Their report includes interesting statistics about a data center housing more than 5000 servers monitored during a six month period.

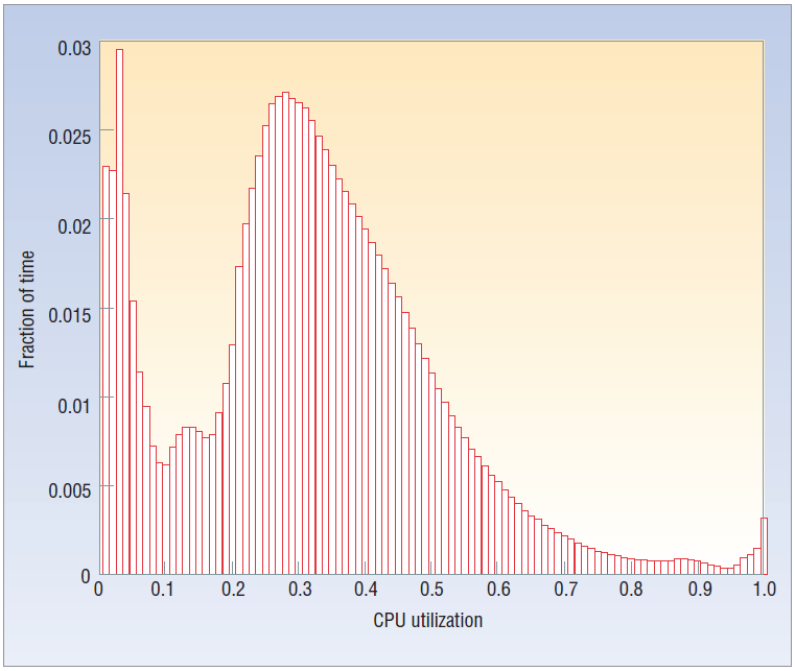

Figure 2.2: Average CPU utilization of more than 5000 servers during a six month period. Servers operate most of the time at between 10% and 50% of their maximum utilization [BH07].

Figure 2.2 details how much time the probed servers spend at the various utilization levels. Interesting to notice is that the servers spend the least time at utilization levels between 60% and 95%, with a minor local maximum at 100%. The global maximum is at 4% utilization, indicating a very high percentage of servers idling for most of the time. Even though this behavior is not accidental and actually designed to account for the peak times where all servers are necessary to handle the load posed upon them, it still results in a substantial waste of capital.

Accounting for server peak usage times is not the only reason why data centers are designed to work this way. Spreading data across multiple machines eliminates bottlenecks

and reduces the likelihood of data loss because of hardware failure. Putting idle servers into energy saving modes or turning them off is not always an option, especially because the most attractive inactive energy savings modes are usually also those with the highest wake-up pentalties, e.g. disk spin-up time.

2.7.2 Server Energy Efficiency Observations

When characterizing a server's power consumption it is also inevitable to look at the corresponding usage statistics. As already mentioned, even an energy-efficient server running in idle is essentially wasting energy because it still uses at least 50% of its maximum power. This figure already considers servers designed with energy efficiency in mind, more often than not servers consume more than half of their maximum power in idle.

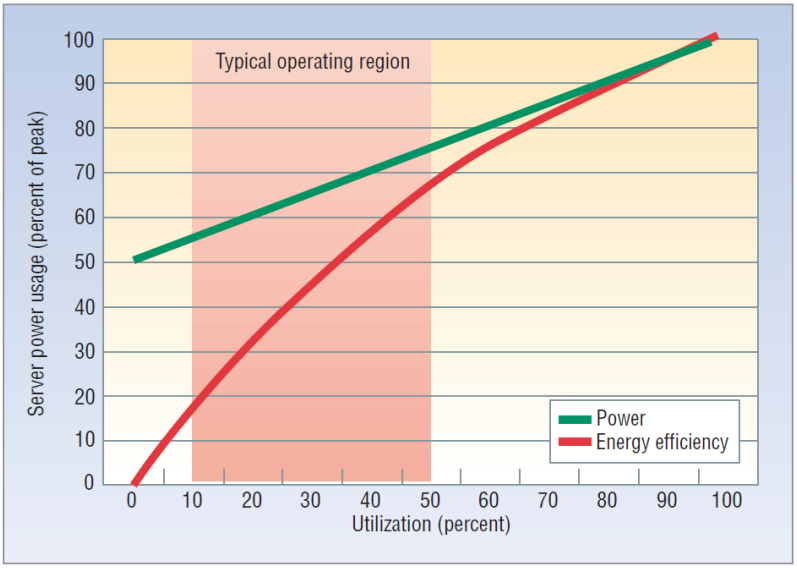

Figure 2.3: Server power usage and energy efficiency at varying utilization levels, from idle to peak performance. The highlighted region shows that servers spend most of the time operating in energy inefficient regions [BH07]. Utilization in this graph is loosely defined as a measure of application performance.

Figure 2.3 shows the power usage of a typical energy-efficient server as a function of utilization. The conclusions that can be drawn from Figure 2.3 are that on an energy-efficient server the dynamic power consumption can be as high as 50% of the maximum power consumption. Currently servers spend most of the time in energy-inefficient utilization regions. Energy efficiency drops very quickly and drastically the less the server is utilized. Notable is also that energy efficiency in the 20% to 30% utilization range is at less than half the energy efficiency at peak performance.

Barroso and Hölzle claim in their article that while the CPU was dominating platform power at peak usage, this is not the case anymore, since processors in modern servers are adopting energy-efficiency techniques more aggressively than other system components [BH07]. It is expected that CPUs contribute an even smaller fraction of peak power in future systems. Responsible for this increased energy efficiency in CPUs are features such as

a wider dynamic power range and active low-power modes. Increasing the overall dynamic range of a server's dynamic power consumption complicates the estimation of the server's power consumption using conventional methods. To correctly predict a server's power consumption it is therefore crucial to have a good understanding about the components responsible for the dynamic power consumption.

2.7.3 Server Component Power Consumption Breakdown

The power consumption of a server or any other system such as a laptop is fundamentally a combination of the power consumption of its individual component's power consumption. For each and every particular component its power consumption is made up of a static and a dynamic amount of power consumed. The static power consumption stays the same regardless of the amount of work carried out by the component, whereas the dynamic power consumption of the component varies by the workload processed by this precise component. To reach a system's overall power consumption for a given point in time (or energy consumed over a time period) we are able to sum up all the component's static power consumption into the system's static power consumption, just like the system's dynamic power consumption is made up by the sum of the ranges of the component's dynamic power consumption.

Based upon previous work by Mahesri and Vardhan [MV05] and Economou et al. [ERKR06] it is possible to investigate into component-level power consumption of information processing systems in order to understand future power consumption trends. Mahesri and Vardhan [MV05] managed to achieve a complete component-wise breakdown of the power consumption of a 2005 era IBM ThinkPad R40 laptop using oscilloscopes and probes and by partially disassembling the laptop. Economou et al. [ERKR06] divide the hardware of an unspecified 2006 era blade server into four different power planes to measure the power consumed in various portions of the server.

Combining the information gained from previous work and our own findings leave us with a number of very interesting findings.

- As already expected, total system power varies considerably depending on workloads. The better the energy efficiency of a system is the higher the extent of the dynamic power consumption is in comparison to the static power consumption.
- For both setups the CPU is still the dominating component for both static and dynamic power consumption, in spite of various energy efficiency optimization methods such as dynamic voltage scaling (DVS). Still, DVS power savings are in fact significant enough to contribute to a lower total system power [MV05].
- On the laptop graphics, wireless, and optical drives are major power consumers only in specific workloads. This indicates that the dynamic power consumption for these components is higher than the static power consumption. This assumption holds specifically for mechanically working parts that also can be switched to power saving modes, e.g. fans. It is to be expected that the dynamic power consumption in servers that make use of arrays of graphic cards is drastically higher than the static power consumption and special attention needs to be paid in these cases.

- Memory usage and its power consumption are extremely complicated to measure due to its dependency on CPU and HDD components. Economou et al. consider memory power consumption to be equally, if not more, important in the future [ERKR06]. Their research shows variations in memory power consumptions of up to 15W, giving memory up to 40% of the overall power consumption of the blade server, depending on the workload.

- Power consumption is not independent of the operating system. Mahesri and Vardhan report a higher idle system power consumption while running Linux OS compared to Windows OS [MV05]. They attribute the difference to the fact that ACPI support was not implemented on the Linux kernel they were using and they were unable to get ACPI working even after upgrading. The difference in power consumption on different operating systems should especially be considered when setting up virtual machines or using power models created with machines running a different operating system.

- In addition to varying power consumption behavior on different operating systems Mahesri and Vardhan observed a mysterious behavior regarding hard drive accesses made by Windows OS when the drive was in idle state [MV05]. These hard drive accesses increased the disk's power consumption by 0.2W, whereas the behavior did not occur under Linux. This highlights that an extensive knowledge about the components and their power consumption behavior is necessary to acquire a good characterization of the system as a whole.

- One of the four power planes the analyzed server hardware was divided into, Economou et al. made the interesting observation that 30% to 40% of the total system power was spent on disk, network, I/O and peripherals, power supplies, and the rest of the glue circuitry in the server [ERKR06]. The single largest contributors to this collection are the disk and the power supply. The difference in the power consumption behavior of these two components lies in the amount of dynamic and static power consumption. Whereas the the power supply has a high amount of static power consumption and very little dynamic changes due to the fact that the energy efficiency in power supplies is higher in regions closer to their peak power, the hard drive can have as much as 30% of difference in power consumption in mechanic hard drives. For solid state drives (SSD) the percentage of dynamic power consumption is even higher; 75% dynamic power consumption is nothing out of the ordinary for SSDs, while the overall amount of power consumed is a fraction of a conventional hard drive. Considering the fact that SSD prices are becoming reasonable enough to be used in servers, the power behavior should be treated as that of different types of components.

- In contrast to most other components, the dynamic range for the motherboard itself is negligible [ERKR06].

- As previously mentioned in chapter 2.6 the inaccuracy of the nameplate power ratings are too far off to be considered relevant [ERKR06]. A comparison of actually measured numbers on absolute power consumption and component-level breakdown reveals that the nameplate power rating overestimates power by almost 50%, and mis-estimates the importance of various components. Considered with the fact that currently it is common practice to use nameplate power when provisioning and optimizing a data center structure this becomes particularly important.

As previously described each of the individual components the system is made of has a static and a dynamic power consumption. The non-trivial task at hand is then to identify the percentage and range of the dynamic power consumption, how it correlates to the the particular component's utilization, and, if there is one, its dependency on other component's utilization.

2.8 Virtual Machines

One way to achieve high energy efficiency in a data center is to constantly operate the available hardware in the energy efficient high utilization regions. Although it is theoretically possible to design and distribute the workload such that every resource on each server is optimally utilized, in practice this approach comes with an uncontrollable amount of drawbacks. A much better and more common solution is to set up virtual machines and run numerous virtual machines on a single physical server.

This technique enables servers to operate longer periods of time in regions of high utilization, where the servers are much more power efficient as previously mentioned. Migrating virtual machines from multiple servers with low utilization to be combined on fewer servers with a higher utilization opens up the possibility to switch some physical servers to a low power state or even shut them down during periods of low data center utilization.

The decision to operate a data center based on virtual machines can not be an afterthought. Since components of physical servers running virtual machines are much more likely to be operating in regions of higher utilization a majority of the time, it is necessary to plan accordingly from the beginning. If a component of a server breaks down, chances are high that much more than just one virtual machine are impacted.

2.8.1 Types of Virtual Machines

Two major virtual machine hypervisor classifications can be distinguished based on their distance to the physical hardware they are running on. These two classifications are represented in Figure 2.4.

1. A type 1 hypervisor runs directly on the host's hardware, controlling the hardware and managing the guest operating systems all with a very small footprint. This model represents the original hypervisor developed by IBM in the 1960s as bare-metal tools. The guest operating system runs on the second software level above the hardware. Software products working as a type 1 hypervisor are Oracle VM Server, Citrix XenServer, VMware ESX/ESXi, and Microsoft Hyper-V.

2. A type 2 hypervisor uses a conventional operating system as the basis for hosting the virtual machines. With the hypervisor layer as the second software level, guest operating systems run another level higher above the hardware than on type 1 hypervisors. Type 2 hypervisor examples are VMware Workstation and Oracle VirtualBox.

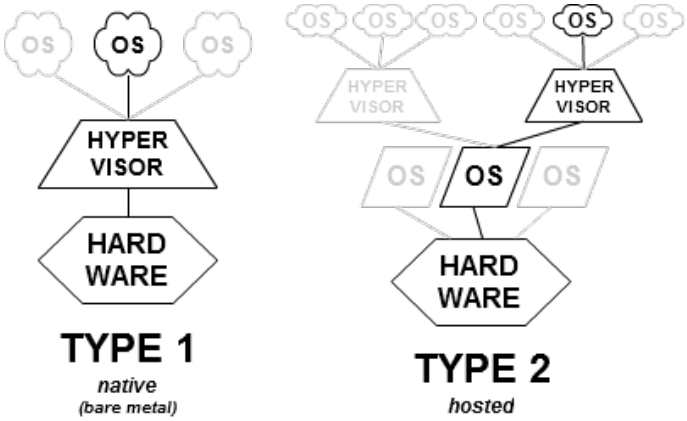

Figure 2.4: Type 1 hypervisors run directly on top of the hardware with the guest operating system on the second software level. Type 2 hypervisors operate as applications in a host operating system, therefore adding a software level and letting the guest operating system run on the third software level.

Type 1 hypervisors have the advantage of not needing to rely on an underlying operating system and generally having a smaller resource usage overhead. Newer implementations are not always easily classifiable, such as Microsoft Hyper-V, where a virtualized Windows Server parent partition is used to manage the type 1 Hyper-V hypervisor. This works by having the Hyper-V hypervisor load prior to the management operating system. Any virtual environments created run directly on the hypervisor and not via the management operating system.

2.8.2 Virtual Machine Migration

In a server environment where physical servers are operating in regions of high utilization due to several virtual machines running on these servers the live migration of these virtual machines on demand is necessary in order to balance workloads. Live migration describes the process of moving a running virtual machine from one physical server without disrupting the service offered by the migrated virtual machine.

Seamless live migration techniques work by copying the memory pages from physical source to destination machine, while the virtual machine keeps running on the source server. If memory pages on the destination become invalid due to changes on the source server they will be copied again until the rate of these re-copied pages is not less than the page changing rate on the source machine. After this warm-up phase the virtual machine will be stopped on the source machine and the remaining pages copied to the destination, where the virtual machine will be resumed. The time frame where the virtual machine is

stopped should only range from milliseconds to seconds, depending on the size of memory and applications running on the virtual machine. This technique allows for a better and more dynamic handling of the available resources.

In most modern data centers with virtualization virtual machine migration inside the datacenter and from and to outside the data center have to be considered. This leads to a dynamic energy footprint of not just the machines, but the whole data center itself. In these cases power supply provision needs to be in sync with the virtual machine migration.

Techniques like the live virtual machine migration open up several possibilities to improve the efficient energy use in IT and data center business. Data center operators that own several data centers world wide are able to balance the load posed upon their service not only inside the data center but view their independent data centers as a dynamic entity. As already mentioned, energy consumption of a data center is tightly coupled with the load posed upon a data center. The amount of services required from a data center can vary greatly over time, similar to the energy consumption fluctuation of a regular household during the day. With several data centers in different places (and most likely time zones) available, data center operators possess the potential to migrate virtual machines to data centers that offer optimal processing of the requests at that given time. Even greater potential to operate with higher energy efficiency comes when data center operators start to consider where the energy for their data centers come from. Especially in areas with a high percentage of renewable energy such as wind power, hydro power, and solar power the power supply is not constantly on the same level. During times where renewable energy is available in abundance live virtual machine migration can be used to shift workload to data centers that are supplied by this peak of available energy. To benefit from this technology a data center operator has to have knowledge over a few key data.

- Knowledge about the energy requirements of the physical server hosting the virtual machine and the resource requirements of the virtual machine providing the service, resulting in energy consumption on the physical server. A good power model for the physical server enables to calculate the energy consumption based on resource usage. The service provided by the virtual machine should be characterized by a resource usage model, which in turn allows to calculate the energy consumption on the physical server.
  - *How much energy will be consumed to provide the requested service?*
- Knowledge about the energy provision, availability, price, and source.
  - *How much does one unit of energy cost?*
  - *Where does the energy come from? Is it renewable energy?*
  - *Is it a good time to switch to that source of energy, e.g. peak times or during a specific time of the day?*
  - *How long will this source of energy be available, e.g. how long will the peak time last?*

- Knowledge about the infrastructure of the data centers providing the service.
  - *Is it possible to migrate the virtual machines? Is the necessary hardware available? Are the systems compatible?*
  - *Is enough bandwidth available to migrate the virtual machines hosting the service?*
  - *Does migrating impact the service in a way the customer will notice, e.g. sudden drop in quality for the customer?*
- Knowledge about the requirements that need to be met by the service.
  - *Is it a time crucial service? Can I afford the down time resulting from the virtual machine migration?*
  - *Are security requirements still being met after virtual machine migration?*
  - *Is the general latency for the service good enough to maintain the same quality of service after virtual machine migration?*

It is easily recognizable that it is essential for data center operators to possess detailed knowledge about their systems on both high and low levels of hardware and software to fully take advantage of the possibilities presented. Energy consumption of server hardware is a necessary key knowledge, yet alone not sufficient enough.

2.9 Data Center Infrastructure Management

Servers, traditional IT and network equipment are not the only thing that comprise data center infrastructure. The data center infrastructure ecosystem now contains air conditioning, power systems, uninterrupted power supplies and generators, and switching equipment. Changing one component can have unintended impacts on one or more of the other components of this ecosystem.

Whereas these components used to be managed using simple spread sheets, this approach is not sufficient anymore. A new market named Data Center Infrastructure Management (DCIM) has emerged, being a juxtaposition of two other markets, systems performance management and building management systems [Cap10].

Data center managers must have the insight needed to properly plan and forecast future data center capacities. Understanding the multiple layers of effects that every change in the data center infrastructure will have on the environment of the data center is essential in order to succeed in capacity planning.

Therefore DCIM is the integration of data center assets and physical infrastructure into a centralized management system with the ability to monitor and collect infrastructure data from the top down to a very low level to enable intelligent analysis and capacity planning.

Monitoring energy consumption and equipment performance has been performed for many years, but almost exclusively on the facilities side of the business, and rarely on IT equipment [Cap10]. Existing building management systems monitor security, power, lighting, and all facets of the day-to-day operations of the building itself, while operations technologies have been used to manage the physical equipment needed to run the business.

Product	Vendor
Asset Point	Align Communications
InfraStruXure	APC
Vista 600	Aperture (Emerson-Liebert)
DSView 3	Avocent (Emerson-Liebert)
Operations Manager	HP
Maximo	IBM (with Tivoli)
Modius	Modius
nLyte Software	nLyte
PI System	OSIsoft
Physical Infrastructure Manager	Panduit
Data Center Manager	Rackwise
dc Track	Raritan
Sentilla Energy Manager	Sentilla
Sentry Power Manager	ServerTech
Synapsense	Synapsense
Viridity	Viridity

Table 2.1: List of DCIM Software Vendors and the products they are offering [Cap10].

IT equipment, on the other hand, was rarely monitored for energy consumption, but rather for performance and availability [Cap10].

Several vendors have been emerging that have begun to integrate some of the tools of building management systems into IT asset monitoring. Table 2.1 lists the most important DCIM software tools compiled by Gartner in 2010 [Cap10].

Cappuccio estimates the use of DCIM tools and processes will become mainstream in data centers, growing from 1% penetration in 2010 to 60% in 2014 [Cap10]. The long-term benefit of this will be the advent of intelligent capacity planning (ICP). If a device's performance characteristics are known, it will become possible to perform predictive analysis of future changes to the environment. ICP will show for new applications if the space and power are available to support its components, as well as predict accurately the potential impact of this new application on the entire infrastructure, essentially providing cascade impact analysis before the application is deployed.

The objectives of a DCIM system is best defined by three keywords: availability, manageability, and economy [FFN12].

Availability denotes the correct operation of a data center every time a service is requested. Several standards are defined by the Uptime Institute to provide guidance for technical solutions and acceptable levels of downtime. DCIM systems should support the data center operator in achieving a high level of availability and improving the physical maintainability of the data center.

Manageability describes the data center manager's capability to administrate the complex data center operations and maintenance. Interdependently and in parallel working subsystems must be running correctly to keep the data center available. DCIM systems should support the management by organizing responsibilities, controlling access permissions, creating forms, reports, and plots from collected data, and providing a comprehen-

sive overview of the data center.

Economy defines how to continuously reduce the cost of data center operation and growth. In this case the previously mentioned PUE is often cited as the main index number to show the data center general energy efficiency. DCIM systems should be able to support the measurement of electrical power consumption of the data center's devices and infrastructure. Beyond only measuring and calculating this information, it should provide means to analyze and compare them to standard limits and dispatch warnings where issues or overloads are detected. DCIM should help on reducing device location mistakes, simulating and defining the best places according to heat generation and energy consumption, optimizing the whole data center distribution and organization.

Typical components of a DCIM system are the following, as outlined by Tranninger [Tra13]:

- Discovering IT assets automatically.
- Physical and virtual environment visualization.
- Data center power usage analysis.
- Modeling data center processes and work flows.
- Future resource capacity planning.

As a result of the overwhelming amount of vendors scrambling to get a product on the market, a lot of the solutions are often not delivering on their promises and end up being far behind the expectations. It is to be expected that in 2013 DCIM will exit the "Peak of inflated expectations" and enter the "Trough of disillusionment", in terms of the Gartner hype cycle [Ver13].

Even though it is a strong recommendation for data center operators to employ DCIM tools and processes, it should be carefully evaluated what the best approach and software for the data center in question is. Faccioni Filho and Neto propose a method to evaluate DCIM tools that do not yet meet a standard or general pattern [FFN12]. The five criteria they propose to evaluate DCIM software on are: data, automation, management, interface, and diagnosis. Their method is dynamical and new evaluation requirements can be added to it at any time as a way to improve it regularly.

Chapter 3

State of the Art

3.1 Server Power Metering

Although server energy consumption had previously been low priority in DCIM tools, it has now become a major influence in determining strategies for data center operators. More and more DCIM tools are looking to integrate power measuring methods into their existing tools. As of now the majority of products only provide a very basic overview over the data center energy consumption. This chapter is dedicated to investigating state of the art approaches to server energy consumption measuring. Some of these methods are part of commercial DCIM software, others are non-commercial scientific research results.

3.2 Server Power Metering Approaches

The gold standard to measuring the energy consumption of a server is to connect an external power meter. Constantly measuring the energy consumption on a number of devices is not feasible due to a number of reasons. Hardware power meters are very expensive and require proper maintenance, as well as the amount of work necessary to physically connect and disconnect these devices are responsible that this approach is extremely costly. Additionally a hardware power meter will only be able to display the energy consumption of a whole physical server, regardless of how many virtual machines are actually being hosted on the device under test.

Even though it is somewhat less accurate to use software to estimate the energy consumption, it is a much more feasible approach. Two major methods can be distinguished when using software to estimate the energy consumption. One approach is to install a small piece of software called agent on the server where the energy consumption is to be measured. Although the other approach is called agent-less, it is technically a misnomer, since it implies that no additional software needs to be installed on the device under test. In reality though the agent-less approach simply uses the agents already provided by the operating system.

3.2.1 Agent Versus No Agent

Several myths surround the topic whether an agent-based or an agent-less approach is better suited to measure energy consumption on a server. Kent and Singh discuss the

approaches, advantages, and challenges of technologies using agents versus agent-less ones [KS11].

As already touched upon, agent-less is actually a misnomer, since the supposed agent-less solution requires services on the operating system already available. In most cases these services need to be manually configured. Most important these services need to be able to provide the information necessary and in the desired granularity. An approach using a custom built agent usually gives much greater control and better tuned information gathering.

An agent is better suited to gather data that is only available locally. In the case of power management, an agent is the only way to identify whether useful work is being carried out on a server. Remote methods make it difficult to impossible to identify if sessions are really active. In contrast to an agent-based approach, an agent-less method relies on what the API provides.

An agent-less solution is entirely dependent on network connectivity to obtain any information from clients, whereas an agent will have the ability to act and react independent of the network connectivity. Should a network problem occur, a managing system using an agent-less approach will have to pass on the information from the disconnected server during the time of the disconnect. On the other hand, an agent on a server has the option to act with a certain degree of autonomy. The agent can cache and store the data to process it on the managing system at a later point in time when the network connectivity has been restored. A built in policy can determine and execute actions on the system the agent is residing on without the managing server. On an agent-less system, this is not possible.

Security is another aspect to be considered in this discussion. Although it is not immediately obvious, an agent-less method needs higher access rights from the beginning. The local security policy on each server to be measured needs to be set up such that the managing server can query the information necessary. This means that a on each machine a local account with administrator privileges has to be set up. It is obvious what security risks this opens, should this account be compromised. In contrast to this an agent-based approach only requires administrative rights on the machine it is installed on. Exchange of information is based on sending packages initiated either by the agent itself or by the managing server. No administrative access to the device under test is necessary otherwise. This corresponds to the security principle of only allowing the absolute minimum of administrative rights necessary.

Network configuration also has to be considered with an agent-less approach. Depending on the way the managing system queries the monitored hardware different protocols have to be used. On Microsoft Windows the Windows Management Instrumentation (WMI) would be used. Inbound firewall connections need to be enabled to allow the querying from the managing server. For network devices Simple Network Management Protocol (SNMP) will be used, which again has to be configured correctly before being useful. In an agent-based approach the agent itself initiates communication using HTTP or SSL. As a stateless and ubiquituous protocol HTTP does not require network devices and firewalls to be configured.

Constant querying of a larger number of servers incurs a higher traffic and puts more strain on the network with every monitored server added. An agent-based solution is much more scalable. Data does not have to be sent synchronously and agents are able to locally

process part of the information to enhance scalability. Sending data in batches instead of as a continuous stream allows to wait for times where the network is less utilized by the main services.

The frequency with which data is recorded can also be much higher by a local agent. In an agent-less system a finer tuned granularity automatically means a higher polling frequency, which adds more traffic to the network. A local agent can start collecting data as soon as the system is powered on and does not even need an active network connection.

In an agent-less environment existing protocols will have to be used. In an agent-based approach on the other hand for each targeted set of platforms a platform specific agent is required. The disadvantage can be minimized by employing existing technologies such as Java, where runtime environments on almost all platforms have been established.

Proponents of agent-less methods claim interference with the operating system and running applications by the agents. In fact a small low level agent running in the background listening to operating system events can have less of an effect on a machine than executing a remote query. Each query using agent-less technology executes calls similar to agent-based systems and therefore utilizes just as much system resources as an agent.

The decision to use an agent-based or an agent-less approach in summary is depending on the tasks required to be accomplished. For power management an agent-based approach tailored to the system being monitored is much more suited for the time being. This agent-based approach delivers the best combination of stability, scalability, granularity, and security, unless operating system vendors start to incorporate much better tools into their products.

3.3 Competitors

In many cases a power management software provides a good and helpful way in reducing an institution's power consumption. Reilly et al. provide an overview over some solutions available at the time [RWAJJ11]. Since then this market has grown and the complexity of the solution has increased. The following chapters can be seen as an updated version to this overview.

The following subsections compare and survey several commercially available products as well as non-commercial research projects. These competitors to Stratergia are examined as best possible, given that a lot of the commercially available products tend to be somewhat sketchy or protective about the technologies employed.

Most of the commercial products presented here are positioned as DCIM tools, with the server power metering as an additional service provided. Stratergia regards the power metering and the good accuracy as one of its key strengths.

3.3.1 JouleX Energy Manager

JouleX specializes in monitoring and control of power consumption of computers and associated devices attached to networks. It began its operations with the founding in 2009. Multinational networking equipment corporation Cisco acquired JouleX in 2013 for US $ 107 million, indicating the growing importance of offering a product in this market.

The products offered by JouleX are all based on an agent-less approach. Their main product, the JouleX Energy Manager, monitors energy usage of all network-connected

devices and systems, such as PCs, Macs, thin clients, VoIP phones, printers, servers, network switches and routers, and storage devices. JouleX promises savings of over US $ 300,000 per year for a 5,000 user network. While their website[1] delivers a high number of buzzwords, it fails to mention anything about the methods or the achieved accuracy in measuring the energy consumption.

Figure 3.1 shows a screenshot of the JouleX Energy Manager (JEM), while its technology is explained in four functions[2]:

- **Discovery and Management.** JEM remotely measures energy consumption, costs and carbon emissions for all network-connected devices and systems across the enterprise network, data center and facilities.
- **Assessment and Simulation.** JEM pinpoints energy reduction opportunities by analyzing energy data (consumption, costs, carbon, savings, etc.) by any grouping (date, time, location, device, application, cost center, business unit, etc.). JEM also enables you to simulate energy saving scenarios.
- **Policy and Control.** JEM enables you to create policies to automatically control energy consumption using JEMs time-based, location-based and robust event-based policies. JEMs execution proxies use existing network and systems management infrastructure to automatically control energy usage of devices and systems.
- **Reporting and Decision Support.** JEM provides interactive, drill-down reporting capability. Use JEM to view energy usage, cost, carbon and potential or real-time savings by location, cost center, business unit, device, system or group. JEM also enables cost and usage comparisons over time for specific devices, locations, etc.

The lack of any real information regarding the methods and no claim on accuracy complicate the overall evaluation of the JouleX Energy Manager software. Given that JouleX employs an agent-less approach using continuous polling of network connected devices, the accuracy of the software can range anywhere from 15% to 50% accuracy on the energy consumption estimation. Overall the JouleX Energy Manager solution seems to be much better suited for offices than for data centers. JouleX therefore provides an integrated solution for small to middle sized businesses. In the case of bigger data centers it leaves things to be desired, even though one could argue that it is still better than not monitoring the energy consumption at all.

3.3.2 Viridity VPower

Viridity VPower is a direct competitor to the JouleX Energy Manager. Its agent-less approach is based on the same technologies, all the information gathering happens through polling the networked devices. The company was founded in 2007, with Schneider Electric acquiring all of it's assets in 2011. Schneider Electrics offers DCIM software with their product StruxurWare. The acquisition allowed them to incorporate energy management into their DCIM offering.

[1] `http://www.joulex.net`
[2] `http://www.joulex.net/products-0/technology/`

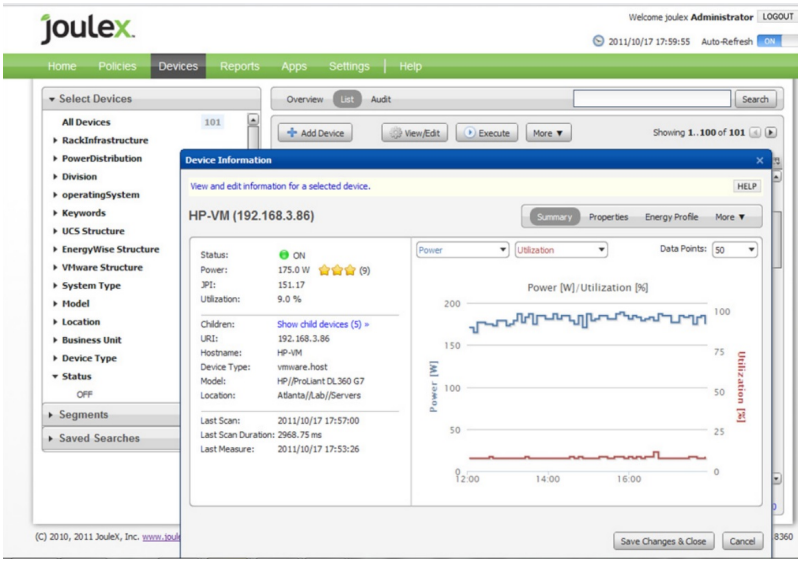

Figure 3.1: Screenshot of the JouleX Energy Manager software. Shown is the detailed summary view of an HP ProLiant server.

The Viridity VPower Energy Monitoring and Analytics product gives insight through the following functions:

- **Energy Dashboard.** With your customized dashboard, you will have quick view of information across all categories. You will see information on current consumption, peak and usage summaries, and an overview of performance corresponding to whatever other programs you sign up for.
- **Energy Monitoring.** Through our Energy Monitoring portal, our clients are able to track real-time generation and load from their utility meter on a five minute or less interval.
- **Energy Trends.** Clients can also examine trends in their usage, providing significant insight into their operational demand and cost of energy.

Viridity claims an immediate improvement of 20% to 40% in data center energy efficiency. Since the technologies employed are based on agent-less methods and no accuracy details are given, it is safe to assume the same power consumption estimation accuracy of 15% to 50%. On one occasion Viridity mentions the nameplate value as the basis of their energy report. We discussed the accuracy of nameplate ratings in chapter 2.6. Based on the information available through their website[3] we are able to come to the same conclusion as with JouleX. A somewhat inaccurate management of energy consumption is still better than having no overview at all.

[3] `http://www.viridityenergy.com`

3.3.3 1E NightWatchman

Based in the United Kingdom, 1E was founded in 1997. The company does not solely focus on DCIM, but has several products on the market. Their energy management product NightWatchman Enterprise manages power use for servers and PCs. It can as well be regarded as a direct competitor to the offerings of Viridity and JouleX.

In contrast to the previously explored competitors, the 1E NightWatchman uses an agent-based architecture. The whole solution is rather targeted to shut down unused office PCs during night time than provide an accurate energy consumption measurement. On a dashboard view the system administrator can get an overview of the savings achieved by shutting down unused machines. However, no claim on accuracy of the energy consumption measurement is given. Even though this solution uses an agent-based approach, we estimate the accuracy of their methods to be on par with their competitors, about 15% to 50%.

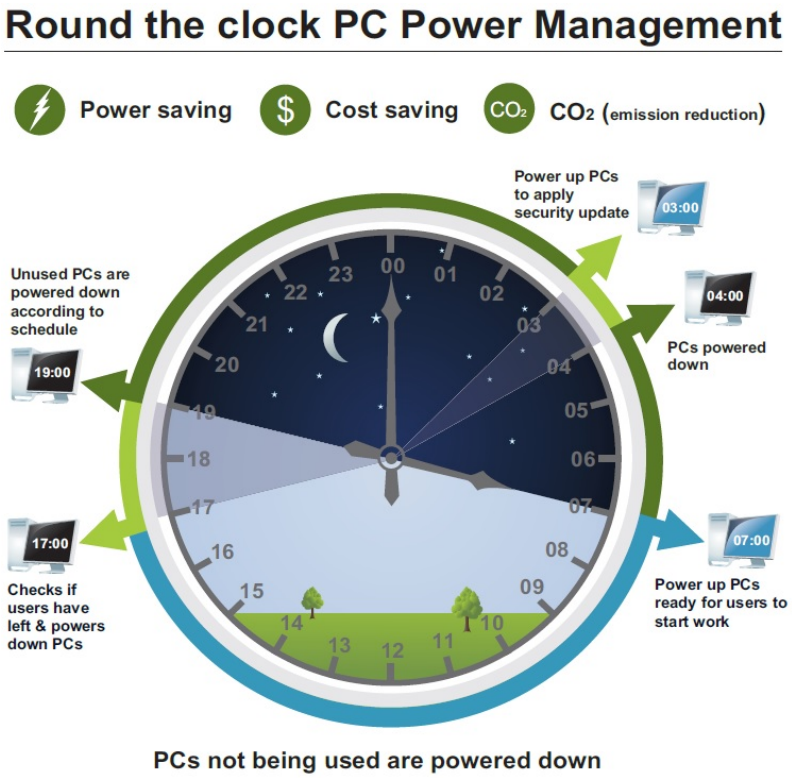

Figure 3.2: PC Power Management plan using the 1E NightWatchman software. Interestingly enough, some modern operating systems already come equipped with the functionality offered by this product, in practice this usually fails to be useful because they are not configured correctly.

According to their website[4], the 1E NightWatchman Enterprise solution is comprised of the following functionality. Figure 3.2 illustrates the PC Power Management according to 1E.

[4] http://www.1e.com

- **Power Management.** Secure power management that is simple to deploy, manage and maintain. Includes ability to sleep and standby, wake with alarm clock, group and report.
- **PC Estate Profiler.** Monitor and report on hardware utilization of all PCs across your estate.
- **WakeUp.** Secure Wake-on-LAN technology for waking machines from shutdown out of hours to apply patches/updates.
- **WebWakeUp.** Enable end users to remotely access a power-managed work PCs outside office hours using a home PC, mobile or tablet.
- **Enterprise Patch Management.** Integration with the patch management process to ensure that patches are successfully applied out-of-band.

The 1E NightWatchman software lays a heavy focus on the office environment, it is less useful in data centers with a high number of servers. This puts it in line with the previously reviewed competitors. It does not seem to be able to provide a claim of accuracy regarding the measured energy consumption. The recommendation here is the same as before, it is still better to have at least some visibility over the power consumption than to be completely in the dark about it.

3.3.4 DataSynergy PowerMAN

DataSynergy[5] is a british company, selling several self-developed solutions for enterprise PC deployment and management. One of their products is the PowerMan Power Manager. This solution complements the built-in PC power management features of Microsoft Windows by providing comprehensive, centralised, configuration of PC power management and web-based, enterprise-wide reporting of PC usage and costs. Figure 3.3 depicts a deployment scenario for this solution.

PowerMAN Power Manager is an agent-based solution, falling in the same category as the 1E NightWatchman. Based on the information available on their website[6] it is clear that it is direct competition to the previously mentioned software. Its method is based on powering down unused PCs during times of low activity or productivity. The software does not seem to take actual energy consumption into account and it is not mentioned how the savings are calculated. Therefore the software does not seem to be at all useful in making accurate predictions or measurements about a server's energy consumption.

3.3.5 Faronics PowerSave

Faronics was founded in 1996 and develops a range of software products for multi-user IT environments. One of their products is PowerSave, released first in 2007. Faronics PowerSave comes in versions for Windows and Mac OS X. The technology behind PowerSave can be seen as agent-based, the software suite has to be installed on all computers to be monitored. Using the Faronics PowerSave Dashboard shown in Figure 3.4 the system manager has an overview over the managed PCs and can get energy consumption reports.

[5] http://www.datasynergy.co.uk
[6] http://www.datasynergy.co.uk/products/powerman

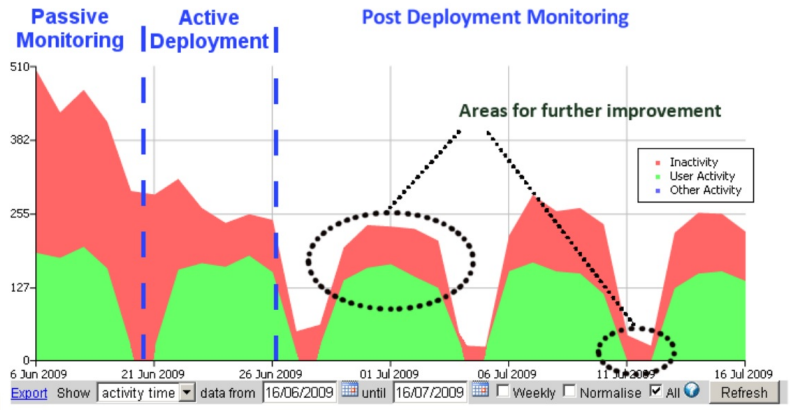

Figure 3.3: A typical deployment scenario for the DataSynergy PowerMAN Enterprise Server. Energy consumption is reduced by powering down PCs during times of inactivity. Savings are based on times of activity or inactivity and not on actual energy values measured.

The Faronics PowerSave competes in the same market as the previously examined products. It offers the ability to employ previously determined power policies according to system utilization. Even though the software calculates savings based on system activity it does not mention anything regarding the accuracy of energy consumption measured. Similar to the previous offerings of other companies this software is more targeted to small to mid-sized businesses using office PCs than large data centers with servers.

3.3.6 Verdiem Surveyor

Privately owned company Verdiem is the producer of PC power management software Surveyor. Surveyor uses an agent-less method to query information from the network controlled devices. Verdiem Surveyor competes directly with systems like JouleX and Viridity, offering basically all of the same functionality. Power management policies have to be determined in the beginning that will then be enforced.

Just like any of the other products reviewed before, Surveyor can be used to automate the software update duties by powering on the PCs monitored during times of inactivity. The software differentiates very little in terms of features compared to the other products. Accuracy of the power consumption of the monitored devices is not mentioned. The savings the company promises are based on standard estimations, giving it the same accuracy of 15% to 50%, putting it in line with the previously reviewed PC power management software. As with all of the previously examined tools, Verdiem Surveyor is much more focused on powering down office PCs in times of inactivity than gathering accurate reports on power consumption, making it more useful in an office environment than a data center.

3.3.7 Competitors Conclusion

After examining the PC power management software presented above the most important question remains unanswered: what advantage does a company gain by installing this

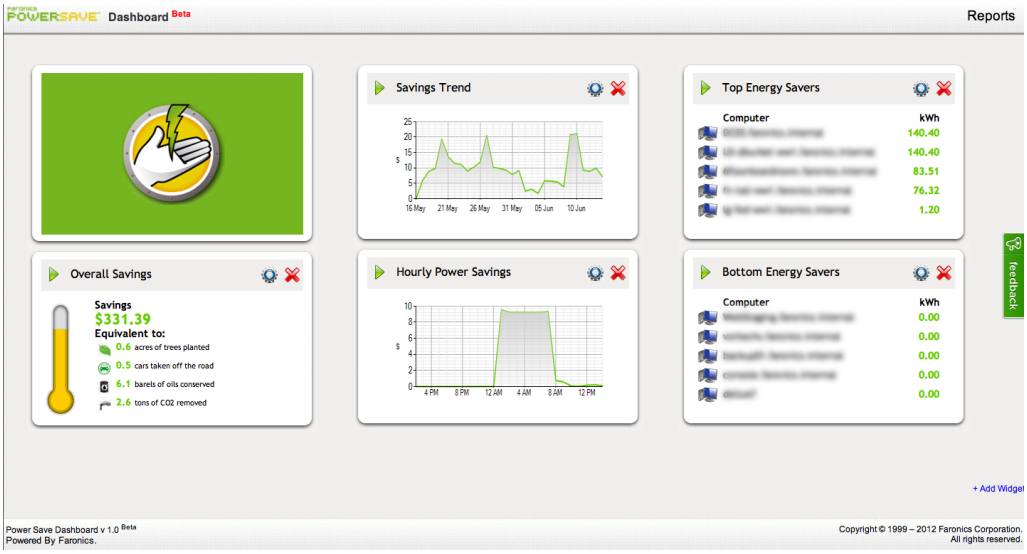

Figure 3.4: Screenshot of the Faronics PowerSave Dashboard. The Dashboard gives insight into the various aspects of power consumption, including savings trends, hourly power consumption, top energy savers, and overall savings.

kind of software versus asking their employees to switch off their computers over night and during times of inactivity? The answer is, that several tasks can easily be automated:

+ Software is more reliable in following a schedule to power down machines when they are not used.

+ PC power management software can enhance productivity by turning on PCs at night to update software instead of having the user wait during their otherwise productive time.

+ A somewhat inaccurate overview of the energy consumption of the IT infrastructure is still better than no overview at all.

+ Most of the products reviewed provide the ability to automatically save documents before powering down the machines.

+ Some of the products allow to remotely wake up the PCs.

However in some instances PC power management software can cause grief for users because it will follow the power policies set regardless of the circumstances for the user:

– Remote login can cause complications when Wake-On-LAN fails to work and the user then has to be physically present to solve the problem.

– The PC power management software might not always be able to recognize system activity as such and shut down a PC doing useful work.

– Failing to recognize non-standard office software or specialized programs with unsaved documents can result in lost work when the PC power management decides to shut down the PC.

For offices the advantages generally outweigh the disadvantages and the installation of such a software is justified except for special cases. But for all that PC power management software is more suited for offices and PCs that have a lot of inactivity than for data centers with a high number of virtual machines and the need of a detailed energy consumption statistic. In addition almost all of the examined software runs on Windows and Mac OS X, whereas Linux is not really considered, which is standard in server and virtual machine environments.

3.4 Methods

Although the topic to estimate the energy consumption on servers using software estimation techniques is fairly new, a good number of methods have already been proposed. The approaches range from very crudely estimating the energy consumption using only the utilization and a linear regression to very intrusive methods with sensor nodes on every hardware consuming energy.

Several steps have to be decided on in order to create a system allowing to estimate the power consumption of a server based on software:

1. **Agent-based or agent-less.** The first decision has to be made depending on the general requirements of the system. As previously mentioned, agent-less will only allow access to already existing services on the operating system and will have to be configured. Agent-based requires the development and deployment of client software, but allows for more freedom and control on the client end of the system.

2. **Select the right system utilization parameters.** Naturally the best parameters to take are the ones that characterize the power consumption behavior of the hardware intended to be described. Some hardware components provide hardware registers to store counts of hardware-related activities such as cache misses or branch misprediction. As for the decision to be as hardware independent as possible hardware counters are not a good choice since they are not always available.

3. **Benchmark the system utilization parameters.** To achieve an accurate estimation of the energy consumption it is essential to cover the dynamic range of the hardware utilization parameters. The best benchmark not only stress tests single components but also simulates real world usage. Usually a number of benchmarks have to be chosen to record the whole range of dynamic energy consumption.

4. **Energy consumption registration.** Recording the energy consumption can either happen on a component-level or on a system-wide basis. This decision has to be made depending on the method employed and accuracy necessary.

5. **Appropriate prediction mechanism.** After benchmarking the system and recording the corresponding system utilization and energy consumption values the prediction method has to be chosen. Several learning algorithms each with their own advantages and drawbacks are available for this step.

6. **Figure of merit or verification.** Although it is optional and most methods forgo this step it is still good to have a way to quantify the method. Two different figures can be distinguished: the quality of the created power model and the accuracy of the prediction without verifying it.

The following subsections take a look on important and interesting work done in this field of research.

3.4.1 Full-System Power Analysis and Modeling for Server Environments

This work by Economou et al. [ERKR06] studies the component-level power breakdown and variation and the temporal workload-specific power consumption of an instrumented power-optimized blade server. Economou et al. introduce a non-intrusive method for modeling full-system power consumption and providing real-time power prediction, called Mantis. After a one-time calibration phase to generate a model by correlating AC power measurements with user-level system utilization metrics, the power consumption can be predicted. The generated models are validated on two server systems with different power footprints [ERKR06].

In order to understand future power consumption trends Economou et al. study the component-level power consumption of a blade server. They obtain the overall server power consumption by connecting a power meter between the system and AC outlet. For the component-level power consumption they organize the blade server into four power planes and measure the voltage and current drop across the different components. The four power planes are as follows:

- A 12V power plane dominated over 90% by the processor and memory.
- A 5V hard disk dominated power budget plane.
- A 5V auxiliary plane.
- A 3.3V plane accounting for network, peripherals, regulators, supplies, and other miscellaneous components.

To measure the system activity Economou et al. used both operating systems performance metrics and hardware performance counters. The OS metrics used were CPU utilization and I/O request rates to the hard disk and network. For this they used the Linux command *sar*[7]. The hardware performance counters used the modules *perfctl* and *perfmon* to provide finer-granularity data for the main memory (off-chip misses).

The benchmarking process in this work uses Gamut[8] (Generic Application eMUlaTion) to emulate applications with varying levels of CPU, memory, hard disk, and network utilization. The calibration phase of the power models consist of an idle run and several different configurations of Gamut stressing the CPU, memory, hard disk, and network.

[7] `http://www.computerhope.com/unix/usar.htm`
[8] `http://http://www.cs.duke.edu/nicl/cod/`

Their linear program relates utilization metrics to power consumption, compiling all of the utilization measurements into a matrix. Mantis relies on a simple linear regression minimizing the absolute error while retaining linearity.

Economou et al. present with Mantis a method to predict the power consumption on servers with the error ranging from 0% to 15% according to their results [ERKR06]. They use performance counters and system activity measurements in combination with a linear regression prediction method.

3.4.2 A Comparison of High-Level Full-System Power Models

Continuing the work of Economou et al. [ERKR06], Rivoire et al. enhance Mantis and provide an updated evaluation on a wider range of server types [RRK08]. The methodology is similar enough to the previously investigated work that we can forgo a discussion. The extended evaluation includes the comparison of five different types of models, which vary in the inputs used and the complexity of the model equations.

The first model simply predicts a constant power C_0 regardless of resource utilization corresponding to the mean power during the calibration suite. This model is valuable both as a baseline for the utilization-based models and it is similar to the method of estimating a system's power based on its manufacturer specifications.

Two other models use only CPU utilization as model input. One model is linear in CPU utilization, yielding an equation of the form $P_{pred} = C_0 + C_1 \times u_{cpu}$. The second model adds an empirical term found to improve accuracy, yielding an equation of the form $P_{pred} = C_0 + C_1 \times u_{cpu} + C_2 \times u_{cpu}^r$, with C_2 and r as additional fitted parameters.

Using disk utilization in addition to CPU utilization in a linear model yields the fourth model. Adding CPU performance counters to the OS-reported CPU and disk utilization gives the fifth and final model.

Evaluated are these models on an 8-core Xeon server, a server with 4 Itanium 2 CPUs, a mobile fileserver with 13 SATA laptop disks, the same with just one disk, and an AMD Turion laptop. Several benchmarks were used namely the SPEC CPU integer and floating-point suites, the SPEC JBB benchmark, the *stream* memory-stress synthetic benchmark, and various I/O-intensive benchmarks. The predicted energy consumption is compared to a wall-plug power meter.

The results of this study are sevenfold:

- **Generally the worst results had the model using the constant energy consumption.** This shows that using the manufacturer provided energy consumption numbers are not reliable in situations with dynamic energy consumption.
- **CPU utilization alone is not enough to accurately predict the energy consumption.** Although the CPU can be considered a first-order proxy for dynamic power consumption, its importance greatly diminished in systems that are not CPU-dominated or where workloads are not CPU-intensive.
- **More information does not always yield a better model.** In cases where a large component of the system's dynamic power is not directly accounted for by the power models a less detailed model may actually yield better results.

- **Performance counters have to be carefully selected.** Blindly applying performance counters will deteriorate the results. If the details of the processors and memory architecture are unknown it is better to do without performance counters.
- **Lack of insight into memory and disk power.** The researchers were able to achieve much more accurate models in CPU-dominated systems. Current CPU performance counters give much more insight into power consumption than their counterparts in disks and memory. More often than not similar high-level interfaces to low-level behavior do not exist for other components. This shows a problem regarding information provided by the hardware itself which is only going to get worse in the future with the rise of systems whose power consumption is dominated by components other than the CPU.

With this study Rivoire et al. give some interesting insight by examining five different full-system power models on several different servers.

3.4.3 Virtual Machine Power Metering and Provisioning

Kansal et al. develop a solution for metering and power capping energy consumption in virtual machines [KZL+10]. The method employed is very similar to the system developed by Economou et al. [ERKR06]. Kansal et al. name three contributions in their work:

1. They present a software named Joulemeter to provide the same power metering functionality for virtual machines as currently exists in hardware for physical servers. Using power models in software they track virtual machine energy usage. They mention explicitly not using any hardware performance counters as an additional challenge.
2. They experimentally evaluate the accuracy of the virtual machine energy metering mechanism using benchmark applications on commonly used platforms. They claim to achieve errors within 0.4W - 2.4W.
3. They show how virtual machine power metering improves power management for cloud platforms. Employing Joulemeter they use power capping in virtual machines to enable significant reductions in power provisioning costs.

The approach taken by Kansal et al. is to track resources used by each virtual machine in software and then convert the resource usage to energy by leveraging the power models of individual resources. This presents two challenges:

1. The resource usage measurements have to be accurate enough in time measurements to distinguish between the energy used by different virtual machines.
2. Resource usage needs to be able to be allocated to a specific virtual machine.

They solve the challenges by creating power models for every component with a high dynamic power profile, namely CPU, memory, and disk. The Hyper-V hypervisor enables tracking of the specific hardware usage by the virtual machines.

For the processor they track active and sleep times. The CPU energy model becomes $E_{cpu} = \alpha_{cpu} u_{cpu} + \gamma_{cpu}$ where α_{cpu} and γ_{cpu} are model specific constant.

For the memory the key factor affecting memory usage is the read and write throughput. Since accurate measuring of this value is strongly dependent on the hardware, they estimate the memory throughput by using the last level cache (LLC) miss counter. The memory energy model therefore becomes $E_{mem}(T) = \alpha_{mem} N_{LLCM}(T) + \gamma_{mem}$ where $E_{mem}(T)$ represents the energy used by memory over duration T, $N_{LLCM}(T)$ is the number of LLC misses during T, and α_{mem} and γ_{mem} are the linear model parameters.

The hard disk power model has the number of bytes read and written as the usage parameters. The disk energy model becomes $E_{disk}(T) = \alpha_{rb} b_r + \alpha_{wb} b_w + \gamma_{disk}$ where $E_{disk}(T)$ represents the energy consumed by the disk over time duration T, and b_r and b_w are the number of bytes read and written respectively during interval T. The parameters α and γ_{disk} are model parameters to be learned.

The full system energy model is the summary of the static and dynamic energy consumption, becoming the following equation: $E_{sys} = E_{cpu} + E_{mem} + E_{disk} + E_{static}$.

Learning the model parameters is done using linear regression with ordinary least squares estimation. The initial observations for the model learning are carried out before any virtual machines are instantiated. For each new virtual machine on a server the linear regression has to be repeated and model parameters are then learned again.

As this work is concerned with predicting power consumption in virtual machines, the verification of the predicted power consumption can not be achieved by comparing it to a ground truth since no physical power meter can be connected. However if the sum of the virtual machine's power consumption equates to the physical server power consumption then it can be assumed that the single virtual machines power consumption is estimated correctly. Kansal et al. claim an error in total power within the range of 1.6W - 4.8W. They expect the error for each virtual machine to be in the 0.4W - 2.4W range. Taking a standard server of about 150W to 250W the mean absolute error would be in the 5% error range.

The method used in this work yields very good results with a promising outlook. Since the method is developed by Microsoft researchers on Microsoft's product Hyper-V hypervisor it has yet to be investigated how these discoveries can be useful on other platforms.

3.4.4 WattApp: An Application Aware Power Meter for Shared Data Centers

In this work, Koller et al. present WattApp, an application-aware power meter for shared data centers that addresses the challenge of dealing with heterogeneous applications in the data center [KVN10]. Their approach is to establish a linear relationship between marginal power and marginal application throughput on a diverse set of enterprise applications and benchmarks.

To build accurate power models they incorporate the number of VMs on the server into the modeling process. This results in the ability to establish that a linear combination of the power models for individual applications can estimate the power drawn by a mix of applications.

Koller et al. name five important requirements for any good power estimation model:

1. **Accuracy.** Since the static power drawn by a server is fixed and does not require a model, the accuracy of a power model captures its ability to predict the dynamic power of a server. The error in accuracy of a power model is defined as the absolute difference between the predicted dynamic power and the real dynamic power, normalized by the real dynamic power in their work.

2. **Usable Parameters.** The input parameters of a power model should be readily available and monitored in server farms. Optimally such parameters require no dedicated instrumentation code, have minimal overhead, and can be collected from user space.

3. **Predictable Input.** In a virtualized data center applications are co-located with each other on a physical server. Live migrating changes the virtual machines depending on availability of the servers and requirements of the applications. Optimally the input parameters do not change with this reconfiguration or can be predicted after reconfiguration.

4. **Speed.** Once generated, a good power model should be able to give an estimate of the power drawn in reasonable time (of the order of a second or less).

5. **Heterogeneity Support.** The power model should be accurate for a diverse set of workloads hosted on the same physical server.

In order to satisfy these listed requirements, Koller et al. explore a power model that takes the throughput (number of jobs executed per second) of an application as input in order to estimate power, as opposed to previously examined application oblivious power models. A calibration run for every application on every type of server the application is being run on is necessary to generate this application throughput based power model. Using the given throughput and a model acquired by simple linear regression learning they are able to predict the power consumption.

The evaluation of WattApp is done using the IBM Active Energy Manager[9] for the blade servers and a power meter for the desktop machine. Koller et al. claim an accuracy of within 5% error of the real power for the applications studied.

3.4.5 VM Power Metering: Feasibility and Challenges

Krishnan et al. use a black box monitoring approach to estimate the power usage of a virtual machine at runtime [KAGS11].

The basic idea behind their VM power metering approach is to establish a power model for the various system resources present on a given platform. This happens by correlating the utilization level of the specific resource to the overall system power, when other types of resources are maintained at extremely low utilization levels. After at runtime the per-VM utilization of various resources is measured. The VM power usage is then estimated based on the power levels corresponding to the appropriate resource utilization.

The total system energy consumption can be expressed by the following equation: $P_{server} = P_{idle} + P_{cpu} + P_{mem} + P_{io}$. The idle power is measured by keeping all cores

[9] `http://www-03.ibm.com/systems/software/director/downloads/v52.html`

and the memory subsystem idle. Krishnan et al. assume the contribution of network I/O utilization to the total system power to be very low. For this reason they focus on the CPU and memory subsystems, showing estimations for benchmarks which are CPU-bound, memory-bound, or combinations of those.

The parameter values taken to measure the utilization on CPU and memory are hardware counter values for instructions retired per second (inst_ret/s) and last-level cache misses per second (llc_miss/s). From these two models are generated, one for CPU and one for memory.

For the evaluation the models are compared to a WattsUp power meter. Depending on the benchmark the error for the predicted value ranges from 5% to 38%.

3.4.6 Evaluating the Effectiveness of Model-Based Power Characterization

The work by McCullough et al. seeks to compare and evaluate several methods [MAC+11]. They present the following contributions to the field of power modeling:

- They show that total system power can be modeled with 1-3% mean relative error across workloads when restricted to a single core and 2-6% mean relative error for multi-core benchmarks.
- They recognize linear models to have a significantly higher mean relative error for individual subsystems such as the CPU: 10-24% error on average, but as high as 150% for some workloads. They additionally employ more complex techniques to improve predictive performance by a few percent.
- They present an analysis of why modeling fails for modern platforms under multi-core workloads. They ascribe this poor prediction performance to effects such as cache contention, processor performance optimizations, and hidden device states not exposed to the OS.

McCullough et al. make a compelling case against simple linear regression models on non-trivial workloads. According to them the poor performance of these models could be explained by one of the following reasons:

- The features being fed into the model contain a certain level of cross-dependency, whereas linear models assume feature independence.
- The features used by previously published models are no longer appropriate for contemporary platforms. There may, however, exist a different set of counters that can still lead to a good model.
- Modern hardware components, such as processors, abstract away hardware complexity and do not necessarily expose all the power states to the OS and are thus fundamentally hard to model since changes in power consumption are not necessarily associated with changes in exposed states.

For their experiments McCullough et al. use the Mantis linear model as a basis [ERKR06]. The linear regression models used are the Mantis linear model and the Lasso regression. The non-linear regression models used are Polynomial with Lasso, Polynomial + exponential with Lasso, and Support Vector Regression.

The evaluation is done on an instrumented Intel Calpella (Centrino) platform and a commercial WattsUp power meter measuring the consumption at the wall. The results are as already mentioned in the range of 1-6%, however in some cases the authors experienced an error above of 80% rendering the models basically useless under specific workloads. Most likely these are due to hidden states not exposed to the OS.

3.5 Comparison

The following tables provide a quick overview of the main features, similarities, and differences between Stratergia Papillon and the previously examined tools and methods.

Stratergia Papillon	**JouleX Energy Manager**
Agent-based	Agent-less
Platform independent	All network connected devices
Not necessary	Supports WMI, WinRM, SSH, SNMP, IPMI, SMASH network services
Supports real-time monitoring	No claim
Not necessary	Based on schedules
Buffers power values in offline mode	No offline monitoring
5% accuracy	No claim
Detects errors higher than 5%	No claim
3rd party software integration possible	No claim
Agent installation necessary	No agent necessary
Application level power visibility	No claim
Virtual machine power metering	No claim
Server power metering software	Complete DCIM Suite for offices

Table 3.1: Comparison of Stratergia Papillon and the JouleX Energy Manager.

Stratergia Papillon	**Viridity VPower**
Agent-based	Agent-less
Platform independent	All network connected devices
Not necessary	Supports WMI, WinRM, SSH, SNMP, IPMI, SMASH network services
Supports real-time monitoring	No claim
Not necessary	Based on schedules
Buffers power values in offline mode	No offline monitoring
5% accuracy	No claim
Detects errors higher than 5%	No claim
3rd party software integration possible	No claim
Agent installation necessary	No agent necessary
Application level power visibility	No claim
Virtual machine power metering	No claim
Server power metering software	Complete DCIM Suite for offices

Table 3.2: Comparison of Stratergia Papillon and Viridity VPower.

Stratergia Papillon	**1E NightWatchman**
Agent-based	Agent-based
Platform independent	Platform independent
Supports real-time monitoring	No claim
Buffers power values in offline mode	No offline monitoring
5% accuracy	No claim
Detects errors higher than 5%	No claim
3rd party software integration possible	No claim
Agent installation necessary	Agent installation necessary
Application level power visibility	No claim
Virtual machine power metering	No claim
Server power metering software	Power management for PCs
No claim	Scheduled system to set energy states of the machines

Table 3.3: Comparison of Stratergia Papillon and 1E Nightwatchman.

Stratergia Papillon	**DataSynergy PowerMAN**
Agent-based	Agent-based
Platform independent	Microsoft Windows 2000 and later
Supports real-time monitoring	No claim
Not necessary	Based on schedules
Buffers power values in offline mode	No offline monitoring
5% accuracy	No claim
Detects errors higher than 5%	No claim
3rd party software integration possible	No claim
Agent installation necessary	Agent installation necessary
Application level power visibility	No claim
Virtual machine power metering	No claim
Server power metering software	Power management for Windows PCs

Table 3.4: Comparison of Stratergia Papillon and DataSynergy PowerMAN.

Stratergia Papillon	**Faronics PowerSave**
Agent-based	Agent-based
Platform independent	Windows and Mac OS X
Supports real-time monitoring	No claim
Not necessary	Based on schedules
Buffers power values in offline mode	Offline monitoring possible
5% accuracy	No claim
Detects errors higher than 5%	No claim
3rd party software integration possible	No claim
Agent installation necessary	Software installation necessary
Application level power visibility	No claim
Virtual machine power metering	No claim
Server power metering software	Power management for Windows and Mac OS X

Table 3.5: Comparison of Stratergia Papillon and Faronics PowerSave.

Stratergia Papillon	**Verdiem Surveyor**
Agent-based	Agent-based
Platform independent	Windows (XP and higher) and Mac OS X (10.5 and higher)
Supports real-time monitoring	No claim
Not necessary	Based on schedules
No claim	Over night update schedule plan
Buffers power values in offline mode	Offline monitoring possible
5% accuracy	No claim
Detects errors higher than 5%	No claim
3rd party software integration possible	No claim
Agent installation necessary	Agent installation necessary
Application level power visibility	No claim
Virtual machine power metering	No claim
Server power metering software	Power management for Windows and Mac OS X

Table 3.6: Comparison of Stratergia Papillon and Verdiem Surveyor.

3.6 Stratergia

Based in the Silicon Valley of Europe, Dublin, Stratergia[10] is an Irish company founded in 2010. Stratergia maintains research centers in several countries, including Austria and Sweden. Their research teams consist of post-graduate and doctorate teams from University College Dublin, Technical University Graz, and the University of Uppsala, having collectively published over 100 peer-reviewed research and technical papers. The teams hold numerous international patents in products and techniques related to the areas of advanced processor design, RFID, and system-level power analysis tools.

3.7 Stratergia Papillon

In collaboration with Compare Test Lab in Karlstad Stratergia has developed a technology to monitor power usage on servers in data centers. Stratergia Papillon is an innovative non-intrusive solution to monitor the energy consumption in real time. Compared to the other offerings reviewed Papillon is less aimed at PC power management in offices, but it targets a market where existing products fail to deliver the promised energy consumption measurement accuracy. Papillon provides the ability to be integrated into an existing DCIM solution to take advantage of the high accuracy of the energy measurement by offering access to an extensive API. Through monitoring a data center with Papillon it is possible to perform the following functions:

- Audit the IT infrastructure of the whole data center for power consumption.
- Devise, implement, and validate a green energy saving strategy for this data center.

[10] `http://www.stratergia.com`

- Identify and quantify the actions that provide the biggest gains with certainty.
- Establish logical and equitable customer or corporate cost centers based on data center usage.

3.7.1 Papillon Methodology

The basis of Papillon are several key technologies including Java, MySQL, Python, and Apache Tomcat. Papillon uses an agent-based client-server architecture to measure the energy consumption of every server the agents are installed on. One master server is set up to which the agents report back after gathering system usage information. On this master server the corresponding power models are stored, for each type of server a tailored power model to guarantee the accuracy of below 5% error. Optionally each agent can be deployed to be responsible for themselves, by basically acting as a master server for their own agent.

Given that the master server has knowledge over the power consumption of all his agents, a web based interface can show detailed reports and results in real time. A data center with Papillon monitoring all servers is equivalent to each server having its own power meter. Figure 3.5 depicts the described architecture of a deployed Papillon system.

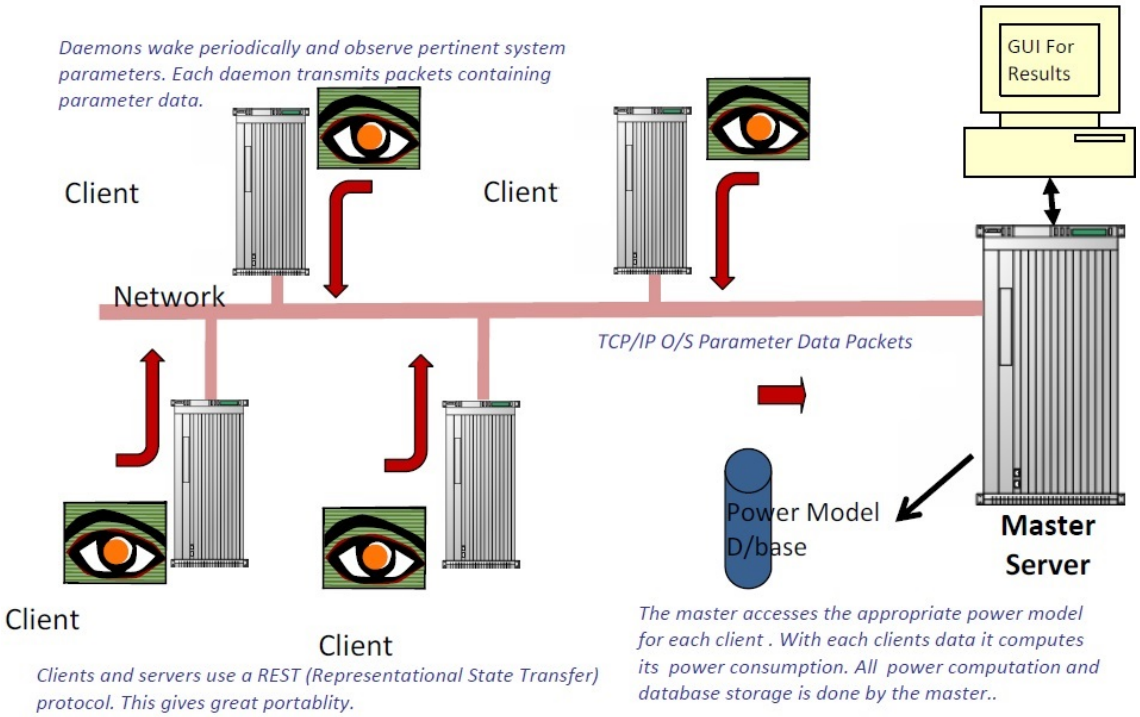

Figure 3.5: Overview of a deployed Papillon system. Each agent with the corresponding power model residing on the master server is equivalent to a connected power meter on each server monitored [Tra13].

The following subsections will take a look at the power modeling method, Papillon master, Papillon agent, and Papillon API.

3.7.2 Papillon Power Models

An accurate estimation of the energy consumption of a server without any hardware meter connected requires a carefully tuned power profile of the server. Therefore a power model based on system utilization statistics corresponding to the hardware has to be generated. This power model can be created once and then used again for every server with the same hardware parts.

The process of power model generation consists of determining and exercising the hardware components mostly responsible for the dynamic power consumption. While the components are being benchmarked system utilization statistics are recorded and stored in combination with power values supplied by a very accurate power meter connected to the server. Several benchmarks are necessary to cover the high range of the dynamic energy consumption of the relevant hardware components. The generated power models are saved as XML files. Each type of server therefore has its own XML file containing the power model. The Papillon master server imports the power models into a database and calculates the power consumption based on the agents reported usage statistics.

Figure 3.6 shows the process of estimating the energy consumption from a generated power model. A more detailed description of the power model generation process follows in the later chapters, as the power model generation is one of the main parts of this thesis.

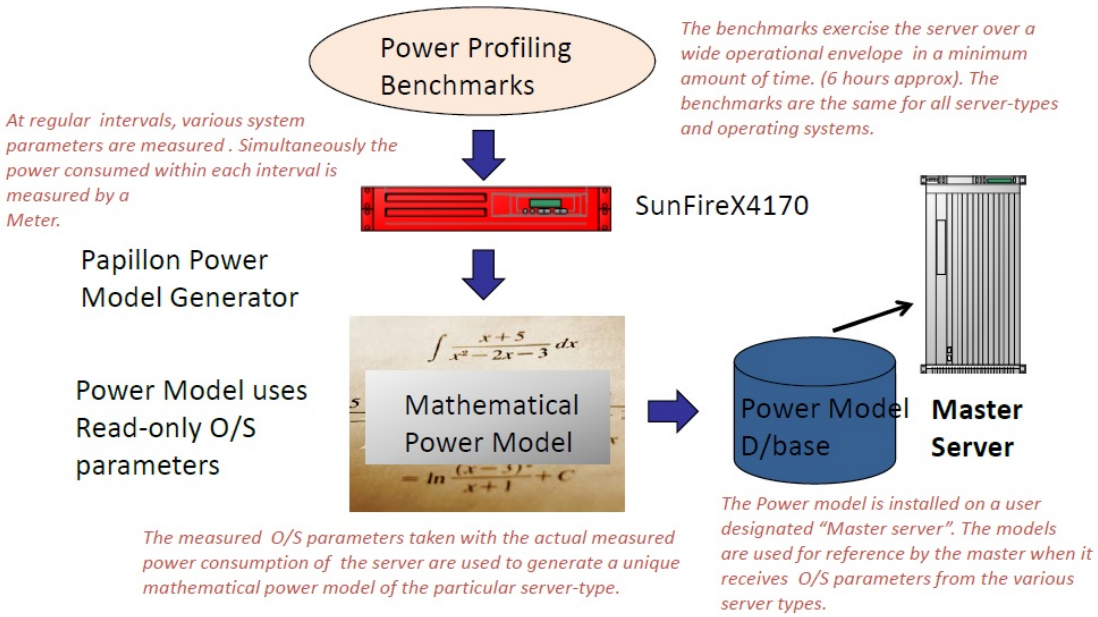

Figure 3.6: Process of power consumption estimation. After exercising the server and creating a power model, the energy consumption can be estimated using the generated power models, usage statistics, and the non-linear nearest neighbors methodology [Tra13].

3.7.3 Papillon Master Server

The Papillon master server includes the whole logic of the Papillon system and has knowledge over the deployed agents. The agents deliver the data they have measured in periodical time intervals, which the master server then uses to estimate the power consumption on each individual server using the previously generated power models. The methodology behind the computation is based on a non-linear nearest neighbor algorithm.

The Papillon master software is programmed in Java, running as a Java servlet on an Apache Tomcat HTTP server. The Papillon master stores all its data in a MySQL database. It provides a Representional State Transfer (REST) able API with methods to get and set data from the master database. The API handles both JSON and XML requests and follows the KISS (Keep It Simple, Stupid) design principle. The API also provides overview over the agents deployed and is able to throw alerts should agents fail to deliver the necessary data in time.

Figure 3.7 shows a diagram overview over the tasks managed by the Papillon master server.

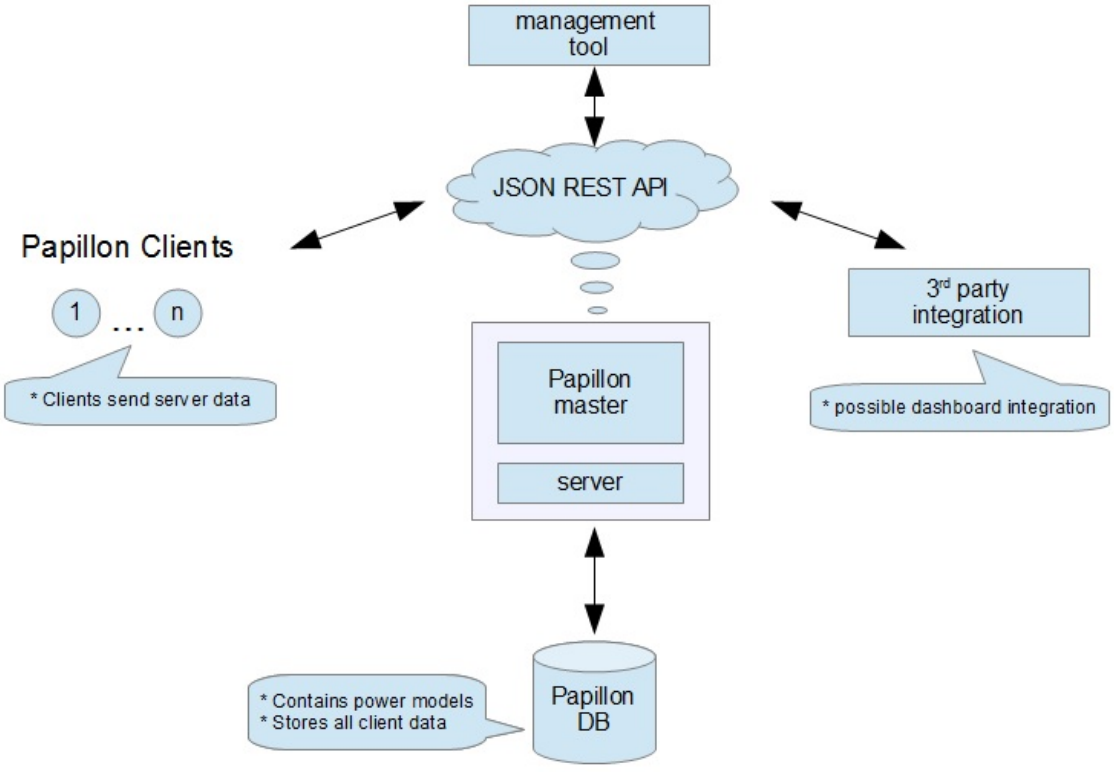

Figure 3.7: Diagram overview over the tasks of the Papillon master server. The master handles communication with the deployed agents, provides integration to third parties via the RESTful API, and holds the power models in a MySQL database [Tra13].

3.7.4 Papillon Agent

The task of the Papillon agent software is to gather the required system utilization information from the server it is deployed on and to report back to the Papillon master. The agent uses a proven and stable Java library called SIGAR[11] (System Information Gatherer and Reporter) to access and record CPU, Disk, and Network usage parameters.

The requirements for such an agent software demand that the software itself has as little impact on the system as possible. Therefore the agent is registered as a service in Windows and as a daemon in Linux. Shifting the processing logic from the agent to the master is instrumental to keeping the agent as light and thin as possible.

Information gathered by the agent is sent via the RESTful API to the master in JSON format. Messages are stored in a buffer in times when the master is unreachable. Therefore the continuous monitoring of the servers is assured without down times even if some component should fail. This can not be guaranteed for systems that rely on a continuously functioning network for querying the system statistics.

3.7.5 Papillon API

A good advantage of Papillon is its easy exchange of information with third party software. Based on the REST principle and the standard HTTP protocol, URLs can be used to send or retreive data in JSON and XML format. The four most used commands supported by the API are POST, PUT, GET, and DELETE.

The base URL to access the Papillon API is as following: `http://localhost:8080/PapillonServer/rest/`

Concatenated to this base URL are then additional parameters to execute various requests from the Papillon API. A URL returning all servers hosted in a data center with DataCenterId 1, on floor 1, in rack 1, would looks as following:

`http://localhost:8080/PapillonServer/rest/datacenters/1/floors/1/racks/1/hosts/`

A more detailed look at the usage of the Papillon API can be found in [Tra13].

[11] `http://www.hyperic.com/products/sigar`

Chapter 4

Design

In order to improve upon the already existing Papillon software and the power model generating process the current implementation has to be analyzed. The main focus of activity during this thesis is the power model generation, the Papillon master and agent software will remain as they are. One of the main goals of this work is the streamlining of the power model creation process and developing a figure of merit for power model accuracy. The following provides a current system analysis to assess the requirements of the new system.

4.1 Current System Analysis

The current implementation of the power model generator tool requires several software packages to be installed. The software is written in several Python scripts, requiring a Python installation on the server that the power model will be created on. The Python scripts generate the model into a SQLite database, which in turn requires SQLite to be installed on the server. After the generation of the power model database file a different software tool written in Python needs to be executed to transform the database file into an xml file.

4.1.1 Power Model Generation Work Flow

A power model generation work flow usually consists of the following steps (visualized in Figure 4.1):

1. Install the required software on the server the power model will be created of. This includes Python and SQLite.
2. Connect the power meter to the server via serial port (RS-232).
3. Update the power model settings Python script with the data of the server and the time period the server will be profiled.
4. Start the power model generation process by running the power model generator Python tool.
5. Subsequently run the benchmarks to stress test the specific hardware.

6. After the power model generation process is finished copy the SQLite database file into the xml generator tool directory and run it to create an xml file containing the power model.

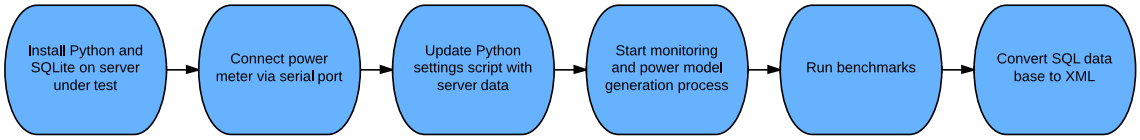

Figure 4.1: Flow diagram of the power model generation process visualized with the currently employed system.

4.1.2 Python

The three main scripts responsible for the power meter test, power model generation, and xml file conversion are written in Python. Python is a general-purpose, high-level programming language. It is designed to enable the programmer to express concepts in very few lines of code and has an emphasis on code readability. Python supports several different programming paradigms, of which object-oriented and functional programming are one of the most popular. Both small and large projects scale well with Python.

The functionality of standard Python can be extended by importing and using a wide variety of modules. Important modules used in the power model generation tool are:

- **serial.** This module is necessary to communicate with the power meter using the RS-232 serial port.
- **os.** Accessing the system utilization parameters is made possible through this module.
- **time.** Since multiple operations of the software are time critical, this module is essential for the functionality.
- **sqlite3.** The power model generation tool creates a SQL database file containing the power model generated.
- **xml.** This module is necessary in order to enable the conversion from the SQL database to an xml file containing the power model.
- **math.** This module provides the necessary mathematical operations for the various calculations.

The Papillon master and agent software are written in Java. Porting the power model generation to Java unifies the structure and removes the dependency on Python.

4.1.3 SQLite

The power model generation process uses the Python tool suite to generate the resulting power model in an SQL database. SQLite is a relational database management system available as a small library. In contrast to MySQL, it is not running as a separate process providing data to the application but instead is embedded into the application. The strengths of SQLite are in the quick and easy setup and development. Its weaknesses are scalability, user management, and lack of performance features. This makes it very suitable for temporary databases such as the one used in the power modeling generation process.

However, since the end result of the power model generation process is a power model that will be sent over the network, portability is an important factor. The power model generator in Python writes a temporary SQL database, after which a second program has to be executed to convert it into a portable and human readable xml file containing the power model. Writing the xml file directly instead of a temporary SQL database file skips an extra unnecessary step and keeps the power model human readable during all times of the power model generation process without the necessity of installing additional software.

4.1.4 XML

The final power model is stored in an xml file. XML stands for Extensible Markup Language, which defines a set of rules for encoding documents in a format that is both human readable and machine readable. Its free and open standard is specified by the W3C[1] (World Wide Web Consortium).

XML emphasizes simplicity, generality, and usability over the Internet as design goals. It is widely used for the representation of data structures in web services, as well as other general documents.

The characters making up an xml file are divided into *markup* and *content*, distinguishable by application of simple syntactic rules. Listing 4.1 shows a sample xml file containing a power model with 3 data points.

The xml essentially contains two parts. First the general information regarding the hardware and the server type description and second a list of data points collected during the power model generation process. One of the advantages of the power model in xml form is the easy integration into a 3rd party software.

[1] `http://www.w3.org`

Listing 4.1: Sample XML file containing a power model with 3 data points

```
<?xml version="1.0" encoding="UTF-8"?>
<powerModel>
  <id>1</id>
  <modelVersion>1</modelVersion>
  <modelType>HP-DL380-G7</modelType>
  <powerMode>AC_DEFAULT</powerMode>
  <name>Sample XML</name>
  <description>General Description: This is a sample xml file
      containing a sample power model.</description>
  <manufactureId>Stratergia Ltd.</manufactureId>
  <cpu>Intel Xeon E5620 Dual Core</cpu>
  <memorySize>16GB</memorySize>
  <diskSize>410GB</diskSize>
  <networkCardSpeed>1GB</networkCardSpeed>
  <powerModelStatisticsList>
    <powerModelStatistics>
      <id/>
      <modelId>1</modelId>
      <power>1.6551</power>
      <stat1>5.97</stat1>
      <stat2>704.0</stat2>
      <stat3>884555.0</stat3>
    </powerModelStatistics>
    <powerModelStatistics>
      <id/>
      <modelId>1</modelId>
      <power>1.6627</power>
      <stat1>6.91</stat1>
      <stat2>412.0</stat2>
      <stat3>4452282.0</stat3>
    </powerModelStatistics>
    <powerModelStatistics>
      <id/>
      <modelId>1</modelId>
      <power>1.6577</power>
      <stat1>6.79</stat1>
      <stat2>2464.0</stat2>
      <stat3>2869556.0</stat3>
    </powerModelStatistics>
    <powerModelStatistics>
    <id/>
  </powerModelStatisticsList>
</powerModel>
```

4.1.5 Power Meter Connection

Connecting the power meter to the server under test is an integral part of the system. In order to ensure correct operation of the communication between server and power meter a tool providing a command line has been developed. The Meter-Test Python software consists of the following scripts:

- **test-meter.py.** This is the main class that provides the command line waiting for user input. User given input is relayed to the Power Meter class and output from the power meter is printed on the screen.
- **PowerMeter.py.** This class provides the communication with the power meter. On instantiation it establishes a connection with the power meter on serial port 0. This uses the Python module pyserial. Several write and read commands can be executed using this class.

4.1.6 Power Model Generator

The power model generation software consists of several Python scripts, each providing a specific functionality:

- **PowerModelGenerator.py.** The main class to be executed. It calls upon all the other classes necessary for the process.
- **PowerMeter.py.** This class provides the communication with the power meter. It is the same class providing the same functionality as in the Power Meter Connection tool.
- **SystemInfo.py.** This class provides the system utilization values. Specifically it reads the CPU usage values from the Linux virtual file system /proc/stat, the disc utilization values from /proc/vmstat, and the network utilization values from /proc/net/dev.
- **PowerData.py.** This class contains the main logic behind the storage and use of the power and system utilization values. It correlates the values from the power meter with the system utilization values and calculates the distance, upon which then a power value can be estimated.
- **XmlWriter.py.** This class contains the necessary routines to output the generated power model to an xml file.
- **PowerModelSettings.py.** It is necessary to modify values in this class before generating the power model. Information about the sample period and benchmark time is to be set up in this file.
- **PowerMeterMock.py.** In case a power meter is not available, this class serves as a testing class providing random power values.
- **SystemInfoMock.py.** In case the Python script is not run on a Linux environment, this class serves as a testing class providing random system utilization values.

4.1.7 System Utilization Parameters

As already mentioned, the methodology to estimate the energy consumption during a certain time interval lies in correlating the system utilization values with a measured power value. The system utilization parameters employed are the the CPU usage, hard disk usage, and network usage. The utilization values are read from the Linux virtual file system /proc.

The /proc virtual file system is a process information pseudo-file system. Instead of containing real files it provides the ability to access runtime system information, such as system memory, devices mounted, hardware configuration, etc. Many system utilities are in fact simply calls to files in this directory. The system utilization statistics used are the following:

- **/proc/stat.** These statistics are kernel and system statistics. These are used to describe CPU utilization. The values taken from here are *user*, *nice*, *system*, and *idle*.
- **/proc/vmstat.** Detailed memory statistics from the kernel are available through this file. The values interesting here are *pgpgin* and *pgpgout*, corresponding to paging in and out from and to disk, essentially indicating I/O activity.
- **/proc/net/dev.** Here is the network device status information contained. The number of received and sent packets, the number of errors and collisions and other basic statistics can be seen. Important here is the number of bytes in and out.

The described parameters are recorded and stored in correlation with the associated power value.

4.1.8 Prediction Algorithm

After correlating the system usage statistics and the metered power value, the power prediction requires some sort of learning method. Two major problem classes can be distinguished in the field of machine learning, classification and regression. Whereas classification is the problem of identifying to which of a set of categories a new observation belongs, a regression analysis is the process for estimating the relationships among variables. Predicting the energy consumption when only the utilization statistics are given therefore equates to solving a regression problem.

Regression analysis is widely used with success for prediction in forecasting and many techniques for carrying out regression analysis have been developed. Probably the most popular ones are linear regression and ordinary least squares regression, which are also used in discussed work in section 3.4.

Contrary to linear regression and ordinary least squares regression, the k nearest neighbors algorithm is a non-parametric method, which can be used for classification and regression. This algorithm predicts values or class membership of observations based on the k closest training example in the feature space. Even though the k nearest neighbor algorithm is amongst the simplest of all machine learning algorithms it performs very well for certain multi-dimensional problems like the one detailed in this thesis.

Using the k nearest neighbors algorithm for regression works by assigning the property value for the object to be the average of the values of its k nearest neighbors. Using weights to control the contributions of the neighbors allows to let nearer neighbors contribute more to the average than the more distant ones.

The prediction of the energy consumption is based on the 10 nearest neighbors algorithm. The distance between the observations are calculated using the Euclidean distance between all features. The weighting of the neighbors follows equation 4.1. This weighting equation has been empirically determined to yield good results. The implementation in section 5.3.2 details the reasons for abandoning this weighting method.

$$weight = (\sqrt{3} - distance)^2 \tag{4.1}$$

4.1.9 Benchmarking

In order to attain a high range of the dynamic energy consumption of the server it is necessary to stress test all the components of the server responsible for the energy consumption.

The benchmarks used to generate the power models are tiotest[2], nbench[3], Phoronix Test Suite[4] with the server options, and SPECpower 2008 [5]. Tiotest stands for Threaded Input and Output Test. It is a file system benchmark especially designed to test I/O performance with multiple running threads. Nbench is designed to expose the capabilities of the CPU, graphics, and memory. The Phoronix and SPECpower test suites evaluate several power and performance characteristics of server class computers. Additionally the system running in idle is captured.

4.2 New Java System Analysis

Having analyzed the current system we are able to determine the requirements of the new system. The new system should not in any way try to reinvent the wheel and reuse the existing code base wherever possible. However this restriction is lifted whenever it poses a disadvantage to the new system.

The system design was focused on the following principles:

- **Encapsulated Classes.** Every part of the software should do one thing only and do it well. It should be easily recognizable from the name of the class what its purpose is. This class should have defined interfaces, making it possible to be reused in other software. Fragmentation and splitting in too many small subclasses should be avoided.

- **Easy maintainability.** Although this principle follows from the first, it is a principle in its own right. Source code should be written in a consistent style. Good documentation has a high priority.

[2] `http://linux.die.net/man/1/tiotest`
[3] `http://www.tux.org/~mayer/linux/bmark.html`
[4] `http://www.phoronix-test-suite.com/`
[5] `http://www.spec.org/power_ssj2008/`

- **Easy expandability.** In the always changing environment of hardware, making the power generation tool extensible is a must. Classes should be designed in such a way that they can be augmented however deemed necessary.

- **Portability.** Power model generation should be as platform independent as possible. Using technology that is available on operating systems covering more than 99% of the market share allows for great portability of the software. This principle includes using as few software dependencies as possible.

While sole purpose of the current system is to record and correlate system utilization statistics and power meter measurements, there are several important features that are missing. The design of the new software should be such that these missing features are implemented and features that are necessary later on can easily be added. Therefore the decisions are as follows:

- Porting the power model generation software to Java provides several advantages. Java is available on every major operating system as well as on numerous embedded solutions. Since the Papillon master and agent software are already implemented in Java, having the power model generation written in Java as well solves the problem of having to maintain the software in two different languages.

- Several important features regarding the power model generation process need to be added to the software. These include a way to resume the power model generation process after canceling for any reason. A functionality missing is to verify or even compare generated models side by side.

- Using Matlab we are able to develop a method to visualize models and quickly experiment with the models without the need to go into the Java source code and recompile. This means porting the main methodology of the power model prediction to Matlab and working there with the generated models.

- Instead of temporarily saving the generated power models to an SQL data base file and then converting it to an xml file, the xml file is now directly written by the power model generator. This avoids an unnecessary additional step.

- In a real use case scenario where there is no power meter present anymore the Papillon system has no means to figure out the true accuracy of the energy consumption estimation. A figure of merit independent from a connected power meter is developed to serve as a means of quality control of the the estimated energy consumption value.

With these decisions in mind a logical view of the new system can be illustrated. Figure 4.2 shows this logical view of the power model generator software. Figure 4.3 shows the work flow diagram of the new system.

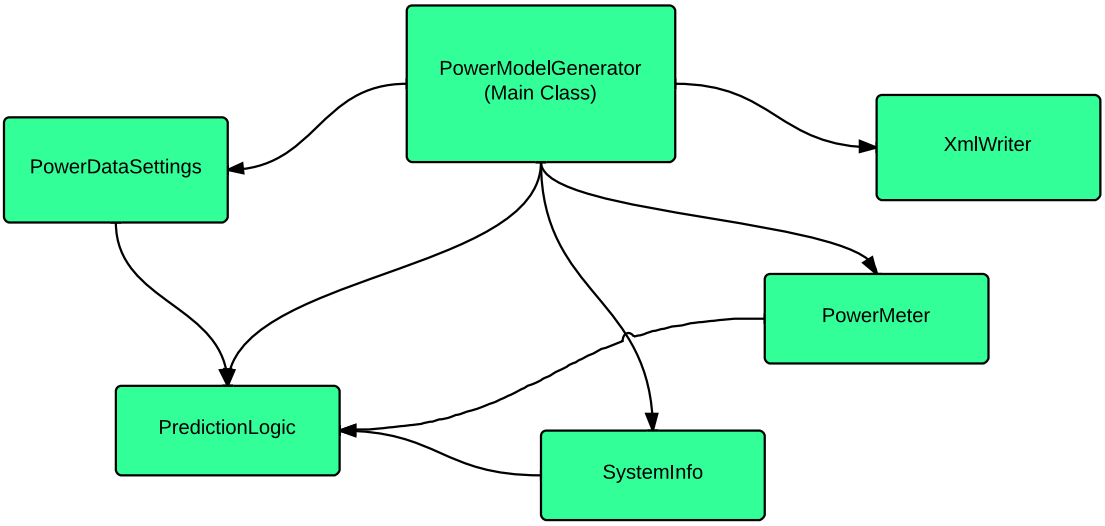

Figure 4.2: Logical view of the power model generator. As defined every class serves a specific purpose. The main class is responsible for controlling the program flow.

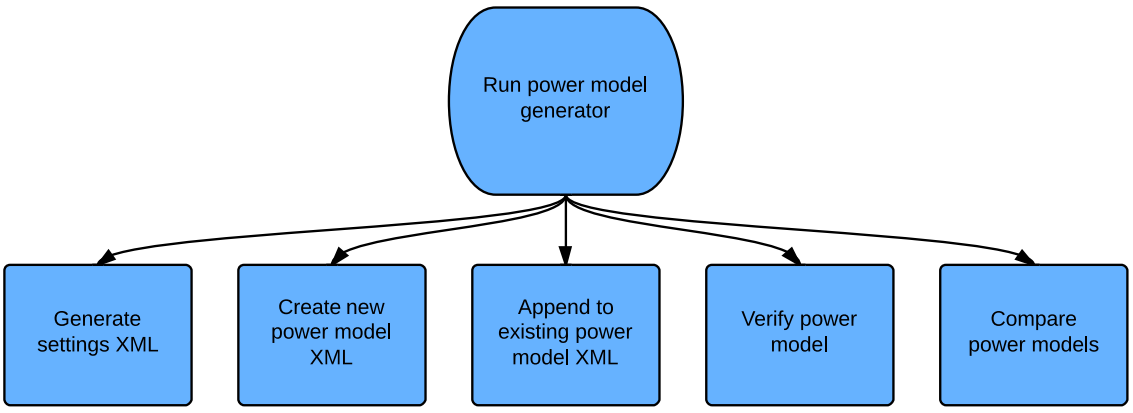

Figure 4.3: Flow diagram of the power model generation process visualized with the new system.

Chapter 5

Implementation

The following chapter details the implementation of the power model generation tool in Java, the visualization method in Matlab, and solutions to problems that presented itself during the implementation.

5.1 Java Implementation Details

The main development was done using the Eclipse IDE with the Oracle Java Development Kit. Eclipse is the standard environment for developing Java applications.

The language Java is object-oriented, class-based, and concurrent. It is designed for applications that follow the "write once, run anywhere" principle. This is made possible by compiling the written software not to machine-specific code but to bytecode that can run on any Java Virtual Machine regardless of the computer architecture. This allows the software to be extremely portable since Java Virtual Machines are available on virtually every operating system.

One of the first challenges faced was the serial port communication to the power meter using Java. While Python has an official module allowing for easy communication and control of the serial port, Java's official serial port library has been discontinued by Oracle. We reviewed several Java serial communication libraries and after a meticulous testing process found two libraries that were not either unstable or did not provide the required amount of features. These two libraries are the jSSC[1] (Java Simple Serial Connector) and serial-comm[2]. We finally settled for the jSSC solutions since this project is still actively developed and the user base is higher.

Java Simple Serial Connector

jSSC supports a wide array of operating systems, including Windows 32-bit (Windows 98 to Windows 8), Windows 64-bit (all versions), Linux (x86, x86-64, ARM), Solaris (x86, x86-64), Mac OS X 10.5 and higher (x86, x86-64, PPC, PPC64). This is essential in keeping the power model generation process as platform independent as possible.

[1] `https://code.google.com/p/java-simple-serial-connector/`
[2] `https://code.google.com/p/serial-comm/`

5.1.1 Power Meter Connection

The first step was to port the power meter connection test command line tool. The functionality of this tool is to relay commands from the computer to the connected power meter and wait for the response of the power meter (if there is one). This tool is also used to test which of the Java serial port libraries provide the most stable experience. The class diagram in Figure 5.1 depicts the structure of the software.

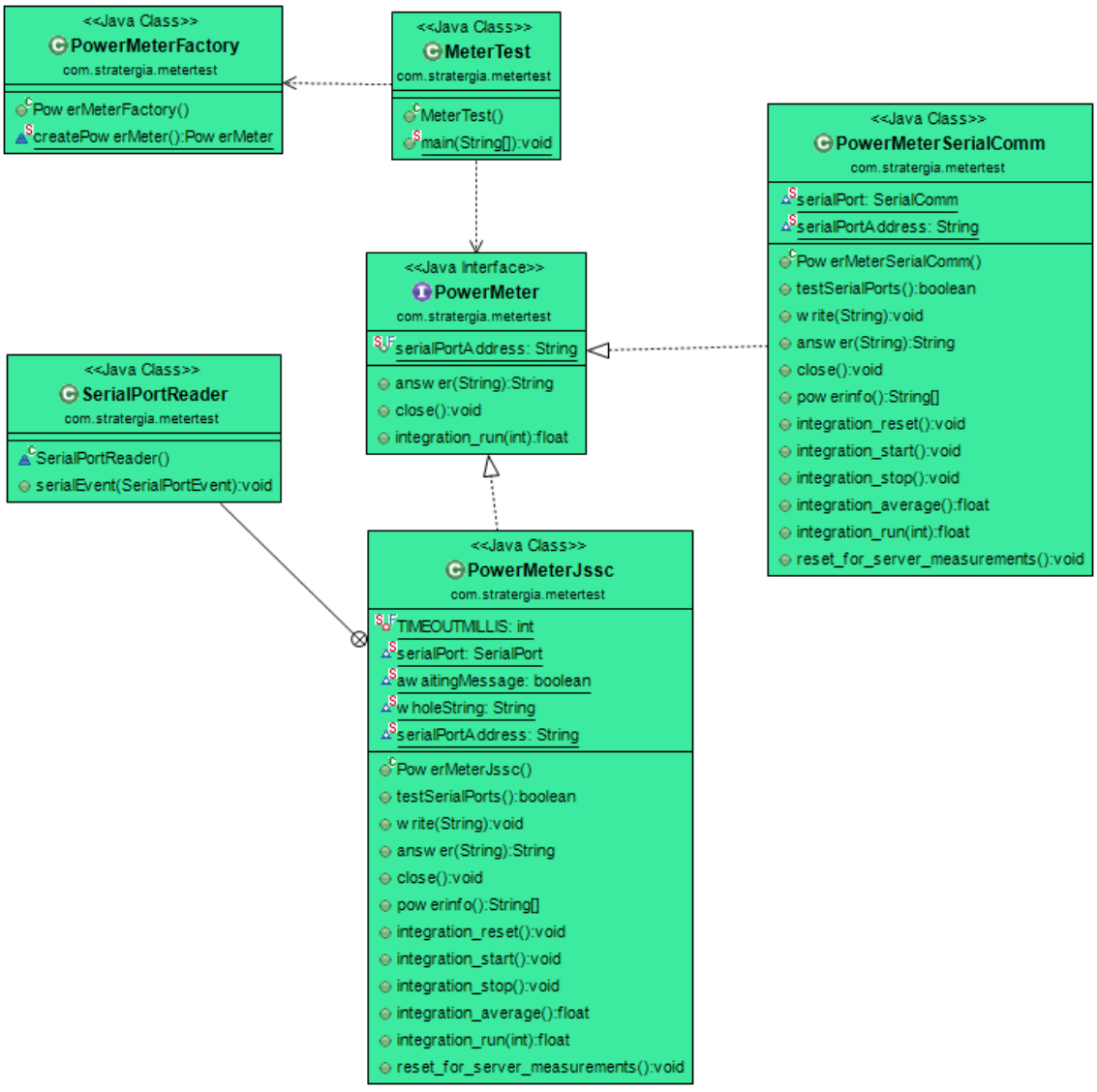

Figure 5.1: Class diagram of the MeterTest power meter connection test utility.

Upon porting the software from Python to Java all of the original features were retained and further additional functionality was implemented. Instead of failing to function when the power meter was not plugged into the right port, the new software now scans all the serial ports available on the system. It automatically selects the right port where the power meter is connected on. If no power meter is connected at all, it returns the user

that no power meter could be detected and the connection should be inspected.

The power meter can be addressed using a variety of different commands, some of which request an answer while others simply issue an instruction to be carried out. The new implementation can now differentiate between them using two different write commands. Using *write* the command will be relayed to the power meter without expecting any return value at all, *answer* writes the message to the power meter and expects a response from the power meter. In case the power meter does not return any data a time out gives the user feedback that the device did not respond.

5.1.2 Power Model Generator

The main functionality of the power model generator software is to record and correlate system utilization statistics and measured energy consumption values from the connected power meter. This part of the functionality was already available in the old Python scripts and has been carried over to Java with the new implementation. Additionally previously missing features have been added:

- Before executing the power model generation process certain settings need to be specified. Instead of editing settings in code, they are being read from an xml file in the directory called settings.xml. To enable easy handling of the software even without knowledge of the underlying structure, the power model generator tool lets the user know that it requires this file, should it not be in the directory. If this is the case, the user is presented with the option to automatically create a sample xml file, that the user then will have to modify. The power model generator then only allows to proceed with the process if the settings.xml exists and is correctly configured.

- With an existing settings.xml the user is given a choice of options. The first is to create a new and therefore overwrite an existing power model xml file in the directory if one already exists.

- If a power model xml file is already present in the directory, the user also has the option to continue the power model generation process. This will append the data to the existing power model. The power model generation process can be interrupted or exited at any point in time without data loss, since the power model file is written to the hard disk after every new data point.

- If the power model generation process is completed and no new data needs to be collected for the power model, the user has the option to verify the generated power model. This option takes the existing power model as an input and compares its estimated energy consumption to the true value measured by the connected power meter. This is an option to manually verify the quality of the power model.

- The last option is an experimental feature which does not require a physical power meter to be present, but two power models. One power model is assumed to be the perfect power profile of the server and takes the role of the training data, while the other power model is to be tested. This option does a cross correlation of the test power model on the training data and is available for development purposes on power model research.

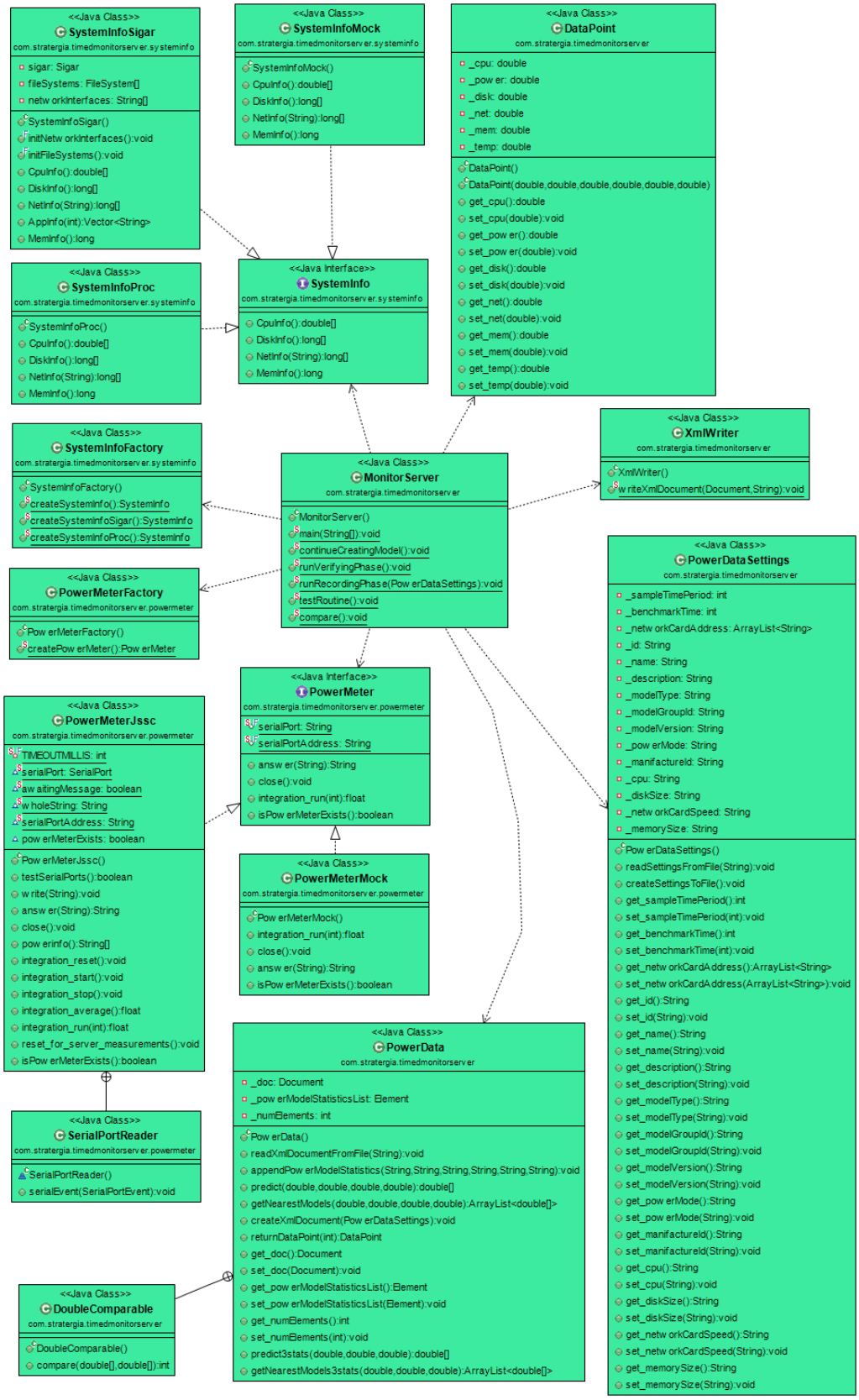

Figure 5.2: Class diagram of the Power Model Generator Java software.

While these are the main improvements in terms of functionality available with the power model generation software, improvements have also been made in terms of portability of the software. To be able to create power models on operating systems that do not provide a /proc file system to access system information statistics a solution was necessary to gather system information independent from the operating system.

Hyperic Sigar

Hyperic is a company specializing in systems monitoring, server monitoring, and systems management software.

SIGAR[3], short for System Information Gatherer and Reporter, is a cross-platform API to collect data about a system using the operating system's built in methods. The following operating systems are supported by SIGAR:

- Linux
- FreeBSD
- Windows
- Solaris
- AIX
- HP-UX
- Mac OS X

SIGAR provides an extensive API to access and monitor data including the following:

- System memory, swap, CPU, load average, uptime, logins.
- Per-process memory, credential info, state, arguments, environment, open files.
- File system detection and metrics.
- Network interface detection, configuration information and metrics.
- Network route and connection tables.

The advantage of using SIGAR is that it enables the software to become highly operating system independent, while still maintaining a very good level of accuracy regarding the system utilization statistics.

Initially our efforts were focused on OSHI[4] (Operating System and Hardware Information), which aims to provide the same functionality. We deemed this library as not mature enough for our requirements.

Also retained from the original power model generation Python scripts was the possibility to use mock system information and mock power meter classes to test the system. Given the fact that several options are available to choose from when picking the SystemInfo and PowerMeter classes (refer to Figure 5.2) we decided to use a design pattern to facilitate the development.

[3] `http://www.hyperic.com/products/sigar`
[4] `https://github.com/dblock/oshi`

5.1.3 Factory Method Design Pattern

The Factory Method design pattern is an object-oriented creational design pattern that deals with the problem of creating objects without specifying the exact class of object that will be created. In our case this relates to two types of objects: the SystemInfo class providing system utilization statistics and the PowerMeter class providing the measurements from the power meter.

The main point of this design pattern is to define an interface for creating an object, but let the classes that implement the interface decide which class to instantiate. Therefore the factory method lets a class defer instantiation to subclasses [JHVG95].

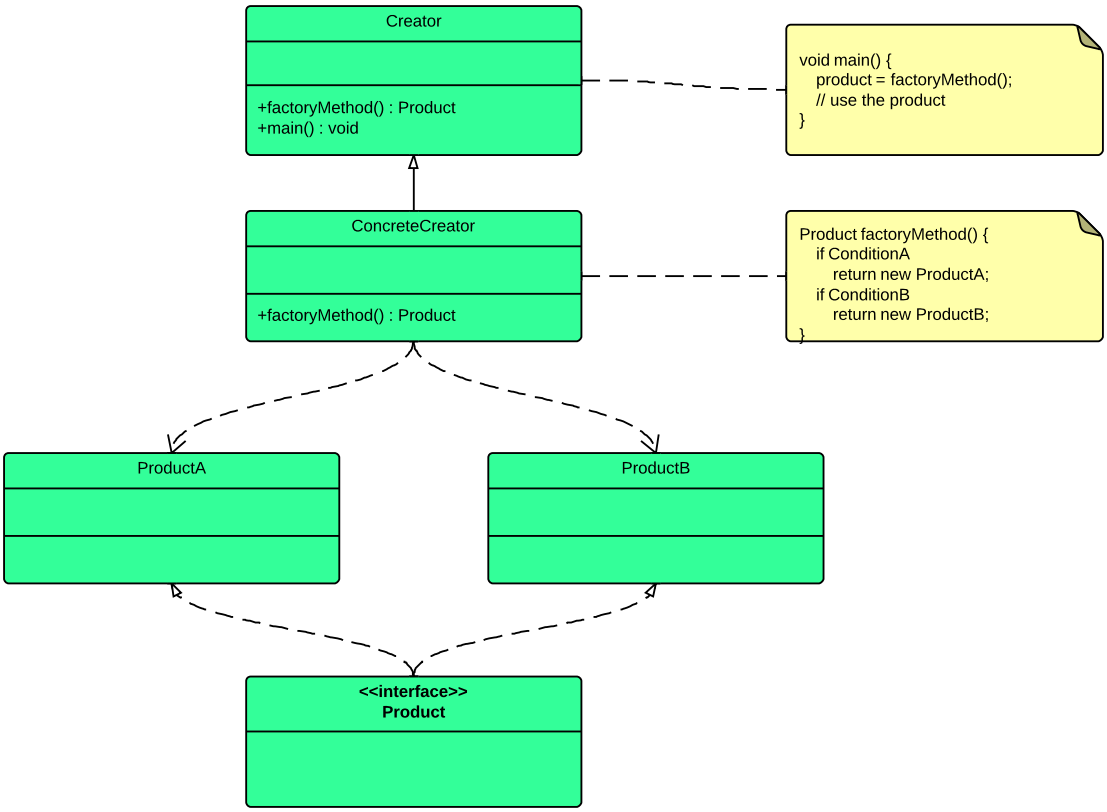

Figure 5.3: UML diagram of the factory method design pattern.

The usage of the factory pattern is usually a good idea for the following three cases [JHVG95]:

1. The creation of an object makes reuse impossible without significant duplication of code.
2. The creation of an object requires access to information or resources that should not be contained within the composing class.
3. The lifetime management of the generated objects must be centralized to ensure a consistent behavior within the application.

For us the first two use cases are interesting. Further we benefit from being able to switch between the classes easily without changing code but by simply being able to instantiate the correct object at runtime. Figure 5.3 shows the design pattern's UML diagram, while Figure 5.4 shows the class diagram of our implementation of the design pattern.

The concept of this design pattern works as shown in the UML diagram in Figure 5.3.

- **Product** defines the interface for the objects the factory method creates. This interface shows the functions and attributes the classes will have to implement.
- **ProductA** and **ProductB** are concrete objects (ConcreteProduct) that implement the Product interface.
- **Creator** (often referred to as Factory because it creates the Product objects) declares the method factoryMethod(), which returns a Product object. It may or may not call the generating method for creating Product objects.
- **ConcreteCreator** overrides the generating method for creating ConcreteProduct (ProductA and ProductB) objects.

This design pattern is used twice, once for the SystemInfo and once for the PowerMeter classes. The classes *SystemInfo* and *PowerMeter* correspond to the *Product* interface. *ConcreteProducts* are *SystemInfoProc*, *SystemInfoSigar*, and *SystemInfoMock* for the SystemInfo and *PowerMeterJssc*, *PowerMeterMock*, and *PowerMeterSerialComm* for the PowerMeter. The *Creator* is the *MonitorServer* for both patterns and the *ConcreteCreator* is *SystemInfoFactory* as well as *PowerMeterFactory*. This is visualized in the Figures 5.1 and 5.2, but probably best illustrated in Figure 5.4.

The benefits of using this design pattern are the easy switching between real PowerMeter and SystemInfo classes and the mock versions of them. Additionally the software only needs one universal version to be compiled which then can run on any operating system, automatically switching between the appropriate system utilization statistics libraries.

5.2 Matlab Implementation Details

While the generation of the power models is done using the Java software presented in the previous sections it is not possible to verify or visualize the generated models in an easy and sensible manner. Experimenting with the energy consumption prediction algorithm requires changing the source code of the power model generator and recompiling the software, turning even the smallest changes in the source code into a cumbersome task.

For these reasons we developed a new method in order to to visualize the power models and implemented the prediction algorithm again in Matlab. Matlab is very well suited for these tasks as it is a numerical computing environment allowing matrix manipulations, plotting of functions and data, and implementation of algorithms by default. Additional packages can be used to extend the capability of this fourth-generation programming language.

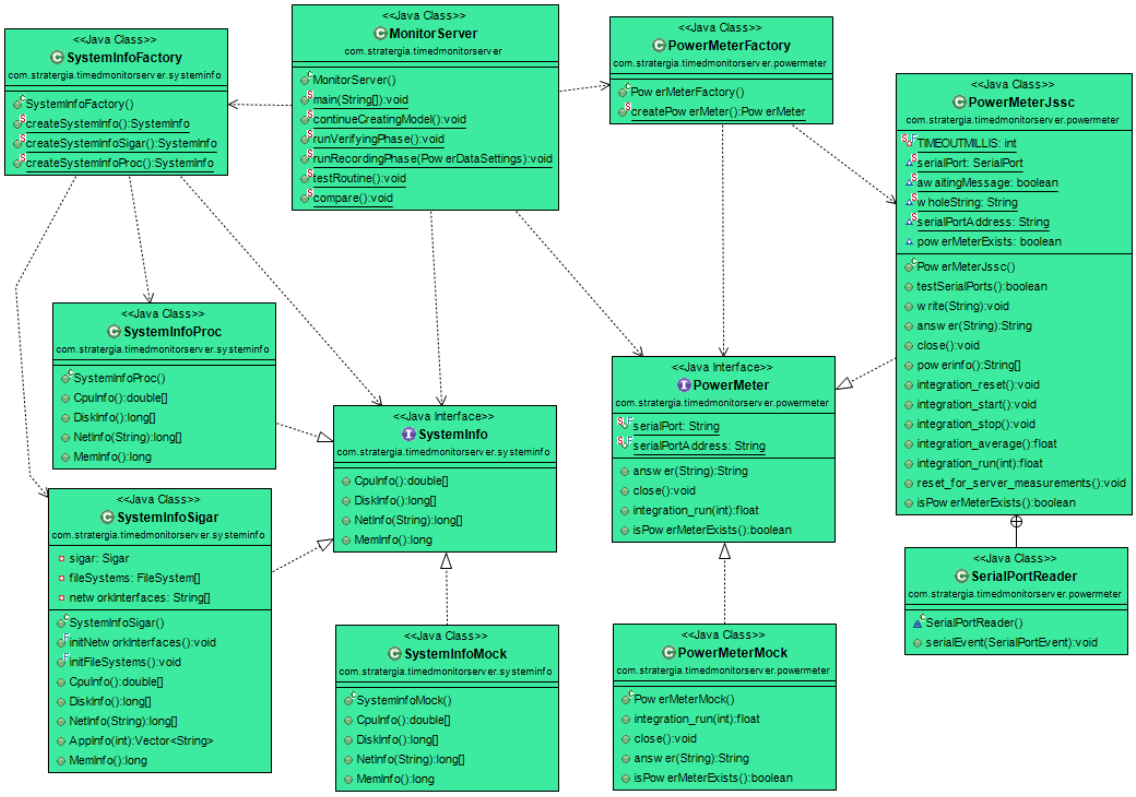

Figure 5.4: Class diagram of the factory method design pattern implemented in the power model generator.

With the same principle in mind to write encapsulated scripts and reuse them whenever possible we generated the following Matlab scripts and functions:

- **readXmls.m.** The purpose of this script is to read the power model from an xml file and import it into a matrix into Matlab.
- **readXmlVerification.m.** Mainly a test script, this is to verify that the Java implementation and the Matlab implementation do not differ in the prediction of the power value.
- **calculateVisualizationkNN.m.** This function takes as input a power model from an xml file, the amount of the nearest neighbors the prediction will use (standard is 10) and the granularity of the grid the visualization will use. The output is a 3D model of the power model for easier visibility.
- **call_visualization.m.** This script specifies which power model to use and the settings with which the visualization will be executed.
- **call_visualization_multithreaded.m.** Since the computation of the visualization is a very CPU intensive task, this script takes advantage of the matlabpool command and uses multiple CPU cores (if available), speeding up the process.

- **knn_LOO_crossvalidation.m.** This is the main script used to develop the figure of merit and test and verify different methods regarding the prediction algorithm. The input is either one or two power models, with which either a cross-validation or leave-one-out cross-validation is performed.

- Several other scripts were implemented serving different purposes, e.g. a prediction using regression, support vector machine regression, showing the nearest neighbor data points for a specific point, etc. Given that the visualization Matlab script is the main focus, these scripts will only be superficially touched upon.

5.2.1 Power Model Visualization

A created power model is essentially a long list of recorded data points. It is very hard to review the quality of the generated power model manually. For this reason we developed a visual 3D representation method of the power model.

The Figures 5.5, 5.6, and 5.7 show one power model. These Figures were taken from three 3D model representations, which allow for easy 3D manipulation to easily inspect the model from all sides.

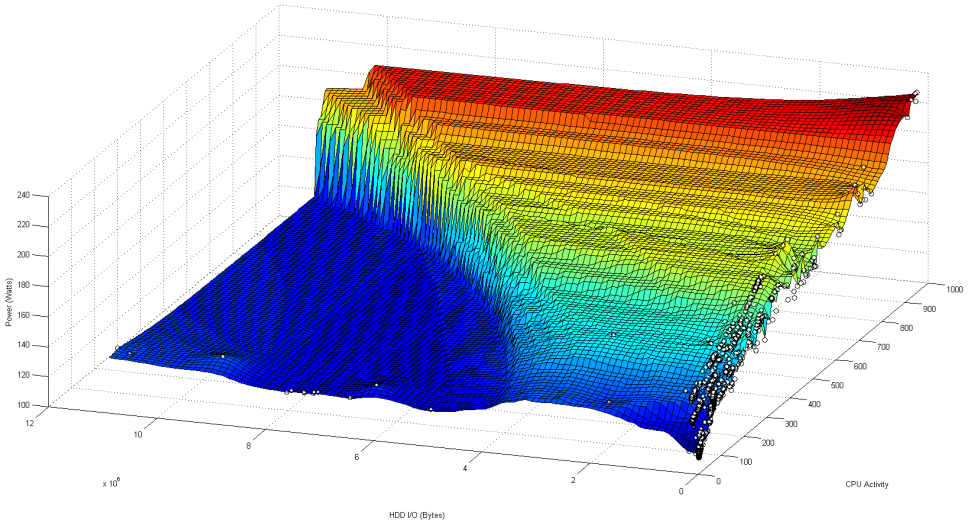

Figure 5.5: Sample power model visualization HDD/CPU. Visible on the x-axis is HDD I/O utilization (bytes), on the y-axis CPU utilization (percentage times ten), and on the z-axis the power consumption (Watts).

Depending on the number of parameters, a power model has a higher dimensional feature space. Using the three parameters CPU, HDD, and network gives the power model a 4D feature space with the power consumption measurement. This requires three 3D visualizations of the power model. Each representation omits one of the parameters to show the two other parameters in relation to the corresponding power measurement.

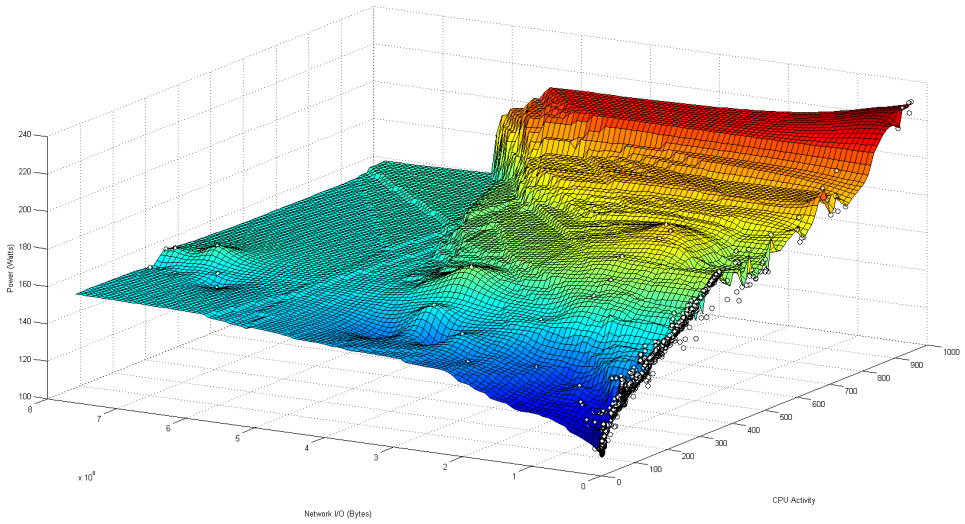

Figure 5.6: Sample power model visualization NET/CPU. Visible on the x-axis is Network I/O utilization (bytes), on the y-axis CPU utilization (percentage times ten), and on the z-axis the power consumption (Watts).

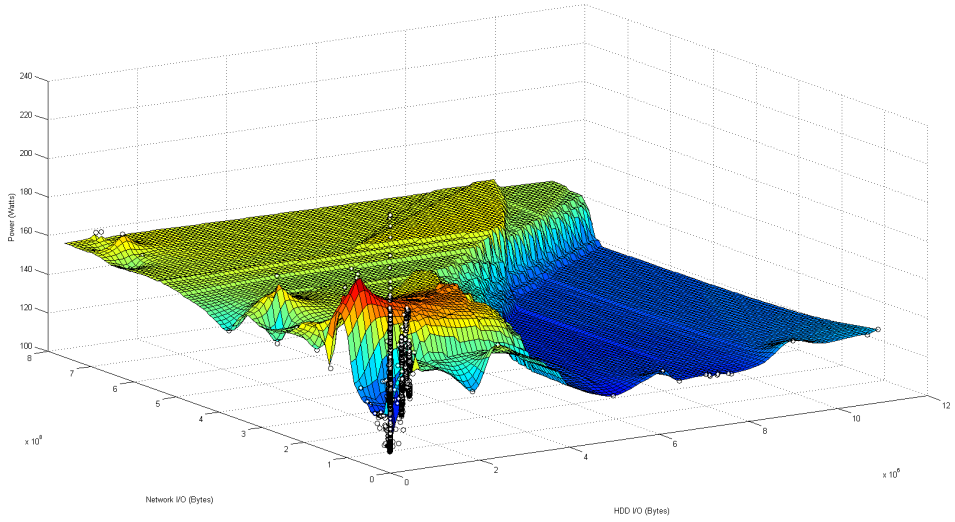

Figure 5.7: Sample power model visualization NET/HDD. Visible on the x-axis is Network I/O utilization (bytes), on the y-axis HDD I/O (bytes), and on the z-axis the power consumption (Watts).

Using this representation a few interesting observations can be made:

- **Coverage of the power model.** Looking at the positioning of the data points in the model gives an overview over the areas the server usage is well covered. Areas with little to no data points either point to no usage or very little coverage of the benchmarks.
- **Server energy consumption profile.** While the power model itself already characterizes the energy consumption of a server, using this representation gives a better overview over the dynamic versus static energy consumption of a server.
- **Energy consumption of the parameters.** Even though each of the power models representations necessarily omits one or more parameters it can be deduced from the remaining parameters in which situation which of the parameters contribute to the energy consumption.
- **Comparison between power models.** In addition to comparing the models using the mean absolute error the power models can be visually compared.

The visualization is intended as an additional way of identifying the quality of a power model. It is not intended to be the sole measure of the quality of a power model.

5.2.2 Power Model (Leave-One-Out) Cross-Validation

As previously mentioned we also implemented a way of quickly testing the generated power model such that the parameters of the learning algorithm could be changed without compiling the power model generator over and over. To test the models we used leave-one-out cross-validation.

Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. Its main purpose is in settings where the goal is prediction and an estimate on how accurately a predictive model will perform in practice is needed.

The Matlab script has two methods of performing the analysis. Using two models, one of them is used as a training model whereas the other is the testing model. With only one model a leave-one-out cross-validation is performed, with each data point of the model serving as a test data while the other data points serve as the training data.

Using this method several parameters can be compared via the mean absolute error.

5.3 Methodology

While the goal was to keep the simplicity of the existing software several improvements were made during the implementation of the power model generator. A figure of merit was developed to gain a form of quality control without a connected power meter. A change in the prediction algorithm corrects a potential source of errors. Additional benchmarks increase the range of the coverage of the power models.

5.3.1 Figure of Merit

Since the accuracy of an energy consumption prediction can not be verified without a connected power meter it is generally very hard to assure that the predictions stay within a certain accuracy. After several dead ends we developed a method to detect energy consumption predictions that have a high probability of not being within the required error accuracy.

The idea behind this figure of merit is based on the fact that a distance between each data point in the power model's feature space can be calculated. Several computed values were considered regarding this figure of merit, e.g. standard error, standard deviation, relative standard error, relative standard deviation, accumulated distance between data point, relative standard error times accumulated distance, etc.

The standard deviation of the power value of the 10 nearest neighbors turned out to be the best and most reliable indicator for the accuracy of the energy consumption prediction. The standard deviation of the power value calculates as shown in Equation 5.1. The calculation for the mean power value is shown in Equation 5.2.

$$PowerValueStandardDeviation = \sqrt{\frac{\sum_{i=1}^{k=10} power_i - mean}{k}} \tag{5.1}$$

$$mean = \sqrt{\frac{\sum_{i=1}^{k=10} power_i}{k}} \tag{5.2}$$

The threshold for power value standard deviation has been empirically determined to be under 0.15 for accurate estimations. Table 5.1 shows the tabularised relations between good (error below 5%) and bad (error above 5%) predictions tested on a real world power model.

	Correctly identified	**Wrongly identified**
Identified as below 5% error	440 (91.5%)	41 (8.5%)
Identified as above 5% error	63 (37.9%)	103 (62.1%)

Table 5.1: Type I and type II error relations of the figure of merit. The power model consisted of 647 data points.

5.3.2 Prediction Algorithm

Most of the energy consumption estimation algorithm did not need any changing from the design to the implementation. However we improved upon the existing 10 nearest neighbors method by changing the weighting of the neighbors.

The weighting distance previously used and shown in Equation 4.1 turned out to be a possible source of errors. The weighting of the data point is decreased with increasing distance up to $distance = \sqrt{3}$. However distances further away than this have their weighting increased by a quadratic factor. Figure 5.8 shows the old weighting versus distance in a plot, whereas the new weighting versus distance plot is shown in Figure 5.9.

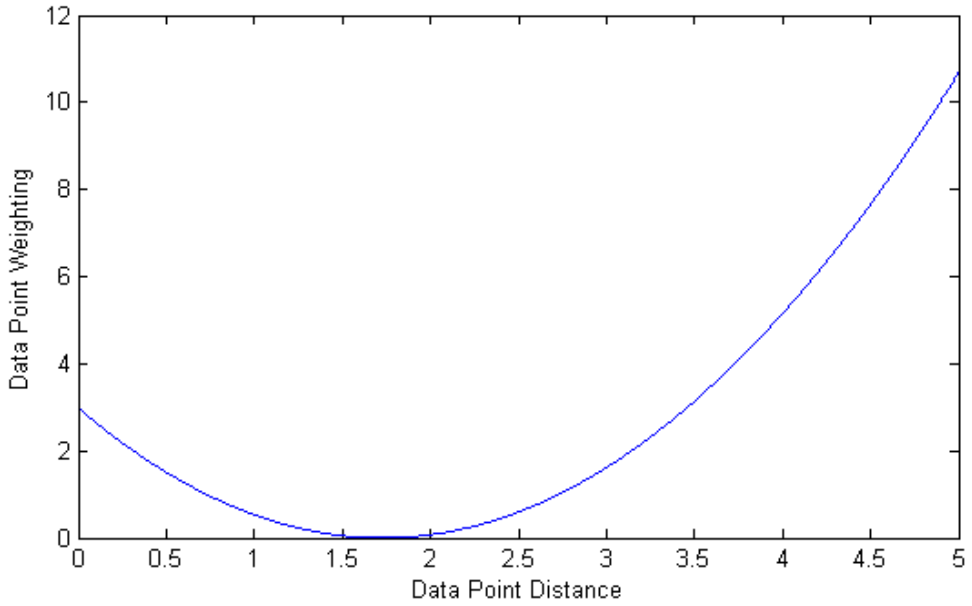

Figure 5.8: The weighting is decreased with increasing distance up to $distance = \sqrt{3}$, after that the weighting is increased with increasing distance.

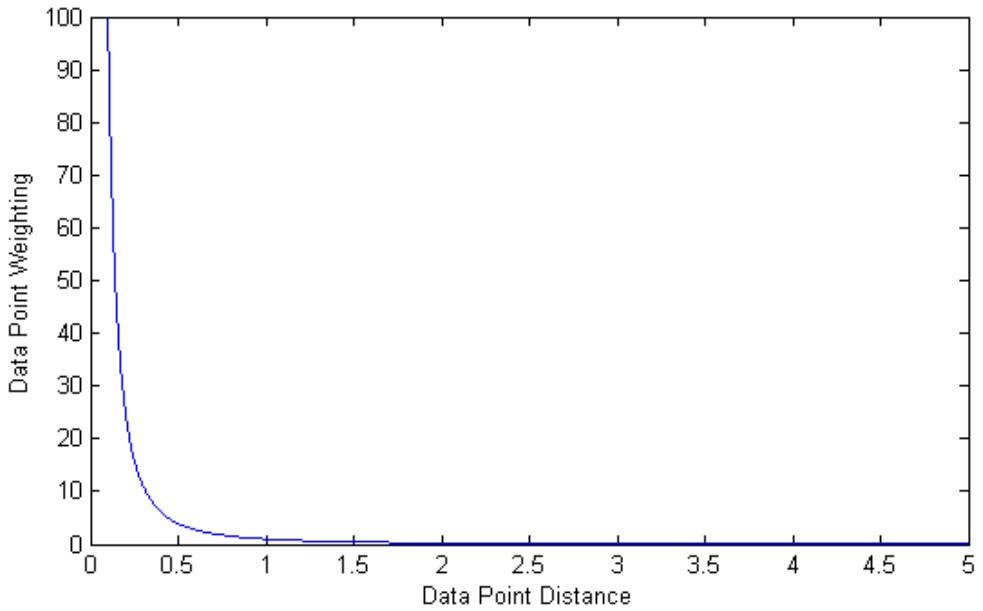

Figure 5.9: The weighting is decreased with increasing distance. Only $distance = 0$ requires special attention.

Fortunately given enough data points the distance between the 10 nearest neighbors never crossed or even came close to that boundary. However in order to prevent future errors we switched to the better alternative of computing the weighting shown in Equation 5.3. This only required the treatment of the special case where $distance = 0$. In all other cases an increasing distance results in a decreased weighting.

$$weight = \frac{1}{distance^2} \tag{5.3}$$

We did not change the number of neighbors because after experimenting $k = 10$ turned out to be a number that gives good results for a wide range of power models.

We did not exchange the prediction method from 10-nearest neighbors to a linear regression or a support vector machine based regression because the conducted experiments did not show any signs of improvement while at the same time the simplicity of the method would have been lost.

5.3.3 Benchmarking

Several experiments have shown that the power models can be improved by adding more benchmarks, especially real world test situations. In addition to the previously used benchmarks described in section 4.1.9 we expanded the benchmarking process using video streaming sites to account for a real world application.

The real world benchmarks consisted of opening several browser windows on Google Chrome and Firefox and streaming multiple videos in high definition from Youtube[5] and Vimeo[6]. These additional benchmarks added coverage in areas different from the ones covered by the previously used benchmarks as can be seen by the results chapter 6.

5.3.4 System Utilization Parameters

The system utilization parameters on Linux systems are read from the /proc pseudo file system just like before. The parameters used did not change in this case. Additionally we added the possibility to generate power models on other operating systems such as Windows and Mac OS X by using the Sigar libraries. The system information taken from Sigar are corresponding to the system information taken from the /proc file system. This allows for power models to be operating system independent.

We investigated into adding more to the existing three system utilization parameters. Initially our efforts were focused on adding memory utilization, but several problems in regards to acquiring the correct utilization values prevented this parameter from being useful. Our and previous research show that for memory intensive operations the power contribution in low CPU utilization situations range can from 20-30%. However this is an issue that needs to be investigated separately. Fortunately the design and implementation of the power model generator allows for easy expandability regarding system utilization parameters. Parameters can easily be added at a later time.

In addition to quickly changing system parameters we also planned on adding slowly changing system parameters such as temperature and fan speed. The influence of these slowly changing parameters is very small and needs to be investigated as well.

[5] `http://www.youtube.com`
[6] `http://www.vimeo.com`

Chapter 6

Experiments

Several experiments were completed in order to verify that the method and implementation for the generation of power models described in this thesis was completed correctly.

6.1 Evaluation Hardware

To accurately measure the power consumption of the hardware under test it is not enough to use a conventional electricity usage monitor such as Kill-A-Watt. To allow the measured values to be recorded alongside the correlating hardware utilization values a much more sophisticated device with a connection to the device under test is necessary.

6.1.1 Digital Power Meter Yokogawa WT210

The Yokogawa WT210 (depicted in Figure 6.1) is a digital power meter and an Accepted Device by the Standard Performance Evaluation Corporation (SPEC) used for Power Efficiency Benchmarking. A high frequency range and improved precision make this power meter an excellent tool, albeit not inexpensive with a price point above $2500 [NNTS03]. The Yokogawa WT210 needs to be calibrated before reaching its basic accuracy of 0.1%, which has to be done properly and can cost up to $600.

Figure 6.1: The Yokogawa WT210 Digital Power Meter as seen from the front side.

The Yokogawa WT210's features place it several levels above a conventional Kill-A-Watt power meter.

- **0.1% basic accuracy.**
- 26 A maximum input with assured accuracy.
- 0.5 Hz to 100 kHz frequency range DC measurement.
- 5 mA range for very low current measurements.
- High speed data update with up to 10 readings per second.
- Line filter function and harmonic measurement function.
- **Integration timer of 10.000 hours maximum with a resolution of 1 second.**
- Supply voltage 100-120 VAC / 200-240 VAC.
- **Serial communications RS-232-C with a Baud rate of 1200-9600 bps.**

The connection between the Yokogawa WT210 power meter and the server (equipment under test) is illustrated in Figure 6.2. Using a custom set up the power for the server is provided by a power outlet via the power meter.

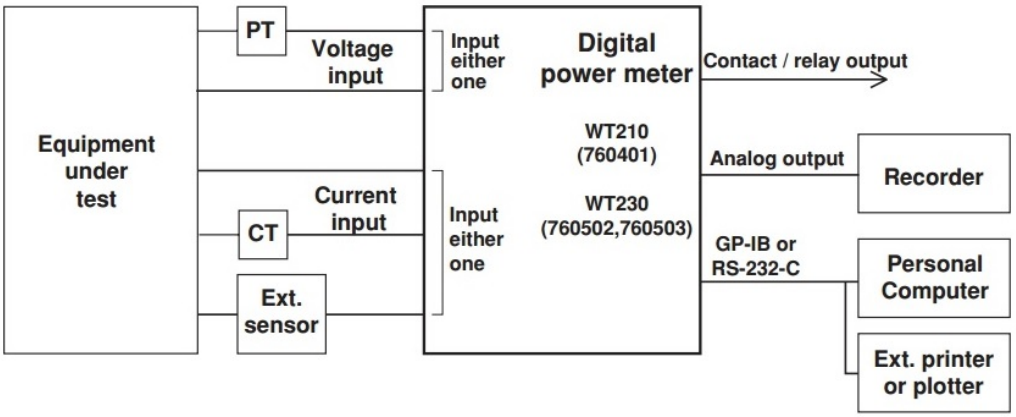

Figure 6.2: Yokogawa WT210 measurement system configuration.

6.1.2 Server Hardware HP ProLiant DL380 G7

As the device under test to carry out the experiments on functioned a 2010 era Hewlett Packard ProLiant DL380 Generation 7 (G7) rack size server. This type of server has undergone several hardware revisions and is a very common server to be found in current data centers. Good documentation from the manufacturer allows us to compare the validity of our own findings. Table 6.1 lists the server specifications, with a detailed look at the CPU in Table 6.2. Figure 6.3 shows the dynamic energy consumption of the server corresponding to CPU usage. This figure correlates with the energy consumption measured by our power meter.

CPU	2x Intel Xeon E5620
Memory	16 GB DDR3 1333 MHz (3x4GB, 2x2GB)
HDD	1x SCSI 410GB HDD
Power Supply	460 Watt Standard
Network	5x Broadcom BCM5709C NetXtreme II GigE

Table 6.1: System specification of the server under test.

Launch Date	Q1'10
Number of Cores	4
Number of Threads	8
Clock Speed	2.4 GHz
Max Turbo Frequency	2.66 GHz
Intel Smart Cache	12 MB
Instruction Set	64-bit
Max. TDP	80 W
VID Voltage Range	0.750V-1.350V

Table 6.2: Specifications of the Intel Xeon E5620 CPU used in our version of the HP ProLiant DL380 G7 server. Notice that our configuration uses two CPUs, effectively doubling the cores and power consumption.

6.2 Experiment Methodology

The power models created were all generated with the new power model generator using the evaluation hardware previously presented. If not otherwise specified the power models were generated with the following parameters:

- The operating system running on the HP ProLiant server was Ubuntu 12.10 64-bit.
- The hardware utilization parameters are taken from the /proc pseudo file system.
- The measurement time interval was one minute. This has empirically shown to be a good middle ground between keeping the measurement impact on the system as low as possible and the real-time capabilities still high.
- The overall system profiling time was 10 hours. This allows for all benchmarks to run as well as including idle time and some real world usage patterns.

6.3 Error measurements

The main measurement of quality in regards to the specified methodology is the mean absolute error (MAE). This quantity is generally used to measure how close forecasts or predictions are to the eventual outcomes. The MAE calculates according to Equation 6.1.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |prediction_i - observation_i| \tag{6.1}$$

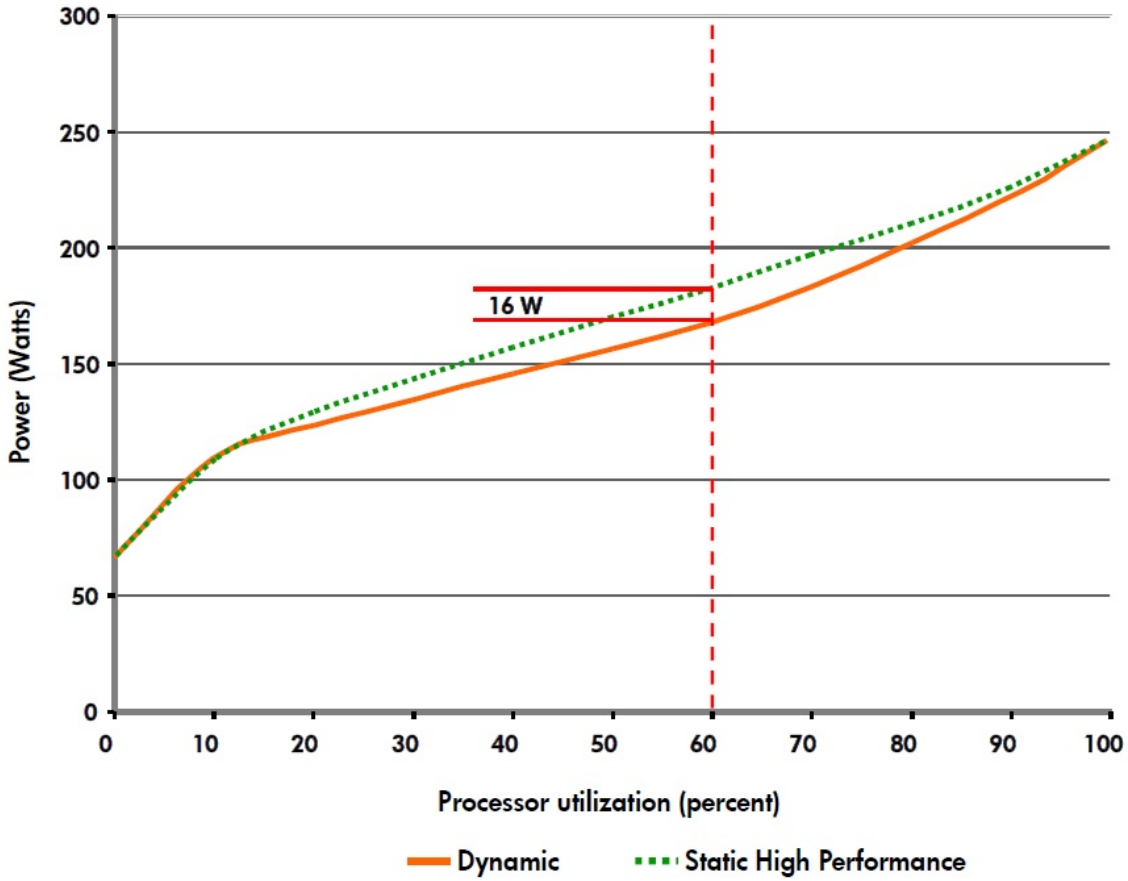

Figure 6.3: Effective power consumption of a DL380 G7 server at various system loads [HP11]. These specifications correspond to the measurements of our power meter.

The MAE can only be used for prediction values for which a hardware power meter measurement was recorded.

6.4 General Method Validity and Error Distribution

In order to validate the method we generated a power model on the evaluation hardware using all of the benchmarks. This power model then was validated using a leave-one-out cross-validation using the previously discussed Matlab scripts.

Using the absolute error from each prediction compared to the power meter measured value we were able to gather the results shown in Table 6.3. Figure 6.4 visualizes the error distribution. Interesting to note is that more than three fifths of the predictions are below an absolute error of 2%, with a quarter of the predictions below the very low error threshold of 0.5%.

More than four fifths of the predictions are below the error threshold of 5%. This shows that the performance of the estimation of the energy consumption is very good. Only very few predictions (below 4%) are worse than 20% error.

Error threshold	% predictions with error below threshold	% predictions with error above threshold
0.5%	25.6%	74.4%
1.0%	41.6%	58.4%
2.0%	61.3%	38.7%
3.0%	72.6%	27.4%
4.0%	79.4%	20.6%
5.0%	83.9%	16.1%
6.0%	88.2%	11.8%
7.0%	89.9%	10.1%
8.0%	91.6%	8.4%
9.0%	92.1%	7.9%
10.0%	92.6%	7.4%
15.0%	94.9%	5.1%
20.0%	96.6%	3.4%

Table 6.3: Error distribution of the power model generated with all benchmarks. The power model consists of 647 data points.

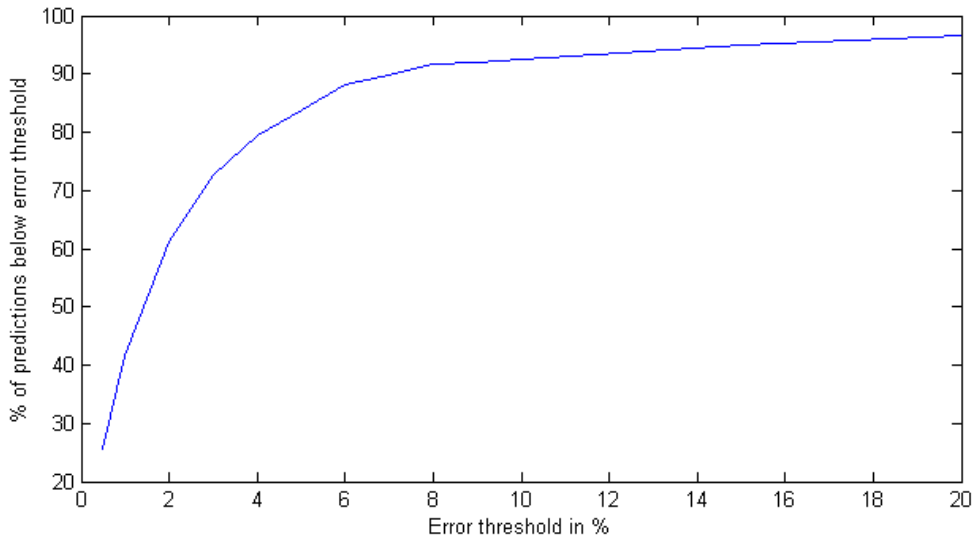

Figure 6.4: Plot of the power model error distribution.

The mean absolute error of this power model in the leave-one-out cross-validation is 3.22%.

6.5 Power Model Benchmark Dependency

The quality of a power model is strongly dependent not only on the parameters describing the model but also on the benchmarks used to generate the power profile. Optimally a power model should be generated with the software running during the profiling process that will later then also run on the server the power model will be used on to estimate the energy consumption. This approach is not always possible because the use of the server might not always be known in advance to the data center administrator. This experiment is designed to identify the impact the benchmarking has on the power model error accuracy.

For this reason we generated two power models using different sets of benchmarks. Whereas one power model (PM1) was generated with the use of only the benchmarking tools (tiotest, phoronix, nbench, and idle), the other power model (PM2) was generated without the use of the benchmarking tool and only includes real world usage of the server (continuous streaming of high definition videos from Youtube and Vimeo, opening, using, and closing software, copying files, etc.). A third model (PM3) was finally created simply by copying the data points from PM1 and PM2, thusly producing a power model containing benchmarks and real world usage.

The evaluation of the three power models is done via Matlab using cross-validation. In the case where training and testing power model are the same we employ leave-one-out cross-validation. The mean absolute error for each of the power models is shown in Table 6.4.

	PM1 as testing	**PM2 as testing**	**PM3 as testing**
PM1 as training	4.05%	6.38%	5.14%
PM2 as training	2.88%	1.81%	2.38%
PM3 as training	4.13%	2.18%	3.22%

Table 6.4: Table listing the mean absolute errors of different power models. PM1 stands for the benchmark power model, PM2 stands for the real world usage power model, PM3 is the combination of PM1 and PM2.

The interesting findings gained from this experiment was that the real world power models have a higher accuracy even when the test input is the power model generated using the benchmarks. This highlights the need for a server stress test process that takes into account the work load the server will be operating under.

The best results in terms of prediction accuracy was achieved using the real world power model in the leave-one-out cross-correlation. As expected the worst of the results was testing the benchmark power model PM1 in a real world situation using the power model PM2. However, even the mean absolute error of 6.38% is still considered a very good accuracy.

The three power models PM1, PM2, and PM3 are represented in the Figures 6.5 to 6.13. The most interesting findings here are that the benchmarks used on PM1 have very little impact regarding the network usage. This is particularly visible in the PM1 Figures 6.6 (NET/CPU) and 6.7 (NET/HDD) compared to the PM2 Figures 6.9 (NET/CPU) and 6.10 (NET/HDD).

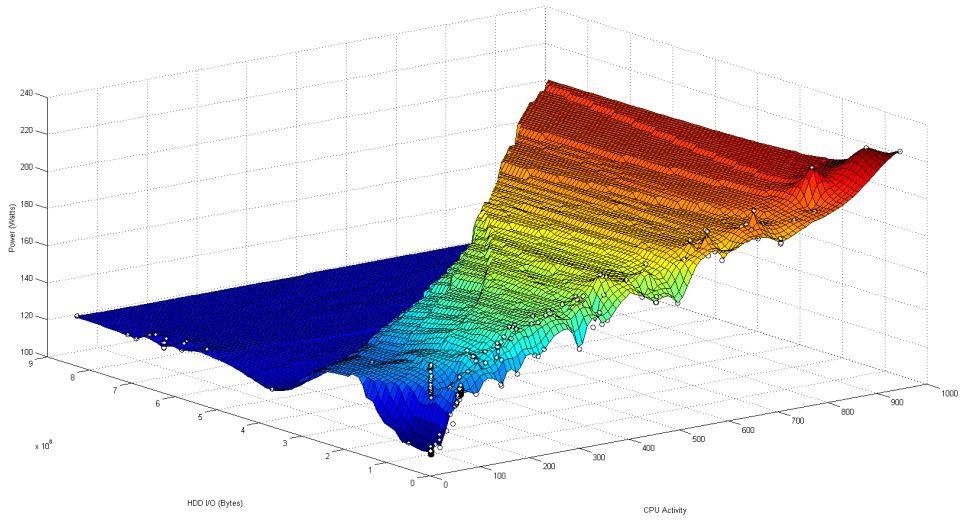

Figure 6.5: PM1 power model visualization HDD/CPU. Visible on the x-axis is HDD I/O utilization (bytes), on the y-axis CPU utilization (percentage times ten), and on the z-axis the power consumption (Watts).

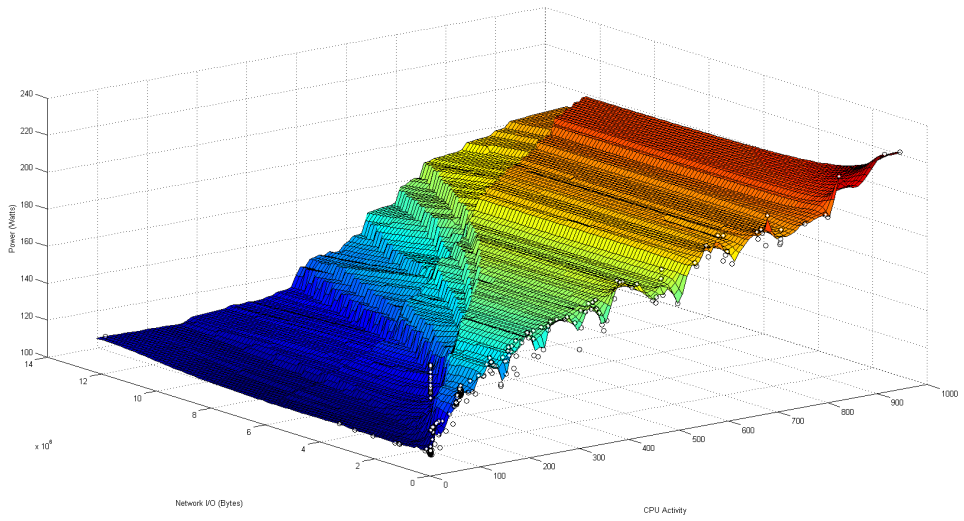

Figure 6.6: PM1 power model visualization NET/CPU. Visible on the x-axis is Network I/O utilization (bytes), on the y-axis CPU utilization (percentage times ten), and on the z-axis the power consumption (Watts).

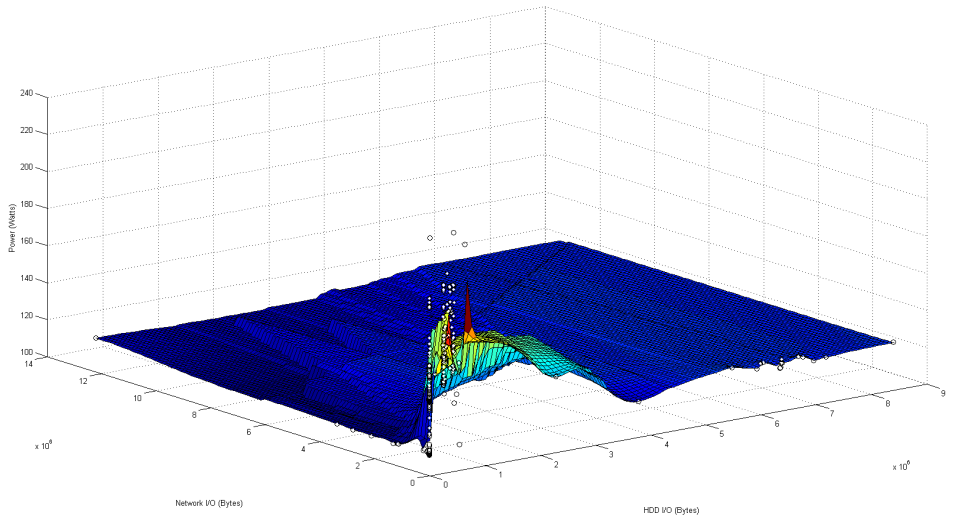

Figure 6.7: PM1 power model visualization NET/HDD. Visible on the x-axis is Network I/O utilization (bytes), on the y-axis HDD I/O (bytes), and on the z-axis the power consumption (Watts).

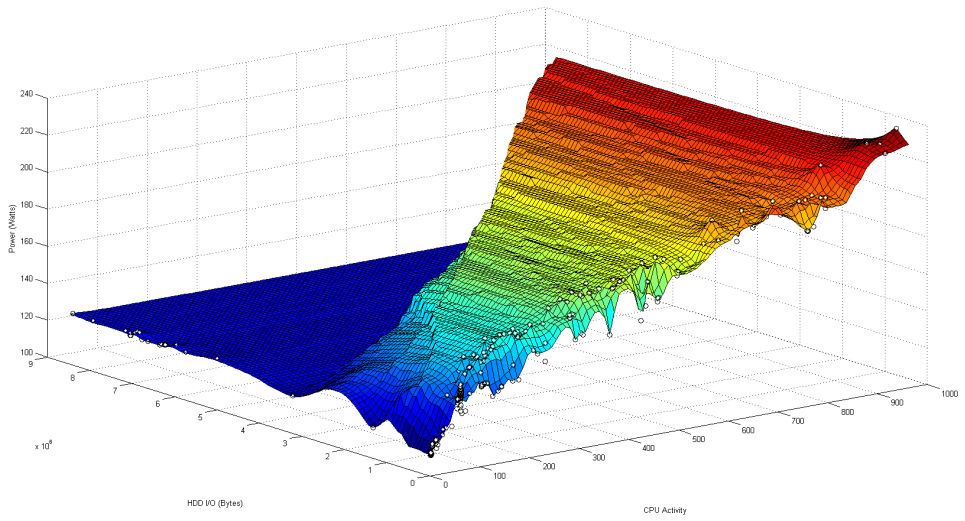

Figure 6.8: PM2 power model visualization HDD/CPU. Visible on the x-axis is HDD I/O utilization (bytes), on the y-axis CPU utilization (percentage times ten), and on the z-axis the power consumption (Watts).

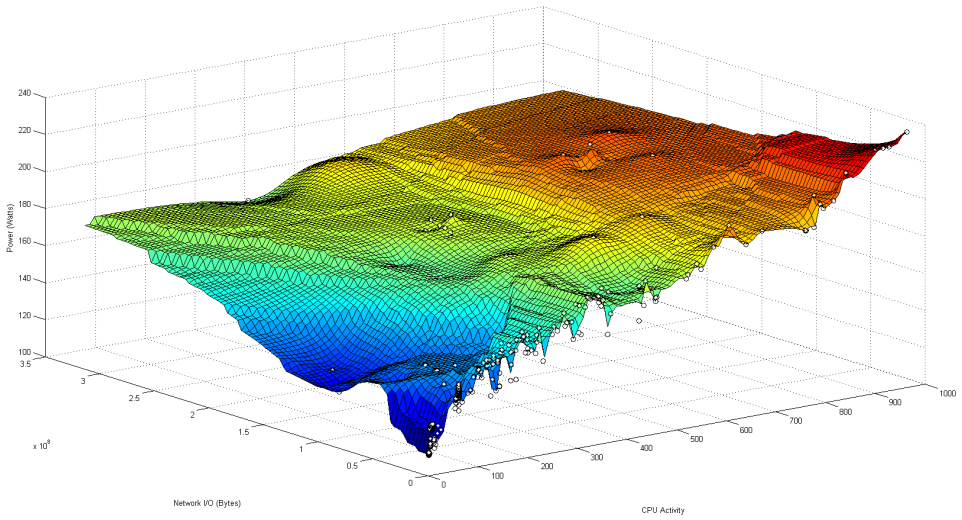

Figure 6.9: PM2 power model visualization NET/CPU. Visible on the x-axis is Network I/O utilization (bytes), on the y-axis CPU utilization (percentage times ten), and on the z-axis the power consumption (Watts).

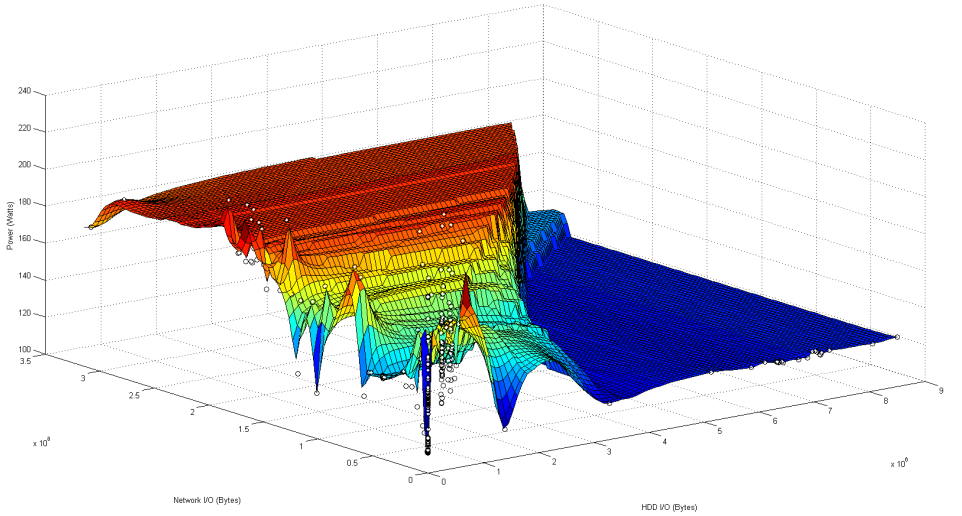

Figure 6.10: PM2 power model visualization NET/HDD. Visible on the x-axis is Network I/O utilization (bytes), on the y-axis HDD I/O (bytes), and on the z-axis the power consumption (Watts).

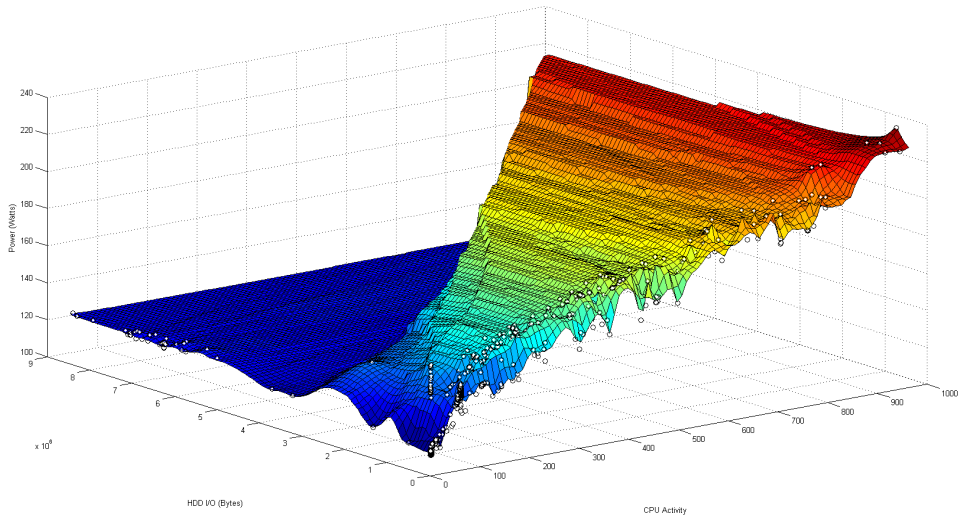

Figure 6.11: PM3 power model visualization HDD/CPU. Visible on the x-axis is HDD I/O utilization (bytes), on the y-axis CPU utilization (percentage times ten), and on the z-axis the power consumption (Watts).

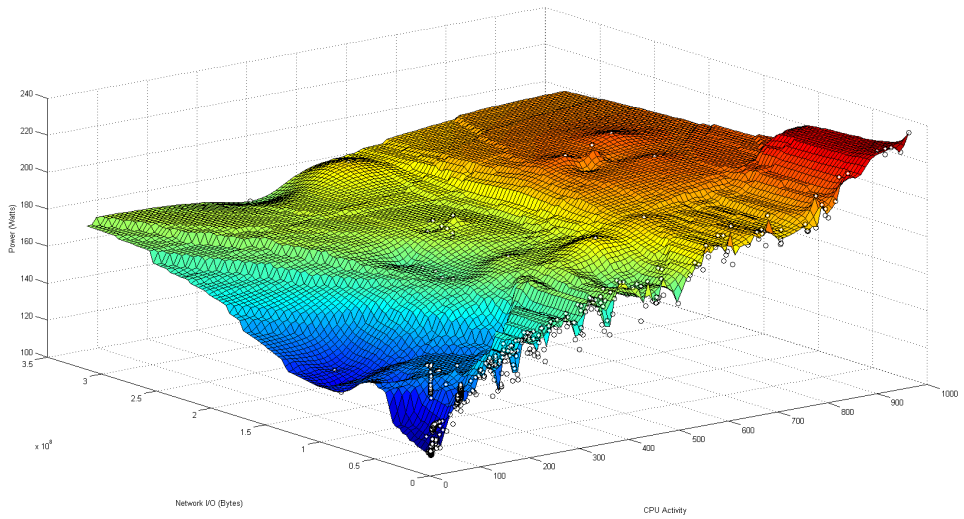

Figure 6.12: PM3 power model visualization NET/CPU. Visible on the x-axis is Network I/O utilization (bytes), on the y-axis CPU utilization (percentage times ten), and on the z-axis the power consumption (Watts).

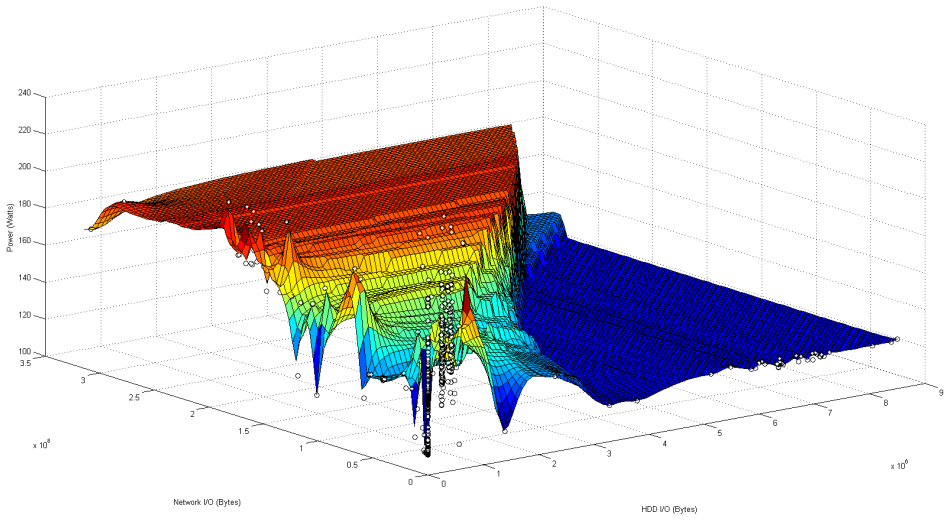

Figure 6.13: PM3 power model visualization NET/HDD. Visible on the x-axis is Network I/O utilization (bytes), on the y-axis HDD I/O (bytes), and on the z-axis the power consumption (Watts).

6.6 System Utilization Parameter Combinations

Another experiment conducted was to execute our method with system utilization parameters separately and in different combinations. This experiment highlights the influence of the various parameters on the energy consumption. This experiment was conducted using PM3. Table 6.5 shows the results. As expected the influence of the CPU on the dynamic energy consumption is the highest, with the HDD second and NET the least important parameters.

	CPU	**HDD**	**NET**
CPU	3.68%	3.07%	3.82%
HDD	3.07%	10.14%	9.45%
NET	3.82%	9.45%	18.48%

Table 6.5: Table listing the mean absolute errors of different power model parameter combinations. The combinations tested were CPU only, HDD only, NET only, CPU+HDD, CPU+NET, NET+HDD. The unlisted combination of all three parameters CPU+HDD+NET yielded a mean absolute error of 3.22%.

This shows that a very good prediction value can already be achieved using only CPU parameter in CPU intensive work loads. However the results are strongly work load dependent. Using additional parameters improve the prediction results.

6.7 Sigar vs. /proc

Since the implementation included not only the system utilization values taken from the Linux /proc file system but also the Sigar libraries we conducted an experiment to verify that this would decrease the quality of the method. Therefore we generated a power model where system utilization values were recorded using both methods. After performing a leave-one-out cross-validation with our Matlab scripts we determined the results from the power model gained via Sigar to be slighty worse (by 0.25%) than the power model using /proc values.

The reason for this is to be found in the way disk operations are being recorded by each method. However for real world applications this slight decrease in performance should be unnoticeable.

We also tested and verified the power model generation using Microsoft Windows Server 2008 R2 as an operating system. With this OS only Sigar was available for the system utilization statistics, as Windows does not offer the /proc pseudo file system. The results did not significantly differ from the results of the experiment done on Ubuntu.

6.8 Figure of Merit Evaluation

The development of a figure of merit allows us to improve the percentage of which a prediction can be considered good (<5% error). The experiment we conducted is using the PM3. By taking the threshold of 0.15 on the standard deviation of the 10 nearest neighbors power value we were able to increase the amount of predictions considered to be good from 77.7% (503 <5% out of 647) to 91.5% (440 <5% out of 481).

6.9 Regression-based vs. Nearest Neighbor

Even though it was not the focus of this work it should be mentioned that we conducted some experiments using regression-based methods to compare with our method of using the 10-nearest neighbor prediction algorithm.

For this experiment we used the libsvm library[1] as well as the Matlab built in command *regress*. The results of both the support vector machine regression and the least squares linear regression were both very close (1-2% MAE worse) to the 10-nearest neighbor method after experimenting with the parameters. Given the complexity of support vector machine regression however, it took much longer to find appropriate parameters and also the computation of the estimation took a longer time.

However this experiment is only noted for the sake of completeness and future research would be necessary in order to include this method as a viable alternative to the currently well-tuned 10-nearest neighbors prediction algorithm.

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Chapter 7

Conclusion

7.1 Conclusion

Innovative solutions are necessary to cope with the rising energy consumption in the IT industry. One step towards solving the energy consumption related problems is to determine where the energy consumption happens. An old management proverb tells us "You can't manage what you can't measure".

Stratergia presents a methodology and software that is able to accurately estimate the energy consumption of servers without additional hardware. This work focuses on characterizing and advancing the methodology with which the power models are generated. We designed and implemented a software with the given requirements and improved upon the methodology.

The main advantage of the Papillon system is the high accuracy and the ability to determine estimations with a higher error. The new implementation of the Papillon power model generator is also focused on expandability, allowing for system information parameters to be added easily.

As can be seen in the experiments the power model generator and the visualization provide the amount of information required. The system benefits from a generally very solid accuracy (mean absolute error of under 4%) and more than a quarter of estimations having an error of under 0.5% error. However there is still room for improvement and future work to be done in the field of IT energy efficiency.

7.2 Future Work

7.2.1 System Component Energy Consumption Information

One of the improvements necessary in order to better measure and define the energy consumption of entire data centers is to more openly reveal the energy consumptions of the single components in a system. While for a lot of the components we are already able to profile them independently of the manufacturer's specifications (which are often not very helpful anyways) for other components such as memory it still is very hard to gather information about the dynamic energy consumption. This problem is only going to become more important with CPUs that internally switch states and hide this power management from the operating system.

7.2.2 Energy Consumption on Operating System Layer

While Papillon is a software that can be employed right away to measure the energy consumption of servers with existing (and most likely future) operating systems it is an abstraction level higher than the operating system itself. Optimally a software measuring the energy consumption would run on the same layer as the operating system itself, especially in the case of virtual machines, in order to gain a higher accuracy on the energy consumption estimations.

A glimpse of where operating systems are heading can be seen with the latest release of Apple's Mac OS X 10.9 Mavericks. Even though this solution is still not as sophisticated as the Papillon system and can not estimate the energy consumption in a defined measurement such as Watts, it is a step in the right direction by giving each running application a measurement called *Energy Impact*. This measurement is described as a number that is a relative measure of the energy impact of an app or process, taking into account factors such as overall CPU utilization, idle energy draw, and interrupts or timers that cause the CPU to wake up.

7.2.3 Papillon Future Work

The underlying prediction algorithm powering the Papillon system is simple, elegant, and quick. We have proven within this thesis that the 10 nearest neighbors algorithm gives very good results. However little research has gone into the question whether there are better alternatives to this prediction algorithm. Therefore this should also be considered in future work.

The system information parameters we use have also shown to be quite accurate for current generation server power models. However with every advancement in hardware these should be evaluated if they capture the component's energy consumption behavior correctly. One instance of this could be the switch from conventional hard disks to ever improving solid state disks, where the hard drives energy consumption profile could be different.

With our experiments we have also shown that the performance and accuracy of the power models is strongly dependent on the benchmarks used during the power model generation process. Real world power models can have a high impact on the accuracy of the power model energy consumption estimation. This makes obvious the need for knowledge over the applications running on the monitored server. Incorporating this information into the power model generation process will only improve the accuracy of the system.

The last field of improvement lies in enabling Papillon to carry over the system to virtual machines. While the whole power model generation process can already be applied to a virtual machine, we have not yet done any extensive testing using virtual machines. Therefore this remains a field that needs investigation in order to retain the powerful position Stratergia is in.

Bibliography

[AAF12] V. Avelar, D. Azevedo, and A. French. PUE: A Comprehensive Examination of the Metric. `http://www.thegreengrid.org/en/Global/Content/white-papers/WP49-PUEAComprehensiveExaminationoftheMetric`, 2012.

[ACD+08] D. Anderson, T. Cader, T. Darby, N. Gruendler, R. Hariharan, A. Holler, C. Lindberg, G. Verdun, and J. Wallerich. A Framework for Data Center Energy Productivity. `http://www.thegreengrid.org/Global/Content/white-papers/Framework-for-Data-Center-Energy-Productivity`, 2008.

[BAP+12] A. Bérut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider, and E. Lutz. Experimental Verification of Landauer's Principle Linking Information and Thermodynamics. *Nature*, 483(7388):187–189, 2012.

[Bel10] C. Belady. Carbon Usage Effectiveness (CUE): A Green Grid Data Center Sustainability Metric. `http://www.thegreengrid.org/Global/Content/white-papers/Carbon_Usage_Effectiveness_White_Paper`, 2010.

[BH07] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *Computer*, 40(12):33–37, 2007.

[Bla10] M. Blackburn. The Green Grid Data Center Compute Efficiency Metric: DCcE. `http://www.thegreengrid.org/en/Global/Content/white-papers/TheGreenGridDataCenterComputeEfficiencyMetricDCcE`, 2010.

[Cap10] D. J. Cappuccio. DCIM: Going Beyond IT. *Gartner Inc, Stamford, CT, USA G*, 174769, 2010.

[CH11] G. Cook and J. V. Horn. How Dirty is Your Data? A Look at the Energy Choices that Power Cloud Computing. `http://www.greenpeace.org/international/en/publications/reports/How-dirty-is-your-data/`, April 2011.

[ERKR06] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan. Full-System Power Analysis and Modeling for Server Environments. In *Proceedings of Workshop on Modeling, Benchmarking, and Simulation*, pages 70–77, 2006.

[FDOR12] P. Faist, F. Dupuis, J. Oppenheim, and R. Renner. A Quantitative Landauer's Principle. *ArXiv e-prints*, 2012.

[FFN12] M. Faccioni Filho and M. F. Neto. Data Center Infrastructure Management and Automation Systems: an Evaluation Method. *(CAINE2012) 25th International Conference on Computer Applications in Industry and Engineering*, 2012.

[FWB07] X. Fan, W.-D. Weber, and L. A. Barroso. Power Provisioning for a Warehouse-Sized Computer. *ACM SIGARCH Computer Architecture News*, 35(2):13–23, 2007.

[Goo13a] Google. Data Center Efficiency: How others can do it. `http://www.google.com/about/datacenters/efficiency/external/`, July 2013.

[Goo13b] Google. Google Green. `http://www.google.com/green/`, July 2013.

[Gre13] Greenpeace. Greenpeace Green IT Challenge. `http://www.greenpeace.org/international/en/campaigns/climate-change/cool-it/`, July 2013.

[HP11] HP. Power Regulator for ProLiant servers. `http://www.hp.com`, February 2011.

[JHVG95] R. Johnson, R. Helm, J. Vlissides, and E. Gamma. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1995.

[KAGS11] B. Krishnan, H. Amur, A. Gavrilovska, and K. Schwan. VM Power Metering: Feasibility and Challenges. *SIGMETRICS Perform. Eval. Rev.*, 38(3):56–60, January 2011.

[Koo11] J. Koomey. Growth in Data Center Electricity Use 2005 to 2010. *Oakland, CA: Analytics Press. August*, 1:2010, 2011.

[KS11] S. Kent and R. Singh. Agent or Agentless? What are the Approaches, Advantages and Challenges of Deploying Technologies that use Agents versus Agentless ones? `http://www.1e.com/`, 2011.

[KVN10] R. Koller, A. Verma, and A. Neogi. WattApp: an Application Aware Power Meter for Shared Data Centers. In *Proceedings of the 7th international conference on Autonomic computing*, ICAC '10, pages 31–40, New York, NY, USA, 2010. ACM.

[KZL+10] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. Virtual Machine Power Metering and Provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing*, SoCC '10, pages 39–50, New York, NY, USA, 2010. ACM.

[MAC+11] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta. Evaluating the Effectiveness of Model-Based Power Characterization. In *USENIX Annual Technical Conf*, 2011.

[MLB+11] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi. Cloud Computing - The Business Perspective. *Decision Support Systems*, 51(1):176 – 189, 2011.

[MV05] A. Mahesri and V. Vardhan. Power Consumption Breakdown on a Modern Laptop. In *Power-aware computer systems*, pages 165–180. Springer, 2005.

[NNTS03] H. Nakanishi, N. Nishigaki, K. Tachibana, and T. Shinagawa. WT210/WT230 Digital Power Meters. Technical report, Yokogawa Test and Measurement, 2003.

[Pat10] M. Patterson. ERE: A Metric for Measuring the Benefit of Reuse Energy From a Data Center. `http://www.thegreengrid.org/en/Global/Content/white-papers/ERE`, 2010.

[Pat11] M. Patterson. Water Usage Effectiveness (WUE): A Green Grid Data Center Sustainability Metric. `http://www.thegreengrid.org/en/Global/Content/white-papers/WUE`, 2011.

[RRK08] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A Comparison of High-Level Full-System Power Models. *HotPower*, 8:3–3, 2008.

[RWAJJ11] D. Reilly, C. Wren, D. Al-Jumeily, and A. Jones. Development of a Power Saving Framework for use in Large Campus Networks. *Developments in E-systems Engineering*, 2011.

[Sau92] H. D. Saunders. The Khazzoom-Brookes Postulate and Neoclassical Growth. *The Energy Journal*, 13(4):131–148, 1992.

[Tra13] H. Tranninger. Application Specific Power Estimation for Virtualized Data Centers. Master's thesis, Graz University of Technology, May 2013.

[Vel13] D. Vellante. Cloud Computing 2013: The Amazon Gorilla Invades the Enterprise. `http://wikibon.org/wiki/v/Cloud_Computing_2013%3A_The_Amazon_Gorilla_Invades_the_Enterprise`, July 2013.

[Ver13] J. Verge. Ten Predictions: Why 2013 Will Be a Big Year for DCIM. `http://www.datacenterknowledge.com/archives/2013/01/03/ten-dcim-predictions-for-2013-25-penetration-by-june`, January 2013.