Thomas Kriechbaumer, BSc

Optimisation and Analysis of Full-System Power Models for Servers in Data Centres

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor:

Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger

Institute for Technical Informatics

Advisors:

Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger
Dr. Damian Dalton, University College Dublin

Graz, May 2015

The only footprints I will leave
will be my inventions.

– Lee De Forest

Abstract

The trend of cloud computing and moving services into data centres underwent an exponential growth in the last decade. More and more companies rely on the benefits a large data centre can provide. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) are popular products in the current technology area. Such an increase in computational density in data centres around the world comes at a not to be underestimated prize in terms of energy, cooling, and resource management. Reducing the total energy consumption and increasing the efficiency of a data centre is a huge research area under the term Green IT, which focuses on sustainable data centres and computing devices. Data Center Infrastructure Management (DCIM) is a key component to increase efficiency and maximise the output of a data centre. A key feature of DCIM is to provide accurate power values of individual devices in real time. Monitoring every server with their own hardware power meter is financially not feasible with the number of total servers in a data centre exceeding 100,000.

Therefore the Papillon system was created to monitor the energy and power consumption without the need for physical meters. Papillon uses a mathematical model to estimate the energy and power consumption in real time. Each server reports specific usage metrics to a central Papillon Server which then estimates the power consumption for the server and individual applications running on it. The model is specific to each server type and hardware configuration. The power model only needs to be created once before the server goes into operation. This work contributes to the existing Papillon technology stack by improving the model generation and evaluating different approaches to increase the accuracy of power estimations. The power model generation workflow is optimised and an artificial workload generator, which replicates a live running server with real activity, is implemented. The new workflow and components are numerically evaluated in terms of accuracy, various error metrics, and performance against various other modelling techniques. This thesis concludes with a comparison between the Papillon system and simplified mathematical models proposed by other researchers working in the Green IT field.

Kurzfassung

Der aktuelle Trend zu Cloud Computing und die Zentralisierung von Diensten in Rechenzentren ist von exponentiellen Wachstum in den letzten zehn Jahren geprägt. Immer mehr Firmen verlassen sich auf die Vorteile von großen Rechenzentren und den Angeboten in der Cloud. Infrastructure as a Service (IaaS), Platform as a Service (PaaS) und Software as a Service (SaaS) sind stark nachgefragte Produkte in der gegenwärtigen IT-Ära. Dieser enorme Anstieg in Rechenkapazität für weltweit agierende Rechenzentren hat starke Auswirkungen auf den globalen Energieverbrauch, sowie die damit verbundene Abwärme und Kühlung. Die Verringerung des Gesamtenergieverbrauchs und die gleichzeitige Effizienzsteigerung von Rechenzentren ist ein aktuelles Forschungsgebiet welches unter dem Begriff Green IT behandelt wird. Green IT versucht den Fokus auf nachhaltig operierende Rechenzentren und effiziente IT Ausstattung zu legen. Data Center Infrastructure Management (DCIM) ist eines der zentralen Komponenten um genau diese Effizienzsteigerung zu erreichen. Eines der Hauptmerkmale von DCIM ist das Bereitstellen von genauen Stromverbrauchswerten für einzelne Geräte in Echtzeit. Die Überwachung von jedem einzelnen Server mit dezidiertem Strommessgeräten ist aus finanzieller Sicht nicht durchführbar. Moderne Rechenzentren beinhalten über 100 000 Server.

Aus diesen Gründen wurde das Papillon System konzipiert, um die Energie- und Stromverbrauchswerte permanent ohne Messgeräte zu überwachen. Papillon verwendet ein mathematisches Modell um die Verbrauchswerte in Echtzeit zu bestimmen. Jeder überwachte Server meldet spezifische Nutzungsmetriken zu einem zentralen Papillon Server, welcher anschließend diese Daten mit dem Modell kombiniert und den Stromverbrauch für den Server und einzelnen Applikationen ermittelt. Das Modell ist spezifisch für jeden Servertyp und Hardwarekonfiguration. Für jeden Servertyp muss daher nur einmal ein Strommodell erstellt werden bevor dieser im Rechenzentrum eingesetzt werden kann. Der Beitrag dieser Master's Thesis ist eine Erweiterung und Analyse des vorhandenen Papillon Systems, im Besonderen Verbesserung der Strommodellerstellung und die Analyse der Genauigkeit von Stromverbrauchswerten. Die Arbeitsabfolge zur Erstellung von Strommodellen wird optimiert und ein synthetischer Belastungserzeuger für diverse Nutzungsmetriken wird vorgestellt. Der neue Arbeitsablauf und neue Komponenten des Papillon Systems werden anhand der Genauigkeit und Effizienz von Fehlermetriken analysiert und verglichen. Zusätzlich zu den Verbesserungen zum Papillon System werden bereits existierende Energiemodelle und deren mathematische Algorithmen, aus dem Green IT Umfeld, analysiert und mit dem neuen Papillon System verglichen.

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.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

Contents

List of Figures

List of Tables

List of Listings

Acknowledgements

I am indebted to my colleagues in Graz, as well as my co-workers in Dublin, where I conducted the main part of this thesis. Their support and advise proved to be invaluable. I am deeply thankful for the opportunity to work with the School of Computer Science and Informatics at University Collage Dublin and the Institute for Technical Informatics at Graz University of Technology.

Special mention goes to my advisors Christian Steger, Damian Dalton and Abhay Vadher for their essential comments, constant feedback and valuable annotations during drafting sessions.

Last but not least I want to thank all of my family, my friends in Austria, as well as the new friends I met in Dublin, who supported me throughout my studies and writing this thesis. They endured my never-ending monologues and brainstorming sessions, for which I am very thankful.

Thomas Kriechbaumer
Graz, Austria, May 2015

1

Introduction

This thesis describes improvements and evaluations to the Stratergia Ltd. Papillon system. The Papillon technology stack is used in a data centre environment to monitor energy consumption of individual servers and higher level topological groups of devices, such as racks and floors. The key component is the power and energy estimation by using a mathematical model without the need for hardware power meters on each device. The specific additions to the Papillon system are described in this thesis and the power estimation accuracy is evaluated and compared to different model types.

1.1 Motivation

Estimating the power consumption of a server is a non-trivial task. Usually data centres put in hardware power meters to gather energy consumption values at higher levels, like a full rack or floor in a data centre. Compared to measuring every single device independently, this reduces the costs significantly, but at the same time the granularity is lost to distinguish individual devices and servers. Since most of the hardware in data centres is produced in masses there is only a limited number of different manufacturers and model configurations. Using a non-invasive approach to estimate power consumption without the need for any additional hardware is a key feature of the Papillon system created by Stratergia Ltd. This technology stack allows the data centre operator to gain fine-grained energy and power consumption values for individual servers. Incorporating this information into a larger system allows the data centre to reduce its overall costs significantly.

Power estimation is the key part to increase the efficiency of Information Technology (IT) equipment and reduce environmental impact. For each type of server a power model is created before the server goes into use in the data centre. This mathematical model consists of various correlating performance metrics from a running server and with power readings from a connected power meter. The Papillon system then can use this model to estimate power values for running servers in the data centre which report their performance metrics on a regular basis to the Papillon system.

The motivation for this thesis is to improve the workflow of creating these power models and increase the accuracy for different workloads and system utilisation levels. Various similar techniques and approaches to power estimation are proposed and researched in academia, all promising different levels of accuracy and usability. Currently no comparisons between the Papillon power estimation process and other systems exists. Therefore it is necessary to prove the validity and accuracy of the gathered power estimation values from a data centre environment.

1.2 Goals

The goals for this thesis can be divided into four main parts, which are reflected in individual components of the Papillon system:

Power Model Generation
The Papillon system is a collection of multiple tools and applications. The code base should be unified and duplicated libraries and tools removed. The power model generation should be as compact and portable as possible to allow for usage on different architectures and operating systems.

Workload Generator
The current workflow for power model generation is a result of non-deterministic processes which yield a non-structured power model. By using an algorithmic approach to create an artificial workload on the server the repeatability and resulting accuracy of the power model should be improved significantly.

Post-Processing and Optimisation
The newly introduced workload generator will most likely result in an upsurge of computational effort for each power estimation. Due to the centralised approach of the Papillon system with a Server-Client architecture (see Section 4.1) an additional goal is to improve the estimation performance for the central Papillon Server while simultaneously increasing the accuracy.

Comparison and Accuracy Evaluation
Many researchers proposed simple linear power models. The Papillon system needs to be compared and evaluated to support the accuracy claims and provide a solid empirical basis for various error metrics. Various mathematical models as well as the usage of different activity data gathered from the running system can be compared to each other and the Papillon technology.

The work can therefore be split into a design, implementation, and evaluation section. The implementation chunk of work includes the modifications to the existing Papillon system as well as the extensions for new features like the workload generator and the post-processing pipeline. The evaluation piece then uses the newly created tools to gather and evaluate power estimation with different error metrics. The Papillon power estimation process will be compared against other popular power modelling techniques found in related publications.

1.3 Outline

The first part of the thesis covers a brief introduction into the necessary fundamentals in Chapter 2. Chapter 3 outlines previous research areas and related work, as well as other publications in this field. Chapter 4 describes the Papillon technology stack, in particular the power estimation and power model generation workflow as well as the state of the art in terms of model generation and validation.

The second part of the thesis covers the contribution of this work in terms of design and implementation in Chapter 5 and 6. The new power model generation workflow with a workload generator and a post-processing and optimisation pipeline is explained and design decisions are discussed. Finally, Chapter 7 presents the results of various experiments conducted to evaluate the accuracy of the proposed changes to the Papillon system. This thesis concludes with some ideas for future research and work in Chapter 8.

2

Fundamentals

In order to discuss this thesis a certain amount of vocabulary and terms are explained in this chapter. The given definitions are specific to the energy and power modelling field, but may be used in a broader scope with similar meaning. Since the field of energy efficiency in data centres and power modelling only emerged in the last decade most of the research publications uses slightly different definitions and meanings for new phrases. Therefore this thesis will use exact definitions and refer to different meanings if needed.

2.1 Energy and Power Consumption

The term energy refers to the electric energy which is transferred and consumed by electric consumers. For usages in electric application energy is measured in joules. Energy can only be transferred and converted, but never be created or destroyed. This means that all energy has to be accounted for after going through a device. For electric energy the most common conversion is into thermal energy by heating up surrounding matter such as air, heat sinks, and radiators. Usually only a small portion of the consumed electrical energy is actually used for electric operations.

Power is the amount of energy consumed within a given time. Therefore power is measured in joules per second, which is also abbreviated as watt. Using this definition leads to the derived form of a joule, which can be expressed as a watt-second. Reversing this definition results in a watt-hour, which is 3600 joule.

It is common to use kilo-watt-hours to measure electric consumption of devices, households, or data centres. Using kilo-watt-hours yields handy figures for the electric company to be used for billing purposes and such. From a pure physical and mathematical point of view this unit is not very intuitive, since it is only a strange form of 3600 kilo-joule.

A list of common units and their corresponding alternative definition is given as follows:

$$\begin{aligned} 1\,\mathrm{J} &= 1\,\mathrm{N\,m} = 1\,\mathrm{kgm^2/s^2} \\ 1\,\mathrm{W} &= 1\,\mathrm{J\,s} \\ 1\,\mathrm{kWh} &= 1000\,\mathrm{W\,h} = 3600\,\mathrm{s\,kJ/s} = 3.6\,\mathrm{MJ} \end{aligned}$$

Power describes an instantaneous rate at which energy is consumed, whereas energy refers to a time-sensitive value. Stating or comparing energy values without mentioning the time frame does not produce meaningful figures. Therefore the terms energy and power are used interchangeably in this thesis with the definition of a 60 second measurement period, unless

stated otherwise. Additionally the Papillon system, described in Chapter 4, does not use a strict definition in their power models and estimation workflow. The measurement in calorie and electronvolt is not common for regular energy consumption in typical data centre or household environments.

2.2 Power Estimation

Most IT devices in data centres consume a significant amount of energy depending on the current load on the device. Therefore the data centre operator needs to keep track of the constantly changing energy consumption throughout the various areas in a data centre, such as floors, containers, racks, or even individual servers inside a rack or blade system.

Power estimation is only necessary for devices with load-depended power consumption. This might be the case for servers, networking equipment, and storage devices. To a second degree cooling is affected as well, but this is usually not part of the IT equipment, but instead is accounted and managed by the infrastructure pool itself. Load-independent devices for example are physical security measurements consisting of electronic gates, door access control systems, and video surveillance equipment.

Therefore power estimation for the purpose of this thesis is always in respect to computational activity on a device. An incoming request to a web server creates load on the network switch, the server handling the request, a database server used for storage, a load balancer handling the connection, etcetera. The basic approach is to use special metrics from the device and feed them into a power model to get an approximated power value for the current sample. This means the used metrics must be highly correlated to the real power consumption, otherwise the estimation accuracy will be very low. With this definition a mathematical representation can be given as:

$$P_{\text{estimation}} = f(model_X, m_1, m_2, \ldots, m_n)$$

$\mathbf{P_{estimation}}$ is the estimated power consumption for the given sample.

$\mathbf{f}$ is the power estimation function, some form of algorithm or mathematical formula.

$\mathbf{model_X}$ is the static power model for a specific device X.

$\mathbf{m_1 \ldots m_n}$ are the metrics of a given sample gathered from the device.

This defines the three requirements for accurate power estimation derived from the components involved in the above mathematical representation:

- An algorithm to transform the given inputs into actual power values
- A power model representing the real power consumption of a specific device
- A set of activity metrics derived and gathered from the device

The resulting power estimation accuracy is solely based on the quality of these requirements. The accuracy can be increased by providing better components or changing the composition of one of the independent components.

2.3 Power Models

A power model correlates usage metrics with power values. Usually a static set of recorded metrics is combined with power readings from a physical power meter connected to the System Under Test (SUT). This data is commonly represented in a table-based structure and maps a tuple of metric values to a power value.

All values are stored using the same units for each metric. This allows for simple comparisons between different entries. Depending on the power correlation the entries can be sorted by the strongest power-relevant metric.

Depending on individual metric definitions these values can either be gathered directly or already contain combined values from multiple sources. This allows the power estimation to work on a smaller set of metrics with a better power correlation.

Ideally the power model serves as look-up table during the power estimation. If the metric values do not exactly match an entry in the power model the power estimation function has to interpolate the correct power value solely based on existing entries.

3

Related Work

3.1 Existing Publications

Data centres are spread across the world and account between 1.5% and 2% of the global electricity consumption according to Program [Pro07]. In recent years the growth rate was approximately 12% and is predicted to increase steadily over the next decade according to Cook and Horn [CH11]. Typical data centres take up an area of 30,000 sqft, consume 10 MW, house 1000 racks, each consuming 10 kW. This sums up to annual electricity costs of 8 million USD, annual air conditioning costs between 2 and 5 million USD, as well as additional operation costs for cooling between 4 and 8 million USD per year [Pat+03].

Modern data centres house over $100,000$ servers, consuming a total amount of up to $20MW$ according to Katz [Kat09]. The number of servers increased by one or two orders of magnitude since the late 1990's and early 2000's - and the number is still increasing at a fast rate. Due to the usage pattern of servers the data centre rarely reaches a centre-wide energy peak, because the servers tend to be underutilised the majority of the time, as shown by Ranganathan et al. [Ran+06] and Barroso and Holzle [BH07]. According to Hamilton [Ham08] power infrastructure costs account for about 80% of total data centre facility costs, but of which only 40% are used for operational costs and IT equipment. Therefore power consumption is a first-order design constraint in data centres [Dav+11].

By resizing the power supply units to a more likely nominal value Wang et al. [Wan+14] were able to install up to 50% more server into their data centres. Using such approaches the energy consumption has to be closely monitored to prevent the equipment from failing during peak usage, which might lead to expensive repairs and downtime.

Monitoring and controlling the power consumption of servers in a data centre is an important task to avoid peaks in power supplies and also to stay within the agreed power budget of the electric company. To ensure that the power consumption stays below a certain threshold each system can be limited with a power cap. Before reaching a certain power value a server could be capped by reducing the Central Processing Unit (CPU) frequency and voltage with Dynamic Voltage and Frequency Scaling (DVFS) or by limiting the memory access speed. Chen, Wang, and Li [CWL11] proved that such techniques can successfully reduce the risk of power consumption peaks in a data centre-like environment.

A data centre-grade server usually consumes between 40% and 60% of the maximum peak power during system idle periods [Kat09]. Typical utilization values for servers are 15% or less on average. This creates the possibility to consolidate workload onto a single server with virtualisation by using multiple virtual machines. With different virtualisation approaches one can increase the utilization up to a value of about 80% - which leaves a small window of operation

to account for usage variations [Kan].

Comparable metrics to characterise data centre efficiency are introduced by Belady [Bel07]. The Power Usage Effectiveness (PUE) describes the ratio of energy used for IT equipment to the total energy consumption of a data centre annually. It gives a simple comparable number which can be used in a broad spectrum. The best value would be 1.0, meaning every bit of energy consumed went straight into IT equipment (meaning servers and computation), current data centres have a PUE between 2.0 (first generation data centres) down to 1.2 (state of the art). The Data Center Productivity Index (DCPI) takes a slightly different approach by defining the ratio between the amount of useful work over the total consumed energy by the facility. The PUE has a disadvantage in the sense that if one upgrades only the IT equipment, but leaves the rest of the infrastructure untouched, the PUE will in fact increase, which is counter intuitive for a data centre energy efficiency metric. As Patterson et al. [Pat+13] stated "PUE is an infrastructure measure, to trend changes in the infrastructure over time, not to trend changes in the IT equipment. Changing the IT equipment is changing the baseline."

In typical Platform as a Service (PaaS)-based environments the provider usually wants to bill certain resources proportionally to each customer. Billing energy consumption for total servers is comparable easy, as long as each server is attributable to exactly one customer. In virtualised systems this becomes more and more difficult. Therefore tracking of individual applications is necessary to collect usage statistics for billing purposes. Such techniques are proposed by [Sno+09], [Zen+02], and [FS99].

Measuring power and energy consumption can be quite expensive if the number of devices is very high. Installing a power meter on every server in a data centre is financially not feasible and would lead to an increased number of maintenance costs to operate such devices. For these reasons software-based modelling of power and energy consumption is considered as alternative solution. Economou, Rivoire, and Kozyrakis [ERK06] introduced the concept of using a computational model to estimate the power consumption of a running system. Similar approaches with different use cases and refinements are proposed by [Riv08], [BJ07], and [FWB07].

According to Economou, Rivoire, and Kozyrakis [ERK06] power modelling can be divided into two broad classes by either using power measurements with a hardware power meter, as shown by [CM05], [CFR02] and [FS99], or by using simulated components and models the power consumption of individual parts in a pure software-based solution, as shown by [Bel00], [CKE00], [Gur+02], and [BTM00].

Measuring the energy consumption of physical components can be used to get an aggregated energy estimation for a specific Device Under Test (DUT). Most researched components are CPU, volatile memory (including bus communication), and hard disks (storage) - which make up for most of the consumed energy as shown by Kansal et al. [Kan+10] and Rivoire, Ranganathan, and Kozyrakis [RRK08].

Hardware manufacturers are aware of the trend to increase energy efficiency across components and systems. Minas and Ellison [ME09] describe several sources of power consumption in a data centre and servers by using techniques to "break down power consumption into its constituent parts" and try to "systematically provide guidance for minimizing each draw on electrical power" [ME09].

Castañé et al. [Cas+13] and Rivoire et al. [Riv+07] provide some insight into modelling different components which can run in various power states during runtime. Modern CPUs can operate on multiple frequencies and voltages, each state consuming different amounts of energy. Hard disks and volatile memory can be operated in similar ways to either increase throughput or energy efficiency [BJ07].

The CPU is one of the largest contributors to energy consumption and thus widely investigated in [Nat+09], [Bel00], [IM03], [SH01], [BTM00], and [SC01]. Increasing the energy

efficiency of processors was one of the easiest solution to reduce the overall energy consumption in the early days. Using techniques such as DVFS most modern servers are far more efficient than their predecessors as shown by Orgerie, Lefevre, and Gelas [OLG10]. The CPU can be used as a first-order proxy for energy estimation [RRK08].

Lent [Len13] considered the volatile memory, like Random-Access Memory (RAM), to be part of the CPU in their model.

Lin et al. [Lin+08] claim that a large amount of RAM can consume similar amounts of energy as the system main CPU, which would therefore be a non-negligible component.

Although there are various studies investigating and modelling energy consumption of volatile memory ([Raw04], [Jan01]), and hard disks ([Zed+03], [KGS06]), Zhang et al. [Zha+13] came to the conclusion that hard disks have almost constant power consumption. A similar conclusion can be drawn for networking devices, such as a Network Interface Controller (NIC), which consume power at an almost constant level, independent of the workload according to Heath et al. [Hea+05] and Nedevschi et al. [Ned+08]. Never the less these components can still contribute a large amount to the total idle system power.

Newer storage technology like Solid State Drive (SSD) consume up to 75% less energy than their spinning counter-parts [EKR02].

Using power models to predict the power consumption without actually measuring it can be done with various approaches. Most commonly hardware performance counters [Mai+14] and Operating System (OS)-based utilisation metrics [RRK08] are used. The simplified approach is to find a correlation between a (sub-) set of these metrics to the power consumption of the SUT. The SUT is connected to a wall power meter and an application on the SUT records various metrics and performance counters. This data set is then fed into a mathematical model to find correlations and compute a power prediction model based on the recorded data. Linear regression models are very common and widely adopted [Che+08], [Che+10], [Hea+05], [Ber+10], [LTG12], [FWB07], [SF10], [LB06], [SBM09], [LPF10], [Len13], and [Dav+11].

Steinder et al. [Ste+08] and Fan, Weber, and Barroso [FWB07] claim that CPU fits a linear relation to power consumption, but according to Orgerie, Lefevre, and Gelas [OLG10] this is not the case for most systems. OS design can have a big impact on energy efficiency. Real-time, embedded, or full-stack OSs will result in a major impact on system energy consumption [Bay+01]. Furthermore Ruth [Rut09] proof that "the link between what we use (load and performance) and what we pay (electric consumption) is not linear".

Non-linear-based power models were recently published by Li et al. [Li+12] and Dhiman, Mihic, and Rosing [DMR10], which prove to be more accurate over a larger range of server utilisation, without too much computational overhead.

Due to the increased adoption of virtualisation in data centres measuring and modelling of power consumption for physical machines is not sufficient any more, therefore predicting a virtual power consumption of virtualised machines was proposed by Yang et al. [Yan+14], Bertran et al. [Ber+12], and Chengjian et al. [Che+13].

This thesis builds on the previous work of Tranninger [Tra13], Kampl [Kam13], and Abraham [Abr15]. All of the mentioned work was performed in cooperation with Stratergia Ltd. The Papillon system is described in [Dal+12].

3.2 No Hardware Required: Building and Validating Composable Highly Accurate OS-based Power Models

Approximately 80% of data centre facility costs and 40% of operating costs account for power infrastructure in state-of-the-art data centres [Ham08]. Davis et al. [Dav+11] created an automatic framework for modelling power consumption in single nodes and clusters of node servers. The model uses only OS-level performance counters and is therefore cross-platform compatible. The validation of the created models "yields highly accurate predictions without the intrusiveness and/or the correctness and portability problems of hardware performance counters or board-level measurements", according to their publication [Dav+11].

In addition to the new modelling framework, the authors defined a new error metric called Dynamic Range Error (DRE), which gives a tighter bound for dynamic systems compared to existing metrics. According to their research, this is the "most complete study of system power modeling". Well-known error metrics, such as mean-squared-error and median error, are "difficult to compare across platforms" depending on large differences in operating power of each machine. Results might get skewed if the overall power contains large static components, e.g. idle power consumption. Therefore the only relevant part, in terms of objective accuracy, is the dynamic variation in power of a machine. The Dynamic Range Error (DRE) is defined by [Dav+11] as "the root-mean-squared error divided by the dynamic range".

The measurement infrastructure used for their tests and model creations includes an individual power meter for every machine in all clusters. The power measurements are recorded and stored on each machine over a USB port. As software platform the authors chose Microsoft Windows Server 2008 R2, which provides a standardised interface for OS-level performance counters using the Windows *perfmon* utility.

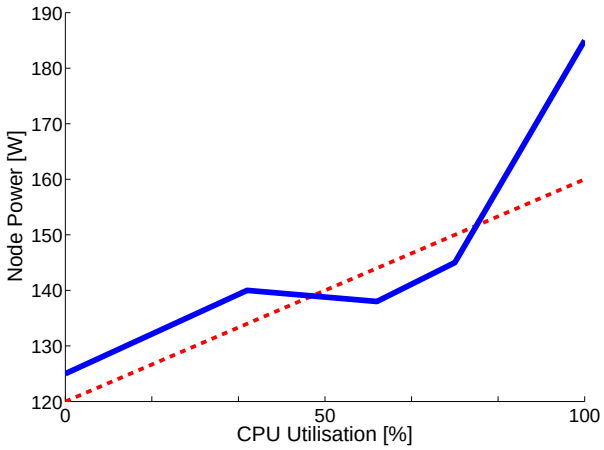


Figure 3.1: The piecewise linear model (full line) can predict the full-system power range, while the trained model (dashed line) cannot. Adapted from Davis et al. [Dav+11].

The results shown by Davis et al. [Dav+11] clearly indicate that piecewise linear power models outperform purely linear CPU-based power models, as visualised in Figure 3.1. The full dynamic range of a machine cannot be modelled with a linear model using only CPU utilisation. The authors state that the results "are typical across all platforms" and therefore only data for a single cluster and workload type was presented in their publication. The achieved power modelling accuracy ranges from 0.5 to 2.5% for median error and rMSE. The newly introduces error metric DRE settles below 10%.

3.3 Demystifying Energy Consumption in Grids and Clouds

Orgerie, Lefevre, and Gelas [OLG10] address some theoretical assumptions about energy efficiency in large-scale distributed systems, such as large data centres. By conducting multiple small- and large-scale experiments in regard to these assumptions and theories they can correct a few of the most common misconceptions.

Energy consumption of a system depends on the OS in use. The monolithic Linux kernel has full control over all hardware-related aspects and can regulate various parameters to manipulate activity and energy consumption. Comparing multiple different Linux kernel versions clearly shows variations in power consumption with identical workloads. Technologies such as Dynamic Voltage and Frequency Scaling (DVFS) allow the OS to change and adapt energy levels on-demand, but are "rarely exploited by default" according to Orgerie, Lefevre, and Gelas [OLG10]. Recent developments in kernel capabilities reduce the number of wake-ups significantly by using a tickless kernel mode, also called *Dynamic ticks*. This keeps the processor idle for as long as possible and only triggers a wake-up if required, but not periodically as without *Dynamic ticks*.

The authors state that "most facilities lack energy sensors and online power-monitoring tools", which is due to financial reasons in large data centres. This limits the capabilities in terms of monitoring and saving energy. Experiments conducted by Orgerie, Lefevre, and Gelas show that the general assumption of direct proportional CPU to energy consumption relation is actually wrong for most systems.

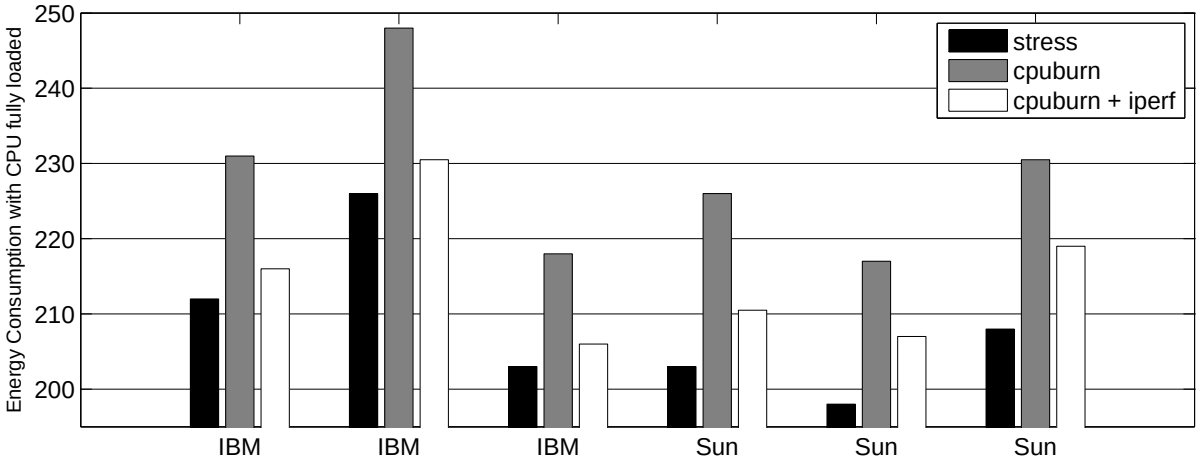


Figure 3.2: Comparison of the energy consumption of three applications that fully load the CPU. Adapted from Orgerie, Lefevre, and Gelas [OLG10].

In addition to the non-linearity of CPU and energy, the actual workload or application running on the CPU affects the total energy consumption significantly. Figure 3.2 shows 6 different systems, each running three types of applications that fully load the CPU to constant utilisation level of 100%. The bar chart clearly shows that different applications yield multiple different energy consumption values depending on the machine. The authors conclude, that "the difference can reach 14% (compared to the idle consumption of the node), which is not negligible", and that CPU utilisation impact on power consumption is not linear as stated in various other publications. Total CPU usage reported by the OS is not a valid indicator for actual work performed on the chip, since 100% usage does not necessarily relate to maximal power consumption. The authors try to explain this observed behaviour with a dependency of power to "the [currently executed] instructions [...] or the quantity of RAM accessed by the running applications". Similar experiments yield the same result for networking devices and disks.

3.4 Full-System Power Analysis and Modeling for Server Environments

Economou, Rivoire, and Kozyrakis [ERK06] introduce a power modelling system called *Mantis*. According to their own definition, *Mantis* is a "non-intrusive method for modeling full-system power consumption [... and provides] real-time power predictions". This systems uses a one-time calibration phase, during which user-level system utilisation metrics are correlated with power readings from a connected AC power meter.

Mantis uses a similar approach as Stratergia Ltd. Papillon, described in Section 4.1. Figure 3.3 shows the different stages used to create a Mantis power model and how to use it for power predictions. The authors state that power predictions in Mantis work by "correlating a few user-level utilization metrics or hardware performance counters with power consumption during a calibration phase". Depending on the update frequency the utilisation metrics are recorded and evaluated, the power estimations can be nearly instantaneous.

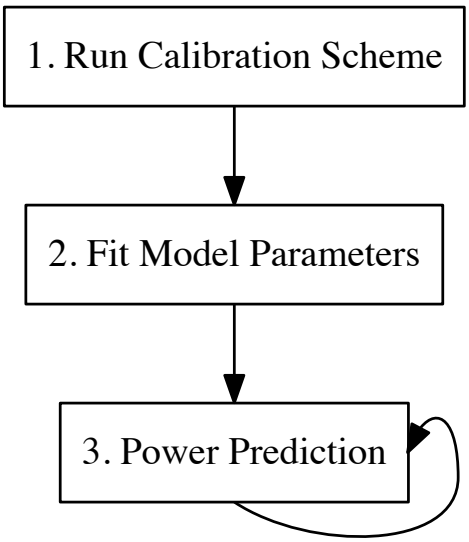


Figure 3.3: The stages of Mantis model development and use. Adapted from Economou, Rivoire, and Kozyrakis [ERK06].

The first stage consists of a calibration process by running "benchmarks that individually stress each major component of the system under test in order to derive the basic correlation between its utilization and power consumption", according to the Mantis definition of the authors. The second stage uses a linear program to fit special model parameters to the data. This process is only performed once for every new system type, "most likely by its vendor", as reported by the authors. Finally the third and last stage is the actual power prediction during which a server is continuously monitored and the live utilisation metrics are used to derive "accurate predictions of overall and component-level power consumption".

In addition to the *Mantis* system, the authors studied the power consumption of a blade system. In order to get power values for various components of a server, multiple power meters were connected to the four power planes inside the server. The resulting power consumption distribution can be seen in Figure 3.4.

In order to get a detailed list of power consuming components inside a blade system, the existing power planes with different voltages have been observed. The four power meters measured the following components:

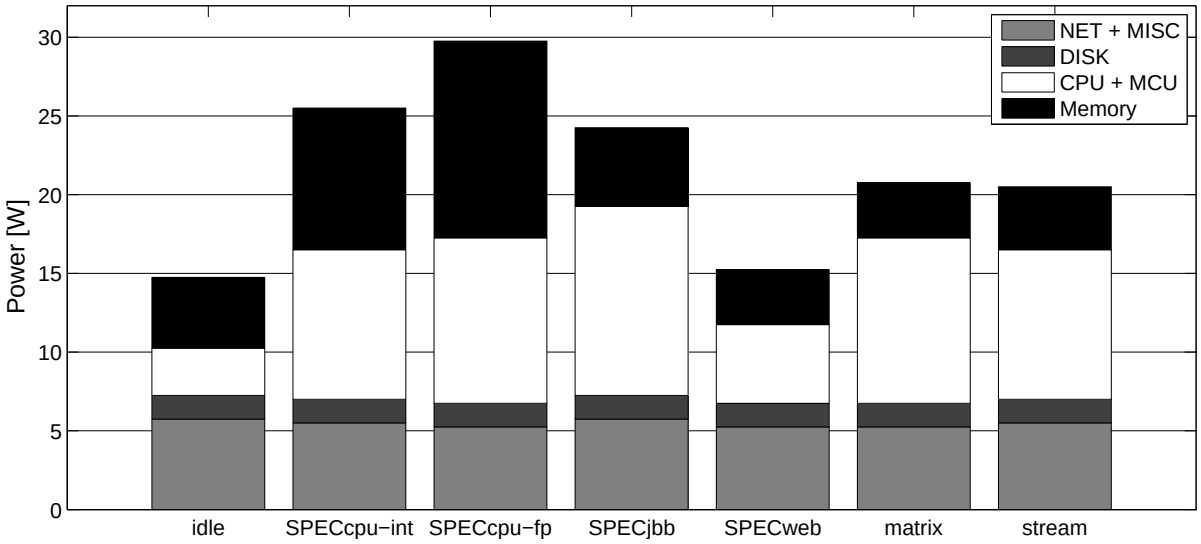


Figure 3.4: The component breakdown of the blade's measured power consumption. Adapted from Economou, Rivoire, and Kozyrakis [ERK06].

Processor connected to a 12V power plane is expected to be the largest power consumer in the system. Depending on the mode of operation in regard to DVFS the actual power reading might change significantly over a given time period.

Memory connected to the same 12V power plane as the CPU was physically extended with an extra sense resistor. According to the research conducted by the authors, memory is still an uncertainty in terms of power consumption, compared to other components.

Disk connected to a 5V power plane include spinning hard disks, as well as SSDs which can consume significant amount of power as well.

Network and other peripherals connected to a 3.3V power plane includes regulators, supplies, and other parts of the remaining system.

The full blade system was utilised by running different benchmarks and applications. The resulting power fragmentation across all power planes can be seen in Figure 3.4 for each application. The authors conclude that networking and other peripherals consume an almost constant level of power across all applications and workloads. The same assumption can be made for the disks used in their blade server. As expected and assumed by previous publications, CPU and memory contribute the most to the overall power consumption. These two components are basically the only ones with varying power values on different applications. This indicates a stronger correlation to power than other components.

4

State of the Art

4.1 Stratergia Papillon

The work of this thesis is based on the existing Stratergia Papillon system. Stratergia Ltd., an Irish-based company developed the Papillon system and tries to improve data centre workflows and energy efficiency. According to their own definition[1]:

> Stratergia is the global leader in non-intrusive, meter-less, data centre power measurement and management systems.

The introduction and definition of the Papillon system and the mission statement can be obtained from their website[2]:

> Profiling and Auditing of Power Information in Large and Local Organisational Networks (PAPILLON) is a unique and innovative, comprehensive power management system, measuring server and application - level power consumption in continuous time without the requirement of any additional hardware. It is simple and intuitive to use, operates in any common operating system and can be downloaded and installed ready from the Stratergia website.

This section was jointly written by Thomas Kriechbaumer and Matthias Vierthaler, who both worked at the same time on individual parts and experiments in the context of Papillon.

4.1.1 Main Features

The needs of a data centre operator are spread over a broad band of Data Center Infrastructure Management (DCIM) functionalities and components. Papillon provides a uniform access pattern to the most important areas in relation to energy, efficiency, and infrastructure management in general. The main features can be outlined as follows:

- overview of registered servers
- current and past estimated power usage per server
- current and past top three processes with respect to power and CPU usage

[1] http://stratergia.com/

[2] http://stratergia.com/papillon/introduction/

- reports with different metrics
- give power related information without the need of an power measuring device

Since every component and feature build on top of a central database it is easily extendable and configurable to the specific needs of the data centre operator. In addition to the software solution provided by Stratergia, it is possible to create extensions utilising the available data to provide custom solutions.

4.1.2 Architecture

In order to keep a flexible approach that scales well with many servers a server-client approach was chosen. In this scenario a client is one of the machines whose power demands are in question. The client periodically sends its necessary utilisation characteristics to the server. This is also referred to as *active client*, since the client is the one sending data messages. The server is only receiving and storing the information in a database for later use. For each new client utilisation message the server computes the corresponding power value and stores them together with additional information about individual application usage metrics.

Every Papillon-monitored environment has one designated Papillon Server which keeps track of all clients and their power-related values. The communication between the server and client, as well as other components, is handled via a single interface which builds on a Representational State Transfer (REST) web Application Programming Interface (API) with Hypertext Transfer Protocol (HTTP) as exchange protocol. This means that for the default HTTP calls RESTful API endpoints are implemented which handle `GET`, `POST`, `PUT` and `DELETE` according to the REST software architecture style.

Finally a user interface is provided in form of the Papillon Dashboard, which visualises the gathered information and manipulates database entries via API calls on the server. This component is optional and is normally installed on the same machine as the server software is running on. Since the API is well documented and accessible to everybody each customer can add functionality specific to their use case and data centre structure.

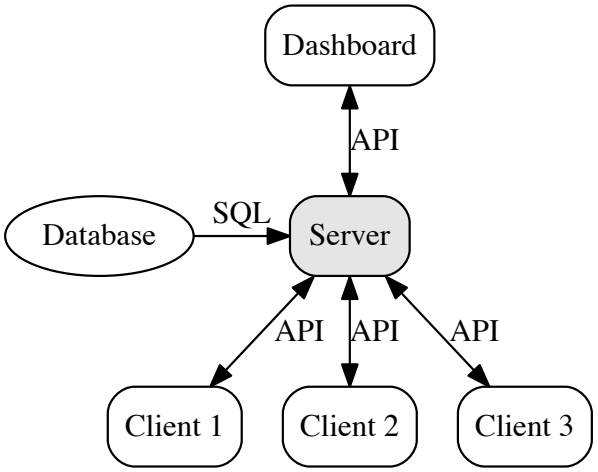


Figure 4.1: The relationships between Server, Client and Dashboard in the Papillon system.

The relationship between these components is depicted in Figure 4.1. In the currently deployed Papillon systems the clients only use a single API call to report data to the server. The server stores and manages data entries in the database and serves as data source for the dashboard or other user interfaces.

4.1.3 Power Estimation and Application Power Consumption

Since it is cumbersome for a data centre operator to install physical power meters for all servers and devices the Papillon system provides a virtual power meter to monitor the devices. The power consumption for each server can be viewed in one minute intervals. Additionally to the total power consumption per server each data entry contains the applications running on individual servers. The total power consumption can be split up to indicate the shares contributed by each application. With this feature it is possible to compute a specific power consumption for individual applications and process groups, such as database, email, or web servers.

To allow Papillon to estimate the used power for a device, a power model is created specifically for this hardware configuration and server type. The first stage is to generate such a power model once before the server goes into live operation within the data centre. This power model is then used by the Papillon system to estimate the needed power based on live data collected from each monitored server. Since most data centres tend to have a very homogeneous hardware equipment and a limited list of server manufacturers the number of different power models which have to be created will be manageable.

4.1.4 Papillon Server

The Papillon Server is the central information hub in a Papillon environment. It provides an interface to the database entries and services API requests for host clients and other Papillon applications. Each Papillon Client sends data to the central Papillon Server which computes power values and stores all the messages in a database.

The server application is a Java-based web application deployed to a single Apache Tomcat [Pro15] instance. The back-end database store is accessed via an interchangeable Java Structured Query Language (SQL) adapter and uses MySQL in the default installation. Since Apache Tomcat is available on all modern platforms the Papillon Server can be deployed to almost all OSs and web-capable platforms. The interface provided by Apache Tomcat is an open source project which exactly fulfils the purpose of executing Java code an a web-based architecture.

The publicly documented API accepts either JavaScript Object Notation (JSON) or Extensible Markup Language (XML) data messages and is capable to distinguish between those formats on a request-by-request basis.

The currently implemented Papillon Server API does not support authentication or authorisation. Therefore it is necessary for the data centre operator to provide a suitable security model on top of deployed Papillon system. This is likely to change in the upcoming versions as development advances.

4.1.5 Papillon Client

The Papillon Client is a small application installed on every monitored host. Its small footprint does not affect normal operation and does not require special privileges. Running as background service allows it to silently collect usage information and report it to the Papillon Server.

Usage data is gathered every minute for individual metrics and each sample is sent to the server automatically. Therefore it must be supplied with necessary configuration about Papillon environment:

- domain name or Internet Protocol (IP) address of the Papillon Server
- Transmission Control Protocol (TCP) port on which the server is listening

- Host ID, individual to each server

Some usage metrics need higher granularity on their reading, therefore the Papillon Client may perform a small amount of work more often then the regular one-minute interval. Since the only communication between client and server is done via the public API, there is no limitation if both are installed on the same machine. It is therefore possible for a Papillon Server to monitor itself as well as other servers only running the Papillon Client application.

Similar to the Papillon Server, the client is designed as Java-based application. This allows the data centre operator to monitor Microsoft Windows, Linux-based, and Mac OS X-based systems in a unified infrastructure using the Java Virtual Machine.

Due to the regular monitoring interval the Papillon Client is expected to run during all times a server is operating in the data centre. This will generate a steady stream of usage information stored in the Papillon Server and can be accessed and analysed at any time by the data centre operator.

In addition to the main usage metrics a set of application-specific information is recorded and stored together with the regular entries. Currently the three CPU-heaviest processes on the system are recorded and transmitted to the Papillon Server to indicate which application cause power spikes or are responsible for high power consumption periods.

4.1.6 Papillon Database

The Papillon Server is the only component which requires direct access to the database. All data manipulation operation pass through an Structured Query Language (SQL) interface to the MySQL database management system running in the Papillon environment. Since the queries are all defined as prepared statements the actual database system could be swapped out for a different engine, such as PostgreSQL or MariaDB using a different Java database connector library.

All data entries are stored in a relational database. Depending on the access pattern various indices are added to improve performance and querying speeds. Since many hosts are monitored at the same time it might be necessary to implement sharding to reduce performance bottlenecks. The basic database schema contains the following data models and tables:

Data Centres, Floors, Racks provide a simple way to topologically group multiple hosts into individual subgroups based on the actual physical placement inside a data centre.

Hosts are servers with an active Papillon Client running and reporting data back to the Papillon Server. The *Host ID* from this database entry must be configured in each Papillon Client to identify the data messages during transmission.

Power Model Groups and Power Models store the information necessary to estimate power based on activity metrics. These power models are created and grouped by different server manufacturers, server models, and hardware configurations.

Activity and Application Usage Entries contain all the activity information a Papillon Client has transmitted to the Papillon Server on a regular 60 second interval. These values are used to estimate the power value with the power model assigned to the host. Each entry is timestamped and stored in the database after the power estimation finished.

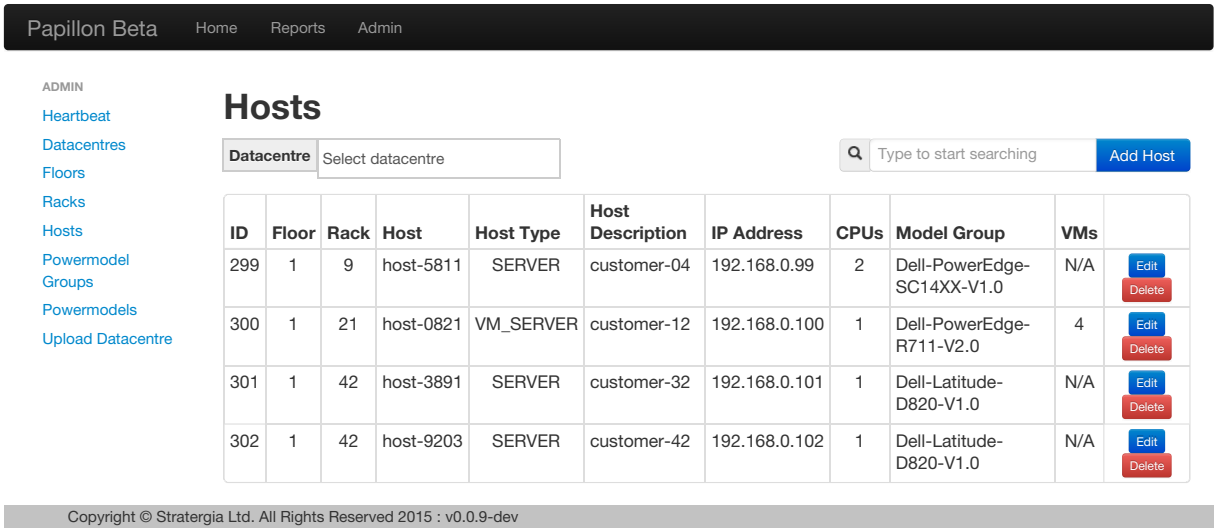


Figure 4.2: The dashboard shows the host administration page for the data centre operator to manage, add, and modify host entries in the Papillon database.

4.1.7 Papillon Dashboard

Since every monitoring system needs an interface to visualise and present the collected data the Papillon system provides the Papillon Dashboard as default user interface for data centre operators. It provides easy access to all management functionalities and database access for basic add, modify, and delete operations. Since the dashboard uses the public API of the Papillon Server it is easily exchangeable for a customised dashboard or even completely different application sitting on top of the API.

The dashboard can be used to add new hosts and view the power consumption and application-specific power values over a selectable time period. Different evaluation methods and reports can be created and downloaded via the browser. It is an optional component that is not required for regular operation, but acts merely as a user interface. The dashboard is a single-page web application and uses AngularJS and Bootstrap as web frameworks. The key features provided to the user are:

- manage infrastructure in data centres, racks, floors, and hosts
- manage power model groups and power models for different hardware configurations
- assign hosts with power models
- show inactive hosts and hot-spots
- render a power consumption graph per host and applications
- give visual feedback regarding the power value of hosts

Figure 4.2 shows the host administration page. This page provides information about all Papillon-monitored hosts and power model mappings. The *Host ID* can be acquired from this list to configure individual Papillon Clients. Basic search and filtering functionality is accessible via the top right corner menu. On the left an access menue to various other data models is visible.

4.2 Power Estimation

The current state of power estimation used by the Papillon system needs to be defined and evaluated in terms of accuracy and efficiency. This is done in the following chapters of this thesis. To provide a uniform power estimation across platforms and server types a strict specification of all involved metrics, values, and their units is needed. Currently the Papillon system uses different definitions on Microsoft Windows and Linux-based OSs.

A Papillon power model combines CPU, disk, and network metrics with a power reading from a connected power meter during the power model generation. To achieve a uniform power estimation these values and their usage must be strictly defined in terms of measurement technique, units, sampling rate, and if required raw value processing of individual metrics.

The estimation algorithm used by Papillon is based on the k-Nearest Neighbours (k-NN) with a neighbourhood size of 10. Due to different metric units all values are normalised to the interval $[0, 1]$ before being used. This can be achieved by dividing all values of a given metric with the maximum value. To ensure consistency and prevent runtime errors a check for 0-values is in place. The resulting data points are then used to compute a weighted average power value which is the final power estimation for a given input. The weighting is based on the Euclidean distance of the ten neighbours to the input point. Since Papillon uses three metrics for estimations this can be represented by a three-dimensional space. The Euclidean distance d of two points p and q in three dimensions is defined as

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}$$

```
1  // input with arbitrary values
2  input = (cpu=42.314, disk=123456, net=987654)
3
4  // get power model for current server type
5  model = get_model_for_server
6
7  // find nearest neighbours, see Listing 4.2
8  neighbours = find_neighbours(model, input)
9
10 // square distance difference between input and neighbours
11 input = normalise(input, model)
12 neighbours = normalise(neighbours, model)
13 weights = (sqrt(3) - euclidean_distances(input, neighbours)) ** 2
14
15 // apply weights to power values
16 weighted_power = neighbours.power * weights
17
18 // average power with weights
19 power = sum(weighted_power) / sum(weights)
```

Listing 4.1: The Papillon Power Estimation Algorithm takes a tuple of activity metrics as input and estimates a fitting power value based on the assigned power model.

Listing 4.1 shows the basic power estimation code using vector- and matrix-based computations. The function to find the nearest neighbours uses the power model and the input point. Before actually searching for the neighbours the input point must be normalised into the value range of the power model. This is accomplished by using the maximum value of each metric and dividing the corresponding input value with it. Ideally the resulting values should be within 0 and 1. Below 0 would indicate an error, since negative values are not useful. A value above 1

would indicate that the power model has poor coverage, meaning that it does not cover this region of usage. The neighbours are then used to compute the weights by computing the distances of the input point to each neighbour and subtract it from the maximum distance of $sqrt(3)$. In order to deal with negative values the result is squared. This only happens if the power model coverage is too small for the given input point. The ten weights are then multiplied with their corresponding power values from the power model. The final step is to average the total sum of power with the total sum of weights.

Using a reasonably suited programming language or library capable of vector computations, this algorithm can be implemented in less than 15 lines of code. Therefore this is a very simple and efficient power estimation algorithm and which is easily replicable.

In order to find the ten nearest neighbours a simple version of the k-NN algorithm can be implemented using a sorted map data structure. The basic implementation can be seen in Listing 4.2.

```
1  // input and model are given as above
2
3  map = <new empty map>
4
5  for each point in model
6    normalised_point = normalise(point, model)
7    normalised_input = normalise(input, model)
8    distance = euclidean_distance(normalised_point, normalised_input)
9
10   map[distance] = point
11 end
12
13 return map.sublist(1, 10)
```

Listing 4.2: The k-NN algorithm used for finding the neighbours in a given Papillon power model using normalised values in the interval [0, 1].

This simplified implementation lacks basic error handling and fault tolerance, but indicates the algorithmic goal clearly. Using a data structure which is guaranteed to be sorted after each manipulation the final retrieval of the ten nearest neighbours can be achieved by simply returning the first ten entries in the map. Since the power value is relevant for the power estimation (see Listing 4.1) the map contains the distance as key and the corresponding point, including the power value, as value.

A caveat worth mentioning is that in case a point distance is an exact match to an already existing entry in the map, the point will be overwritten with the current point. This might falsify the result and would require a safe guard to prevent this error. A possible solution would be to append the new point to a list and once finished average all points and power values for a given distance entry if there is more than one point attached to it.

5

Design

In the following chapter the design of the performed work is described. The power model generation in general, as well as the new design for a workload generator and the model generation with the Papillon Client is explained in detail. The post processing step for generated power models is outlined, including a proposed workflow on how to integrate the new functionality into the existing system.

The current workflow to acquire a power model for a specific System Under Test (SUT) (physical server) described in Figure 5.1 can be divided into four steps:

Step 1
Running a special application on the SUT which records activity data and relates those values to an energy value from a connected power meter. The power meter must be hooked into the main power supply of the SUT to be able to read potential, current, power, and energy.

Step 2
Running a set of workload generating applications on the SUT to artificially create load and activity. Currently multiple different applications and predefined benchmarks are used to perform this step.

Step 3
After the artificial activity program run is completed, the data acquisition program can be stopped, which will then save the gathered values into a power model file.

Step 4
The newly generated power model is imported into the Papillon Server, to be used for power estimations. The import workflow depends on the used Papillon interface, which can either be a Graphical User Interface (GUI), Command Line Interface (CLI) or with raw access to the HTTP API.

Section 5.1 proposes a simpler approach on a combination of *Step 1* and *Step 2*. Additionally Section 5.2 introduces a more robust way on how to create artificial activity while also increasing the overall coverage of the power model. The recording of power values in combination with activity data will be incorporated into the Papillon Client to allow power model generations on live running systems without disrupting normal operation. During the simplification of *Step 1* a more suitable approach on generating artificial activity is introduced. The workload generator will cover a larger operational range in all dimensions of recorded activities, therefore leading to a higher accuracy during power estimation. This results in a significant amount of data points which would render a live power estimation computationally infeasible.

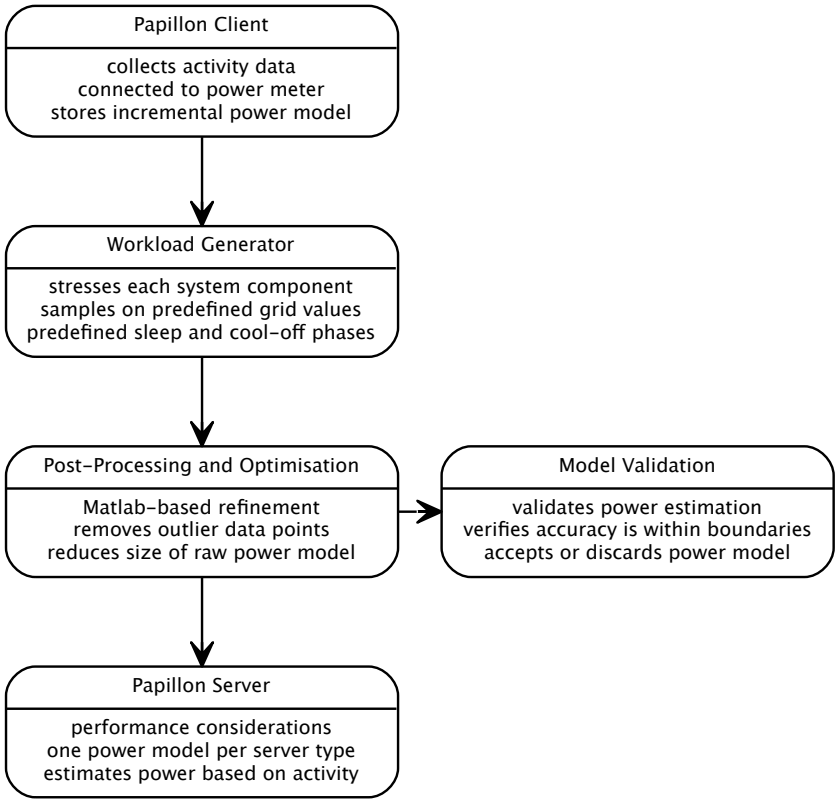


Figure 5.1: The basic workflow to create a power model which can be used later by Papillon in order to estimate power values for a given SUT.

The work proposed in Section 5.3 will introduce a post-processing step to optimise the raw data gathered for the power model. This post-processing will be inserted between *Step 3* and *Step 4*. The number of data points in the power model generated with the modification of Section 5.1 is about a magnitude larger than current power models. Using such power models without any optimisation would significantly decrease the performance of the Papillon Server during power estimation. This work will reduce computational complexity during power estimations without significant loss in accuracy.

5.1 Power Model Generation

The above mentioned *Step 1* is a simplified version of the actual workflow currently used by Stratergia Ltd. A significant part of the Papillon Client started out as stand-alone application to record system activity data and power values from a connected power meter. During the birth of the Papillon Client large portions of this code were reused, which created some sort of code duplication across projects. During multiple iterations of development and releasing new versions the code bases became more and more separate, without any actual change to the underlying data acquisition.

As a first step, to simplify the workflow, the mentioned applications are merged into a single code base with a proper interface to address the different modes of operations. During this merging process multiple parts of the existing client are refactored to streamline the behaviour.

Independent from the actual tool, which records the data during power model generation, more focus lays on the applications used to create artificial activity on the SUT. If the server is not utilised in a wide range of operational states, the resulting power model will not produce

accurate power estimations. Currently existing power models were created by using different benchmark-like applications, like *toibench*[1], *phoronix*[2] or multiple instances of *cpuburn* and *iperf* in various different configurations. The workload generation tool can be triggered from the Papillon Client to start and stop the power model generation with a single command invocation.

5.1.1 Activity Metrics

The main idea behind the modelling methodology of the Papillon system is to use activity metrics from a running server to estimate its power consumption in real time. This approach only works if the monitored activity is strongly correlated with power consumption. If the metric is not an accurate proxy for power consumption the total accuracy decreases. Therefore it is important to identify and select a set of activity metrics to get highly accurate real time power consumption estimations. The influence of each activity can vary significantly, therefore it is important to choose metrics with care.

Modern server system have a variety of sources to gather activity metrics from a running or halted system state:

Sensors
A very common sensor type measures temperature in specific areas of a device. Air temperature inside a rack or device enclosure are common sensor placements. Main power supplies usually detect overload conditions and might switch off if the temperature reaches a certain threshold. CPUs can scale their frequency and voltage according to the core temperature on the chip to prevent damage to the silicon die. These types of sensors are usually accessible through the OS or similar interfaces to be read by user applications. Other sensor types can be more limited or elementary. Device enclosures and openings in a rack frame can be monitored if an air vent is opened or closed.

Performance Counters
CPUs and memory controllers contain multiple registers to count certain events, e.g. cache misses, context switches, or interrupt rates. These types of counters are also called Performance Monitoring Counters and depending on the hardware are user selectable. Usually a restricted set of hardware registers is available to the user to be used as counter for a large list of possible events. Due to the hardware implementation this monitoring technique does not introduce any kind of overhead or additional computations to the main system.

Operating System Runtime Information
The OS needs to keep track of all processes and resources in normal operation. This creates a centralised structure where all information and metrics can be gathered and retrieved. In Unix-based systems this data pool is accessible via the `/proc` pseudo file system[3]. Microsoft offers a similar feature for their virtualisation hypervisor technology which allows for easy access to runtime information about resources and individual process-based metrics. A common type of metric would be the current CPU time consumption or memory usage per process. In addition to trivial metrics more complex queries can be created by combining multiple metrics.

External Sources
If a certain metric is not directly observable via the SUT itself the data can be gathered by an external device and sent to the SUT or Papillon Server directly. Some manufacturers

[1] `http://linuxperf.sourceforge.net/tiobench/tiobench.php`
[2] `http://www.phoronix-test-suite.com`
[3] `http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html`

embed physical power meters in their enterprise-grade servers and rack devices. Depending on the accuracy of these sensors the data can be used either instead or in addition to the already discussed sources.

The Papillon system currently uses three metrics in their models to provide power estimations. All metrics are measured and gathered by the SIGAR library[4] which provides a uniform interface across Linux and Microsoft Windows.

The active CPU time in percent is measured for each core and a total sum is used. Therefore the possible range for CPU values is from 0% to e.g. 800% for a 8-core CPU. Due to the way SIGAR operates and computes the internal CPU values, an average value is computed asynchronously to the main execution. Once every second the value from SIGAR is read and saved in a ring buffer for the last minute. Every time the execution path from the main Papillon Client requests a CPU value the ring buffer is queried and an average value over the last 60 entries is computed.

The second metric Papillon uses is hard disk throughput measured in bytes. Read and write operations are counted equally and therefore summed up. If the SUT is equipped with more than one hard disk, the total sum of all storage volumes is used. This might be topic for further research into Redundant Array of Independent Disks (RAID) systems (hardware- and software-based), as well as the unknown power consumption profile of SSD storage devices.

As third metric the network throughput in bytes is used. The Papillon Client can be configured to either sum up all networking interfaces currently active and connected or just use a single interface for the power model. A power model is therefore only valid for the same configuration of networking interfaces. Similar to hard disk throughput the network throughput measures incoming and outgoing traffic and treats them equally.

The correlation between these metrics and power consumption will be evaluated in Chapter 7. Additionally to this small set of metrics Mair et al. [Mai+14] compared various other metrics in terms of power correlation and therefore power estimation accuracy. A very low-level approach was taken by Yang et al. [Yan+14] by performing a Principal Component Analysis (PCA) to find the best metrics for power estimation.

Using memory consumption or throughput to predict power consumption was proposed by Chen, Wang, and Li [CWL11] and found to be a main contributor for servers with significant memory capacities. Depending on the real workload the power consumed by the main memory might be attributed to a combined CPU-memory metric.

Depending on the system the list of possible activity metrics can contain over hundreds of performance counters and sensors. To just name a few probably interesting metrics: CPU pipeline stalls, Arithmetic Logic Unit (ALU) operations, Memory Management Unit (MMU) accesses, fan speeds (if variable), temperature in various locations, etcetera.

5.1.1.1 Data Gathering in Live Systems

Depending on the OS used on the SUT there are multiple ways to access activity metrics. Data gathering is an important aspect for accurate power models because this task is also performed during the normal operation in the data centre. Therefore reading activity metric values must not interfere with regular workload on the server. The computational effort to collect these values must not be significant in a way that it would change the power consumption significantly. This means the Papillon Client must be as small as possible in terms of computational footprint and consumed resources. Due to the fact that the power model generation is performed as independent mode of operation of the Papillon Client these restrictions are fulfilled automatically.

[4] http://sigar.hyperic.com

The only additional work to be done during power model generation is the creation of a file with the collected activity and power values, which is considered to be a trivial task.

On Microsoft Windows systems the OS provides a common interface to access a wide range of performance metrics and other related date of a running system. Windows Management Instrumentation (WMI) is an interface and extension[5] to the Windows Driver Model which provides low-level access to system components. Special drivers can extend and add new information to be accessed via the WMI. Microsoft provides a variety of programming language bindings to allow for easy access to external applications.

Unix-based systems usually follow the Unix philosophy and provide access to performance counters and runtime information via special files located in the `/proc` pseudo file system. These files contain information in a plain text format and are therefore easily accessed and parsed by external applications. Depending on the security measurements access to these files might be restricted for certain users. Using the guidelines for pseudo file systems this approach is available on almost all modern Linux-based systems in a uniform and standardised way.

In order to provide a cross-platform solution the Papillon Client uses the above mentioned SIGAR library. This tool provides an abstraction layer on top of platform-specific interfaces and programming language bindings. A uniform interface is published and different value units are automatically converted to ensure a homogeneous programming experience. SIGAR uses the WMI on Microsoft Windows and the `/proc` pseudo file system on compatible Unix-based systems. This allows for a very simple implementation of the Papillon Client, because cross-platform capabilities are achieved by using a single interface and library instead of adapting the code base to each platform.

Hardware performance counters are usually accessible as special CPU instruction to read a custom set of registers holding information about pre-defined events. These events can be selected by the user and evaluated at any time. Due to space limitations on the chip the number of registers to record events is much smaller than the number of possible events. Therefore the user has to know exactly which events are important. Chen et al. [Che+10] make heavy use of these hardware performance events and counters by evaluating different sets of events to be used as input for a performance and power model.

The existing Papillon Client implements the system and runtime information gathering in different ways on Microsoft Windows and Linux-based OSs. For Linux-based systems the pseudo file system `/proc` and a combination of various command line-based activity monitors are used to read activity metrics, while using the SIGAR library on Microsoft Windows. Since the usage of SIGAR is already established in the Papillon Client it is of no benefit to implement the same functionality again using `/proc` on Linux-based systems. SIGAR uses `/proc` already, therefore the current Papillon Client duplicates this functionality unnecessarily and introduces unit mismatches between Microsoft Windows and Linux-based systems. A Papillon Client on a Linux-based system reports CPU usage in a value range from 0% to 100%, independent of the actual number of cores. This is a violation of the Papillon power model specification stating that CPU activity metric values must for example be within 0% and 800% for an 8-core CPU.

This results in the fact that power models created for one OS are not valid on the other. Simply removing the custom `/proc` reading and parsing in favour of SIGAR on all platforms will solve these problems and allows for the generation of a cross-platform power model with correct activity metric units.

[5] `https://msdn.microsoft.com/en-us/library/aa394582`

5.1.2 Test Setup During Model Generation

The SUT is connected with a physical hardware power meter to the main power supply of the data centre. Only the necessary components and devices for this specific SUT should be connected to the power meter to reduce measurement error introduced by non-related power consumers. Depending on the actual power meter various different cables must be connected to measure voltage and current between the wall socket and the SUT power supply.

All fans related to this system must be connected and set to their default speed if possible. Any fans with variable speed, which are controlled by the system, might influence the power estimation accuracy by up to 20 Watts depending on the maximum power consumption of each fan. Spinning hard disks must be monitored as well. Some consumer-grade hard disks are equipped with special energy saving states which affect the accuracy as well. Screens and input devices should be on a separate power supply because they are usually not active in regular operation mode. If the SUT is determined for a highly environmentally controlled area of application the test setup should resemble these temperature and humidity conditions as close as possible.

The running OS should not be relevant to the power consumption of the device, but different kernel versions and driver vendors might implement various energy saving states which can affect the accuracy slightly. It is recommended to choose the same OS during the power model generation as used for regular operation in the data centre. Automatic background processes such as downloading and installing updates, executing of cron jobs, backup systems and any other non-essential applications should be disabled and terminated prior to the power model generation. Only the bare minimum OS and purely essential processes are allowed running during the measurement period.

Creating a new power model for a given SUT consists of the above mentioned preparations and running a single application on the server for an extended period of time, up to multiple days for CPU-focused devices. The estimated time needed to generate a full Papillon power model is described in Section 6.2. The measurements, consisting of the actual power value and the recorded system metrics, are saved incrementally to a predefined location on the SUT or any reachable server over the connected network. This allows for pausing and resuming the power model generation if needed. The final power model is saved as XML file which can be used in the Papillon Server for power estimations after running it through the post-processing and optimisation pipeline described in Section 5.3.

5.2 Workload Generation

Step 2, of the current workflow to acquire a power model, is to create artificial activity on the SUT. This means that all components relevant for the recorded activity metrics must be stressed in various combinations and load factors.

In case of Papillon the recorded component metrics are CPU busy-time, disk throughput, and network throughput. There are multiple approaches on how to address the issue of generating activity. This work focuses on a dense and full coverage in the operational space.

The operational space can be seen as a three-dimensional space (related to CPU, disk, and network) in which a number of data points are distributed. Each data point holds its own coordinates and additionally the recorded power value. The coordinates for each point originate in the SUT and are read via hardware and OS performance counters. The power value is integrated with an external power meter connected to the SUTs power supply while also hooked into the system itself to transfer data over a serial connection.

5.2.1 Sampling

In order to generate a power model with high accuracy the workload used during generation must cover a wide range of operational zone (3D space). This includes a vast amount of points, spread over as much space as possible, which are distributed in a semi-ordered way. For purposes of simplicity and effectiveness a grid-like approach is chosen to allow sampling of data points in a structured and procedural fashion.

The operational zone is defined by the three recorded metrics, ranging from 0 (meaning no utilisation at all) up to the maximal value for each metric. To illustrate this definition the following example is given:

CPU from 0% to 800%
(in case of a 8-core processor)

Disk from 0 bytes to 48.318.382.080 bytes (46.08 GB)
(in case of a 6Gbit SAS attached disk over 60 seconds)

Network from 0 bytes to 8.053.063.680 bytes (7.68 GB)
(in case of a 1Gbit network interface over 60 seconds)

Some of these values a theoretical maximums, which are probably not reached with a real workload due to protocol overhead, OS-specific implementations and so on. This mainly applies to the network metric, but as well as the disk metric on some minor scale. The maximum value for the CPU metric usually can be reached comparably easy.

One minor caveat regarding the CPU utilisation is the metric itself. For power-related experiments the total CPU time is usually considered. In some cases one might only want to measure the utilisation for user-space applications (e.g. a database or web server), or only time for kernel-space applications (hardware drivers or OS-maintenance itself).

Due to increasing performance in respect to parallel computing on a single core of a modern processing unit another consideration must be taken into account for technologies like Intel's HyperThreading implementation, which basically allows a single CPU core to handle two OS threads to run at the same time (in theory). This means a single physical core can amount for up to 200% of virtual computing power.

The design for a workload generator to be integrated into the Papillon workflow includes a total CPU time metric (including kernel-space and user-space) as well as the number of bytes per minute for the disk and network separately. Although as previously stated the disk and network utilisation might not be fully covered, the results discussed in Chapter 7 will explain the deviation from the theoretical maximal values.

Statistical anomalies during measurements and non-deterministic influences of the OS might interfere with the power model data acquisition. Therefore it is necessary to sample multiple data points for each grid point with a specific required utilisation level. The number of data points for each grid point can be reduced later on to reduce computational efforts during power estimation, as described in Section 5.3.

5.2.2 Resolution

Depending on the used tools and techniques to generate the individual workloads for each component different granularities of each activity metrics can be applied to distribute the data points across the operational space.

CPU
The basic principle of cycling through different levels of utilisations, characterised by a percentage value, is encapsulated in an outer layer of cycling through the number of virtual cores. This allows a full utilisation of all cores, including virtual HyperThreading-cores, with various levels of usage. Utilising a given core is straight forward and can be done by simply running algorithms on random data, e.g. multiplications of large numbers.

Disk
Due to the different nature of Input/Output (IO) for disk devices the generation of workload is more delicate than compared to CPU. The conceptual notion of operation is the same. To create artificial workload on a disk device, the disk has to perform operations, which means simply to read and write a file from or to disk. The total number of read or written bytes should then correspond to the overall measured throughput. File system, OS-level, and hardware-level overhead will reduce the maximum value to a certain degree. This does not affect the final power model, because the same restrictions and limitations apply during normal operation of the SUT in a data centre.

Network
Disk and network are very similar in their nature of metric, both measure an amount of bytes transferred between the component and the remaining system. For network one must send and receive data via one or multiple NICs to generate workload. Again, the overall achievable maximum value for this metric might be (significantly) below the theoretical maximum, which does not affect the outcome in the power model and estimation accuracy.

The coverage and granularity of the resulting power model is solely based on the intervals in which the individual metrics are exercised. Assuming a range of 0% to 100%, one reasonable interval might be to use 10% steps, resulting in 11 separate steps of 0, 10, 20, ..., 90, and 100%. Depending on the desired density and coverage of data points different intervals and steps can be chosen on a per metric basis.

Due to the integrating nature of the power measurement over a period of one minute, the workload itself should be applied equally over this time. Otherwise power spikes and other non-deterministic effects might impair the power model. This means the utilisation should be as uniform as possible on the SUT, without large chunks of time just idling. If the workload requires a certain amount of idle time to achieve a specific level of utilisation, this idle time should be split into smaller chunks and distributed evenly over the available measurement time window.

5.2.3 Outline for Papillon

For the Papillon integration of a new workload generator the following values have been chosen after conducting initial trials and evaluations:

Sampling
Each grid point must be sampled four times, resulting in 240 seconds spent on each utilisation level. To minimise the effects of interference between the utilisation of neighbouring grid points each grid point must be prefaced with a cool-down period of 90 seconds. This allows the SUT to settle down from previous workloads (temperature, fans, different power states) and provides a common base line as starting point of each grid point sampling.

Resolution for CPU activity
For modern CPUs with highly parallelised computation pipelines a total number of two

workload threads per physical processing core must be used. Due to the fact that the OS will only see virtual cores, the workload generator must only start one utilisation thread per available core. Each thread must then cycle through levels with 25% steps from 0 to 100%.

Resolution for Disk & Network activity
IO is domineered by multiple levels of caching and optimisations on the OS and hardware level. It is therefore necessary to balance these effects over the measurement time frame. The workload generator should use 25% steps from 0 to 100% as well as for CPU. On account of the different utilisation techniques available it must do so by writing multiple large files for a predefined amount of time, then idle and start over with writing a large file again. This relates to a Pulse-Width Modulation (PWM) approach by cycling through states of writing and sleeping in different time intervals.

Before and after the actual workload is run an idle period of 20 minutes is added in which the system and power meter can warm up and settle into a steady-state without affecting the measurements in any way. This idle period can later on be used to determine a system idle power consumption.

5.3 Post-Processing And Optimisation

After a power model is produced through the previously described steps it is necessary to perform multiple operations on the data to reduce the amount of data points, and therefore increase the estimation performance. A Matlab-based pipeline is used as prototype to illustrate all the procedures described in the following sections.

Depending on the hardware configuration of the SUT a single power model can consist of multiple thousands of data points (up to low 5-digit numbers). This results in a high computational effort during power estimation. Computing point distances in three dimensions, as well as finding and weighting the nearest neighbours to get a final estimation value is therefore a non-trivial task, in respect to mathematical labour. In contrast, the algorithmic complexity is well understood and easily manageable in different programming languages and frameworks.

The Papillon system is designed to monitor many thousands of physical and virtualised servers with a single Papillon Server, which holds a power model for each server type and performs the power estimation each minute for each server. Thus it must be a main goal to make a single power estimation as fast and as easy on resources as possible, without decreasing the accuracy of estimated power values in any way.

The overall workflow of post-processing is shown in Figure 5.2 and can be described as pipeline with a single data input and a final optimised power model as output. This pipeline can be implemented as part of either the Papillon Server itself or as a stand-alone tool for easier access and usability. An automated integration into the Papillon Server makes sense from an overall system workflow perspective. This would reduce the number of needed steps and manual interventions for new server types.

5.3.1 Pre-Computing Values and Metrics

Due to the way the Papillon power estimation algorithm is designed, it makes sense to pre-compute certain values and metrics of the power model, to reduce the amount of repeated work done for each estimation. This includes the range normalisation in the three dimensions of the data point cloud, as well as finding the minimum and maximum values in each axis. This work

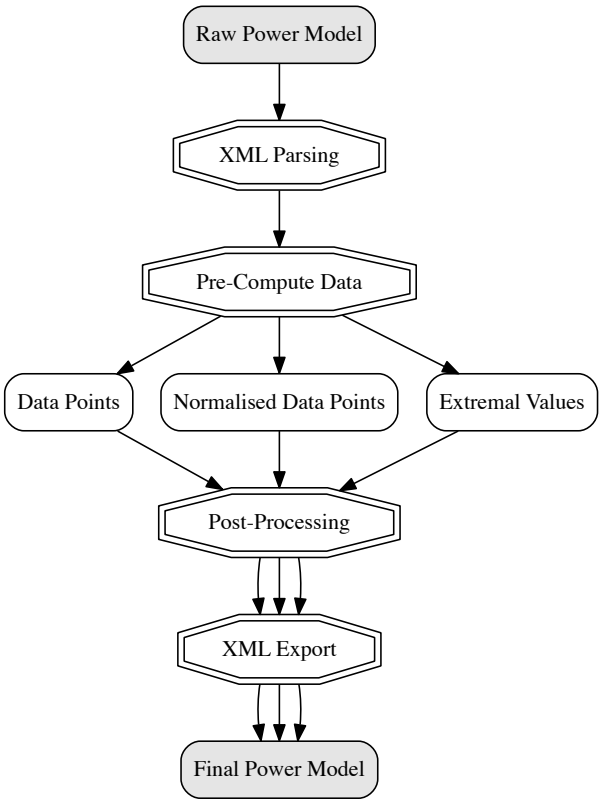


Figure 5.2: The basic algorithm to run the post-processing and optimisation on a raw power model. This outlines the data flow through various algorithms and steps before the final power model is generated.

can be computed once, as part of a post-processing and optimisation step, and then the results can be included in the power model, before sending it to the Papillon Server for estimation purposes.

5.3.2 Outlier Removal

During the long-running workload generation described in Section 5.2 it may happen that a low number of data points will not fit well into the surrounding environment. This can happen for multiple reasons, many of them not deterministic or controllable. For instance the OS might perform certain background jobs, such as defragmentating the hard drives or downloading application updates. If possible such automated tasks should be disabled beforehand. In the case of specialised hardware components it is not feasible to disable this functionality without interrupting normal operation, e.g. random RAID integrity checks.

During various experiments performed as part of this thesis it happened to be nearly impossible to control everything, therefore the most feasible approach was taken by simply identifying and removing these outliers from the data set.

The value range in each axis is taken under consideration to be used as indicator if a data point is part of the regular environment (within the expected operational space) or not (a non-deterministic event occurred).

The metric for outlier detection is based on the point-to-point distances in the point cloud. For n data points there are $n \times n$ euclidean distance values to be compared against each other. The sum of distances of one point to all other points can be computed and compared against each other easily. The higher this total distance sum per point is, the more unlikely this point

is to be part of the regular environment. Therefore the 95th percentile can be used to divide the points into valid data points and outliers.

Due to the fact that each axis can have different maximum values it is necessary to perform a range normalisation beforehand to get comparable metrics, e.g. in the range of $[0, 1]$.

With empirical data from early experiment runs an outlier removal strategy based on the 95th percentile seemed appropriate. This might be specific to the used measurement tools and hardware, but can be easily adapted to achieve stable results.

5.3.3 Reduction of Redundant Data Points

The number of data points in a raw power model might be not suitable for large Papillon environments and should therefore be reduced as much as possible, without decreasing the accuracy in a significant way. The reasons for a high number of data points in power models are in direct relation to the sampling design decisions described specifically in Section 5.2.1. The effects of this high sampling rates can be controlled to some extent by thinning out the point cloud in the power model at hand. This means identifying similar data points and pruning or combining them from the final power model.

Similarity of data points is defined by plain position coordinates of each point in 3-dimensional space. The power value has no direct impact on similarity, but might be used as a second-order labelling approach.

An abstract point of view on identifying redundant data points can be seen as a well-known clustering problem. A given point cloud must be divided or labelled in such a way that each point is assigned to a single cluster. The number of clusters should be defined beforehand. After the clustering is completed each cluster should contain one or more points which are considered similar, according to the above definition. It is therefore safe to combine all points from within a single cluster into a single data point, without loosing too much information or decreasing final power estimation accuracy. In an ideal power model and measurement environment each grid point (see Section 5.2.1) should be represented by a single cluster containing exactly all data points sampled for this workload. This leads to the safe assumption that combining and removing these points will not deteriorate the final power model.

In case of Papillon this means that each cluster should contain four data points which can be replaced by a single point. The new data point can be computed by the same method used for power estimation, using a weighted sum approach with the power value. This results in a theoretical power model size reduction of 75%. The effects on accuracy using this ambitious simplifications will be discussed in Chapter 7. For clustering techniques the following two algorithms are considered:

(Multivariate) Gaussian Mixture Models [FJ02]

This kind of probability density function is parametric and represented by multiple single Gaussian distributions superimposed onto each other to form a larger probabilistic model. The model to fit a power model point cloud would be to represent each grid point as a supposed Gaussian centre point and check with which parameters the nearby data points would fit into each component. The mean values and variances of each component must be evaluated and checked against a threshold range to determine if a finished fitted Gaussian Mixture Model (GMM) performs well under the given parameters. The GMM is usually considered in unsupervised machine learning problems with non-labelled input.

A requirement for plausible results is that all input data is represented by observations with a normal distribution. This might not be the case for Papillon power model data points, since initial trials indicate a tendency to underestimate power, i.e. a bias towards

lower than average values. Therefore using a GMM might result in large deviations from the actual raw data distribution. Although the clustering algorithms for GMM are part of most mathematical software packages and distributions, they are commonly not included in standard application frameworks, which makes an integration into a tool chain more complex.

k-Nearest Neighbours [Alt92]

The k-NN algorithm is mainly used in pattern recognition tasks. In the application of Papillon the k-NN is used in a regression-based fashion to use the output as power estimation value. This non-parametric method is simple to implement and yields good results for broad application purposes.

Due to the nature of k-NN the locality and grouping of data is of great importance to the expected outcome. In the case at hand with a normalised 3-dimensional feature space the Euclidean distance is a safe metric for comparison. The final value is a combination of the inverse weighted distances of the chosen neighbouring points. The neighbourhood size is the only parameter to chose, which affects the outcome drastically.

Since the Papillon power estimation is based on the k-NN it makes sense to reuse this algorithm in the post-processing to eliminate redundant data points. The output of the regression is used to replace all points by their respective k-NN centre point and the corresponding average power value. A possible future research topic for Gaussian Mixture Models is outlined in Section 8.3.

6

Implementation

In order to create a working prototype of the proposed system, including the changes described in the previous chapters, the detailed implementation of the adapted power model generation workflow, the workload generator, as well as the composed post-processing steps are described in this chapter. For each part a detailed structural analysis, as well as explanations for minor changes to the outlined design are given. The implemented system is then evaluated and used to get experimental results to compare the accuracy and overall performance in Chapter 7.

6.1 Power Model Generation

The Papillon Client and the power model generation tool are currently two separate applications and projects, although their code base is rather similar and large portions of functionality is implemented in both projects. The new design for the Papillon Client therefore merges the existing two projects and removes redundant code by restructuring and simplifying the overall component diagram.

The responsibilities of the new Papillon Client can be isolated into four main components, each defined with a fixed and small interface for interaction between them:

Main
: The main component holds the basic program functionality of the Papillon Client. This includes parsing of command line parameters and configuration options. Depending on the activated features the other components are initiated and started as independent threads. For inter-thread communication a small set of shared data structures is created and initialised.

Data Collector
: The collector component periodically queries the activity metrics using the SIGAR library and adds the resulting state entry into a queue for further processing. A state entry consists of the CPU, disk, and network activity metrics as well as a current time stamp. Additionally application entries are collected to allow power estimations for individual user application.

Data Sender
: The sender component periodically reads a state entry from the queue and sends it over the network to the Papillon Server. Each message is encoded in XML or JSON depending on the configuration. If the queue is empty the sender thread waits for the next state entry.

Power Model Generator

The power model generation component combines the state entries created by the data collector with a real power value read from an attached physical power meter and writes the result into a power model file encoded in XML. Since the power meter is directly connected to the SUT a software abstraction is in place to provide a well defined interface to query the power meter for integration values over 60 second time intervals.

Due to the simple structure and interconnection of these components they can be implemented directly as Java classes and threads. Figure 6.1 shows a basic Unified Modeling Language (UML) diagram of the above discussed components with the shared resources and available methods.

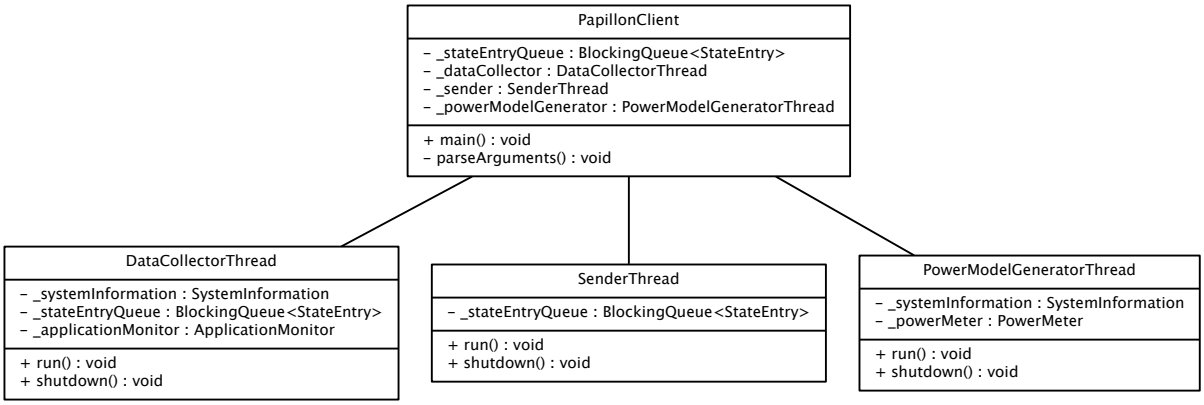


Figure 6.1: The main architecture in terms of classes for the Papillon Client is shown in a simplified UML diagram.

Additionally to the above mentioned components various helper modules are in place to reduce the complexity in the individual modules. Each of them is encapsulated into Java packages and tested separately with unit and integration tests.

6.1.1 Data Encoding and Formatting

Due to different customer requirements the Papillon Client is capable of communicating to the Papillon Server in multiple formats. The Papillon Server currently provides a REST API endpoint for either XML or JSON data. This means any client communicating with the server needs to encode and format the data accordingly. In normal operation the Papillon Client performs only a single REST call to send a new state entry with current activity values to the server.

This fact is represented by a `DataFormattter` singleton in the code base, which uses a strategy pattern to switch the different encoding and decoding formats. A Java interface is provided to easily extend the required methods and abstract the components. The strategy to be used is determined by the current configuration of the Papillon Client and can be switched at any time.

6.1.2 System Information and Metrics

The SIGAR library is encapsulated in a `SystemInformation` class which combines and transforms SIGAR values to a Papillon-usable format. Due to the long list of supported platforms and OSs by SIGAR the custom Linux implementation using the `/proc` pseudo file system is removed from the Papillon Client and only the SIGAR library is used on all supported systems.

Therefore small portions of the Java code need to be rewritten for the Linux system to get a uniform interface across all platforms. Kampl [Kam13] performed an extensive comparison and analysis on the accuracy of SIGAR and the `/proc` pseudo file system. The mentioned publication could not find a significant difference in the values and accuracy tested on Microsoft Windows and Linux-based systems. Therefore this code duplication is removed in the Papillon Client and only the SIGAR library is used for all available platforms.

The CPU activity metric is averaged over the last time period of 60 seconds. To accomplish a smooth average value without any major spikes a special thread continuously records the CPU metric every second and stores the value in a queue which only holds 60 entries at the most. Every time the data collector or power model generation component request a CPU value from SIGAR, the request is actually answered by the `AverageCPUInfoThread` which computes the average of all items in the queue which always contains the last 60 entries - after being fully initialised. This allows for a smoother CPU metric. Before this code refactoring the SIGAR library only reported the CPU utilisation value within the very last second, which did not yield accurate results over a 60 second measurement period.

6.1.3 Configuration

The configuration of a running Papillon Client is determined by three different possibilities in the following decreasing priority order:

- CLI parameters
- plain-text-based configuration file: `papillonclient.properties`
- hard-coded default values

All configuration options are documented and accessible with well-known command switches. The CLI as well as the configuration file provide the same set of options to configure a Papillon Client instance on a single machine. There are only three mandatory parameters which need to be present and valid:

Papillon Server IP defines the IP address or resolvable domain name under which the Papillon Server is reachable.

Papillon Server Port defines the port on which the Papillon Server is accepting new connections.

Host ID defines the ID of the current host machine. This parameter must be different on each system to distinguish and assign state entries correctly in the database.

The Papillon Client has two basic modes of operation, the regular mode for reporting activity metrics to the Papillon Server, and a second mode which generates a power model. The two modes can be run together or independently at the same time. While the regular mode is available at any time, the power model generation mode only works if a compatible power meter is connected and hooked up to the SUT properly.

6.1.4 Self-Registration of Clients

Currently every Papillon Client must be configured with the correct *Host ID* corresponding to the database entry in the Papillon Server. This is necessary to assign each state entry to the correct host and therefore power model for further processing. This information must be

included in each request, due to the active client structure. An active client periodically sends activity information to the server, whereas an active server would periodically pull information from all clients. This design decision prevents a data centre operator from simply installing the same pre-packaged Papillon Client on all devices in a data centre, because each one must be configured with a different (e.g. the correct) *Host ID*.

For this reason the self-registration feature was added to the Papillon API. A Papillon Client would register itself with a Papillon Server during start-up and initialisation. The Papillon Server would then respond with the assigned *Host ID* which can then be used for all further communication of state entries. This shifts the problem of client identification from the initial installation to each time the Papillon Client is started. The initial implementation expects the Papillon Client to send its own IP address for initial identification, due to the fact that each host entry in the database contains the IP address and *Host ID*.

Contrary to the intended idea, this solution only pushes the problem one level deeper into the service hierarchy. Using the IP address as pseudo unique identifier in the Papillon host list produces multiple unsolved issues. Despite the implementation of a working self-registration prototype, this feature is not yet ready to be deployed on a data centre-grade scale due to the following issues:

IP addresses may not be unique
: Depending on the network topology in a data centre, or even across data centre boundaries in a globally distributed cluster, different networking schemas can be applied. This could create duplicate IP addresses within a single Papillon-controlled environment. This therefore limits the size of a Papillon-controlled environment to a single uniquely defined IP address space.

IP addresses can change dynamically
: Using techniques such as Dynamic Host Configuration Protocol (DHCP) or IPv6 prefixes a physical device might not be identifiable at every given moment in time. This leads to partial mismatches and overlapping *Host IDs* on different Papillon Clients. Depending on the DHCP system in use the Papillon database must be kept in sync with the currently assigned IP addresses.

Servers with multiple IP addresses
: Data Centres usually employ redundant network interfaces and routes to their connected devices. Therefore a single server might be connected via two or more interfaces to the same network using different IP addresses. This leads to confusion on the Papillon Server, as well as the human operator side, who has to keep track of the registered address for each server and network interface. Prioritising between networking interfaces must be kept in sync with the Papillon database and in case of a fail-over solution the *Host IDs* will mismatch.

IPv4 and IPv6 have different notions of valid addresses
: The definition of a reachable IP address is not very strict in terms of IPv4 and IPv6. In IPv4 one must consider private networks[1] and public route-able addresses. On the other hand IPv6 distinguishes between various types of interface-local, link-local, and site-local scopes[2]. All these caveats complicate the self-registration feature further.

The above described issues with self-registration are a fundamental problem of using an IP address as identification mechanism for hosts in a Papillon-controlled environment. Finding

[1] https://tools.ietf.org/html/rfc1918
[2] https://tools.ietf.org/html/rfc4007

a stable and unique identifier is a non-trivial task depending on an arbitrary chosen level of complexity.

6.1.4.1 Proposed Solution

To re-use as much as possible from the existing API endpoint and server-client communication path the requirements for a simple solution are as follows:

- A single Unique Identifier (UID) for each Papillon database host entry.
- Papillon Client can derive the UID from hardware and/or software configuration during runtime.
- Installation packages are universal and do not require the configuration of *Host ID*.

Since each host is connected to the network via one or multiple network interfaces a UID is already present for each of these interfaces: the Media Access Control (MAC) address.

A MAC address is assigned to a network interface and qualifies[3] as UID. Therefore most of the problems and issues outlined above are solved by simple switching from using IP addresses to MAC addresses. The one remaining issue is to deal with multiple network interfaces in a single host. There are multiple solutions to this, for instance:

Using the first one
Every OS keeps track of the available network interfaces and provides a simple way to enumerate them. Usually these interfaces are labelled like `eth0`, `eth1`, etcetera. By using simply the first interface in the list the UID should be stable unless the hardware configuration changes.

Using the lowest/highest address
If the enumeration of network interfaces by the OS is not stably ordered or changes randomly, the list of available interfaces can be sorted lexicographically by their MAC addresses. This always yields a stably ordered list, from which either the lowest or highest MAC address can be chosen as UID.

Using a fixed interface
Depending on the management and structure of the data centre it is possible to guarantee that each host is equipped with a network interface called e.g. `eth0`. With this prior knowledge the data centre operator can pre-package the Papillon Client to specifically target this network interface and use the MAC address of the `eth0` interface as UID for self-registration.

The necessary changes to the Papillon Server are minimal because only the field for storing the UID must be renamed from IP to MAC address. The Papillon Client must be adapted to choose the correct MAC address at any time. The last option (using a fixed interface) was implemented to demonstrate the problem and possible solution to the self-registration feature. Therefore the Papillon Client has an configuration option to select the network interface.

[3] `https://tools.ietf.org/html/rfc5342`

6.2 Workload Generation

The foundation for the workload generator is straightforward and can be described with the code snippet in Listing 6.1.

```
for cores = 1 upto total_cores with step 2
  for cpu = 0 upto 100 with step 25
    for disk = 0 upto 100 with step 25
      for net = 0 upto 100 with step 25
        // run actual workloads in parallel
      end
    end
  end
end
```

Listing 6.1: The workload generation algorithm used to deterministically create workload based on grid points.

Inside the nested **for**-loops the individual tools for workload generation are invoked and run in parallel. Concurrent to this a countdown is started which goes off once a grid point is completely sampled (e.g. after 4 minutes). The workload tools are terminated and the cool-down phase is initiated. After cooling down the next grid point can be sampled.

Due to the vast amount of platforms and different OS-level support for different languages the most portable environment was chosen to be POSIX shell, which should be present on all Linux distributions in a standardised and portable implementation. All used tools are available for this system and can be called and manipulated to the need at hand.

It particular for each component one specific workload generation tool is chosen to create artificial utilisation during power model generation. For CPU and disk activity a tool called *stress-ng*[4] is used. Network activity is created with *iperf*, which needs a second host acting as data source and sink. Both tools expose a vast amount of configuration options and command line switches to tune their respective behaviour. This combination of both tools allows the new workload generator to utilise all recorded metrics simultaneously. Listing 6.1 provides the basic scaffold which then uses *stress-ng* and *iperf* to run the actual workloads inside the nested loops in parallel. Since the workload generator is a POSIX shell script this parallelism is achieved by running the workload-generating tools and functions in the background using the `&` operator to detach it from the shell.

The online manual[5] of *stress-ng* states the intended use case as follows:

> stress-ng will stress test a computer system in various selectable ways. It was designed to exercise various physical subsystems of a computer as well as the various operating system kernel interfaces. stress-ng also has a wide range of CPU specific stress tests that exercise floating point, integer, bit manipulation and control flow.

In case of CPU activity the tool *stress-ng* comes with a broad list of possible actions to execute called *stressors* in their own terminology. Each stressor uses a different technique to create artificial workload, ranging from different mathematical to various memory access operations. Due to the nature of modern CPU architecture it might lead to substantial differences if certain regions of a CPU are not active or only run for a short time period. See the evaluation in Section 7.4 for a more detailed explanation on this behaviour. Since *stress-ng* already comes with

[4] http://kernel.ubuntu.com/~cking/stress-ng/

[5] http://kernel.ubuntu.com/~cking/stress-ng/stress-ng.pdf

command line switches to specify the number of cores and their desired utilisation percentage the implementation as workload generator is straight forward and without any caveats.

6.2.1 Duty Cylce for Non-Continuous Metrics

Generating a deterministic workload for disk and network is a more challenging task with the given tools. Compared to CPU it is not possible to simply set a desired percentage value and keep the system utilised at that level. For these throughput-based metrics, such as disk and network, there is only a single setting, either active or non-active (on or off). The device will work at full capacity once it is turned on, and will settle down into a sleep state once turned off. It is therefore necessary to apply a technique mostly found in electric motor management called Pulse-Width Modulation (PWM) or the related term *duty cycle*. A duty cycle describes the percentage of an active continuous signal in one period. PWM used as mode of operation works by turning the device on and off in short pulses which will result in short bursts of activity. These bursts will then average itself out over a certain time period under observation, see Figure 6.2. Applied to the problem at hand the disk or network device is not turned on for the full time period, but on a per-second basis. The ratio of on-off phases can be easily calculated to achieve a specific total percentage value.

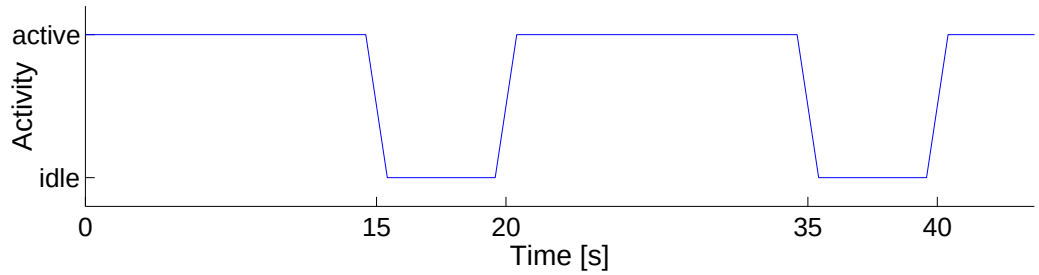


Figure 6.2: The workload goes through an active and idle phase in certain repeating intervals to achieve a total duty cycle in percent. The on-off periods average out after a longer exposure time. This graphs shows a duty cycle of approximately 75%.

Workload in terms of disk throughput can be accomplished by reading or writing a large file to the disk. Due to the fact that various OS drivers and file systems might verify each written block by re-reading it, it is not necessary to differentiate between read and write activity. Preliminary experiments show that there is virtually no difference between different modes of operations (read or write in various sized blocks). A workload therefore consists of writing a single large file to the disk. The file size must be as large as needed for the operation to take up more time then the sampling period. It is worth noting that the location of the written file system is actually on the selected disk, and not in a specially mounted location like a memory-mapped device (*tmpfs*) or special OS file system like `/dev/null`. If the disk is part of a RAID the overall outcome should not be affected, but one has to keep in mind that the RAID itself should be used during the power model generation, because this setup will likely be used in normal operation as well.

Similar demands hold for network throughput. PWM can be used again to create a deterministic percentage value of utilisation. Depending on the used protocol, TCP or User Datagram Protocol (UDP), slight variations in the test environment must be made to account for the different workloads and required external equipment. Packet loss and re-transmission might be an issue during power model generation and should be calibrated before actual usage.

Preliminary empirical tests yielded that the following values perform well under a four minute sampling period. The mapping for percentage values to PWM steps in seconds is as follows:

$$\mathbf{25\%} = \begin{cases} 5\,\text{sec} & \text{active} \\ 15\,\text{sec} & \text{idle} \end{cases} \qquad \mathbf{50\%} = \begin{cases} 10\,\text{sec} & \text{active} \\ 10\,\text{sec} & \text{idle} \end{cases} \qquad \mathbf{75\%} = \begin{cases} 15\,\text{sec} & \text{active} \\ 5\,\text{sec} & \text{idle} \end{cases}$$

6.2.2 Cool-Down Phase Misalignment

Due to the cool-down phase after each sampling the measurement period gets misaligned with the actual workloads which leads to the desired effect of better covered power models and more evenly distributed data points. If the sampling would be perfectly aligned to the workload the resulting data points would very likely end up in a very small space leading to virtually no information gain. This technique was first proposed by Mair et al. [Mai+14] to reduce thermal effects on the SUT. The manual for the hardware power meter states a similar requirement to achieve accurate measurements. Depending on the environmental situation the server might need a warm-up phase in the beginning and end of the power model generation as well as in between individual sections of workloads.

This is independent of the global 20 minute cool-down phase in the beginning and end of the power model generation used to determine the idle power consumption of the SUT.

6.3 Post-Processing and Optimisation

The post-processing of raw power models is implemented as Matlab-based script collection with a single input (i.e. the raw power model), and a single output (i.e. the optimised power model). A streamlined workflow is provided as part of the optimisation tool chain. A second collection of scripts and functions is implemented to visualise power models to be viewed and inspected by a human operator. An optimised model is marked so after successfully completing the tool chain. Both file formats (input and output) are plain Papillon power models, represented in XML. In the current state of implementation a single command is provided, which gives an optimised power model file as output after processing the data. As part of future work the algorithms and optimisation computation can be integrated in the Papillon Server to streamline the usability of the tool chain.

Matlab provides a quick and easy-to-use Domain-Specific Language (DSL) for data manipulation. Due to the fact that a Papillon power model can be represented as vectors and matrices of floating point values this approach fits perfectly into a Matlab-based prototype. An implementation in a more structural or object-oriented language would not allow for the fast and simple usage patterns Matlab provides. Therefore implementing the post-processing and optimisation pipeline as prototype in Matlab is suitable and lays a good foundation for further improvements and tinkering. Once a solid code base is established the Matlab-based pipeline can be ported and integrated into the Java-based Papillon system.

6.3.1 Pipeline Stages

Figure 6.3 shows the flow of data through the various post-processing and optimisation steps. The first stage is parsing the input file into easy accessible Matlab data structures for further manipulation. During the XML parsing all additional values like extremas and normalised activity values are computed and stored together with the raw input data. Most of the following algorithms operate on the normalised activity value. Therefore every time the data changes the data set is re-normalised to keep raw and normalised data in sync to each other.

Outlier removal is done by comparing the point-to-point distances in normalised space, computing the total sum of distance per point and discarding all points with a sum above the 95th

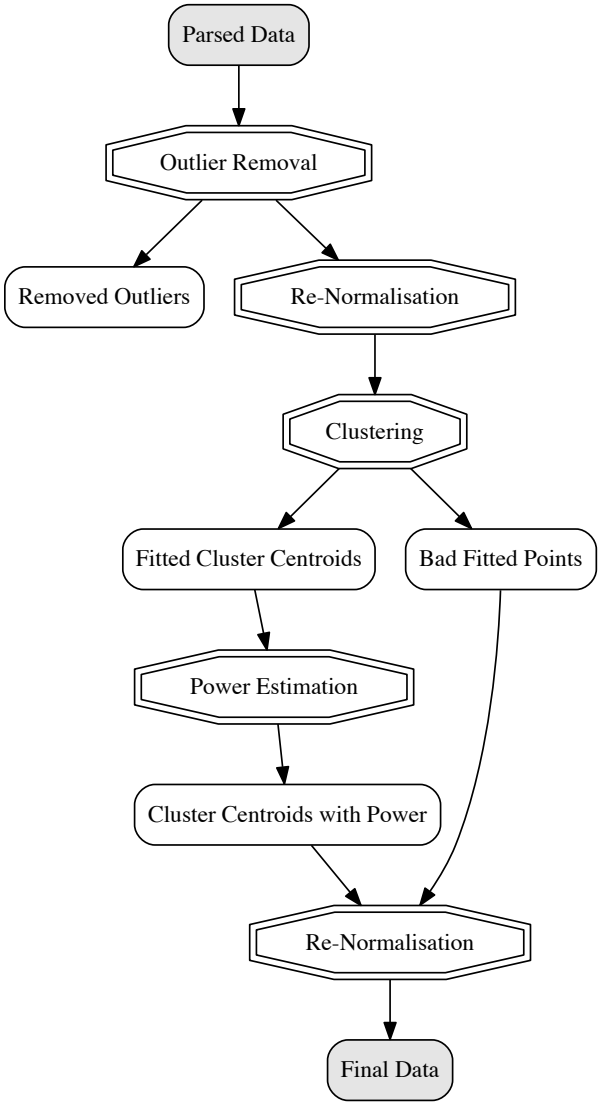


Figure 6.3: The basic algorithm to run the post-processing and optimisation on a raw power model. This outlines the data flow through various algorithms and steps before the final power model is generated.

percentile. To finish the outlier removal the power model must be re-normalised to account for potential missing extremal values.

In order to remove redundant data points the point cloud is subject to a clustering stage. Due to the four minute sampling period per grid point, as described in Section 5.2.3 the ideal number of cluster centres is approximately 25% of the total number of recorded data points. The k-NN is used in regression mode to fit clusters into the point cloud. Some points will not be fitted well by using k-NN. These points can be identified by applying a threshold to the within-cluster sums of point-to-centroid distances and filtering out bad fitting points. Due to the artificial creation of new data points, these points do not contain a valid power value yet. In order to assign a correct power value to these cluster centres the same algorithm is used as in the regular power estimation in the Papillon Server (k-NN with weighted sum). The data points in a cluster are used as source and the new cluster centroid point is subject for power estimation. Well fitted points are now represented by their cluster centroid point and can therefore be discarded.

The final power model consists of the cluster centroid points as well as the bad fitted points which are simply copied over. A final round of re-normalisation is applied to account for changed

extremal points.

6.3.2 Proposed Improvements to the Papillon Server

Additionally to the removal of outliers and the reduction of redundant data points the optimised power model contains pre-computed values for extrema and normalised activity values to speed up the power estimation in the Papillon Server. This requires a small change in the Papillon Server to use a different XML element as value source for the activity metrics. At the moment the Papillon Server computes the extrema of all three activities every time a new power estimation is requested. These extremal values are then used to normalise all data points in the power model which then can finally be used to estimate the power of a given sample point. All this computational effort is repeated for every new power estimation and can completely removed from the regular operation mode. These values are pre-computed during the post-processing and optimisation pipeline and saved into the XML of the optimised power model for later use.

7

Experimental Results

[Model validation is defined as] substantiation that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model.

– Schlesinger et al. 1979, SIMULATION 32

In order to evaluate the modifications to the existing Papillon system in terms of accuracy, performance, and overall consistency, a statistical analysis was performed on different data sets. Multiple data sets are rated with a set of metrics to compare and rank them to one another. A brief comparison to existing systems, mentioned in Chapter 3, is given. In order to provide accurate and replicable results the recommendations of Mair et al. [Mai+14] are followed as close as possible.

7.1 Test Equipment

The specification for the SUT and the power meter can be seen in Table 7.1 and 7.2. An additional data sheet for the server model is available on the HP website[1].

Manufacturer	HP
Model	ProLiant DL380 G7
CPU	2x Intel Xeon E5620
RAM	8x 2GB 1066 MHz
Hard Disks	5x 146GB SAS 6G DP 10K
Network Interfaces	4x 1G RJ45, only one active
Power Supply	230V, 50 Hz, max. 460 W
Form Factor	Rack 2U, Depth: 70 cm
Operating System	Ubuntu 12.04.5, x86_64, Kernel 3.13

Table 7.1: The SUT hardware specification and software versions used.

The results and comparisons in the following chapter for various power models, power estimation algorithms, and error metrics do only apply to the above defined SUT. A generalisation for other types of hardware requires additional evaluations and experiments.

[1] http://www8.hp.com/h20195/v2/GetDocument.aspx?docname=c04199811

Manufacturer	Yokogawa Electric Corporation
Model	WT210 digital power meter
Power Accuracy	0.1% of reading + 0.1% of range
Integration Accuracy	Power Accuracy + 0.1% of reading

Table 7.2: The power meter used for recording power readings of a connected SUT during power model generation and verification.

7.2 Validation Data Set

In order to evaluate the proposed system from Chapter 5 a validation data set is needed to have an objective baseline to compare to. The SPECpower benchmark is an established utility in the community and used throughout various research fields in [Riv+07], [Mai+14], and [Zha+13].

Due to the relative short runtime of the SPECpower benchmark of approximately 90 minutes the need for a larger validation set can be simply accomplished by running the SPECpower multiple times. With this approach a final validation data set with 15.000 data points can be used to evaluate the proposed system in a meaningful and comparable way. According to Zhang et al. [Zha+13] the SPECpower benchmark is mostly CPU-bound and does not contain a high amount of disk or network activity. Depending on the actual use case in operation of the SUT this validation data set should be adapted to represent the expected real workload more precisely.

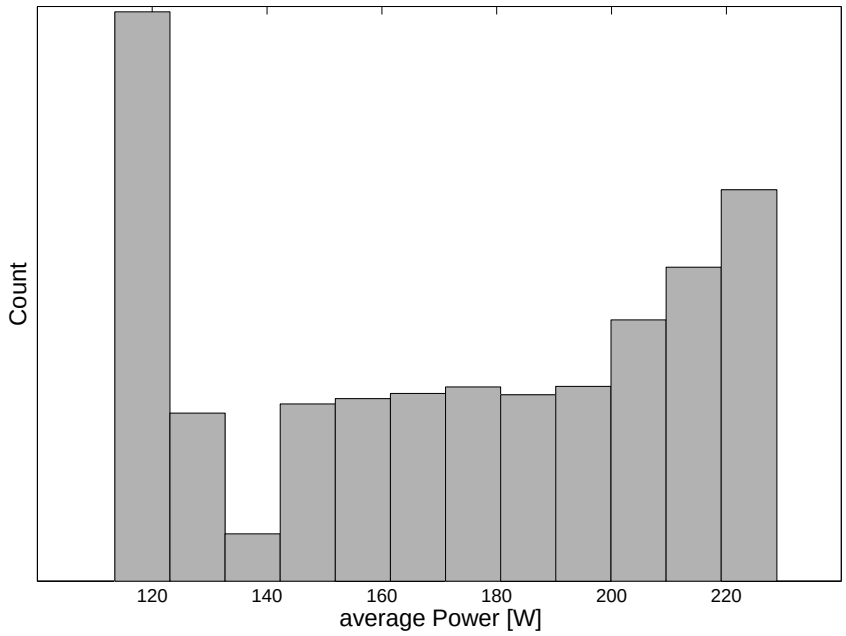


Figure 7.1: The distribution of power values contained in the validation data created by multiple runs of the SPECpower benchmark is shown in this histogram. Low and high values (correlated to low and high activity) are more common than values in between. There are no outliers or accumulation points visible.

Figure 7.1 shows the distribution of power values in the validation data set. The obvious spikes in the beginning and end describe the idling system and a fully utilised system. Between these two extrema the power values are evenly distributes.

7.3 Post-Processing and Optimisation

In order to evaluate the effectiveness of the implemented post-processing and optimisation techniques, various power models have been created and run through the optimisation pipeline. The results are discussed and various features and peculiarities of the resulting graphs and figures compared against the baseline and validation data set.

As discussed in Section 6.2 the workload for CPU-related activity can be comprised of various actions. For a qualitative comparison two different modes of operation for the *stress-ng* utility were chosen. The `sqrt-rand` stressor computes the square-roots of large random integer values over and over again. This mainly creates load in a single unit of the processor. To evaluate this stressor a baseline stressor is used to get an overall average utilisation. The `all` stressor cycles through each available CPU-related stressor and repeats the loop once finished. There are 46 different stressors available[2] ranging from integer, float, and mixed type computation, various control flow operations, random number generation for different types, matrix multiplications, and prime number tests.

To get a statistical sample for each of the two stressors a total number of four power models were created with each stressor using the test equipment described in Section 7.1. Additionally to these eight power models a single power model for the same SUT was used in the evaluation. This last power model was not created with the techniques and tools described in this thesis, but was a pre-existing power model considered to be state-of-the-art prior to this work. In the following qualitative evaluations these power models will be addressed with the following terms:

raw
: A power model is raw if it was saved right after the workload generation is completed. No further manipulation on the data was performed.

optimised
: A power model is optimised if its data went through the post-processing and optimisation pipeline. The data points might not correlate to real observations but rather to cluster centres.

old
: A pre-existing power model created without the techniques and tools described in this thesis is addressed as old.

cpu-sqrt-rand
: A power model created with the `sqrt-rand` stressor is identified by this label.

cpu-all
: A power model created with `all` stressors activated is identified by this label.

7.3.1 Data Point Distribution

The accuracy of a power model is in direct relation to the data point distribution due to the nature of the used power estimation algorithm (k-NN). Measuring objective metrics of power models is difficult because they typically represent data in higher dimensional space, which is hard to fit into words or single metrics. Two important terms which are highly relevant to the accuracy of power models are:

[2] as of *stress-ng* version `v0.03.00`

Coverage

The term coverage describes the size of covered operational space of a given power model. The larger the coverage the higher are the maximum values in each dimension. Since all used metrics have zero as their minimum value the coverage can only be increased by putting more load onto the SUT in each recorded data axis. Simply speaking a power model with CPU values ranging up to 95% is better than a power model ranging only to a CPU maximum of 80%. The same statement holds for more abstract metrics like disk and network which are measured in throughput of bytes.

A power model with optimal coverage would contain data points at the extremal coordinates of the operational space. Depicting the operational space as a 3D cube the optimal coverage would be achieved by placing data points at each corner and on the six surfaces of the cube. The size of the cube itself is determined by the actual hardware limitation of the SUT. A 2-core CPU has a maximum load of 200%. Hard disks attached via SATA III can only account for a theoretical maximum of 36 GB (4.8GBit/s for 60 s).

Given the dependency on the actual SUT hardware for this metrics it is not possible to compare coverage values for different hardware specifications. Nevertheless coverage provides a meaningful abstract metric in terms of first-order accuracy approximation.

Density

The term density is similar to the common definition. It describes how many data points are within a given area or volumetric space. Due to time restrictions during power model generation a balance between density (or resolution) and overall duration has to be found. Increasing the density means sampling more grid points in closer proximity and intervals. This increases the needed time for power model generation significantly. It is important to note that a high density metric can only be achieved if the operational space is covered in a regular fashion without any larger gaps or skipped regions.

Outliers and misplaced data points negatively affect the density significantly. As part of a post-processing and optimisation it is therefore a good indicator for improved accuracy if the density increased during post-processing.

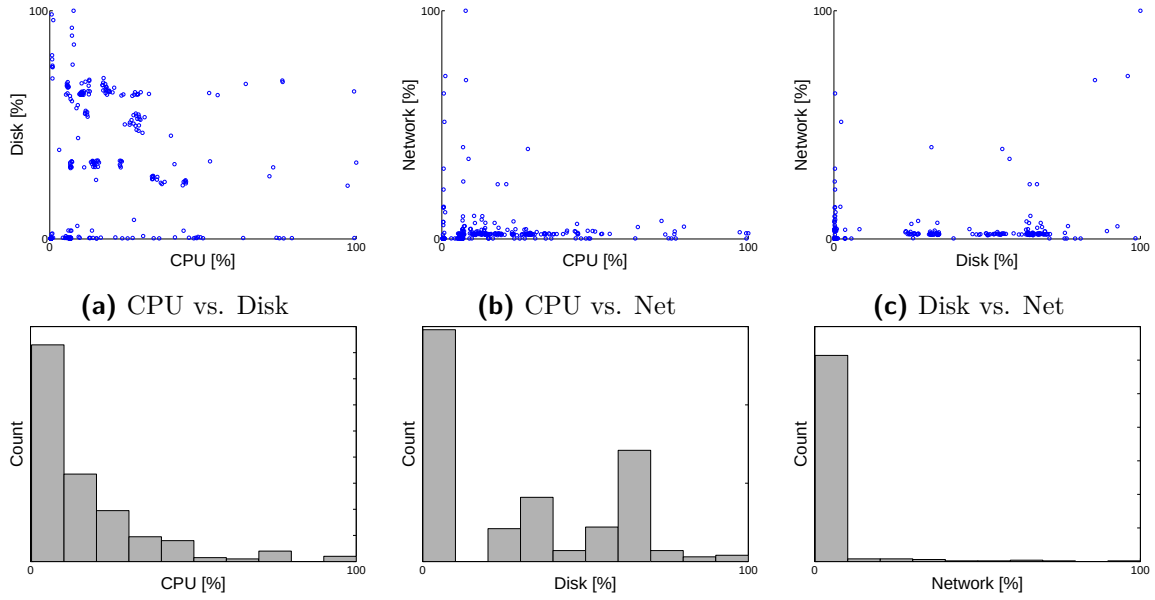
If the density is used as an actual value it is only meaningful if computed from normalised coordinates, otherwise the hardware specification would skew the metric.

Both terms are used to evaluate the results of this work. Although a strict value-based comparison would be possible it is not in the best interest of publishing specific numbers without a widely established system. Therefore the generated power models and their optimisation results will only be discussed and compared in a comparative style.

7.3.1.1 Old Power Model

In Figure 7.2 an old power model is visualised in the three side views. The displayed area ranges from zero to the maximum value in each axis. Each entry in the graph represents a single data point in the power model. The shown power model consists of 324 data points and was created with the SUT and power meter specified in Section 7.1. The total amount of generation time for this model is approximately 5.5 hours. The generation process involved running various applications and 3rd-party benchmarks to create artificial activity. There was no clear check list or sequence on which applications to run for how long. Therefore this power model lacks certain key requirements needed to allow for accurate power estimations.

It is clearly visible from the figures that large areas of the operational space are only poorly or not covered at all. The histograms indicate large gaps in the disk utilisation distribution and



(a) CPU vs. Disk **(b)** CPU vs. Net **(c)** Disk vs. Net

(d) Histogram CPU **(e)** Histogram Disk **(f)** Histogram Net

Figure 7.2: This visualisation shows the data point distribution and histograms of an old power model.

almost no data points spreading along the network axis. High CPU utilisation is only covered with a couple of data points, without covering a large area.

Concluding, although this power model represents the basic structure and distribution indicators, it will not be suitable for servers with a wide range of operation. Feasible power estimation accuracy can only be achieved for CPU-bound tasks, in a low utilisation region of approximately below 45%. Although this power model does include values for network utilisation, these values are of almost no merit to the actual power estimation process.

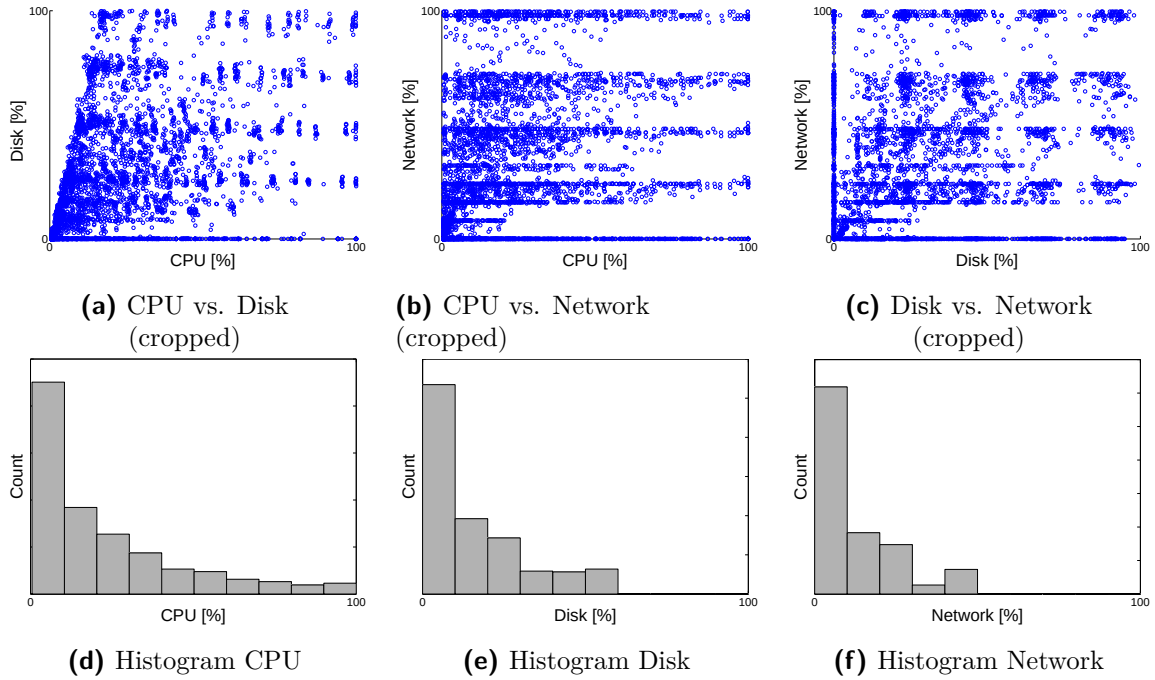
7.3.1.2 Raw Power Model

A power model created with the proposed workload generator and techniques can be seen in Figure 7.3. In order to facilitate visual comparison between the raw and optimised power model, the visualisations 7.3a, 7.3b, and 7.3c are cropped to hide outliers in the data set. The outliers are clearly visible in the histograms for disk and network, in which a large portion of the higher value x axis is basically empty of data points, where as the histogram for CPU shows values for the full spectrum.

This raw power model consists of approximately 6000 data points, which corresponds to a generation time of circa 4.2 days. The overall time needed for a finished power model can be easily deducted from the way the workload generator works. It requires five steps for each resource metric, four minutes for each grid point and an additional cool-down phase of 90 seconds per grid point. Due to time limitations during the conduction of these experiments the 16 CPU cores were iterated with a step range of two. This results in a total time of $t_{total} = 5 \cdot 5 \cdot 5 \cdot (16/2) \cdot 4 \cdot 1.5 = 6000\,\text{min}$.

It is clearly visible that the data set covers all three axis in an evenly distributed fashion. Due to the nature of the workload generator the lower utilisation regions are covered more extensively. This behaviour is represented in the histograms, and also represents common utilisation values of servers, which typically operate most of the time in low utilisation.

Figure 7.3a shows the coordinates of data points for CPU and disk axis. One can observe a



(a) CPU vs. Disk (cropped) **(b)** CPU vs. Network (cropped) **(c)** Disk vs. Network (cropped)

(d) Histogram CPU **(e)** Histogram Disk **(f)** Histogram Network

Figure 7.3: This visualisation shows the data point distribution and histograms of a raw power model.

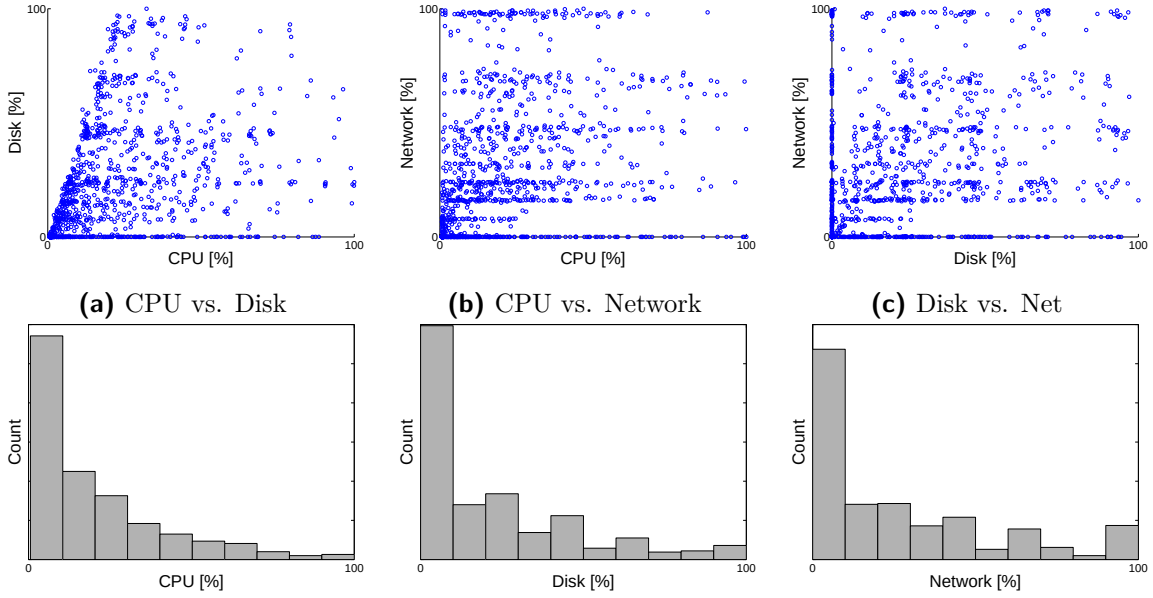
skewed axis for disk utilisation at 0% CPU. This empty triangular area is an effect of hardware and computing restrictions. It is not (feasible) possible to create artificial disk activity while keeping a 0% CPU level. For every byte written to disk a certain amount of CPU time is required to perform this task. Therefore it is physically not possible for the SUT architecture to operate with 0% CPU and 100% disk. This fact seems not to hold for networking, Figure 7.3b shows no such effect.

Due to the way the workload generator operates a common pattern is visible in all point distribution visualisations. Each metric (CPU, disk, and network) is utilised by the workload generator in five steps: 0%, 25%, 50%, 75%, and 100%. These five levels of utilisation are clearly visible in the graphs and represent the accuracy with which the artificial workload is created and applied on the SUT. This effect represents itself in Figure 7.3a and 7.3b as five horizontal lines (areas with increased point density). In Figure 7.3c the same five horizontal lines are visible with the additional outcome that each of the lines is clustered into five groups. Due to the nature of workload generation these lines and groups match the five levels of utilisation pretty well.

7.3.1.3 Optimised Power Model

Going through the post-processing and optimisation pipeline allows the raw power model to be stripped from outliers and redundant data points, which can be seen in Figure 7.4. Outlier removal, as described in Section 6.3, reduces the coverage because data points with maximum values in a raw power model are most likely outliers. The histograms show that the resulting range of values is covered with a considerate amount of data points, in contrast the histograms for the raw power model in Figure 7.3 showed large uncovered areas for disk and network. Due to the nature of the workload generator it is clearly visible that the five steps of 0%, 25%, 50%, 75%, and 100% are reflected in the histogram as well with small spikes at the corresponding intervals.

The overall point distribution is not affected by the post-processing steps and compares quite



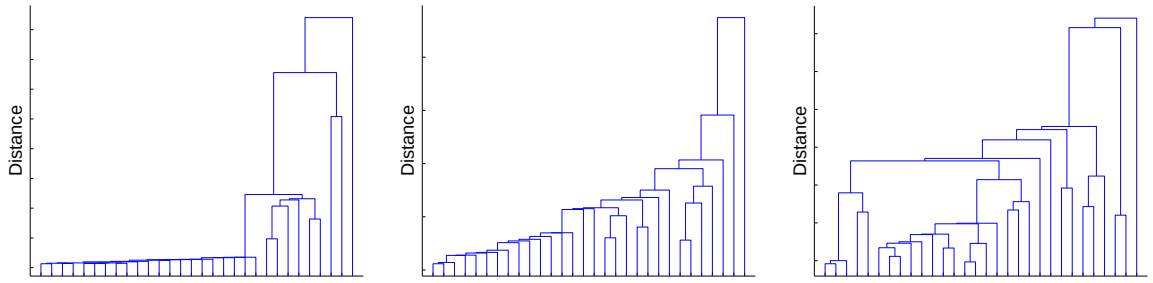
(a) CPU vs. Disk **(b)** CPU vs. Network **(c)** Disk vs. Net

(d) Histogram CPU **(e)** Histogram Disk **(f)** Histogram Network

Figure 7.4: This visualisation shows the data point distribution and histograms of an optimised power model.

well to the raw power model. The triangular space and the approximated five steps are still visible in the reduced data set. As a result of the point reduction, down to approximately 1350 points, the density decreases significantly. The effects of this will be discussed in Section 7.4. The final power model after the post-processing and optimisation steps is about 22% the size of the raw power model, considering the number of data points as primary indicator for storage size. As part of the optimisation pipeline the normalised point coordinates are computed and stored inside the power model. This means the Papillon Server can reuse these values and save computing cycles on each power estimation. Along with the normalised data points additional qualitative information about the power model is computed and stored, such as the overall minimum and maximum values which reflect coverage.

7.3.2 Clusters and Point Distances



(a) Raw Power Model **(b)** Optimised Power Model **(c)** Old Power Model

Figure 7.5: This dendrogram shows the cluster distances of the tree power models used for comparison.

In order to further evaluate the quality of a power model in terms of density a dendrogram can be seen in Figure 7.5 which illustrates the distances between clusters of data points in a given power model. The clusters are determined with hierarchical clustering using an agglomerative

strategy.

Comparing the dendrograms of a raw and an optimised power model indicates that outliers (present in a raw power model, but not in an optimised power model) cause larger distances between clusters. The optimised power model shows more uniformly distributed cluster distances with a clearly visible, almost linear, increase in distances. This reflects the fact that there are much more low utilisation data points then in higher utilisation regions, yielding increased cluster distances. Evaluating the old power model yields a non-reasonable result due to the large gaps and poor coverage of this specific power model.

7.3.3 Power Correlation

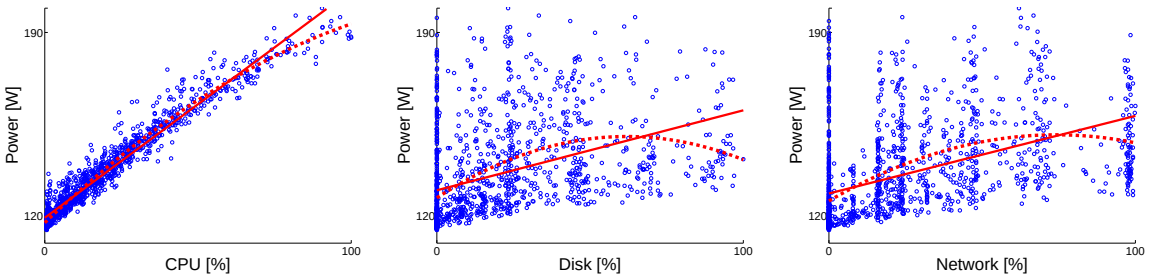
Typical servers used in data centres usually are bound to operate on a specific resource-bound activity load. CPU-bound servers are probably the most common use cases. Large file servers depend to be disk-bound, which means most of the operational workload is reading or writing to disk. Servers with 16 cores and 64GB of RAM show a different power profile then a file server with 32 high-speed hard disks used for storage.

In order to generate an accurate power model for a specific server model it is important to consider the expected use case and operation. For the SUT and an optimised power model the power correlation for the three different metrics can be seen in Figure 7.6. Each graph shows a single data point with the current metric on the abscissa and the power value on the ordinate. Additionally to the data point plots a best fitting polygon with the order 1 (solid line, linear) and order 2 (dashed line, quadratic) is approximated to fit the data.

From Figure 7.6a it is clearly visible that CPU strongly correlates to power with a small variance. Either linear or quadratic polynomials fit reasonably well. It may be noticed that with higher CPU values (85% and above) the best fitting line seems to level off. This means the additional power required from 90% to 95% is typically less then the same step from 35% to 40%. With this knowledge the point of most efficient computing can be deducted quite easily by looking at the gradient of the fitted line. This observation is reflected by the work done in [BH07] and [RRK08].

On the contrary disk and network seem to be not correlated to power at all. The variance is very high compared to CPU and fitting a polynomial line is not feasible. Due to the requirement of certain CPU levels for disk and network activity (triangular area) the same observation is possible for Figure 7.6b and 7.6c. Apart from the triangular correlation there can be no obvious dependency deducted from this data set.

These findings correspond to the research published by Orgerie, Lefevre, and Gelas [OLG10] and confirm the non-linearity for disk and network, as well as a possible non-linear CPU power correlation, although this depends strongly on the SUT.



(a) CPU **(b)** Disk **(c)** Network

Figure 7.6: Each of the graphs show the correlation between power and the specified activity metric, including best-fitting polynomials.

7.4 Power Prediction Evaluation

The most important task in terms of the overall goal of the Papillon system is to provide accurate power estimations based on live activity data from connected servers. The accuracy then is a key factor for higher level estimations like CO2 consumption, PUE [Bel07], and TUE [Pat+13] values for data centres.

Using the validation data set described in Section 7.2 and a variety of different power models and estimation algorithms the following section gives detailed information about power estimation accuracy of the proposed changes to the Papillon system.

Each power model is used to estimate the power of all 15.000 data points in the validation data set. The resulting power values are then compared to actual power readings from a power meter, which recorded a calibrated power value for each validation data point. The resulting statistical analysis is visualised in box plots and as raw values in the tables below.

The box plot shows the absolute error values in percent. The 25th and 75th percentile (Q_1 and Q_3) of error values are outlined by a box, with a straight horizontal line in the box indicating the median error value. The whiskers below and above the box indicate the extremal error values, which are not considered outliers. Outliers are defined with a threshold of above $Q_3 + 1.5 \cdot (Q_3 - Q_1)$ or below $Q_1 - 1.5 \cdot (Q_3 - Q_1)$. In a simplified description the box should be as close as possible to the baseline (0% error). The size of the box and the length of the whiskers are an indicator for the variance of the error values and should be as small as possible to describe an accurate power model.

The results table shows three different error metrics with the following meanings:

Median Error indicates the actual error value dividing the lower half from the higher half of error values.

Mean Error indicates the numerical average of all error values.

Dynamic Range Error (DRE) indicates a relative metric of the root-mean-squared error divided by the dynamic range [Dav+11]. This metric allows for comparison across platforms because the actual error is weighted against the idle and maximum power consumption of the SUT. Due to the proposed comparability this metric will be mainly used for the following model comparisons.

7.4.1 Papillon Power Models with Different CPU Stressors

Stressor Type	Median Error [%]	Mean Error [%]	DRE [%]
sqrt-rand	6.26	6.00	2.05
	5.88	5.93	2.07
	7.33	7.21	2.01
	6.10	5.77	2.55
all	1.07	1.43	0.73
	1.36	1.57	0.72
	1.44	1.67	0.74
	1.66	2.01	0.95

Table 7.3: The power estimation error metrics for different types of CPU stressors.

In order to get an understanding about the accuracy of different Papillon power models a total number of eight power models were created with the modifications and additions outlined

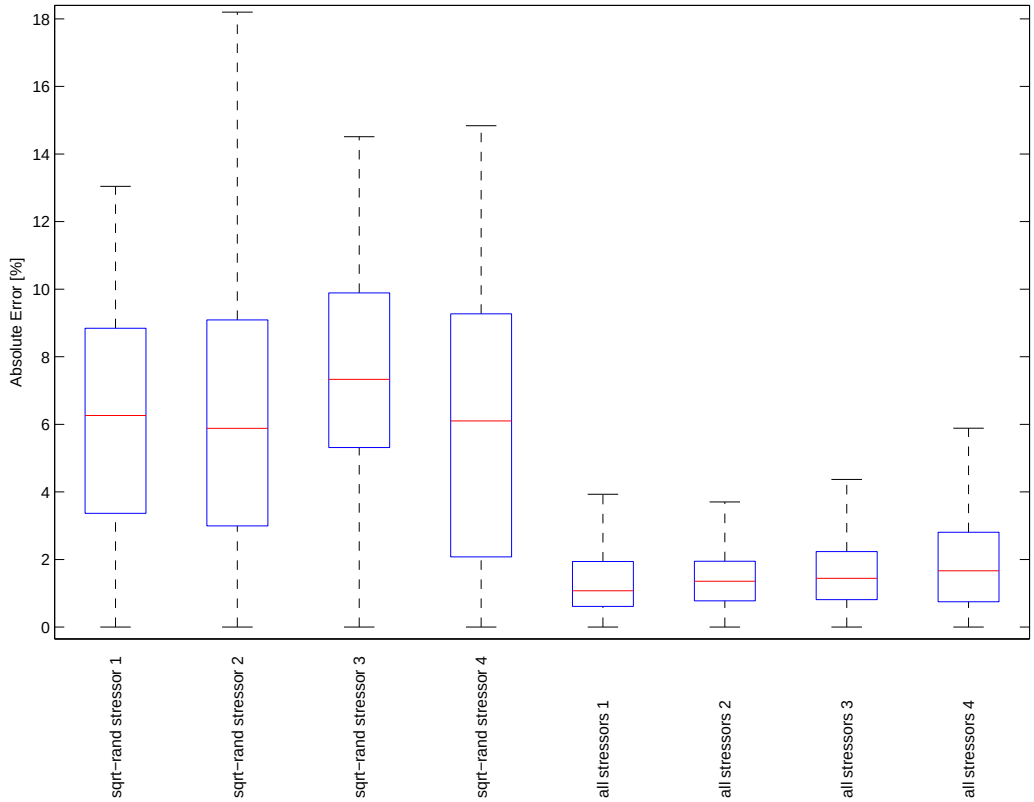


Figure 7.7: This box plot shows multiple raw power models, each created with a different CPU stressor.

in Chapter 5. According to the data sheet of the SUT it is expected to be a mainly CPU-bound power consumption. Therefore it is important to compare different CPU-related workloads. Section 7.3 introduced the concept of stressors, in particular the *sqrt-rand* and *all* stressors. Figure 7.7 and Table 7.3 show the power estimation accuracy for those eight power models. Each entry reflects a raw and untouched Papillon power model used for power estimation with the Papillon 10-Nearest-Neighbour algorithm. Two groups consisting of four power models each were created to compare different CPU stressors.

The results clearly show that using only the *sqrt-rand* stressor to create artificial CPU workload leads to a significant worse accuracy compared to the *all* stressor, which invokes all available stressor types involving the CPU in any way. The average absolute error indicates that the overall power estimation accuracy is approximately 6% for the *sqrt-rand* group and below 2% for the *all* group. The same trend is visible in the median error as well as the dynamic range error.

Therefore it can be concluded that for the given SUT the *all* stressor for the CPU workload generator yields the most accurate power model with an average estimation error of below 1% of DRE.

7.4.2 Papillon Power Models: Raw and Optimised

The post-processing and optimisation pipeline is a key component to reduce the computational efforts needed for each power estimation done by a Papillon server. Estimating more power values in the same time frame allows the Papillon Server to handle more clients with a given infrastructure. Therefore its goal is to increase the performance of each single power estimation and reduce redundant computations.

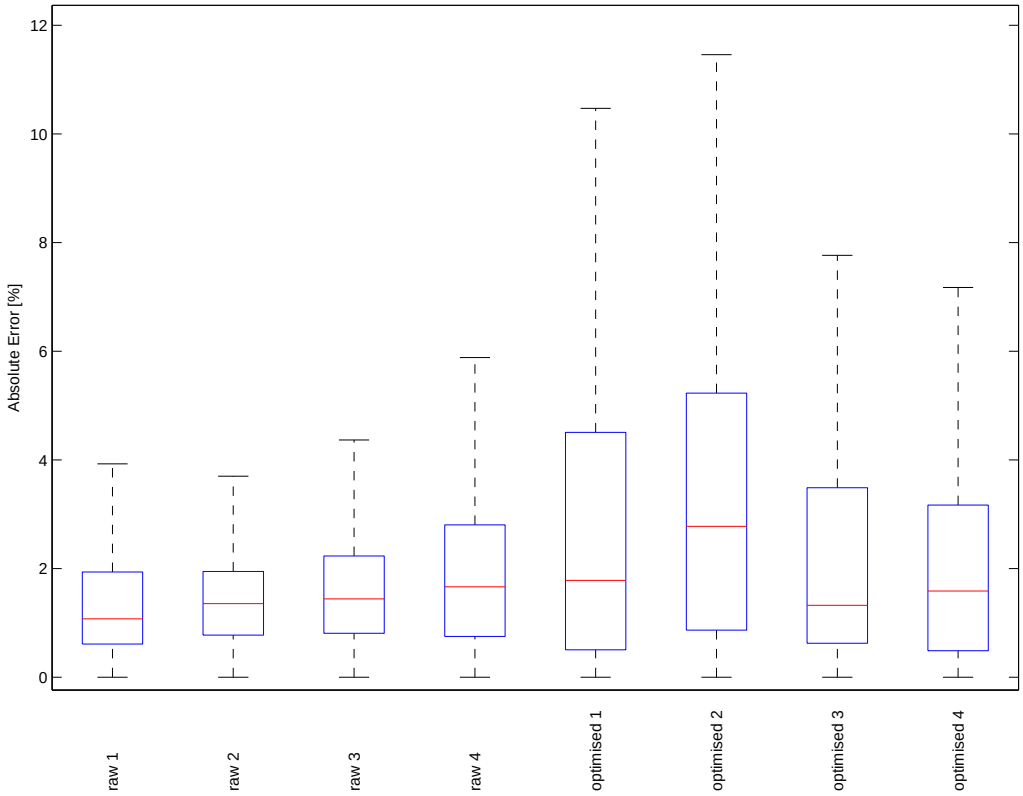


Figure 7.8: This box plot shows multiple raw power models and optimised power models side by side.

An evaluation on the effects in terms of accuracy can be seen in Figure 7.8 and Table 7.4. The four raw power models from above (with *all* stressor) are used as baseline in this comparison (labelled *raw*). The post-processing and optimisation pipeline was applied to these power models and the resulting error values are labelled as *optimised*.

The error metrics indicate that the post-processing pipeline did effect the accuracy of the power estimation on the validation data set negatively. The mean error of all four power models per group increased from approximately 1.67% to 2.55%. Similar effects can be observed for the DRE metric, which changed from 0.79% up to 1.25%. This is an expected result given the fact that each raw power model consists of about 6000 data points, compared to an optimised power model with only 1350 data points. With a size reduction down to about 22% of the raw power model, the power estimation is still yielding similarly accurate results. The variance of error

Name	Median Error [%]	Mean Error [%]	DRE [%]
raw	1.07	1.43	0.73
	1.36	1.57	0.72
	1.44	1.67	0.74
	1.66	2.01	0.95
optimised	1.78	2.85	1.52
	2.78	3.11	1.31
	1.32	2.27	1.24
	1.59	1.95	0.95

Table 7.4: The power estimation error metrics for a raw and an optimised Papillon power model.

values increased considerably for the optimised power models, while still keeping a relatively low average error value.

The maximum error for an optimised power model increased from approximately 6% up to 11%. This is the largest deviation from the raw power models and is strongly influenced by the non-deterministic post-processing and optimisation pipeline. It would be recommended to run a validation after the post-processing to ensure a given power model still fulfils the requirements. A simple cross-validation can be performed by comparing two individually created power models against each other in terms of coverage, density and accuracy.

7.4.3 CPU-only Power Models

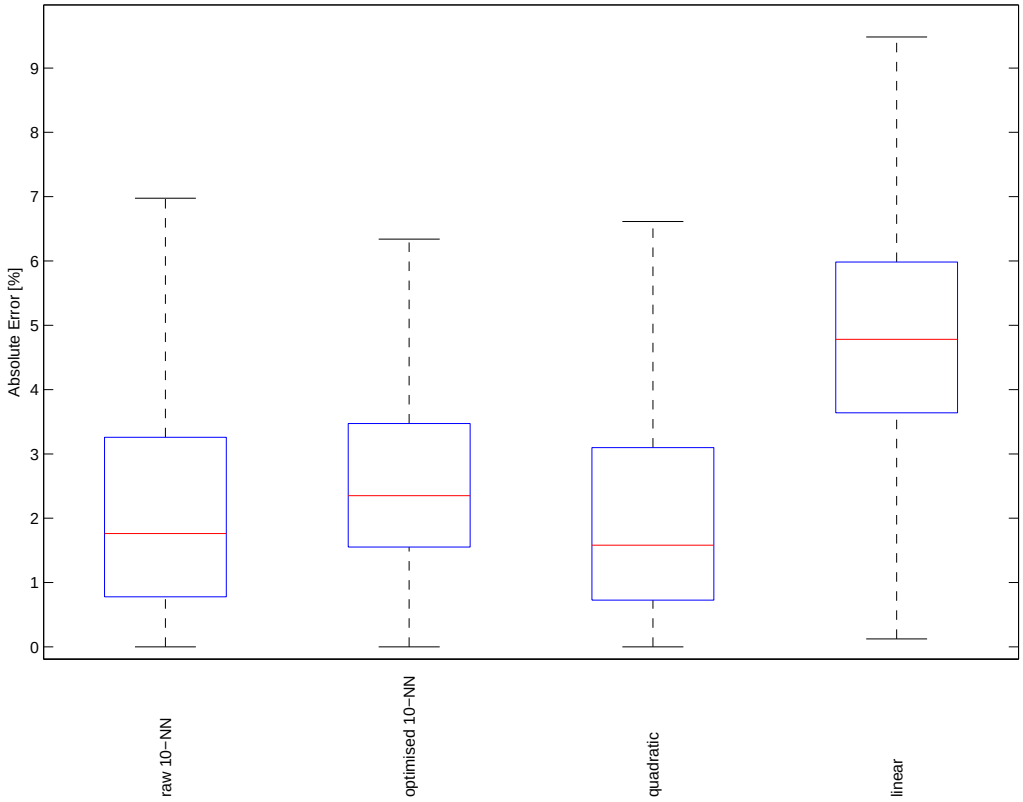


Figure 7.9: This box plot shows a comparison between various CPU-only power models.

Name	Median Error [%]	Mean Error [%]	DRE [%]
raw 10-Nearest-Neighbour	1.76	2.20	0.93
optimised 10-Nearest-Neighbour	2.35	2.62	0.85
best-fit quadratic	1.58	1.99	0.81
best-fit linear	4.78	4.59	0.94

Table 7.5: The power estimation error metrics for various CPU-only power models.

Various different power model techniques exist and are being adopted into the industry. The most common type of model is a pure CPU-only linear power model, as proposed in [FWB07], [LTG12], and [Li+12]. Although various other researchers have found that this simplification is not accurate enough, [Zha+13] and [OLG10], it is still used widely. The Papillon system has not yet been compared to these simple linear models and will therefore be subject to an evaluation together with a simple linear and a quadratic power model.

Due to the fact that for this comparison only CPU activity is used the names for the two power models under evaluation are described as *10-Nearest-Neighbour*, because the term Papillon power model refers to a full power model consisting of CPU, disk and network data. The linear and quadratic model are derived from the raw CPU-only Papillon model by computing the best fitting polynomial with order 1 (linear) or order 2 (quadratic). The best fit is determined with a plain least-squares approach.

Figure 7.9 and Table 7.5 show the numerical evaluation of the validation data set against each of the four CPU-only power models. From the box plot and error metrics it is clearly visible that a quadratic power model yields the most accurate results. This result is similar to the research published by Zhang et al. [Zha+13].

The raw 10-Nearest-Neighbour with about 6000 data points is almost as accurate as the quadratic power model, with a median absolute error of below 2%. The optimised 10-Nearest-Neighbour power model yields similar results, although the absolute error is slightly higher, but the variance of the error values is smaller with smaller extreme values. The least accurate power model in this comparison is the linear model with an absolute error between 4% and 5%.

A slightly different conclusion can be drawn by looking at the DRE. All four power models are below 1% of DRE, with the quadratic model still being the most accurate, followed by the optimised 10-Nearest-Neighbour model. Again the least accurate model is the purely linear power model.

7.4.4 Comparison

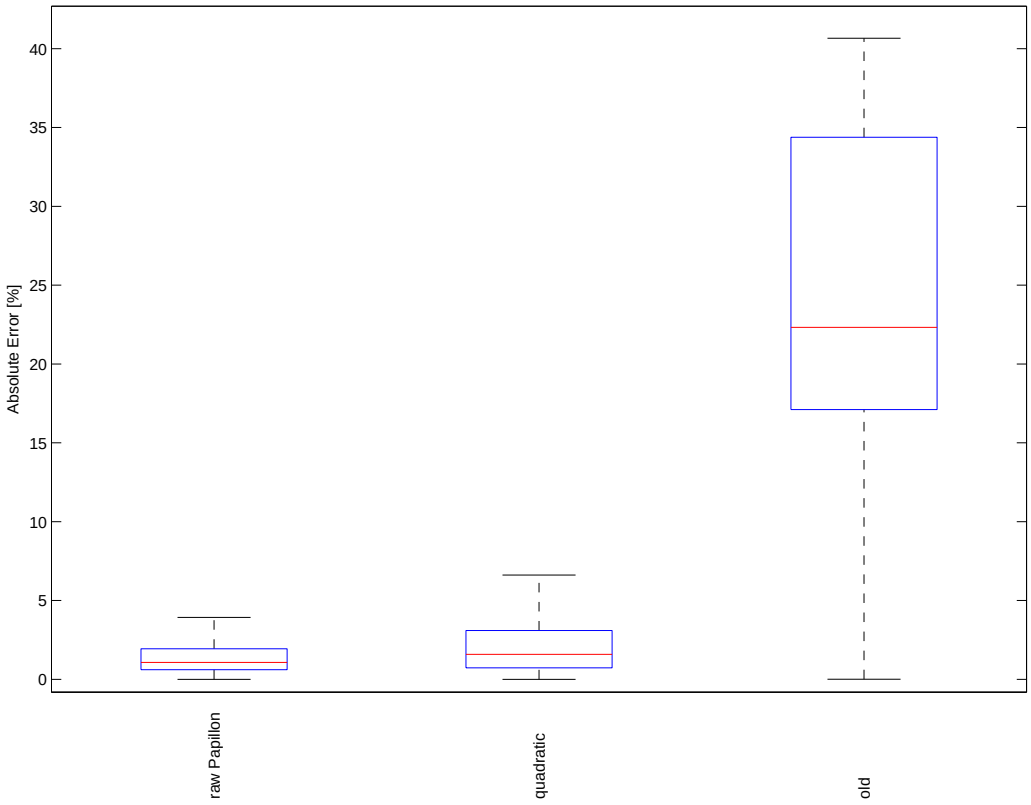


Figure 7.10: This box plot shows a comparison between a raw Papillon, a simple CPU-only quadratic model, and an old Papillon power model.

To draw a conclusion between the different power models and proposed generation techniques a raw Papillon power model is compared against a purely CPU-only quadratic power model. To

Name	Median Error [%]	Mean Error [%]	DRE [%]
raw Papillon	1.07	1.43	0.73
CPU-only best-fit quadratic	1.58	1.99	0.81
old	22.32	23.59	5.57

Table 7.6: The power estimation error metrics for a raw Papillon, a CPU-only best-fit quadratic, and an old Papillon power model.

put these error values into perspective an existing power model currently used in the Papillon system is added to the comparison. This old power model was created with simple workloads and benchmarks, without the techniques and pipelines outlined in this thesis. The changes in accuracy to the old power model can therefore be used as an overall improvement criterion.

It is clearly visible from Figure 7.10 and Table 7.6 that the raw Papillon model, as well as the CPU-only quadratic power model outperform the old power model significantly. Prior to the proposed changes of this thesis the Papillon system was using power models with an average absolute accuracy of 24%, with extreme deviations up to 40% of error. The numbers for the raw Papillon model and quadratic model are already described above.

The CPU-only quadratic power model yields similar accuracy as a raw Papillon power model, both surpass the existing power model significantly. The numbers indicate a total improvement of over 625%.

8

Future Work

During the work on the described endeavours certain aspects surfaced which are out of the scope of this thesis. A brief definition of upcoming topics and proposed enhancements are given to lay the ground for prospective tasks to be done. The listed topics are only considered to be a digest of additional research fields.

8.1 Workload Generation on Windows

The currently implement workload generation described in Section 5.2 and 6.2 is targeting solely Linux-based OSs. Since the Papillon system is designed to be cross-platform compatible a future task is to port and adapt the Linux workload generator for Microsoft Windows systems.

One might expect that a power model generated on Linux should be valid on Microsoft Windows as well. At the current stage this claim has not been verified and power models need proper evaluation to support such assertions. The experimental evaluation performed in Chapter 7 was using done using a Linux-based OS and might not be valid on Microsoft Windows. Various hardware drivers and power modes might not be implemented or available on one of the platforms and therefore could yield different power estimations.

Due to the different ecosystem and available tools the architecture needs to be changed to accommodate for possible incompatibilities. The currently used tools *stress-ng* and *iperf* are not available on the Microsoft Windows platform and need to be replaced. Further research needs to be done if there are cross-platform tools available to keep the Linux-based and Microsoft Windows-based workload generator as similar as possible.

8.2 Evaluation of Different Neighbourhood Sizes

The Papillon power estimation uses a 10-Nearest Neighbour algorithm. Depending on the amount of data points, coverage, and density of the used power model various neighbourhood sizes could yield better estimations in terms of accuracy and computation performance.

A simple evaluation of existing power models and validation data sets with different neighbourhood sizes can be performed to find the optimal size vs. accuracy trade-off. Common numbers for the neighbourhood size are 5, 10 (currently used), 15, and 20. Smaller sizes would increase the power estimation performance because less computations need to be performed. Although this is only a small factor because in order to find the closest neighbours one has to compute a full list of all data points in the power model, unless other complex partitioning approaches are implemented.

The optimisation of existing power models would also benefit from research into different neighbourhood sizes, since the k-NN is used in the power estimation as well as the post-processing and optimisation pipeline. An additional reduction in data points of power models is possible and anticipated by increasing the neighbourhood size during post-processing, while decreasing the neighbourhood size during power estimation.

8.3 Evaluation of Different Clustering Algorithms

In Section 5.3.3 different techniques to reduce redundant data points were discussed. Both described algorithms use a form of clustering and regression approach to detect superfluous data points. Since the Papillon system already uses the k-NN algorithm for power estimation, it was a clear first choice to reuse this workflow in the post-processing and optimisation pipeline.

The second proposed algorithmic approach involves Gaussian Mixture Models for clustering of the point cloud. This would have the benefit of getting a probability distribution for clusters and data points. With this additional information the resulting power values (in either estimation or optimisation) could be further optimised and possibly yield a more accurate result.

8.4 Additional System Metrics

The power estimation in the Papillon system currently works by including CPU utilisation, disk and network throughput values and derive a power value with an existing power model. The number and shape of metrics can differ widely due to the normalisation to a predefine value range. Therefore including additional metrics can be easily accomplished.

The results in Section 7 clearly indicate a CPU-dominate power consumption of the available SUT. Depending on the level of granularity and accessibility of metrics many runtime-related values can be gathered and used for power estimation. Further research can be done to evaluate possible accuracy gains by including one of the following metrics in a Papillon power model:

Memory Access
: MMU and last-level cache misses could be used as indicator for memory access (read and write). Modern system architecture includes multiple levels of caches on different components. A clear distinction between the components must be found to account for memory access patterns caused by the user or the system.

Fan Speeds
: Some systems use a variable speed control on the system enclosure, RAM, CPU, and power supply fans. This could affect overall power estimation accuracy by up to 20 Watts (depending on the fan size and speed).

Temperature
: Ambient temperature is usually controlled by an external climate control systems embedded in the data centre. Depending on the air flow through the rack and device mounting different temperatures could affect different components of a system.

Precise CPU workload
: Currently the only CPU metrics recorded is the percentage of busy-time the CPU experienced. Since modern CPU architecture integrates more and more components onto a single chip, e.g. System on a Chip (SoC), it is not clearly distinguishable which components actually consume power. With strict separation of CPU activity into ALU, MMU,

and Direct Memory Access (DMA) controller the power estimation could be more precise if these metrics are recorded and evaluated independently.

9

Concluding Remarks

In this thesis I have presented modifications to the Papillon power model generation workflow. The Papillon Client was refactored to accommodate a live power model recording in collaboration with the introduction of a workload generator to increase coverage and density of Papillon power models. A number of experiments was conducted to evaluate the contributions of this work in terms of accuracy and system performance. The overall results show a significant accuracy improvement of over 625% for Papillon power models compared to existing models. The average power estimation error of a running server was reduced from 24% to under 2%. The Dynamic Range Error (DRE) was cut down from 5.6% to 0.7%.

A brief introduction to the Papillon system was given in Chapter 4. Chapter 5 and 6 covered the topics of design and implementation of the necessary changes and additions.

The experimental results of the proposed improvements and changes are discussed in Chapter 7. Chapter 8 discussed topics for future research and work currently in progress regarding the Papillon system.

Nomenclature

ALU Arithmetic Logic Unit. 23, 57

API Application Programming Interface. 14–17, 20, 33, 35, 36

CLI Command Line Interface. 20, 34

CPU Central Processing Unit. 6–10, 12, 13, 16, 18, 22–28, 32, 34, 37, 38, 43–47, 49–51, 53–55, 57

DCIM Data Center Infrastructure Management. V, VI, 13

DCPI Data Center Productivity Index. 7

DHCP Dynamic Host Configuration Protocol. 35

DMA Direct Memory Access. 58

DRE Dynamic Range Error. 9, 51, 52, 54, 59

DSL Domain-Specific Language. 39

DUT Device Under Test. 7

DVFS Dynamic Voltage and Frequency Scaling. 6, 8, 10, 12

GMM Gaussian Mixture Model. 30, 31

GUI Graphical User Interface. 20

HTTP Hypertext Transfer Protocol. 14, 20

IaaS Infrastructure as a Service. V, VI

IO Input/Output. 27, 28

IP Internet Protocol. 15, 35, 36

IT Information Technology. 1, 4

JSON JavaScript Object Notation. 15, 32, 33

k-NN k-Nearest Neighbours. 18, 19, 31, 40, 44, 57

MAC Media Access Control. 36

MMU Memory Management Unit. 23, 57

NIC Network Interface Controller. 8, 27

OS Operating System. 8–10, 15, 18, 22–29, 33, 36–38, 56

PaaS Platform as a Service. V, VI, 7

PAPILLON Profiling and Auditing of Power Information in Large and Local Organisational Networks. 13

PCA Principal Component Analysis. 23

PUE Power Usage Effectiveness. 7

PWM Pulse-Width Modulation. 28, 38

RAID Redundant Array of Independent Disks. 23, 38

RAM Random-Access Memory. 8, 10, 49, 57

REST Representational State Transfer. 14, 33

SaaS Software as a Service. V, VI

SoC System on a Chip. 57

SQL Structured Query Language. 15, 16

SSD Solid State Drive. 8, 12, 23

SUT System Under Test. v, 5, 8, 20–23, 25, 27, 28, 33, 34, 39, 42–45, 47, 49–51, 57

TCP Transmission Control Protocol. 15, 38

UDP User Datagram Protocol. 38

UID Unique Identifier. 36

UML Unified Modeling Language. 33

WMI Windows Management Instrumentation. 24

XML Extensible Markup Language. 15, 25, 32, 33, 39, 41

Bibliography

[Abr15] Andreas Abraham. “Power Monitoring of Virtual Machines and their Applications”. 2015 (cited on page 8).

[Alt92] N. S. Altman. “An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression”. In: *The American Statistician* 46.3 (1992), pages 175–185. doi:10.1080/00031305.1992.10475879 (cited on page 31).

[BH07] L.A. Barroso and U. Holzle. “The Case for Energy-Proportional Computing”. In: *Computer* 40.12 (Dec. 2007), pages 33–37. ISSN 0018-9162. doi:10.1109/MC.2007.443 (cited on pages 6, 49).

[Bay+01] Kathleen Baynes et al. “The Performance and Energy Consumption of Three Embedded Real-time Operating Systems”. In: *Proceedings of the 2001 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*. CASES ’01. Atlanta, Georgia, USA: ACM, 2001, pages 203–210. ISBN 1-58113-399-5. doi:10.1145/502217.502253 (cited on page 8).

[Bel07] Christian Belady. *The Green Grid Data Center Power Efficiency Metrics: PUE and DCiE*. Technical report. 2007. http://www.thegreengrid.org/gg%5C_content/TGG%5C_Data%5C_Center%5C_Power%5C_Efficiency%5C_Metrics%5C_PUE%5C_and%5C_DCiE.pdf (cited on pages 7, 50).

[Bel00] Frank Bellosa. “The Benefits of Event: Driven Energy Accounting in Power-sensitive Systems”. In: *Proceedings of the 9th Workshop on ACM SIGOPS European Workshop: Beyond the PC: New Challenges for the Operating System*. EW 9. Kolding, Denmark: ACM, 2000, pages 37–42. doi:10.1145/566726.566736 (cited on page 7).

[Ber+10] Ramon Bertran et al. “Decomposable and Responsive Power Models for Multicore Processors Using Performance Counters”. In: *Proceedings of the 24th ACM International Conference on Supercomputing*. ICS ’10. Tsukuba, Ibaraki, Japan: ACM, 2010, pages 147–158. ISBN 978-1-4503-0018-6. doi:10.1145/1810085.1810108 (cited on page 8).

[Ber+12] Ramon Bertran et al. “Energy accounting for shared virtualized environments under DVFS using PMC-based power models”. In: *Future Generation Computer Systems* 28.2 (2012), pages 457–468. ISSN 0167-739X. doi:10.1016/j.future.2011.03.007 (cited on page 8).

[BJ07] W.L. Bircher and L.K. John. “Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events”. In: *Performance Analysis of Systems Software, 2007. ISPASS 2007. IEEE International Symposium on*. Apr. 2007, pages 158–168. doi:10.1109/ISPASS.2007.363746 (cited on page 7).

[BTM00] David Brooks, Vivek Tiwari, and Margaret Martonosi. "Wattch: A framework for architectural-level power analysis and optimizations". In: *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*. June 2000, pages 83–94 (cited on page 7).

[Cas+13] Gabriel G. Castañé et al. "$E = mc^2$: A formal framework for energy modelling in cloud computing". In: *Simulation Modelling Practice and Theory* 39 (2013). S.I.Energy efficiency in grids and clouds, pages 56–75. ISSN 1569-190X. doi:10.1016/j.simpat.2013.05.002 (cited on page 7).

[CFR02] Fay Chang, Keith Farkas, and Parthasarathy Ranganathan. "Energy-driven statistical profiling: Detecting software hotspots". In: *In Workshop on Power-Aware Computer Systems*. 2002. doi:10.1.1.84.5818 (cited on page 7).

[Che+08] Gong Chen et al. "Energy-aware Server Provisioning and Load Dispatching for Connection-intensive Internet Services". In: *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. NSDI'08. San Francisco, California: USENIX Association, 2008, pages 337–350. ISBN 111-999-5555-22-1 (cited on page 8).

[CWL11] Ming Chen, Xiaorui Wang, and Xue Li. "Coordinating Processor and Main Memory for Efficientserver Power Control". In: *Proceedings of the International Conference on Supercomputing*. ICS '11. Tucson, Arizona, USA: ACM, 2011, pages 130–140. ISBN 978-1-4503-0102-2. doi:10.1145/1995896.1995917 (cited on pages 6, 23).

[Che+10] Xi Chen et al. "Performance and power modeling in a multi-programmed multi-core environment". In: *Design Automation Conference (DAC), 2010 47th ACM/IEEE*. June 2010, pages 813–818. ISBN 978-1-4244-6677-1 (cited on pages 8, 24).

[Che+13] Wen Chengjian et al. "System Power Model and Virtual Machine Power Metering for Cloud Computing Pricing". In: *Intelligent System Design and Engineering Applications (ISDEA), 2013 Third International Conference on*. Jan. 2013, pages 1379–1382. doi:10.1109/ISDEA.2012.327 (cited on page 8).

[CKE00] Todd L. Cignetti, Kirill Komarov, and Carla Schlatter Ellis. "Energy Estimation Tools for the Palm". In: *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. MSWIM '00. Boston, Massachusetts, USA: ACM, 2000, pages 96–103. ISBN 1-58113-304-9. doi:10.1145/346855.346869 (cited on page 7).

[CM05] Gilberto Contreras and Margaret Martonosi. "Power Prediction for Intel XScale® Processors Using Performance Monitoring Unit Events". In: *Proceedings of the 2005 International Symposium on Low Power Electronics and Design*. ISLPED '05. San Diego, CA, USA: ACM, 2005, pages 221–226. ISBN 1-59593-137-6. doi:10.1145/1077603.1077657 (cited on page 7).

[CH11] Gary Cook and Jodie Van Horn. *How dirty is your data?* Apr. 2011. http://www.greenpeace.org/international/en/publications/reports/How-dirty-is-your-data/ (cited on page 6).

[Dal+12] D. Dalton et al. *Power profiling and auditing consumption systems and methods*. US Patent App. 13/180,152, EP Patent App. EP20,110,758,414. Jan. 2012 (cited on page 8).

[Dav+11] John D Davis et al. *No hardware required: building and validating composable highly accurate OS-based power models*. Technical report MSR-TR-2011-89. June 2011.

http://research.microsoft.com/pubs/152100/HotPower_TR_V1.0.pdf (cited on pages 6, 8, 9, 50).

[DMR10] Gaurav Dhiman, Kresimir Mihic, and Tajana Rosing. "A System for Online Power Prediction in Virtualized Environments Using Gaussian Mixture Models". In: *Proceedings of the 47th Design Automation Conference*. DAC '10. Anaheim, California: ACM, 2010, pages 807–812. ISBN 978-1-4503-0002-5. doi:10.1145/1837274.1837478 (cited on page 8).

[ERK06] Dimitris Economou, Suzanne Rivoire, and Christos Kozyrakis. "Full-system power analysis and modeling for server environments". In: *In Workshop on Modeling Benchmarking and Simulation (MOBS)*. 2006 (cited on pages 7, 11, 12).

[EKR02] E.N. (Mootaz) Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. "Energy-Efficient Server Clusters". In: *In Proceedings of the 2nd Workshop on Power-Aware Computing Systems*. 2002, pages 179–196 (cited on page 8).

[FWB07] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. "Power Provisioning for a Warehouse-sized Computer". In: *Proceedings of the 34th Annual International Symposium on Computer Architecture*. ISCA '07. San Diego, California, USA: ACM, 2007, pages 13–23. ISBN 978-1-59593-706-3. doi:10.1145/1250662.1250665 (cited on pages 7, 8, 53).

[FJ02] Mario A.T. Figueiredo and A.K. Jain. "Unsupervised learning of finite mixture models". In: *Pattern Analysis and Machine Intelligence* 24.3 (Mar. 2002), pages 381–396. ISSN 0162-8828. doi:10.1109/34.990138 (cited on page 30).

[FS99] Jason Flinn and M. Satyanarayanan. "PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications". In: *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*. WMCSA '99. Washington, DC, USA: IEEE Computer Society, 1999, pages 2–. ISBN 0-7695-0025-0 (cited on page 7).

[Gur+02] Sudhanva Gurumurthi et al. "Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach". In: *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*. HPCA '02. Washington, DC, USA: IEEE Computer Society, 2002, pages 141– (cited on page 7).

[Ham08] James Hamilton. *Annual Fully Burdened Cost of Power*. Dec. 6, 2008. http://perspectives.mvdirona.com/2008/12/06/AnnualFullyBurdenedCostOfPower.aspx (cited on pages 6, 9).

[Hea+05] Taliver Heath et al. "Energy Conservation in Heterogeneous Server Clusters". In: *Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. PPoPP '05. Chicago, IL, USA: ACM, 2005, pages 186–195. ISBN 1-59593-080-9. doi:10.1145/1065944.1065969 (cited on page 8).

[IM03] Canturk Isci and Margaret Martonosi. "Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data". In: *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO 36. Washington, DC, USA: IEEE Computer Society, 2003, pages 93–. ISBN 0-7695-2043-X (cited on page 7).

[Jan01] J. Janzen. *Calculating memory system power for DDR SDRAM*. 2001 (cited on page 8).

[Kam13] Bernd Kampl. "Characterization and Modelling of Server Energy Consumption". Master's thesis. Graz University of Technology, 2013 (cited on pages 8, 34).

[Kan+10] Aman Kansal et al. “Virtual Machine Power Metering and Provisioning”. In: *Proceedings of the 1st ACM Symposium on Cloud Computing*. SoCC ’10. Indianapolis, Indiana, USA: ACM, 2010, pages 39–50. ISBN 978-1-4503-0036-0. doi:10.1145/1807128.1807136 (cited on page 7).

[Kan] Aman Kansel. *Building a More Efficient Data Center – from Servers to Software*. http://research.microsoft.com/pubs/183609/BuildingMoreEfficientDC.pdf (cited on page 7).

[Kat09] R.H. Katz. “Tech Titans Building Boom”. In: *Spectrum, IEEE* 46.2 (Feb. 2009), pages 40–54. ISSN 0018-9235. doi:10.1109/MSPEC.2009.4768855 (cited on page 6).

[KGS06] Youngjae Kim, S. Gurumurthi, and Anand Sivasubramaniam. “Understanding the performance-temperature interactions in disk I/O of server workloads”. In: *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*. Feb. 2006, pages 176–186. doi:10.1109/HPCA.2006.1598124 (cited on page 8).

[LB06] Benjamin C. Lee and David M. Brooks. “Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction”. In: *SIGPLAN Not.* 41.11 (Nov. 2006), pages 185–194. ISSN 0362-1340. doi:10.1145/1168918.1168881 (cited on page 8).

[Len13] Ricardo Lent. “A model for network server performance and power consumption”. In: *Sustainable Computing: Informatics and Systems* 3.2 (2013), pages 80–93. ISSN 2210-5379. doi:10.1016/j.suscom.2012.03.004 (cited on page 8).

[LTG12] Adam Wade Lewis, Nian-Feng Tzeng, and Soumik Ghosh. “Runtime Energy Consumption Estimation for Server Workloads Based on Chaotic Time-series Approximation”. In: *ACM Trans. Archit. Code Optim.* 9.3 (Oct. 2012), 15:1–15:26. ISSN 1544-3566. doi:10.1145/2355585.2355588 (cited on pages 8, 53).

[Li+12] Yanfei Li et al. “An Online Power Metering Model for Cloud Environment”. In: *2013 IEEE 12th International Symposium on Network Computing and Applications* (2012), pages 175–180. doi:10.1109/NCA.2012.10 (cited on pages 8, 53).

[LPF10] Min Yeol Lim, Allan Porterfield, and Robert J. Fowler. “SoftPower: fine-grain power estimations using performance counters”. In: *IEEE International Symposium on High Performance Distributed Computing*. 2010, pages 308–311. doi:10.1145/1851476.1851517 (cited on page 8).

[Lin+08] Jiang Lin et al. “Software Thermal Management of Dram Memory for Multicore Systems”. In: *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS ’08. Annapolis, MD, USA: ACM, 2008, pages 337–348. ISBN 978-1-60558-005-0. doi:10.1145/1375457.1375496 (cited on page 8).

[Mai+14] Jason Mair et al. “Myths in power estimation with Performance Monitoring Counters”. In: *Sustainable Computing: Informatics and Systems* 4.2 (2014). Special Issue on Selected papers from EE-LSDS2013 Conference, pages 83–93. ISSN 2210-5379. doi:10.1016/j.suscom.2014.03.007 (cited on pages 8, 23, 39, 42, 43).

[ME09] Lauri. Minas and Brad. Ellison. *Energy efficiency for information technology how to reduce power consumption in servers and data centers*. 2009 (cited on page 7).

[Nat+09] Ripal Nathuji et al. “Feedback Driven QoS-Aware Power Budgeting for Virtualized Servers”. In: (2009) (cited on page 7).

[Ned+08] Sergiu Nedevschi et al. “Reducing Network Energy Consumption via Sleeping and Rate-adaptation”. In: *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. NSDI’08. San Francisco, California: USENIX Association, 2008, pages 323–336. ISBN 111-999-5555-22-1 (cited on page 8).

[OLG10] A.-C. Orgerie, L. Lefevre, and J.-P. Gelas. “Demystifying energy consumption in Grids and Clouds”. In: *Green Computing Conference, 2010 International*. Aug. 2010, pages 335–342. doi:10.1109/GREENCOMP.2010.5598295 (cited on pages 8, 10, 49, 53).

[Pat+03] Chandrakant D. Patel et al. “Smart Cooling of Data Centers”. In: *ASME Conference Proceedings* 2003.36908b (2003), pages 129–137. doi:10.1115/IPACK2003-35059 (cited on page 6).

[Pat+13] MichaelK. Patterson et al. “TUE, a New Energy-Efficiency Metric Applied at ORNL’s Jaguar”. English. In: *Supercomputing*. Edited by JulianMartin Kunkel, Thomas Ludwig, and HansWerner Meuer. Volume 7905. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pages 372–382. ISBN 978-3-642-38749-4. doi:10.1007/978-3-642-38750-0_28 (cited on pages 7, 50).

[Pro07] U.S. Environmental Protection Agency: ENERGY STAR Program. *Report to Congress on Server and Data Center Energy Efficiency: Public Law 109-431*. Aug. 2, 2007 (cited on page 6).

[Pro15] Apache Tomcat Project. *Apache Tomcat Website*. Mar. 2015. `https://tomcat.apache.org` (cited on page 15).

[Ran+06] Parthasarathy Ranganathan et al. “Ensemble-level Power Management for Dense Blade Servers”. In: *SIGARCH Comput. Archit. News* 34.2 (May 2006), pages 66–77. ISSN 0163-5964. doi:10.1145/1150019.1136492 (cited on page 6).

[Raw04] Freeman Rawson. *MEMPOWER: A simple memory power analysis tool set*. 2004 (cited on page 8).

[Riv08] Suzanne Marion Rivoire. “Models and Metrics for Energy-efficient Computer Systems”. AAI3313649. PhD thesis. Stanford, CA, USA, 2008. ISBN 978-0-549-62316-8 (cited on page 7).

[RRK08] Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. “A Comparison of High-level Full-system Power Models”. In: *Proceedings of the 2008 Conference on Power Aware Computing and Systems*. HotPower’08. San Diego, California: USENIX Association, 2008, pages 3–3 (cited on pages 7, 8, 49).

[Riv+07] Suzanne Rivoire et al. “Models and Metrics to Enable Energy-Efficiency Optimizations”. In: *Computer* 40.12 (2007), pages 39–48. ISSN 0018-9162. doi:10.1109/MC.2007.436 (cited on pages 7, 43).

[Rut09] S. Ruth. “Green IT More Than a Three Percent Solution?” In: *Internet Computing, IEEE* 13.4 (July 2009), pages 74–78. ISSN 1089-7801. doi:10.1109/MIC.2009.82 (cited on page 8).

[SBM09] Karan Singh, Major Bhadauria, and Sally A. McKee. “Real Time Power Estimation and Thread Scheduling via Performance Counters”. In: *SIGARCH Comput. Archit. News* 37.2 (May 2009), pages 46–55. ISSN 0163-5964. doi:10.1145/1577129.1577137 (cited on page 8).

[SC01] A. Sinha and A.P. Chandrakasan. "JouleTrack-a Web based tool for software energy profiling". In: *Design Automation Conference, 2001. Proceedings*. 2001, pages 220–225. doi:10.1109/DAC.2001.156139 (cited on page 7).

[Sno+09] David C. Snowdon et al. "Koala: A Platform for OS-level Power Management". In: *Proceedings of the 4th ACM European Conference on Computer Systems*. EuroSys '09. Nuremberg, Germany: ACM, 2009, pages 289–302. ISBN 978-1-60558-482-9. doi:10.1145/1519065.1519097 (cited on page 7).

[SH01] P. Stanley-Marbell and M.S. Hsiao. "Fast, flexible, cycle-accurate energy estimation". In: *Low Power Electronics and Design, International Symposium on, 2001.* 2001, pages 141–146. doi:10.1109/LPE.2001.945390 (cited on page 7).

[Ste+08] M. Steinder et al. "Coordinated management of power usage and runtime performance". In: *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*. Apr. 2008, pages 387–394. doi:10.1109/NOMS.2008.4575159 (cited on page 8).

[SF10] Balaji Subramaniam and Wu-chun Feng. "Statistical Power and Performance Modeling for Optimizing the Energy Efficiency of Scientific Computing". In: *Proceedings of the 2010 IEEE/ACM Int'L Conference on Green Computing and Communications & Int'L Conference on Cyber, Physical and Social Computing*. GREENCOM-CPSCOM '10. Washington, DC, USA: IEEE Computer Society, 2010, pages 139–146. ISBN 978-0-7695-4331-4. doi:10.1109/GreenCom-CPSCom.2010.138 (cited on page 8).

[Tra13] Harald Tranninger. "Application specific Power Estimation for Virtualized Data Centers". Master's thesis. Graz University of Technology, 2013 (cited on page 8).

[Wan+14] Di Wang et al. "Underprovisioning Backup Power Infrastructure for Datacenters". In: *SIGPLAN Not.* 49.4 (Feb. 2014), pages 177–192. ISSN 0362-1340. doi:10.1145/2644865.2541966 (cited on page 6).

[Yan+14] Hailong Yang et al. "iMeter: An integrated VM power model based on performance profiling". In: *Future Generation Computer Systems* 36 (2014), pages 267–286. ISSN 0167-739X. doi:10.1016/j.future.2013.07.008 (cited on pages 8, 23).

[Zed+03] John Zedlewski et al. "Modeling HardDisk Power Consumption". In: *USENIX Conference on File and Storage Technologies*. 2003 (cited on page 8).

[Zen+02] Heng Zeng et al. "ECOSystem: Managing Energy As a First Class Operating System Resource". In: *SIGPLAN Not.* 37.10 (Oct. 2002), pages 123–132. ISSN 0362-1340. doi:10.1145/605432.605411 (cited on page 7).

[Zha+13] Xiao Zhang et al. "A high-level energy consumption model for heterogeneous data centers". In: *Simulation Modelling Practice and Theory* 39 (2013). S.I.Energy efficiency in grids and clouds, pages 41–55. ISSN 1569-190X. doi:10.1016/j.simpat.2013.05.006 (cited on pages 8, 43, 53, 54).