Master‘s Thesis

Application specific Power Estimation for Virtualized Data Centers

Harald Tranninger, BSc

Institute for Technical Informatics
Graz University of Technology
Austria

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

Graz, May 2013

Abstract

Cloud Computing has become very popular in the last couple of years. The fast growing market share is seen as a challenge for Data Center technologies. Huge Data Centers are the basis of distributed computing in order to handle the tremendous amount of data. Processing all these workloads requires enormous amounts of energy. Green IT is well known in conjunction with energy awareness in the field of cloud computing. Data Center Infrastructure Management in addition is one approach of making computing more efficient. Most research is done in cooling and resource management. Only few investigations are made in server run time environments itself. Even less research can be found at the application level. This work analyses the power, consumed by applications which are hosted in virtual or physical systems. A state of the art power monitoring tool called Papillon in conjunction with an application stress test tool named vApus, forms a unique test bench. The evaluation part of this work monitors the power consumption of a web-based application under typical workloads in different hosting environments. The final conclusion outlines, that Green IT has to start at the base, with analysing an applications architecture, in order to decide which hosting approach is more efficient. By using smart hosting technologies, tremendous amounts of power can be saved. The work also gives insights on future research topics and challenges to be investigated in.

Kurzfassung

Cloud Computing ist ein Begriff, der in der heutigen Zeit fast nicht mehr weg zu denken ist. Schnell wachsende Märkte und zunehmende Mobilität sind mitverantwortlich für das rasante Wachstum von Server Farmen. Eine Folge davon ist der steigende Energieverbrauch, der weltweit als bedenklich eingestuft wird. Green IT bezeichnet eine Initiative, die durch Methoden, wie zum Beispiel gezieltes Management der Infrastruktur den Energieverbrauch in Rechenzentren optimieren soll. Um langfristige Erfolge im Energiemanagement zu verzeichnen, müssen Server effizient eingesetzt werden. Diese Arbeit untersucht die Energieaufnahme von Applikationen auf physikalischen und virtuellen Servern. Papillon, ein auf Software basierendes Werkzeug, misst den Stromverbrauch von Servern und bietet eine umfangreiche REST API. Aufbauend auf diesem Tool wurde eine Webapplikation entwickelt, welche in Echtzeit den Stromverbrauch ermittelt und grafisch darstellt. Durch die Verwendung von vApus, einer umfangreichen Stress-Test Software, konnten Applikationen entsprechend belastet werden, um ein möglichst realistisches Szenario zu erhalten. Unterschiedliche Energieaufnahmen der Applikationen in physikalischen und virtuellen Systemen wurden im Detail analysiert. Rückschlüsse auf die entsprechende Architektur der Anwendung werden gezogen, sowie mögliche zukünftige Ansätze vorgestellt.

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

Acknowledgements

I am indebted to my colleagues at the UCD, CSI (School of Comupter Science and Informatics), and the ITI (Institute for technical Informatics) who have provided invaluable help and feedback during the course of my work.
I especially wish to thank my advisors, Christian Steger, Damian Dalton and Abhay Vadher, for their immediate attention to my questions and endless hours of toil in correcting draft versions of this thesis.

Special mention goes to my family for all their support during my studies in Graz as well as abroad. Especially to mention my lovely partner Claudia Passrucker which showed a tremendous amount of sympathy during my studies.
I also would like to thank Mario Polaschegg and Rob Quigley helping me with any technical questions during my work at UCD and ITI. A big thanks goes to Andreas Adlung from the Red Bull Mediahouse, helping me to achieve the last missing puzzle in my thesis.

Last but not least I say thanks to all my friends, surrounding me and supporting me in any situation. Giving me hope and faith during my whole studies. Especially Peter Riegler is seen to be a key figure, leading me to where I am now.

Harald Tranninger
Graz, Austria, May 2013

Contents

List of Figures

Chapter 1

Introduction

“ To design and deliver the best innovative, non-invasive and accurate power analysis solutions and service for the data centre environment that support the Green agenda, reduce operational costs and increase efficiencies. ”

[Stratergia Mission Statement]

1.1 Motivation

Cloud environment providers aim to run their data centers as efficient as possible. The PUE (Power Usage Effectiveness) is highly valuable and indicates how efficient a data center operates. Energy usage of a data center constitutes the majority of operating costs and consist of server power and cooling. Cooling typically adds 50% to the server energy bill. Managing energy consumption is a major challenge in the cloud environment. One aspect of this problem can concern the determination of power at the application level. This level of energy resolution permits individual processes on a server to be monitored. Analyzing the power consumption of single different services is seen to be a milestone in data center technology. Once the power consumption of a single application can be determined, cloud environment providers can improve their PUE as well as provide much more flexible charging models.

Stratergia Ltd. offers a powerful tool called Papillon which measures the power consumption of processes running on different servers. In addition, Sizing Servers Lab, provides a unique Benchmark solution called Vapus to stress test services on multiple servers. Both solutions used in conjunction can produce a system whereby the I/O conditions of an application can be emulated in real-time and its corresponding power consumption determined. Based on this investigation, applied stress tests ensure that an application is scalable enough before it becomes available in the cloud.

Cloud providers use several different server technologies to host customer related applications. Depending on the type of application, certain server technologies can be more power effective than others.
By developing a server power analysis tool, the power profile of a reference application workload can be analyzed on physical machines and then compared to a virtual server environment. The development of a SPA (Server Power Analysis) tool, as well as the overall approach description are given in chapter 4.

Every environment has its own power profile while processing certain type of workloads. The goal of this thesis it to determine the power usage of given workloads in different hosting environments. Based on these observations, the following key questions are answered.

- Is there a classification for certain types of application workloads, being processed more efficiently on a physical or virtual system?

- What is seen to be interesting and valuable for the future, based on the approaches made in this thesis?

1.2 Thesis Outline

The term cloud computing has become really popular in the last couple of years. With the fast growing market share, corresponding Data Center technologies are facing new challenges. Huge modern Data Centers are required to handle the workload requested by cloud services. The cloud offers great opportunities but comes with a list of drawbacks as well. If cloud computing would be a reference country, its overall electricity consumption would be ranked on position five on a global chart after the countries like the US, China or Russia. Chapter 2 gives a basic introduction in the fundamental topics of cloud computing, as well as in Data Center Infrastructure Management.

Different server energy monitoring technologies are discussed in chapter 3 and are compared with Papillon. Papillon is a software based server power monitoring approach invented by Stratergia Ltd. in 2010. In addition a stress test software is presented, which offers real world workloads being applied to cloud environments.

Chapter 4 includes the main part of the thesis. The development of a server power analysis tool in order to monitor the power consumption of individual application workloads is documented in detail. The effectiveness of Papillon in conjunction with the SPA tool is evaluated and drawbacks as well as solutions for future work can be found. By isolating applications on a physical or virtual server environment, the power consumption of each application can be monitored by Papillon.

Chapter 5 evaluates the power effectiveness of a given Application. With the developed SPA (server Power Analysis) tool, a cloud application workload is being monitored on a physical and virtual machine. The results give insights on power efficiency depending on the type of application and hosting environment.

Chapter 6, summarizes the the thesis with the essential conclusion and future work to be done.

Chapter 2

Fundamentals

2.1 Green Cloud Computing

Hundred years ago, companies had to build local power plants to operate their machines. With the invention of the electric grid, power generation was outsourced. Building central power plants and supplying the industry via a large power grid became state of the art. Cloud computing is seen as revolutionary innovation in the IT market. Instead of processing and having data storage capacities on every single workstation, large Data Centers can easily supply customers via the internet. The Big Switch, Carr [2009], discusses pros and cons of former technology decisions and gives a short outlook on cloud computing in the future.
Besides the technological progress made in the IT area, environmental questions and aspects need to be considered as well. The society is used to being dependent on power supplier and a well organized power grid. But being dependent on Data Centers and the internet is far from being accepted. Using centralized computing power might save energy, but dependency issues are seen as drawbacks. In April 2011, Cook [2011] published an article about the environmental aspects of data processing. Due to lack of information transparency, Greenpeace could not get detailed information on whether the cloud is more energy efficient than the traditional desktop solution.
Instead of saving energy through technical innovations and increasing efficiency, the rising demand in the efficient technology uses even more. This postulate is called the Khazzoom-Brookes and is discussed in section 2.1.1.
The subsection on carbon footprints of Data Centers outlines the problem of powering IT services with dirty energy. Data Centers consume huge amounts of energy due to inefficient infrastructure. Cooling as well as server hosting can be improved in order to save energy. Within the section Data Center Infrastructure Management (DCIM) possible approaches are listed to improve efficiency.

2.1.1 Carbon Footprint

Carbon dioxide emissions are known to be an issue for the whole environment and should be reduced to a minimum. Dependencies on dirty energy like fossil fuels and nuclear power makes it difficult for the green agenda. Especially in the IT sector, leading companies use dirty energy to power their Data Centers. Experts see a huge potential in the IT business to reduce the use of dirty energy and minimize the carbon footprint. Greenpeace claims that technology of the 21st Century is still powered with 19th and 20th Century energy. The rapid growing IT market struggles with building green powered Data Centers. The cloud is often seen as the green solution, turning dirty computing into green IT. Due to competitive issues, very few companies provide detailed information on efforts turning their Data Centers into energy efficient ones.[Cook, 2011]

Google outlines their carbon footprint and gives samples how the values are related to real life. A single search query in Google search requires 0.0003 kWh which can be translated into roughly 0.2 g of CO_2 emissions. The amount of carbon dioxides being equivalent to a certain amount of electricity depends on the source of energy. Google claims to use 33% of renewable energy by 2011. 100 search queries are roughly equivalent to 20 g of CO_2 emission or ironing a singe T-shirt.[Google, 2013]

Estimating the green house gas emissions of the IT sector is seen to be challenging. The climate group's report published in 2008, Smart 2020, reviews carbon emissions and states that 2% of global green house gas emissions are due to IT usage. [Group, 2008]

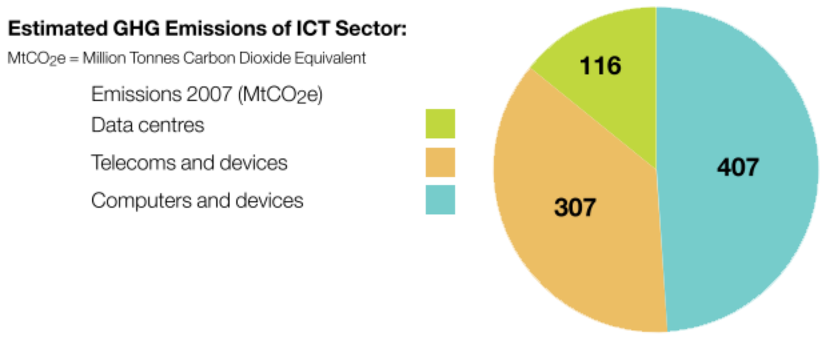


Figure 2.1: Green house gas emissions of IT sector 2007 [Cook, 2012]

In the *Make IT Green* report published by Greenpeace 2010, the electric power consumption of cloud computing is calculated and compared with the annual electricity usage of countries. With a demand of 623 billion kWh in 2007, cloud computing requires more electricity than India. The IT sector is seen to be within the top five most energy hungry countries on the globe.[Cook, 2012]

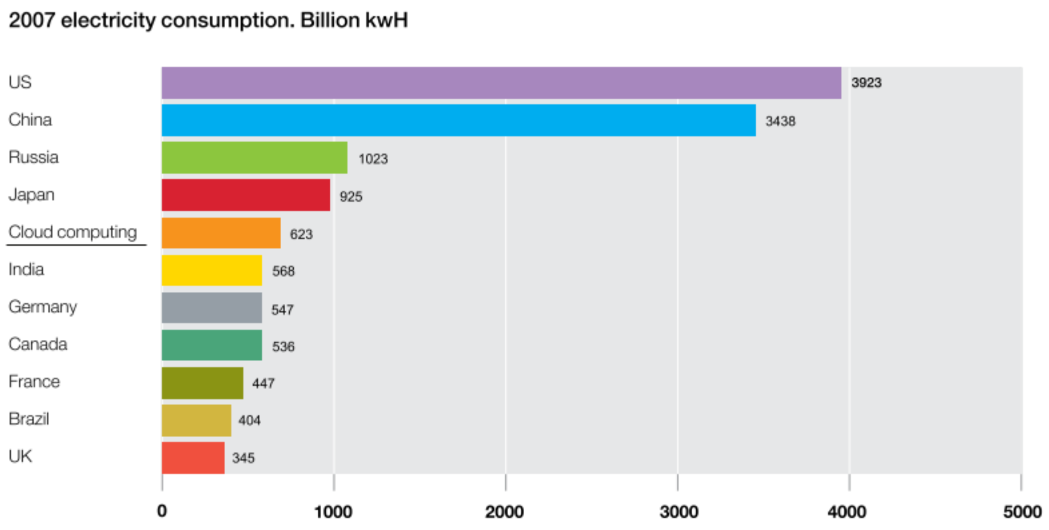


Figure 2.2: Countries's electricity consumption 2007 [Cook, 2012]

2.1.1.1 Key metrics

In order to determine the energy effectiveness of Data Centers, several different key metrics are utilised within the Green IT sector.

- **PUE(Power usage effectiveness)**

  The PUE value gives information on how efficient a Data Center uses its electrical power. The PUE is the ratio between the total facility power and the IT equipment power.

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

  The total facility power consists of everything using electricity in a Data Center like cooling and lighting for example. The IT equipment power covers all servers and network communication devices required to run the Data Center.
  A perfect PUE has a value of 1. This indicates that all the facility's electricity is used for computing and nothing else. The PUE is seen as an efficiency benchmark in the industry. While recently a PUE of 2.0 has been quite good, values with 1.5 and lower are state of the art now. Companies try to get a green image by PUE optimization. The PUE is misused in terms of the green agenda. The value does not give any information on the amount or type of electricity used. Data Centers with a value of 1.1 still might use coal power electricity. In contrast values with 2.0 in combination with renewable energy is seen to be much greener. A simple example shows how the PUE can be misused. Turning servers into idle mode and therefore running virtual machines elsewhere, results in saving computing power. Using less

computing power, but still requiring the same facility power results in a high PUE. Greenpeace suggests to add additional parameters like the carbon emission to the equation. [Cook, 2012], [Belady, 2010b]

	total facility power	IT equipment power	PUE
without VM	24 MW	18 MW	1.33
with VM	24 MW	12 MW	2.0

Table 2.1: Misleading PUE example

- **DCiE (Data Center infrastructure Efficiency)**

The DCiE value is the reciprocal of the PUE. The data center infrastructure Efficiency value faces the same challenges as the PUE. By virtualizing Data Centers, the overall efficiency of the Data Center still might be worse than with physical machines. In order to avoid this scenario, virtualization has to be planned carefully to get the infrastructure into right place. [Belady, 2010b]

$$DCiE = \frac{1}{\text{PUE}} \tag{2.2}$$

- **WUE (Water Usage Effectiveness)**

The water consumption of Data Centers is fairly important in the green agenda. Inefficient cooling or humidification lead to massive usage of water which is neither green nor cost efficient as well. Depending on the geographical area, water can be more or less green. Using water cooling in dry and hot areas can be highly inefficient. In contrast areas with big rivers and lakes are more likely to supply clean cooling to the Data Center. [Patterson, 2011]

$$WUE = \frac{\text{Annual Water Usage}}{\text{IT Equipment Energy}} \tag{2.3}$$

- **CUE (Carbon Usage Effectiveness)**

CUE is a metric measuring the carbon emissions created by the Data Center itself. Initial emissions caused by IT assets being manufactured are not included.The CUE metric has got kilograms of carbon dioxide per kilowatt hour as a unit.
[Belady, 2010a]

$$CUE = \frac{\text{Total CO2 emissions caused by the total Data Center Energy}}{\text{IT Equipment Energy}} \tag{2.4}$$

- **DCP (Datacenter Productivity)**

  Data Center productivity is an indicator that shows the relation between the overall output and the therefore required resources. [Belady, 2010a]

$$DCP = \frac{\text{Useful Work Produced by Data Center}}{\text{Resource Consumed Producing the Work}} \tag{2.5}$$

- **DCcE (Datacenter compute Efficiency)**

  This metric gives an overview whether computing resources are used in an efficient way or not. Idle running servers can be switched of or loaded with tasks in case computing capacity is missing. Depending on the degree of DCIM, computing resource are load balanced and used as efficient as possible. [Blackburn, 2010]

As Khazzoom and Brookes postulated 30 years ago, just making Data Centers more efficient does not include to save energy.[Cook, 2012]

2.1.1.2 Khazzoom Brookes postulate

The economists Khazzoom and Brookes postulated a modern version of the Jevons paradox, describing the problem of increasing energy demand after improving energy efficiency. 1865 William Stanley Jevons found an interesting relation between making technology more efficient and the resulting demand. He observed, that the invention of a new steam engine, using less coal than the old technology raises the demand in the industry. Modern economists call this phenomenon the rebound effect from improved energy efficiency.[Polimeni, 2008]
Improving the efficiency of products, causes a price drop and low costs lead to an increased demand. Khazzoom and Brookes picked up Jevons paradox in the 1980s to describe the societies energy problem. New efficient technology innovations lead to an increasing demand resulting in using even more energy. Critical reports claim that green IT approach suffers from the same issues as Khazzoom and Brookes outlined 30 years ago.[Saunders, 1992]

“ We are not going to solve the climate problem via efficiency - we must move to cleaner sources of energy. ”

[BillWheil,Google Energy Czar (11March 2011 - ClimateOne Forumon Cloud Computing)]

Instead of improving efficiency, Wheil suggests to invest in new clean sources of energy. Choosing a good location during the planning phase of a Data Center offers the oportunity to access clean sources. Areas with wind, hydro, tidal or solar power and low temperatures are of interest. Nontheless leading IT companies Companies are more economically rather than environmentally aware. In the future, companies have to make commitments and stick to policies supporting green and sustainable energy sources.

2.1.2 Data Center Infrastructure Management (DCIM)

Outsourcing computing power, requires larger Data Centers and a robust architecture. Therefore servers process queries non stop all year round. Data Centers already seem to be working to their full capacity. In reality, lots of them waste processing capacity due a lack of Data Center infrastructure management. In former times server housing was less complex than nowadays. Datacenter facility managers used to organize administrative processes with simple spreadsheets. Nowadays server farms are far too complex for such approach. Most Data Center downtime occurs due to human errors. By using smart management tools, downtime as well as operational costs can be reduced. Due to the fact, that existing Data Centers will run out of space in the future, the emerging DCIM sector needs to be investigated. [An, 2011]

Data Center Infrastructure Management (DCIM) is the integration of information technology and facility management systems into a centralized control structure with intelligent capacity planning of Data Center’s critical systems. Managing IT assets, asset deployment as well as resource and space capacity planning is part of the management process. The following Data Center attributes are typical components of a DCIM system.

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

Nlyte, a leading company in the field of DCIM, published the so called DCIM maturity model. This model, seen in Fig. 2.3, outlines different levels of Data Center management complexity. Most server housing companies are to be found at the lowest level of maturity.

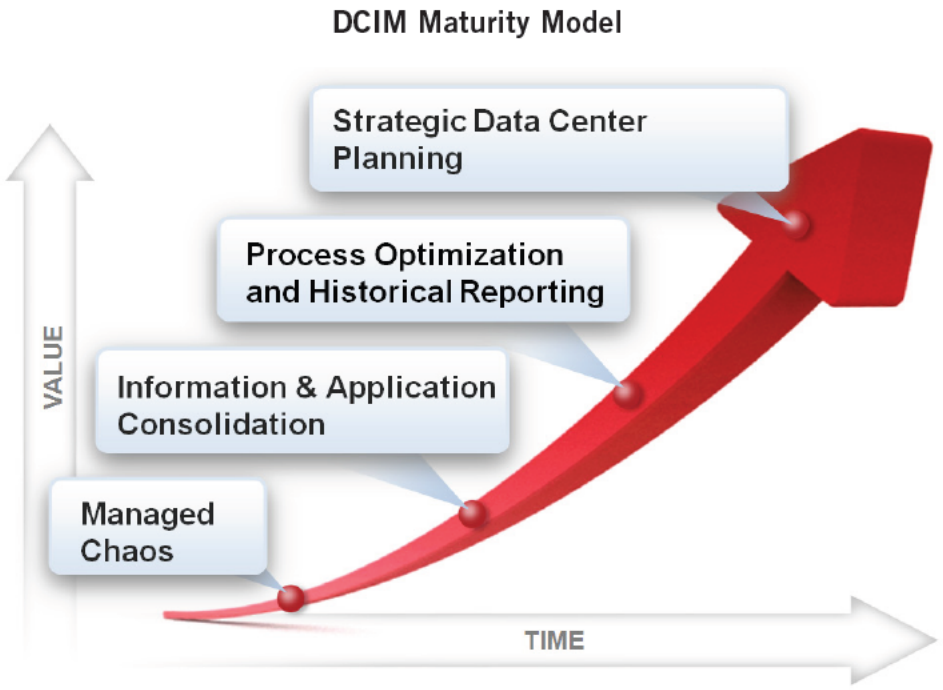

DCIM Maturity Model
Strategic Data Center Planning
Process Optimization and Historical Reporting
Information & Application Consolidation
Managed Chaos
VALUE
TIME

Figure 2.3: Data Center Infrastructure Management Maturity Model

Often they are afraid to make changes in their work-flows and utilities. Using modern management utilities instead of spreadsheets takes the Data Center to the next level.

The phase 2 requires the introduction of an elementary DCIM system. Spreadsheets, reports, floor and rack visualization as well as power monitoring is organized. Accordingly human error rates are lower, which results in less downtime over a year. Phase 3 requires fundamental process optimization. Real time information and historical events are used to control the work-flow. This type of abstraction offers better forecast possibilities and is crucial to take the phase. The top level of the DCMM is called *strategic data center planning*. For large Data Centers, it is crucial to forecast *what-if* scenarios. Potential downtime risks can be detected by analyzing power usage and cooling systems. Space issues are also seen to be a risk and therefore need to be handled. The work-flow is being automated as much as possible to reduce the human error rate to a minimum. [M., 2011]

Gartner Kumar [2012] produced a paper on motives for using DCIM tools and reasons why DCIM is still not used in most Data Centers. Often Data Centers and corresponding facilities are not part of clear company structures. Responsibilities are shared which results in lack of organization. Existing integrated DCIM solutions are often not compatible and cause concern with respect to overlapping processes and additional costs. A large subset of data center infrastructure management tools for different categories are to be found. Evaluating the best tools and using different solutions for various categories seems to be too complex for data center managers. Therefore DCIM should be considered during the initial design stage of Data Centers. [Kumar, 2012]

Efficient space planning leads to a smart cost saving cooling system. A good air circulation keeps the server temperature low and the additional fan cooling to a minimum. DCIM is seen to be important for every level of complexity.

2.1.3 Power Cooling and Space issues

The breakdown of power consumption in Data Centers has been analyzed by Oracle in a white paper Oracle [2010]. The following Fig. 2.4 shows the power decomposition components in a Data Center.

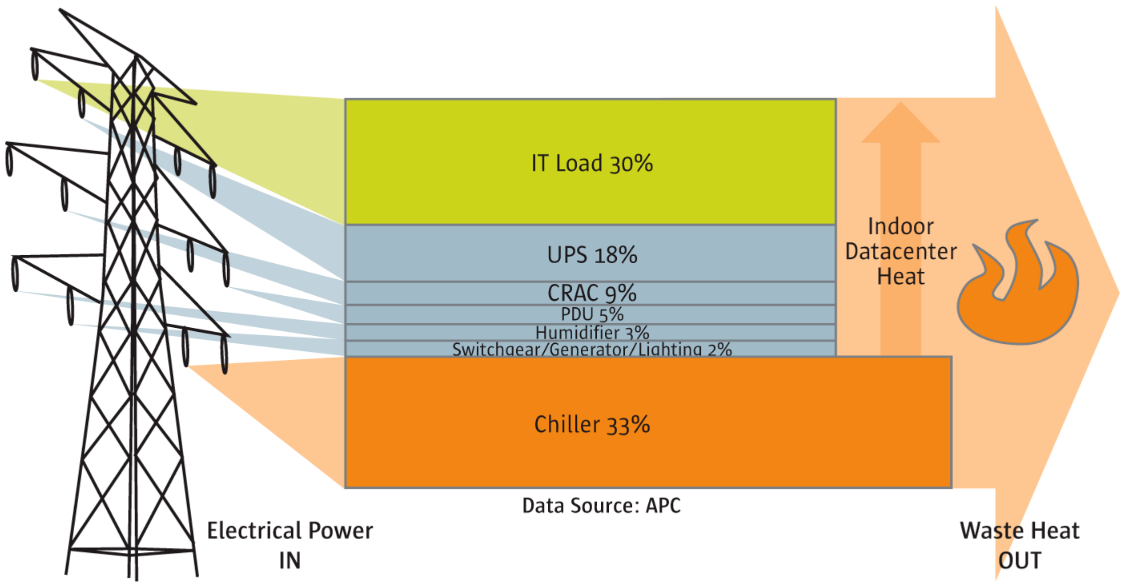


Figure 2.4: Electricity consumption by component in a Data Center [Oracle, 2010]

Chillers use at least 33% of the overall power consumption and tend to require even more if the indoor temperature increases. Older servers, which don´t have state of the art cooling systems, might push the power consumption for cooling and IT even higher than it is stated in the Figure above. Picking servers with a simple hardware structure guarantees a straight air flow and in addition better cooling. Reducing empty rack slots in order to avoid areas where cold and hot air can mix up is seen to be a key as well. A golden rule is to keep the airflow always straight and avoid empty spaces. Special racks and partitioning plastic curtains are used within server rooms to avoid the hot air being mixed with the cold one. Google have roused their Data Centers average operating temperature to reduce cooling costs. Running the servers at an average of 27 degree Celsius leads to significant reduction in the overall facility energy use. Thermal modeling and infrared analysis help as well to visualize the airflow in server rooms and detect possible

hotspots. [Oracle, 2010]
In addition smart cooling systems, controlled by DCIM tools are able to adjust the ventilation on demand which contributes to the overall efficiency. While most of the Data Centers are equipped with air cooling systems, only few take water cooling into account. As a leading Data Center provider Google has done research into efficient cooling mechanisms. The so called *Hot Huts* approach stores heated air which is then blown through water chilled pipes. The air then can be reused to cool down the racks. Water cooling only is suitable for certain locations. Running a Data Center close to the sea might help to use cold sea water. In contrast pumping water across long distances reduces Data Center efficiency.[Google, 2013]
In order to optimize cooling, Data Centers have to be redesigned and structured in a certain way. A large amount of old server farms are the cause for the IT´s footprint being so bad. Lots of them are just about to return the investments done several years ago. They will not be restructured from scratch and so cooling might stay still inefficient. But even small changes in Data Center facility management can lead to more efficiency.

2.2 Server virtualization in Data Centers

The tremendous increase in Data Center electricity consumption is being addressed by several approaches. One of these techniques is virtualization. By applying virtualization in certain IT environments, remarkable energy savings can be achieved.
Beside its obvious advantages the hidden challenges coming with the virtualization should be considered during initial Data Center planning phases. Virtualization is a technique to emulate a full operating server within software. A single physical server is able to run multiple virtual machines, sharing its hardware resources.

Due to its high workload a virtual server consumes much more energy than a physical one. Corresponding IT infrastructure has to be adopted in order to supply the rack with required electricity. Differences between existing hypervisors additionally add a certain electricity overhead to the overall power consumption. Related to the workload various virtual machines are required. The dynamic migration of virtual machines cause the electricity footprint to be alternating. DCIM tools therefore have to take such variations into account in order to provide smart cooling.

Integrating virtualization in a Data Center can be challenging due to required IT infrastructure adoption. The idea of turning each physical server in a virtual one to increase computing capacity is wrong. Not every physical server is a potential virtual one. A virtual server often requires high performance components, to handle continuously utilization close to 100%. By consolidating servers, the hosting environment becomes more critical. In case of a server defect, the virtualized environment affects much more ma-

chines than only a physical one. Existing rack and cooling infrastructure might lead to efficiency problems which could effect the whole virtualization concept. Virtualization is something that has to be taken into account during the early stages of planning Data Centers. Otherwise the overall PUE value will be worse than with physical servers. [Loeffler, 2009] [Jin et al., 2012]
The following example shows, that the PUE value is not a valuable efficiency indicator for Data Centers being virtualized at a later stage. Lets assume, that the total IT equipment power decreases due to virtual server consolidation. The overall facility power could be still the same due to different cooling approaches.

$$PUE = \frac{\text{Total Facility Power}}{\text{IT Equipment Power}} \tag{2.6}$$

$$1.5 = \frac{6000\text{ kW}}{4000\text{ kW}} \tag{2.7}$$

$$2 = \frac{6000\text{ kW}}{3000\text{ kW}} \tag{2.8}$$

In this case, the PUE value is seen to be much higher than in a non virtualized Data Center. As mentioned in Cook [2011], this PUE is not valuable and therefore an additional carbon emission parameter should be added to the equation. Data Centers, which considered strong virtual server consolidation during their early planning phases, have an optimized total facility power value. This leads to a much better PUE than for Data Centers which did not plan virtualization from the early beginning.

2.2.1 Virtualization Technologies

In 1999, VMware extended the basic virtualization concepts from the early 60s and came up with the first virtual machine being able to run on standard x86 architectures. This step was seen to be huge in server virtualization and from there different concurrent operating systems could run on a single x86 machine. By running several virtual machines on a single physical server hardware costs are reduced to a minimum. The only drawback of virtualization is seen in the server being a bottleneck. The system is less robust in case of server crashes. Two types of virtualization technologies are defined in the literature. One is hardware level virtualization, and the other OS level virtualization.
OS level virtualization requires a host operating system to be installed below the virtual one. Most common virtual machines have a host operating system as a base layer. The host OS is responsible to handle resource sharing among the virtual machines. Hardware level virtualization is getting more and more popular for being used in dedicated virtual server environments. In contrast to the host OS, the virtualization layer of the directly interacts with resources on the hardware level. Instead of dealing with the overhead of a host OS, the directly shares CPU, disk and memory on the hardware layer. Typical virtualization software is provided by VMware and Microsoft. VMware offers the so

called ESX and ESXi solution. Microsoft competes with the Hyper-V product.
Research at University of Derby indicated that there is no significant difference in power consumption between OS and hardware virtualization architectures. By using sample workloads within both environments no major power consumption differences could be determined. [Liu et al., 2011]

2.2.2 Virtual machine migration

Server consolidation leads to high server utilization which may require dynamic virtual machine migration on demand. In order to balance workloads, snap shots of disk images are taken and moved to another server environment. This approach comes with a few drawbacks. In order to take a snap shot of the entire disk, the virtual machine has to be suspended for a few seconds or in case of large images even for minutes. Once the snap shot of the image is taken, additional time is required to copy the image to the dedicated server. The only approach to make virtual machine migration more efficient is to increase the shared resources of the parent and child machine. So called Copy on Write (CoW) mechanisms as outlined in [Sun et al., 2009], try to share memory pages of parent and child VM during run-time. By using CoW within a Xen technology, the live migration can be done within milliseconds.

Additional challenges are the determination of which physical server is fully utilized and which virtual machine being hosted, is the best one to migrate. Depending on the utilization of the destination server, a decision has to be made whether the migration is efficient or not. For instance, a highly utilized server might not be able to handle another virtual machine and so the migration process would continue on and on. Virtualization software vendors therefore provide their own live migration concepts.
Xen´s live migration technology and VMware´s Vmotion offer possibilities to live migrate virtual machines. As outlined in [Chuan et al., 2012], the most difficult part in VM migration is seen in workload hotspot prediction. The decision on VM migration should not be made by only using CPU utilization as parameter. Predicting the resources being required by the VM in the future helps a lot to minimize the migration cycles ans is seen to be the key for efficient virtual machine migration.

2.2.3 TCO and ROI

In 1987, Bill Kirwin, research director of Gartner Inc. announced the TCO model. The TCO model is used to determine hidden costs at the beginning of investments.

> *“ Gartner defines total cost of ownership (TCO) as a comprehensive assessment of information technology (IT) or other costs across enterprise boundaries over time. For IT, TCO includes hardware and software acquisition,*

management and support, communications, end-user expenses and the opportunity cost of downtime, training and other productivity losses. "

[Gartner Inc.]

In the IT sector costs seem to be easy determined by only hardware and software parts. Often additional costs for maintenance, support or downtime add up. In this case TCO models are seen to be helpful because they take those costs into account and supply a realistic cost model from the very beginning. Investigations outline, that virtualization in Data Centers can reduce operating costs an therefore lower the TCO. VMware conducted a study, analyzing the TCO within different datacenter environments, running the VMware virtualization software. Virtualization caused the TCO to be reduced to 33% of its original value. Due to physical server consolidation, Data Centers avoided expensive expansions as well as reduced server maintenance costs. The initial server purchase adds only 15% to the overall TCO model costs. This indicates again that maintenance, support, downtime and business administration have to be taken into account.

The second beneficial business indicator for Data Centers is called ROI (Return Of Investment). The ROI sets the benefits in relation to the afforded investments in certain projects.

$$ROI = \frac{\text{gain from investment - cost of investment}}{\text{cost of investment}} \tag{2.9}$$

Understanding TCO is required in order to calculate the ROI. Only when the investments with all hidden costs are determined, can the return on investments be evaluated. Virtualization can reduce TCO and offer therefore a positive ROI if it is done in a proper manner. Integrating virtualization at an early stage of Data Center planning is seen to be the key in reducing TCO. Reduced Hardware needs less maintenance and support. If virtualization was planned during early stages, less cooling is required as well. VMware claims, that customers face a significant positive ROI within the first 6 months after operating with their virtualization software. No matter which virtualization software is running in a Data Center, the most important step to get a fast positive ROI is, sufficient planning during the early stages. [2006, 2006]

2.2.4 Chargeback Models

Creating valuable chargeback models for cloud environments is less than straight forward. As outlined in section 2.2.3, costs can be hidden and therefore easily add up. Determining the total cost of ownership (TCO) helps to expose ongoing costs which have to be taken into account for chargeback. In addition capital expenditures Capex and operational expenditures Opex are a key metric in chargeback models. CapeX include all hardware and facility assets whereas OpeX addresses ongoing expenditures like maintenance, fees and payrolls.
Data Centers consist of a huge set of virtual and physical servers. Each has initial costs and requires additional maintenance, support and energy every month. Calculating the exact chargeback for customers is difficult due to varying energy costs. Depending on the workload, cooling as well as server power, add up in a different way. Cloud providers therefore introduced certain billable resources in order to chargeback. Virtual machines, server blades as well as network and security services are the main assets being taken into account. Based on the usage of these items, customers are charged in different ways. For example some of the chargeback models have a fixed price for virtual machine usage every month being independent from the processed load. Other billing models use GHz units of CPU utilization in order to determine ongoing costs. Using physical servers and virtual machines over certain periods of time is also used as a pricing model. Integrating a proper chargeback system offers a better visibility to the cloud providers and their customers. In order to determine all asset costs, proper monitoring systems have to be set in place. The most important asset to monitor is the energy consumption of the Data Center. While facility power consumption is a fixed value, cooling and server power consumption are flexible. Depending on the server utilization more power is used which also results in higher cooling costs. [Cisco, 2010]
By monitoring server power, based on metering, chargeback models can be improved in the future. Within this thesis a server power analysis tool, using the virtual metering API of Papillon, is used to deliver detailed power consumption profiles. Those can be used to improve complex chargeback systems.

2.3 Related Work

2.3.1 Application power monitoring

A study, [Da Costa and Hlavacs, 2010], done by the Department of Distributed and Multimedia Systems in Vienna investigated the power monitoring of applications running on servers. It gives detailed insights on various approaches to monitor application power and outlines difficulties as well. Their methodology is split into two approaches. The first one addresses the question of how to derive the power consumption of servers by using only OS specific metrics. The second one deals with application identification on the process level.

In order to measure the overall server power, Linux services like *pidstat* or *collectd* are used. *Pidstat* is used to gather information of single processes whereas *collectd* delivers machine-wide statistics. A synthetic workload is used to put a certain load on the server. A Watt meter, measuring the power consumption is connected to a second server in order to keep the disturbance on the server low. The framework presented in this paper has the limitation to only follow a single process. Most applications fork and so have multiple child processes. The research team in Vienna developed certain benchmarks which do not fork. By running a set of different workloads, 165 different processes and a huge subset of system variables were used to create a linear based power model. [Da Costa and Hlavacs, 2010]
As mentioned, the developed framework is only able to measure the power consumption of a single process, which does not help to identify an application‘s power in a multi process environment. Another limitation is seen in the linear power modeling approach. Real world applications do not show linear power characteristics and therefore the linear model might be less accurate. In contrast Stratergia Ltd. uses a non linear power modeling approach by using a sort of nearest neighbor algorithm. The non linear power modeling is seen to be a totally new approach and states a real innovation made by Stratergia Ltd. More on technologies being used for server power modeling can be found in the following section 3.2.

2.3.2 Physical vs. virtual machine

In addition the Politehnica University of Timisoara in Romania, published a paper on power consumption measurements of virtual machines in May 2011. They conducted a fundamental survey on power consumption measured in virtual and physical environments. VMware was used as virtualization software and *Watts up* hardware power meters in order to measure the electricity. By using a proper workload, consisting of a linux implemented blowfish cipher algorithm, virtual as well as physical systems were stressed. The power meters measured the same amount of electricity used during the stress test for both systems. Only by observing the execution time required by both servers, the difference was clear. The virtual system has a lower throughput which reduces the performance efficiency by 10 %. [Marcu and Tudor, 2011]

Using hardware meters to measure the power consumption of servers is only feasible in laboratories. Within this thesis the Papillon metering approach, developed by Stratergia Ltd, is used collectively with a customized monitoring application to measure the power of virtual and physical systems. Based on the fundamental survey conducted in the mentioned paper, the goal is seen to prove these results by using a fully software based power metering approach.
Most of the related work being done in this field used benchmarks to stress the server in order to compare power efficiency, but real world applications often show different

power profiles. A sample web-application will be stress tested with the so called vApus tool developed by Sizing Servers in order to create a real world situation. Instead of just crunching numbers, a realistic traffic is created by simulating several thousand concurrent users interacting with the server.

2.3.3 Green IT Cockpit

As mentioned above, within the field of application power analysis very few related projects are to be found. Most of the research is done in the field of DCIM (Data Center Infrastructure Management). Monitoring approaches are very low level and compete with a handful other methods on the market, whereas DCIM is seen to be of large interest. DCIM tools don´t deal with the question of how to determine the power consumption of servers. They only take the retrieved power values in order to outline the overall power consumption of the racks or even Data Centers. By getting detailed power information on racks, load-balancers are able to balance the workload.

Based on fundamental performance indicators a research team of the TU Berlin developed the so called *Green IT Cockpit* in order to monitor the energy efficiency of information and communication technology systems. The Green IT Cockpit competes with various DCIM solutions on the market. All of them use underlying monitoring solutions in order to retrieve insights on server power consumptions. As outlined in chapter 3.1 several monitoring approaches are available on the Market. Agent-less as well as agent based power monitoring is used within certain DCIM solutions. The research team, investigating in the *Green IT Cockpit* project, executed a state of the art analysis of existing server power monitoring approaches. A corresponding project paper gives details on every single monitoring approach on the market, being relevant for DCIM solutions. Stratergia developed a completely new monitoring approach and faces very few competitors. In the following chapter 3, section 3.1 describes the most relevant monitoring solutions which are related to the Papillon system. A detailed research on other approaches can be found in the published paper written by Koray Erek, Gregor Drenkelfort and Thorsten Pröhl from the TU Berlin. [Erek, 2013]

Chapter 3

Approaches

3.1 Server Power Measurement approaches

Data-center infrastructure management has become a key methodology for data-center managers. The variety of DCIM tools is large and estimated to get even larger. Choosing the right management tools can be challenging due to a fast changing market. These days only a few companies provide a DCIM solution for every single domain. Data center managers have to deal with several different software products and often struggle with their interoperability. Beside managing infrastructure assets, the domain of monitoring a data-center´s power consumption is seen to be very important. Metering a servers power consumption gives detailed information on the utilization. Smart DCIM solutions use this information to shift workloads in order to increase the computing efficiency. Instead of running a machine with only 60 percent load while another server just uses 30 percent, the smart DCIM tool migrates tasks in order to run a server on full load. Stratergia is major in providing software based power metering technologies. Lots of DCIM providers put the focus on an overall integrated data center management solution. Comparing the solution Stratergia is working on with tools that already exist, is difficult. Most of the competitors are major in managing the data center infrastructure and only provide minor information on monitoring server power in real time. The following sections present competitors, having similar metering approaches like Stratergia.

3.1.1 Viridity

Schneider Electric, a mayor French electric engineering company have developed a Data-center Management software called StruxureWare. It enables real time monitoring of the cooling system, security and environmental assets. In December 2011 Schneider Electric announced the acquisition of Viridity Energy Center. Viridity complements Schneider´s DCIM tool StruxureWare and offers new possibilities in the field of server power monitoring. [Keilen, 2010]

3.1.1.1 Architecture

Viridity requires the data-center to be base-lined. The software automatically detects corresponding IT assets and their sub-components. Its monitoring method is seen to be non intrusive and does not require any agents to be installed on any server. By using SNMP, WMI, SSH and the protocol of VMware´s V-Sphere, Viridity collects server specific data in a central database. Those data are correlated with the utilization and corresponding power consumption. Viridity does not require any clients to be installed on the servers, but consequently this approach leads to a minimum amout of information. There is no processes visibility via SNMP or WMI. The calculated power consumption can not be assigned to any application on the server. This makes automated DCIM decisions much more complex. The migration of virtual machines is difficult without knowledge about processes and applications running on the server. In contrast to client based power monitoring approaches, Viridity can´t report power values in real time. Instead, scheduled monitoring is used to sample server power. The central database, collecting the data, uses a patented self learning approach to calculate valuable data more accurately. [Keilen, 2010]

3.1.2 JouleX

The private company JouleX a multi-national company has produced a software package called JouleX Energy Manager (JEM) which offers power monitoring for connected network devices. [JouleX, 2012]

3.1.2.1 Architecture

JouleX is seen to compete with Viridity because of using the same approach. Instead of agents being installed on a server, the JEM solution just listens to network specific protocols.
The following connection proxies are supported by JouleX

- Windows Management Instrumentation (WMI)
- Windows Remote Management (WinRM)
- SSH, SNMP

- IPMI and SMASH
- CISCO EnergyWise
- iLO and DRAC Cards
- Vmware vSpare

The JEM listens to the above mentioned services and protocols in a scheduled manner to retrieve workload utilization for connected devices. In contrast to Papillon, no additional process information can be retrieved. The scheduled monitoring process is seen to be not real time and therefore the JouleX approach is seen as a different technological approach. The focus is set more on management assets than on virtual metering. Even if a physical device can be monitored quite accurate, virtualized systems can´t.[JouleX, 2012]

3.1.3 Nightwtachman

Nightwatchman is a power management software, developed by 1E, a privately owned software company with its roots to be found in the United Kingdom. Beside several different IT efficiency services, the Nightwatchman Management Center offers the possibility of improving power effectiveness within a large client domain.[1e, 2011]

As shown in the architectural overview Fig. 3.1, NightWatchman can be integrated in large data-center environments. Therefore services are split and hosted on different servers. Within small environments, all services can be bundled on a single machine. NightWatchman provides so called wake-up agents which are being installed on various devices. In a scheduled manner, the clients are triggered and power values are transmitted to the central database server. In order to transmit power values, agents are installed on each host. NightWatchman offers power monitoring but does not provide any application specific power mapping. Within the Papillon environment detailed process information on every power value is available. [1e, 2011]

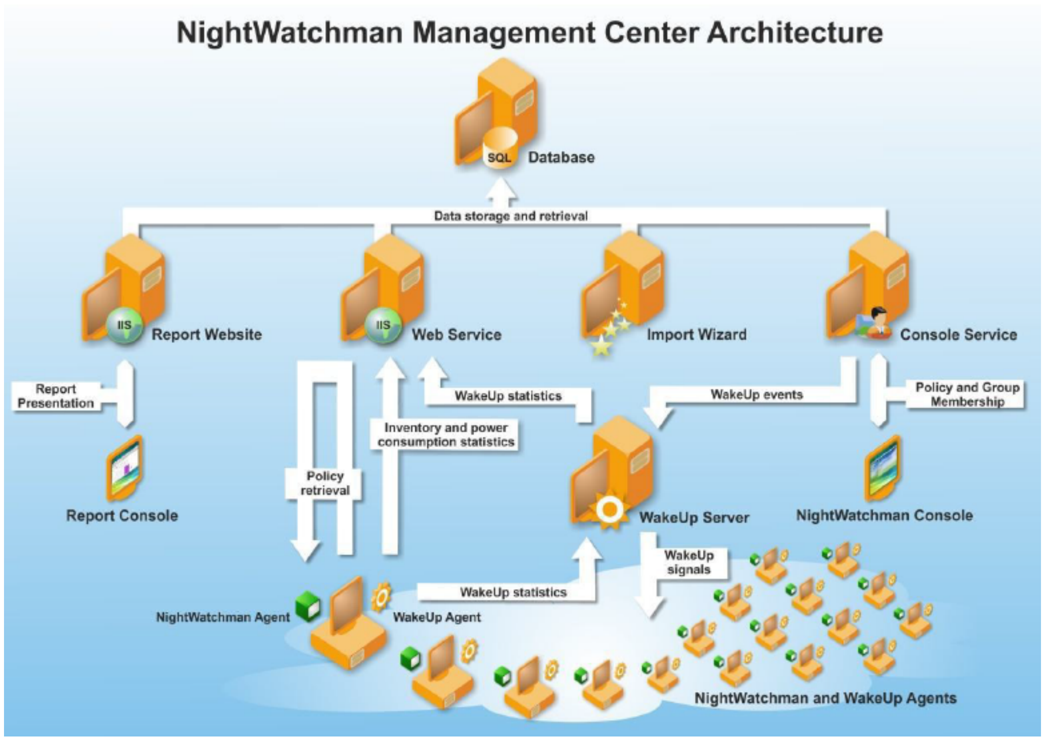


Figure 3.1: Nightwatchman Architecture

3.1.4 Agent-based vs. Agent-less appraoches

“ An agent is like a spy in the ranks, giving you a lot more information than you would get from just looking through a telescope (agentless) ”

[SU KENT / RAJPAL SINGH by 1E]

Every approach has its advantages depending on the field of application being applied to. No general rule makes an agent-less system more effective than the agent-based one or vice versa. But certain domains can be covered better with agent-based approaches than with agent-less ones. When it comes to server power management, only agents are able to access corresponding low levels of the server itself. Data can be preprocessed and only transmit valuable data to a central master system. Agent-less approaches only have access to high level services which do not expose detailed data regarding power management. Another drawback comes with being dependent on certain network services like SNMP. Network connectivity in general is crucial for agent less technology. Whereas agents on the server itself continuously store data in a buffered system. In case of network issues the whole monitoring approach still works and guarantees continuous reporting. [KENT, 2011]

3.1.4.1 Security

Installing an agent on a server is much more complicated than just pulling data via SNMP protocols. So agent less monitoring can be applied in an easy manner. The drawback coming with network based monitoring approaches is seen in security. Retrieving detailed information from servers just by using network services requires much more access rights than sending ordinary buffered data to a central master. In other words agent-less monitoring approaches require much more permission rights in the whole data-center structure than the agent-based one. Whenever a network based approach is taken into account, corresponding security policies have to be checked first. [KENT, 2011]

3.1.4.2 Scalability

Agent-based solutions are more predictable concerning scalability. The amount of collected data being sent to a central master is an absolute term that is multiplied by the number of agents in the system. So the network traffic scales up in a predictable manner. In contrast, agent-less monitoring adds an undefined number of network interactions based on scheduled reporting. In large data-center environments the network traffic is seen to be very high which is a potential risk for the infrastructure. [KENT, 2011]

3.1.4.3 Summary

Depending on the domain requirements a spy can be more effective than just using a telescope. Agent-based approaches guarantee detailed and accurate information, but still need some time to be set up correctly. Agent-less monitoring might be easier to install, but requires certain network services being available in a data-centers infrastructure. Often Windows WMI and SNMP are used withing these approaches and also have to be configured in the data-center IT infrastructure. In case of real time monitoring , agent-based solutions are much more accurate. In addition they provide continuous reporting and have direct access to corresponding power values. [KENT, 2011]

3.2 Power Measurement with Papillon

Stratergia, an Irish company founded in 2010, developed a software called Papillon to monitor power usage on servers in data centers. Stratergia runs research centers in Ireland, Austria and Sweden consisting of post-graduates and doctorates. In cooperation with Compare Test Lab located in Sweden, Stratergia developed an innovative non-intrusive solution, monitoring data centers in real-time. As outlined in chapter 3.1, various technologies analyzing the power consumption of data centers are available on the market. Papillon server power monitoring is a software based approach calculating a server‘s power consumption in a non linear manner. By installing Papillon on common operating systems, the tool is said to achieve a power saving up to 40%. Mathematical models of different server types, are used to calculate the real time power usage within 2% accuracy. Papillon uses a client-master system architecture to gather power information from each single server in a data-center.

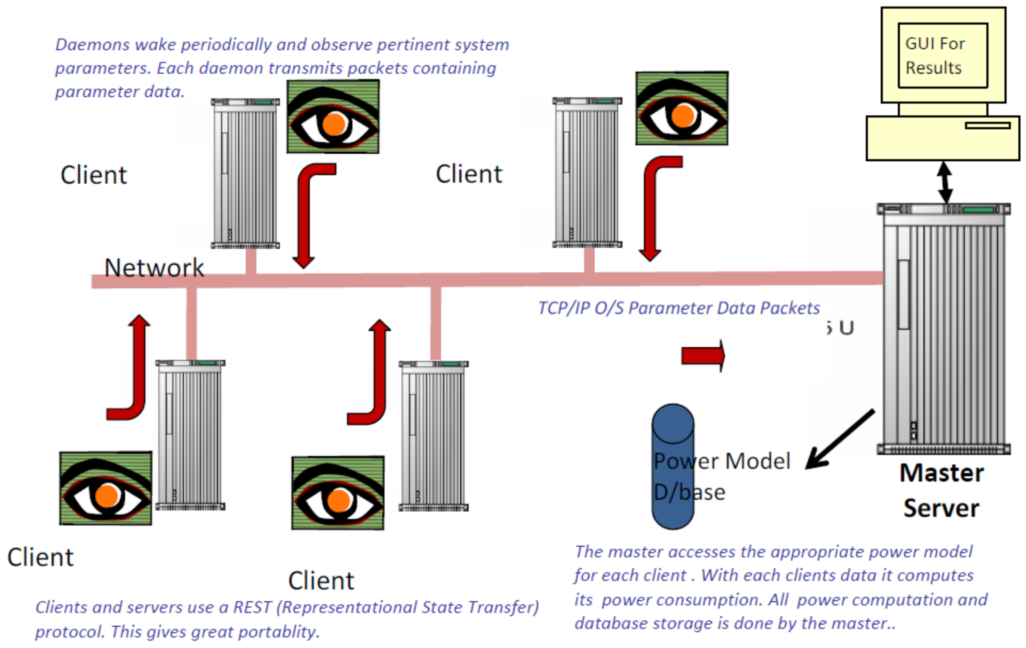

Daemons wake periodically and observe pertinent system parameters. Each daemon transmits packets containing parameter data.
GUI For Results
Client
Client
Network
TCP/IP O/S Parameter Data Packets
Power Model D/base
Master Server
Client
Client
Clients and servers use a REST (Representational State Transfer) protocol. This gives great portablity.
The master accesses the appropriate power model for each client . With each clients data it computes its power consumption. All power computation and database storage is done by the master..

Figure 3.2: Papillon system overview

The installed clients on the server side are designed to be as thin as possible to not influence the power performance of the host. Periodically performance data are sent collectively to a central master, storing the values in a database. Depending on a server´s power model, the mathematical model returns different power values which are displayed in the data management software system. Fig. 3.2 shows an overview on the Papillon software solution. Following sections give a detailed view on the architecture of Papillon, its API and what it offers for this thesis.

3.2.1 Papillon Architecture

The software architecture is kept simple and modular. Papillon is written in Java, to sustain platform compatibility. A MySQL server installation on the master side is powerful enough to handle large data. Data-centers often host thousands of servers, which are monitored by Papillon every minute. Therefore a robust database back end is required. Clients are being installed on a host server, collecting data every minute and returning the values to the central master. The whole logic is based on the master side, to ensure a thin client solution. The master takes the client data and calculates the power by correlating values with a non linear mathematical power model of the dedicated host. Section 3.2.1.1 outlines the procedure on modeling the power characteristics of servers. Providing the data via a RESTful interface offers 3rd parties the possibility of processing the valuable data. DCIM tools use the Papillon API to display the power usage of racks on certain floors in data-centers. By shifting server workloads based on Papillon´s monitoring results, significant power savings can be achieved. As shown in Fig. 3.3, the REST API operates as central interface providing data for additional management tools.

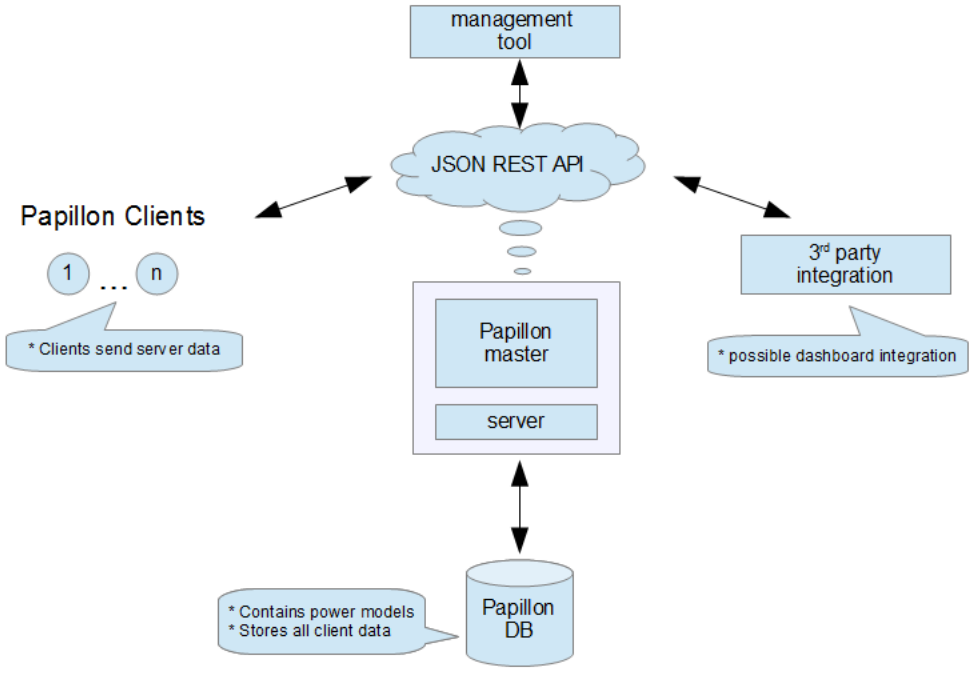


Figure 3.3: Papillon architecture

3.2.1.1 Power Modeling

Data-centers use different types of servers. Every server type has its individual power profile. Based on hardware components and operating systems, servers require different amounts of power. A customers application might run on a certain type of server much more efficient. The initial step is to profile the server. Stratergia Ltd. therefore runs a set of standardized benchmarks while metering the power supply. Benchmarks exercise the server over a wide range of operating conditions so that an accurate model representative to the power behavior of the serer is generated. The power model generation is seen to be the real innovation made by Stratergia Ltd. and leads to a very high accuracy compared other monitoring approaches. The initial power modeling procedure creates a power model file in XML-format. Depending on the amount of different types of servers in a data-center, several XML-power models are required. These models need to be uploaded to the database. The Papillon master uses those models in a non linear manner to calculate power values for the given client. A conceptual overview is shown in the Fig. 3.4.

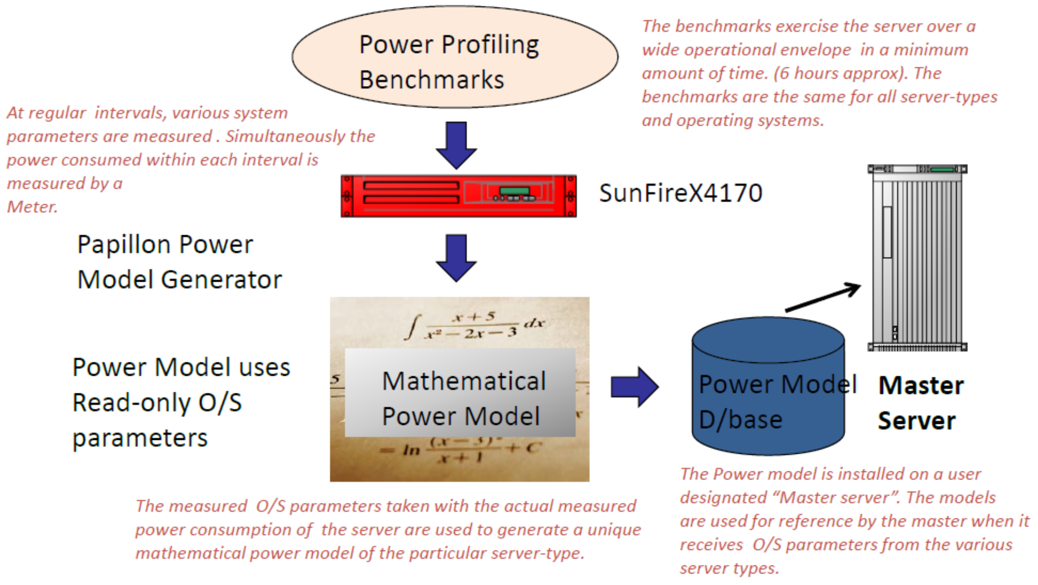

Power Profiling Benchmarks
The benchmarks exercise the server over a wide operational envelope in a minimum amount of time. (6 hours approx). The benchmarks are the same for all server-types and operating systems.
At regular intervals, various system parameters are measured . Simultaneously the power consumed within each interval is measured by a Meter.
SunFireX4170
Papillon Power Model Generator
Power Model uses Read-only O/S parameters
Mathematical Power Model
Power Model D/base
Master Server
The Power model is installed on a user designated "Master server". The models are used for reference by the master when it receives O/S parameters from the various server types.
The measured O/S parameters taken with the actual measured power consumption of the server are used to generate a unique mathematical power model of the particular server-type.

Figure 3.4: Server power profiling procedure

3.2.1.2 Papillon Client

Power measurement requires a thin client that measures resource utilization and using as less resources on the server as possible in order not to constitute to power consumption itself. Therefore the client runs without any reprocessing logic and only gathers server specific data once a minute. To gather system specific data from an operating system, an Apache licensed library called Sigar is used. Sigar (System Information Gatherer and Reporter) provides low level information on hardware and operating systems parameters. Its platform compatibility makes the library widespread in the java developer community. [Wikipedia, 2011]

Papillon uses the library to access CPU, Disk and Network Information on the server. By running the client as a service on Windows or as daemon on Linux, system parameters can be monitored in real time. In order to keep the network traffic to a minimum, only the three top most power consuming processes are sent to the master. The tree top most resource acquiring processes cover 80% of the power usage at a certain point of time.
The Papillon client uses the RESTful API to deliver data in JSON format. In case the Papillon master is out of order, a buffer assures messages to be stored. This guarantees continuous server monitoring over time which is crucial for 3rd party management solutions.

3.2.1.3 Papillon Master

The central processing master runs as java servlet on a Apache Tomcat HTTP-server. A non linear algorithm correlates received client data with stored power models. Using a nearest neighbor algorithm helps to calculate the server power value corresponding to the given parameters CPU usage, disk and network IO. The RESTful application programming interface provides methods to set and get data from the MySQL database. The API handles JSON as well as XML requests and is kept simple to ensure good response times. Whereas the Papillon Master includes the whole logic of the Papillon system, the client only sends data periodically. If any client does not deliver its supposed data in time, the master throws an alert. This mechanism is called life beat and is available via the API.

3.2.2 Papillon API

Papillon provides a RESTful API to develop third party applications. REST is seen as programming paradigm and does not provide a standardized architecture. By using certain URL constraints, data can be received either in JSON or XML format. REST offers a number of typical methods. POST, PUT, GET and DELETE are the main methods supported by the API. The URL consists of a basic part and various additional parameters depending on the data being requested. Due to the fact, that the Papillon master is hosted within a Tomcat environment, the API can be accessed with the following base URL.
Base URL: http://localhost:8080/PapillonServer/rest/

Additional Parameter are added depending on the type of data being requested. Following components are available via the Papillon API structure. **Datacenter**, **Floor**, **Rack**, **Server**
Each data-center consists of different floors, which have a certain amount of racks. Within each rack, servers are hosted. The following snippet shows a relative URI for a GET request retrieving a list of servers in a certain rack.

Relative URI	*GET* /datacenters/DataCentreId/floors/FloorId/racks/RackId/hosts
Description	Retrieves a rack´s servers

Table 3.1: Relative sample URI

A sample URI implemented in the application returning all servers being hosted in a data-center with id 1, floor 1 and rack 1, looks like the following.
Sample URL: http://localhost:8080/PapillonServer/rest/datacenters/1/floors/1/racks/1/hosts/

3.2.3 What Papillon offers for the thesis

With respect to various software solutions monitoring the power performance of servers, Papillon provides a large API exposing details on real time power usage. With its non intrusive measuring method, Papillon gathers accurate power information of physical an virtual servers. Its API offers the possibility to develop various types of applications using power specific data. Within this thesis power characteristics of physical servers are compared with virtual ones. Papillon therefore provides a virtualized measuring methodology. Both, physical and virtual Papillon clients report data to the central database. By correlating power characteristics of different server types, ROI (Return on investment) costs can be discussed in detail.

3.2.4 Competitor comparison

	Papillon	***Viridity***
Strengths	Agent-based approach; Platform independent; Supports real time monitoring; Buffers power values in offline mode; 2% Accuracy due to server power model; Power visibility at application level; Power monitoring of virtual servers;	Agent-less approach; Uses SNMP, WMI, SSH protocols
Weaknesses	Requires agents to be installed on each server;	Relies on network protocols; No real time monitoring; Based on schedules; No offline monitoring possible; No Power visibility at application level;

Table 3.2: Papillon vs. Viridity

	Papillon	***Joulex***
Strengths	Agent-based approach; Platform independent; Supports real time monitoring; Buffers power values in offline mode; 2% Accuracy due to server power model; Power visibility at application level; Power monitoring of virtual servers;	Agent-less approach; Uses WMI, WinRM, SSH, SNMP, IPMI, SMASH and more network services
Weaknesses	Requires agents to be installed on each server;	Relies on network protocols; No real time monitoring; Based on schedules; No offline monitoring possible; No Power visibility at application level;

Table 3.3: Papillon vs. Joulex

	Papillon	***NightWatchMan***

Strengths	Agent-based approach;
	Platform independent;
	Supports real time monitoring;
	Buffers power values in offline mode;
	2% Accuracy due to server power model;
	Power visibility at application level;
	Power monitoring of virtual servers;
		Agent-based approach;
		Platform independent;
		Scheduled system to set energy states of the server;

Weaknesses	Requires agents to be installed on each server;	Requires agents to be installed on each server; No real time monitoring; Based on schedules; No offline monitoring possible; No Power visibility at application level;

Table 3.4: Papillon vs. NightWatchman

As outlined in the section above only few companies are major in server power monitoring. Most DCIM solutions deal with a much higher level of abstraction, e.g. Racks or Floors, and take also facility management into account. Providing low level server power metering with additional process information is seen to be a truly invented approach by Stratergia Ltd. Not any single DCIM provider offers such low level monitoring possibilities what makes Papillon unique and turns a competitor analysis into a difficult mission. Papillon enhances existing DCM solutions to achieve more accurate power values. With a more accurate metering approach DCIM tools could improve data-center efficiency even more in the future.

3.3 Web Application stress testing

Cloud applications are seamlessly integrated into daily life and habits. Therefore Client-server based architecture enables humanity to interact with the cloud and it´s services. Using a cloud-based service in every day´s life requires the application to be fast and reliable. Short server response times even on high load are crucial for user interaction. Due to the world wide availability of cloud-infrastructure, applications face tremendous amounts of users. In order to investigate the performance of cloud based applications, stress testing tools and appropriate test scenarios are required. Various leading tools offer different approaches in this field. The purpose of stress testing client-server based applications is to determine possible bottlenecks leading to critical states.
Stressing applications with real world scenarios is seen to be most valuable for a client-server architecture test. A research lab located in Belgium, called Sizing Servers Lab, put their focus on creating test benches for cloud-based applications using real application workloads. With years of experience and their own stress test techniques, they have created a powerful tool called vApus. vApus is termed to be a stress test solution for any client-server based application. The main goal of vApus is to analyze an applications performance characteristic during different workloads. Recognizing the power of vApus led to the creation of modern benchmarks called vApus Marks analyzing new server technologies. Benchmarks are standardized tests which are executed on different server architectures investigating their performance characteristics. When it comes to virtualization on servers, performance metrics change and therefore benchmarks have to be modified. VMware claims to offer industries first virtualization platform benchmark. Sizing Servers Lab is not seen to be a competitor of the so called VmMark by VMware, but provide their own solution for virtualized platforms. In the following sections an outline is given how vApus works, what it offers for Stratergia and which results were achieved by applying stress tests to Papillon v2.0. Section 3.3.3 describes vApus Mark and how it uses a distributed version of vApus stress testing.

3.3.1 vApus

3.3.1.1 What vApus offers for Stratergia

Stratergia provides a rich API to track the power performance of servers running in a data center. That API will be used to determine the power of single applications running in the cloud. Once an application can be tracked and its power performance is visible, further business cases arise. Using vApus to stress test an application and simultaneously measuring the power usage with Papillon is seen as an unique software analysis for customers. Papillon uses a client-server architecture and therefore is a suitable sample application for stress testing. Stress testing Papillon is a good opportunity for Stratergia to get a performance feedback on their cloud-based application as well as some insights on the architecture of vApus. Using vApus and Papillon collectively in the future requires a basic knowledge in how vApus works and how it can be used to stress a customers application.

3.3.1.2 Architecture

vApus supports all kind of client-server based application stress testing. It is flexible enough for use case dependent modification in order to match the stress testing requirements. Most cloud-based applications use HTTP-requests which require certain connection proxies to be configured. By extending vApus with sniffers and monitors a wide range of client-server applications can be stress tested.
The framework is written in C#, using .Net 4.0 and kept modular to meet the requirements of a flexible stress testing tool. The 64 Bit based vApus is capable to handle up to 10000 active threads on a single client. Each thread simulates a concurrent user action, sending requests to the server-side. Depending on the stress test solution, SQL-statements, HTTP-requests or other types of messages are fired in a scheduled way to the server. A test client, equipped with an Intel Core I7-750 (2.66 GHz), is supposed to require only 20 % CPU load while running 15000 concurrent user threads. [DeGelas, 2010a]

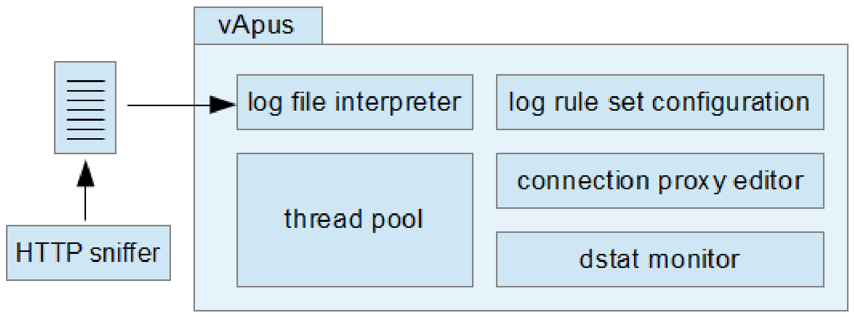


Figure 3.5: vApus modules overview

Applied stress tests, as described in Section 3.3.2, show a maximum of 5000 concurrent users. Theoretically the amount of threads is just a matter of computing power. A powerful server could handle a vApus client with much more than 15000 threads, but corresponding context switches during thread processing would increase dramatically. Calculated throughput and response times for fired requests might be wrong due to the bottle neck on the client-side. To stress test web-applications with more than 5000 concurrent simulated users, Sizing Servers Lab provides a distributed vApus solution. The distributed solution uses a set of vApus slaves being hosted on different test clients. Each slave handles thousands of concurrent users sending requests to the dedicated cloud-application client. Throughput and response times are calculated and returned to a central vApus Master. The master uses gathered metrics collectively to give a detailed report on the performance characteristic of investigated cloud-applications. In Section 3.3.3 the usage of the distributed vApus solution within vApus Mark II is described in detail.

3.3.1.3 How vApus works

Sizing Servers Lab offers an optional HTTP-sniffer to monitor client-server applications. Corresponding log entries are stored in a log file, which can be interpreted by vApus itself. Depending on the application type, a pre-configured log rule extracts actions and items. Database applications require a different log rule set than typical web-applications. A login procedure is seen to be an action and consists of several items. Items are certain queries or URLs which form a login sequence after arranging them in the right order. Stress testing web-applications often require a user to be logged in to proceed with further navigation. vApus therefore allows action arrangement which ensures the virtual test user being logged in before proceeding with further actions.
Creating a stress test for a new application additionally requires new connection proxies to be created. As mentioned each application uses a unique connection procedure to connect a user. vApus offers a built in C# editor to create connection proxy files which are used by the stress test solution. Once a connection to the remote server is established via a valid proxy connection, stress tests can be applied.
As mentioned in section 3.3.1.2, a thread pool simulates user concurrency. Each thread replays the customized log file, triggering a possible login procedure and certain user actions on the server-side. Requests are processed on the server and returned to vApus which calculates the response time as well as throughput. Stressing the server by starting several concurrent user threads, creates the workload. A high concurrency of users causes the CPU usage on the server to rise. An external tool, called dstat, is used with vApus collectively to monitor the server performance characteristics. Analyzing the CPU and memory usage as well as certain other metrics gives detailed information on the application performance during the stress test. Collected results are displayed in the vApus solution and outline the bottle necks of the investigated cloud application. A basic vApus work-flow is given in Fig. 3.6 [DeGelas, 2010b]

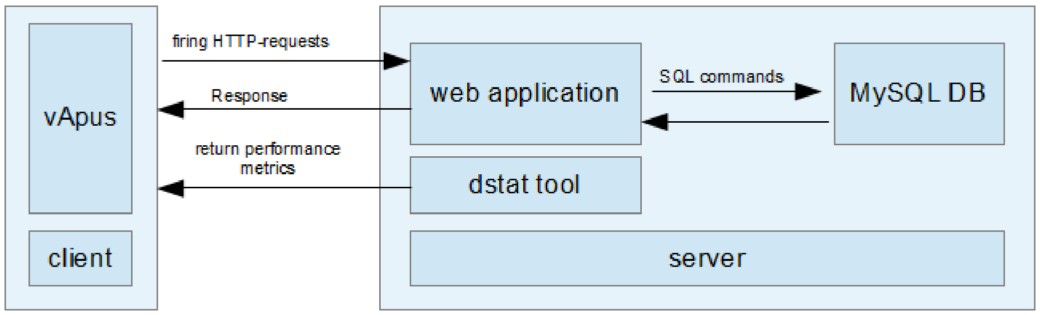


Figure 3.6: vApus client stressing a web-application hosted in the cloud

3.3.1.4 Test bench creation

As outlined in section xx each application being analyzed requires a unique test bench composed of certain log rule sets and connection proxies. The initial part of creating an appropriate test environment is application investigation with traffic sniffers. Sizing Servers Lab provides a sniffer tool which is compatible to vApus. The sniffer is used to monitor user actions which are stored in a corresponding log file. The following sample below shows vApus compatible log entries recorded from a Papillon client sending power information to the Papillon server.

```
/PapillonServer/rest/agents/{L.1}/activity;POST;;<?xml version="1.0"
    encoding="UTF-8" standalone="yes"?><activity><id></id><hostId>{L
    .1}</hostId><power></power><stat1>55.12000000000012</stat1><stat2
    >350084.0</stat2><stat3>47655.0</stat3><powerMode>AC_DEFAULT</
    powerMode><timeStamp>UnixTime</timeStamp><allApps>483.0</allApps
    ><apps><app><name>tiotest</name><cpu>395</cpu></app><app><name>
    java</name><cpu>29</cpu></app><app><name>java</name><cpu>17</cpu
    ></app></apps></activity>;;Java/1.7.0_147-icedtea;application/xml
    ;192.168.35.37;192.168.35.37
```

Listing 3.1: Logfile sniffed from Papillon client

Once a log file is recorded, it can be imported in a new vApus stress test solution. vApus interprets the log file by using configured log rule sets. A cloud-based web-application is most likely composed of HTTP-requests as seen in the log file above. Corresponding HTTP-log rule sets are required to successfully import the file. The snippet from a standard HTTP-log rule file is shown in Listing 3.2.

Database application investigations require a different log rule set than web-applications do. By importing the appropriate rule set file, sniffed log files of database applications are parsed properly. As mentioned, each log entry equals a user action. User actions can be arranged in a random way or sorted manually. Depending on the sequence vApus fires those actions on demand. Creating a valid connection proxy is seen to be the most challenging part to establish a new test bench. Background knowledge of the application

```
            <Rule>
              <Items />
              <RegExp>^GET$</RegExp>
              <IgnoreCase>False</IgnoreCase>
              <ValueType>stringType</ValueType>
              <UsePasswordChar>False</UsePasswordChar>
              <Description>
              </Description>
              <Label>GET</Label>
              <ShowInGui>True</ShowInGui>
              <IsDefaultItem>False</IsDefaultItem>
              <IsEmpty>False</IsEmpty>
              $</RegExp>
```

Listing 3.2: Snippet from HTTP log rule set

being analyzed is crucial to create a valid connection. A built in developer environment offers the possibility to develop within C#.

Each application uses a unique connection sequence. Accessing HTTP-REST APIs, sending login information with certain timeout constraints or connecting to certain databases are the reason for using different connection proxies. After establishing the connection proxy configuration various parameters and settings need to be set in the stress test solution. In order to stress the application, concurrent users can be increased. During the stress test start up, custom user threads are created and the corresponding log actions are fired to the application master. vApus offers a large variety of optional features to be configured which are not discussed here in detail. The referenced documentation [Vandroemme, 2010] gives detailed information on how to use the vApus GUI to start stress testing with customized test benches.

3.3.2 Stress testing Papillon v2.0

Papillon V2.0 has been tested within the Amazon Elastic Compute Cloud (EC2) in order to measure three performance parameters. The CPU load, the idle percentage and the response time are seen to be valuable enough for outlining the overall performance of the stress tested Papillon server. The purpose of the cloud based stress test was to figure out the critical tipping point of the maximum number of possible clients that each master can handle. Running stress tests within the Amazon EC2 is seen to be quite complex and time consuming. With vApus, stress testing becomes much more comfortable. Stress testing results are gathered and give a detailed view on the application performance. Due to the limitation of a single vApus client, tests were processed up to 5000 clients. To verify the critical tipping point of concurrent users on the Papillon master, stress testing with a distributed vApus version is affordable.

```
ConnectionProxy() {

} //ConnectionProxy

#region Functions
public void TestConnection(out string error) {
error = null;
try
{
    _httpWebRequest= (HttpWebRequest)System.Net.WebRequest.Create(new
         Uri(_connectionProxySyntaxItem0 + "/PapillonServer/rest/
        datacenters"));
    _httpWebRequest.UserAgent = "vApus v2 — Test connection function
        ";
    // 2 minutes.
    _httpWebRequest.Timeout = 120000;
    _httpWebRequest.ReadWriteTimeout = _httpWebRequest.Timeout;
    _httpWebRequest.AllowAutoRedirect = false;
    _httpWebRequest.ServicePoint.Expect100Continue = false;

    _httpWebRequest.ServicePoint.ConnectionLimit = 1;

    _httpWebRequest.Method = "GET";
    _httpWebRequest.ContentLength = 0;
    _httpWebRequest.ContentType = "text/plain";
```

Listing 3.3: Snippet from Web HTTP connection proxy

3.3.2.1 Related work

In 2012 the Swedish team of Stratergia conducted a Papillon stress test within the EC2. The Amazon elastic compute cloud was used to run thread pool scripts on virtual machines, firing HTTP-requests to a dedicated Papillon master. Each pool created 60 to 150 scheduled threads. Due to scheduling, threads are pseudo parallel processed and don´t match a real world scenario of 60 concurrent users. In order to achieve a better scenario, each user would have to acquire it´s own virtual machine. Stress testing an application with 15000 virtual machines is seen to be unfeasible. As shown in Fig. 3.7, hosting 150 threads on a single machine affords 100 vms to stress the master with 15000 clients.

vApus is able to create much more than 150 threads in a single instance, but as mentioned above it is less a real-world scenario than the former EC2 cloud test. To get the same granularity vApus has to be used in a distributed way. The overall advantage of vApus is given by a centralized vApus master, collecting performance metrics from all clients. Detailed information on throughput and response time are crucial to investigate application performance characteristics.The following section 3.3.2.2 provides details on applied stress tests using a single vApus instance with varying concurrent users.

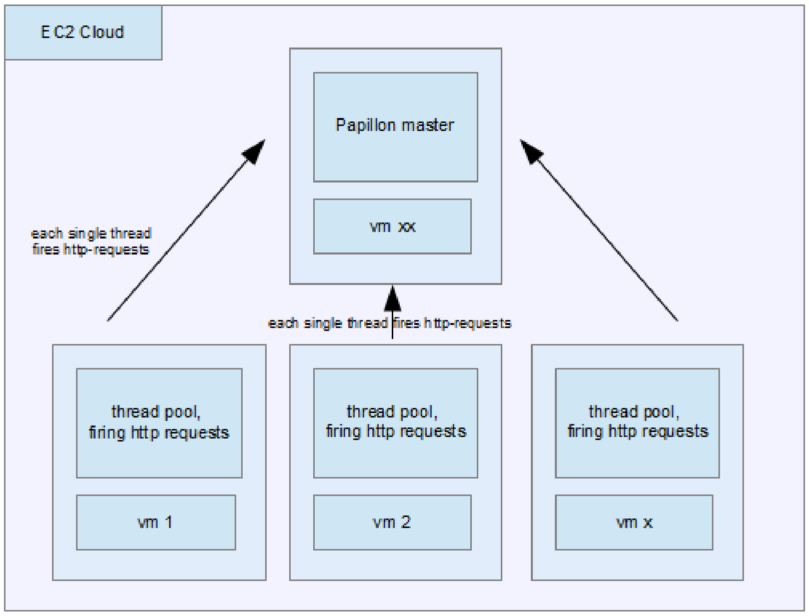

EC2 Cloud
Papillon master
vm xx
each single thread fires http-requests
each single thread fires http-requests
thread pool, firing http requests
vm 1
thread pool, firing http requests
vm 2
thread pool, firing http requests
vm x

Figure 3.7: Stress testing Papillon V2.0 within EC2 cloud

3.3.2.2 Test environment

Using a non distributed vApus version is seen to be valuable for gaining basic knowledge in stress testing. With the possibility of creating thousands of pseudo concurrent users, certain loads can be put on applications. The architecture shown in Fig. 3.8 is used for all applied stress tests on Papillon.

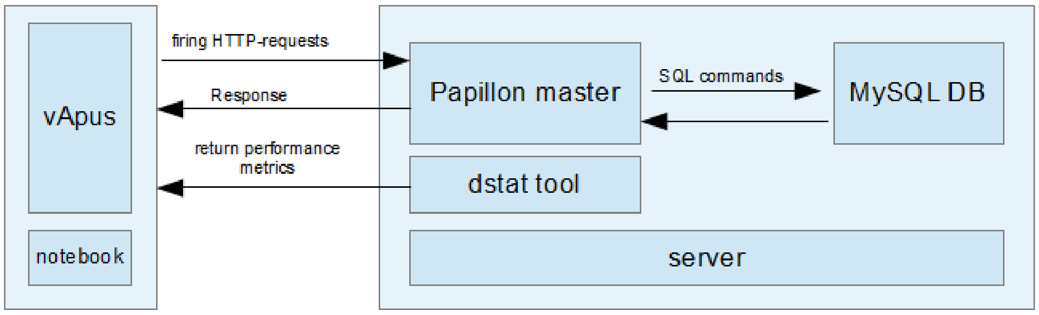


Figure 3.8: Stress testing Papillon V2.0 with single vApus Client

The vApus client is hosted on a notebook running Windows 7 with 64 Bit. In the same network domain, a dedicated test-server runs the Papillon master with it´s MySQL database. During the test start up, vApus creates a customized amount of concurrent users which fire HTTP-requests directly to the Papillon API. An additionally installed dstat tool on the server-side, monitors the hardware performance during the test and returns the results to vApus. The following hardware configurations were used during the tests:

- **Papillon Server**
  Operating Sytem: Ubuntu 12.04 LTS 64-Bit
  Processor: Intel Xeon(R) CPU E5530 @ 2.40 GHz x 8
  Memory: 15.7 GiB
  Disk: 481.8 GiB

- **vApus host server(notebook)**
  Operating Sytem: Windows 7 Professional 64-Bit
  Processor: Intel Core(TM) i5 CPU M430 @2.27 GHz
  Memory: 4.00 GiB

3.3.2.3 Performance metrics

The overall goal with stress testing an application is to get valuable feedback of its performance during workload variation. A set of metrics provided through vApus allow retrospective assumptions on throughput and response time. The following parameter are reported during a vApus stress test.

- **Throughput/s**
  The throughput is highly valuable for analyzing the power performance of an application. Papillon reports every minute power statistics to the master. The corresponding vApus testbench does the same and therefore multiplying the throughput by 60 gives the values per minute. With respect to the reporting interval every minute, the throughput outlines whether all requests can be processed by the server or not. In case too many of them can not be processed and queue up, the server faces a bottleneck.

- **Average Response Time**
  Low response times indicate a good performance on the server-side. A rising amount of concurrent users automatically will increase the response time. High response times generate a low throughput and outline a problem on the server side.

- **95% percentile (Max. Response Time)**
  Due to certain issues on the server side high response times might occur. In order to flatten peaks a 95% percentile of the maximum response times is taken. The five largest response times out of 100 times sending a certain log entry are trimmed. For 100 log entries, being sent 100 times each, the largest 5 % are taken out.[Vandroemme]

vApus uses a performance monitor, based on the versatile resource statistics tool (DSTAT) running on the Papillon master server. Following metrics are seen to be performance indicators during a application stress test.

- **Total memory usage in MB**
  The total memory usage is an important value to be analyzed during a stress test. Depending on parallel processed appliactions on the server, memory usage might mislead. Therefore the relative memory usage measured in several different stress tests, with different workloads is valuable.

- **Total CPU usage in % (usr) (sys)**
  Applications spend processing time in user-space and kernel-space. Some of them spend to much time just switching between both spaces. Context switches create CPU wait time , which also uses resources. According to an applications functionality and workload the type of CPU usage is important to be analyzed.

- **Total CPU waited in %**
  Outlines the wait time in percentage. A low wait value gives a good performance due to fast processing. High wait time occurs due to locked process instructions and unnecessary context switches.

- **Total network data received and sendt in bytes / s**
  Investigation on network traffic is an important part during stress test analysis. High network traffic on low workloads might indicate architecture issues in the monitored application.

Listed performance metrics are used in a collective manner to investigate an application´s performance on a server. Observing only single parameters might lead to wrong assumptions. Expertise in the field of operating systems and stress testing is required to interpret given results and judge an application´s performance characteristic.

3.3.2.4 Stress test results

An initial stress test was applied simulating only a single Papillon client to build build a base line for ongoing investigations. To avoid inaccuracy, vApus provides a custom precision setting. A Papillon agent sends data every minute to the master. Taking 20 log entries, takes 20 minutes for a whole precision run. With a customized precision of three the whole stress test takes approximately 1 hour. Results where verified by Sizing Servers Lab.

- **Single Client Test**

  The initial test with a single client was expected to return excellent results. A throughput of 0.0165 * 60 = 0.99 requests per minute as shown in table 3.5 is seen to be a good value. 1 request per minute would be theoretically possible, 0.99 is perfect. An approximately response time between 63 and 65 milliseconds is fast enough. The test completed without any errors and so is fully valuable.

Clients	Precision	Throughput / s	Avg. Response Time [ms]	95 % percentile	Errors
1	1	0.0165	95.9267	164.509	0
1	2	0.01652	65.6971	74.889	0
1	3	0.01653	63.6815	68.712	0

Table 3.5: vApus precision values for a single client

Table 3.6 outlines performance metrics gathered during the stress test by dstat. The used memory value is fairly high for just one Papillon client being processed. With respect to the low CPU usage and CPU wait, there might have been different applications running on the server as well causing this memory peak. Ongoing stress-tests confirmed the suspicion that Papillon is not the reason for the high memory usage. Additional application running in the background caused the peaks.

Clients	Precision	Mem. / MB	Cpu [%] (usr)	Cpu [%] (sys)	Cpu wait [%]	Received B / s
1	1	2621.326	0.3448861	0.3222453	0.1671213	609.4498
1	2	2636.438	0.3568637	0.3307196	0.1626899	645.2968

Table 3.6: dstat monitor values for a single client

- **1000 Clients Test**

Stress tests executed with several different workloads from one up to 1000 concurrent users were processed without any errors. Corresponding performance characteristics scaled up as expected and are not discussed in detail. In contrast, 1000 concurrent users caused vApus to retrieve errors on requests sent to the Papillon server. Following error message was thrown in all cases: *System.Net.WebException: Unable to connect to the remote server* This Web-Exception occurred at the initial start up phase of the stress test and might be caused by creating 1000 activities in the database simultaneously.

Clients	Precision	Throughput / s	Avg. Response Time [ms]	95 % percentile	Errors
1000	1	16.364	609.2828	748.152	0
1000	2	16.434	352.8098	561.014	10
1000	3	16.426	377.6646	569.348	1

Table 3.7: vApus precision values for 1000 clients

Further investigations need to be done to verify the suspected scenario. The discussed error did not influence the test results and therefore following data are seen to be fully valuable. The throughput with 16.4 * 60 = 984 requests is acceptable and the average response time excellent.

The performance monitor in table 3.8 reports an excellent server performance with respect to CPU usage (usr) and CPU usage (sys). The only critical value is seen to be the CPU wait with a value of about 2%. A wait time of 2% might occur due to the 16 non processed requests being queued. A queue carries a potential risk when it comes to a large amount of concurrent users on the Papillon master.

Clients	Precision	Mem. / MB	Cpu [%] (usr)	Cpu [%] (sys)	Cpu wait [%]	Received B / s
1000	1	3105.691	2.097133	0.5914361	2.64119	12871.75
1000	2	2634.807	1.823885	0.5719722	2.078407	12816.89
1000	3	2563.702	1.836364	0.5789285	2.228823	12741.68

Table 3.8: dstat monitor values for 1000 concurrent client

- **5000 Clients Test**

During the stress test with 5000 concurrent users the errors on processed requests rose significant. Database issues during initial stress test start up are most likely the reason for those problems. The throughput with 80.4 * 60 = 4824 requests per minute causes 176 remaining ones to be queued.

Clients	Precision	Throughput / s	Avg. Response Time / ms	95 % percentile	Errors
5000	1	80.602	1541.0871	8283.629	464
5000	2	80.111	1921.9221	16022.283	329
5000	3	80.495	1624.475	8626.244	473

Table 3.9: vApus precision values for 5000 clients

Confirming the suspicion of the stress-test with 1000 clients, the CPU wait time keeps on rising. While CPU usage is pretty low, the average response time increases. Queued requests are suspected to cause high response times and are seen to be a bottleneck for even much more concurrent users.

Clients	Precision	Mem. / MB	Cpu [%] (usr)	Cpu [%] (sys)	Cpu wait [%]	Received B / s
5000	1	2750.767	8.011715	1.790969	7.197266	62965.82
5000	2	2633.302	7.546957	1.782801	7.60968	62534.4
5000	3	2651.081	7.533314	1.778059	7.210904	62930.11

Table 3.10: dstat monitor values for 5000 concurrent client

3.3.2.5 Summary

Papillon Version V2.0 performs much better than V1.0. The MySQL Database is able to handle a large amount of data and processes requests in time. Stress tests up to 1000 clients reported excellent performance metrics and assume Papillon to scale up. Using a dual quad-core as Papillon master server is seen to be powerful enough to handle much more than 5000 clients. 8 % CPU user time, processing 5000 client requests is seen to be a good value. Only CPU wait might be considered as an issue in the future. Wait time occurs due to locked process instructions. One plausible reason for high wait times might be the MySQL server simultaneously trying to access locked tables. Locked tables lead to connection problems as well which probably causes 8.44 % errors. Further investigations in the MySQL architecture are essential for Stratergia to eliminate potential bottlenecks.

3.3.3 vApus Mark

As mentioned in section 3.3 , vApus was recognized to be a powerful tool, which led to the creation of server benchmarks as well. With expertise in server virtualization and stress testing, Sizing Servers Lab created the so called Vapus Mark, which is comparable to VMmark developed by VMware. vApus Mark uses five virtual machines with different real world applications being hosted. For example a hosted OLAP database in combination with a facility management web application is seen to be a real world application used within the benchmark. To put workload on these applications, a vApus stress testing slave is installed on the virtual machine as well. A vApus master uses gathered performance metrics from all vms collectively to calculate corresponding benchmark results. Fig. 3.9 shows the principle of vApus Mark. vApus Mark uses the distributed vApus stress test solution to gather performance information from different virtual machines. Using real world applications within the benchmark is unique in the field of virtualized server benchmarking and seen to be highly valuable for the future.[DeGelas, 2010a]

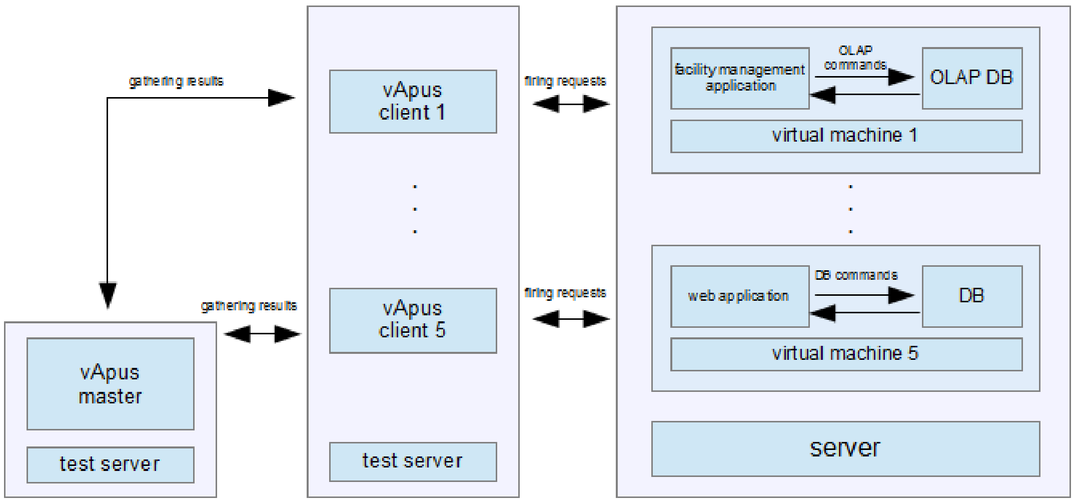


Figure 3.9: vApus Mark II overview

Chapter 4

Methodology

4.1 Application identification

Cloud computing offers the possibility to provide computing resources in a completely new manner. Cloud services provide external storage, application hosting and much more. Depending on the Data Center technologies, physical or virtual machines are in use. In addition resource sharing and virtual machine migration increases rapidly and adds additional challenges for cloud providers. Beside the challenges concerning infrastructure, also the question of an adequate chargeback model arises.
With physical servers, the hardware, corresponding electricity power usage as well as additional software costs are directly associated with the customer. As a result an all inclusive price model can be calculated. Virtual IT environments make it even harder to trace back a customers resource usage. Application workloads and the resulting costs are related in a complex manner. Certain high workload peaks as well as idle modes might have an influence on the average resource pool which makes it even more difficult to trace back the charging. Different approaches of how to create more accurate and transparent charging models are outlined in the following paper.[Gmach et al., 2011]

The three main categories of resource usage are:

- Direct resource consumption by workload
- Burstiness for a workload and for a server
- Unallocated resources for a server

Within the mentioned paper those three categories are used in weighting formulas to get a sum of the total workload costs. The results of applying smart weighting algorithms show, that those approaches give a better granularity and provide a more accurate chargeback model for shared cloud resources.[Gmach et al., 2011]

Instead of using approximation algorithms, a much more accurate approach could revolutionize the chargeback models. Due to dynamic application shifting not every server is dedicated to a certain customer. It is not sufficient enough to just add up the workloads of a server in order to calculate a customers usage. If the application itself could be tracked in a acceptable manner totally new charging models would arise. Within this theses this approach is being discussed as well as all its difficulties coming with it.

4.1.1 Identification Approach

Papillon is unique in its capability to monitor virtual machine power. The real patented innovation is seen in not only gathering accurate power values of each server but to also get detailed information on which processes most computing capacity is spent. This approach offers a range of new possibilities to track an applications workload profile. Once this is visible, the exact electricity usage can be calculated for the overall charging model. As described in section 3.2, by default Papillon provides visibility to the top three power consuming processes being executed on the server. The difficulty for identifying an application just by three given processes is a major issue. Papillon has the possibility to increase the amount of processes being exposed to provide a full process tree. By having full visibility, an application, theoretically, could be identified by assigning corresponding processes. Having the full spectrum of sub processes, the workload and its resulting power consumption are exposed. Once an application is identified by its process tree, it is easy to track the migration within a virtualized environment as well. This approach is seen as completely new due to the fact that no agent software was able to deliver such detailed information of servers before.

The first pie chart in Fig. 4.1 shows the top three processes using 80% of computing resources on the server. By analyzing 80% of the usage and breaking it down in just three processes is sufficient enough for monitoring the power usage in general. For identifying an application in detail a higher granularity is required. The second pie chart in Fig. 4.1 outlines a better granularity which includes much more processes. Papillon by default does not support such high granularity at the moment but can be amended to give unlimited resolution. Another challenge is seen in adding relations between single processes listed in the pie chart below. Gathering parent processes with corresponding child processes would be available in the future as well. Despite a full process tree, application identification is seen to be still difficult.

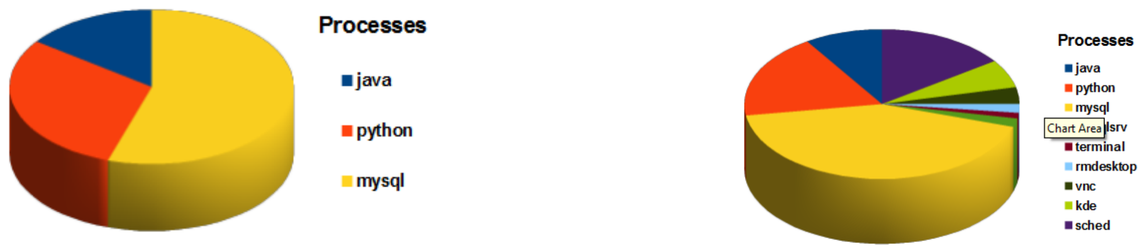


Figure 4.1: Process granularity delivered by Papillon

4.1.2 Identification Difficulties

As mentioned in section 4.1.1, theoretically all processes could be exposed by Papillon. Even if single process trees are available, the application itself can´t be identified in a trivial manner.
Depending on the complexity, an application might be totally identified by just a single process tree, but also could share different other trees as well. Especially web applications are difficult to identify due to their spread in resource allocation. Each web application is hosted on a web server, which has its own process tree. In addition data are often stored in a database. MySQL therefore runs its own services with its own process trees. Most of the web-based applications require java as underlying service which adds another process tree to the list. Detecting relations between those given process trees turned out to be a non trivial challenge.

Fig. 4.2 gives an example of a fully identified web application consisting of three different process trees. Although a fourth tree is available only three of them are used by the application itself. Mapping those trees to the corresponding application requires additional relation information. Investigating relations between single process trees was seen as too complex to handle within the scope of this thesis. The following section provides an outlook on a possible approach for the future.

4.1.3 Future work

Identifying an application by observing a large distributed cloud environment is assumed to be too complex due to large process trees. In order to reduce the complexity only a single physical server should be taken into account. Following two use cases outline a possible approach of application identification.

- Run the server for a given time-span in idle mode to detect the default process tree constellation

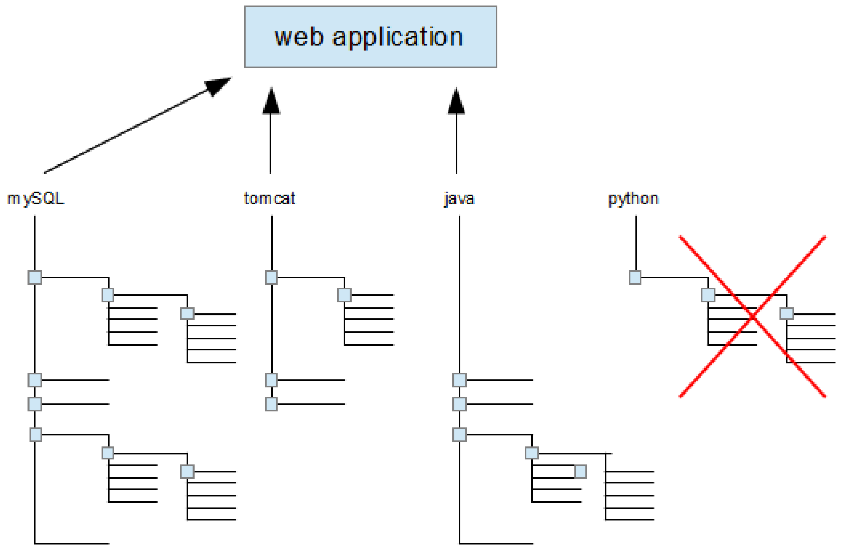


Figure 4.2: Process trees related to web-application

- Run the given web application for a given time-span with different workloads to investigate additional launched process trees

Use case one is seen to be a base-lining method to characterize a servers process tree constellation. Once the server is base-lined, use case two can be executed. By running the application in an isolated manner, the exact amount of additional required processes can be determined. By applying both use cases, the relations between the single process trees are expected to be fully exposed. Once this initial process has been applied, the application can be hosted in a complex system. By analyzing the process trees, the application can be identified by its unique process tree footprint.

4.1.4 Future Issues

Web applications often share resources like the tomcat server or databases. By hosting several similar applications within a cloud environment process trees might be similar. To handle such scenarios, a smart weighting algorithm could be developed. Depending on an applications workload, shared resources might have to be weighted different. In Fig. 4.4 an application A shares some of its resources with application B. Based on workload estimation, an applications resource usage can be weighted. In case the first application A allocates much more of the MySQL process, the weighting algorithm then assigns most of the power consumption to A.

Despite the identification via process trees, applications still can have the exact same tree constellation. In order to solve the mentioned issue, the algorithm has to approxi-

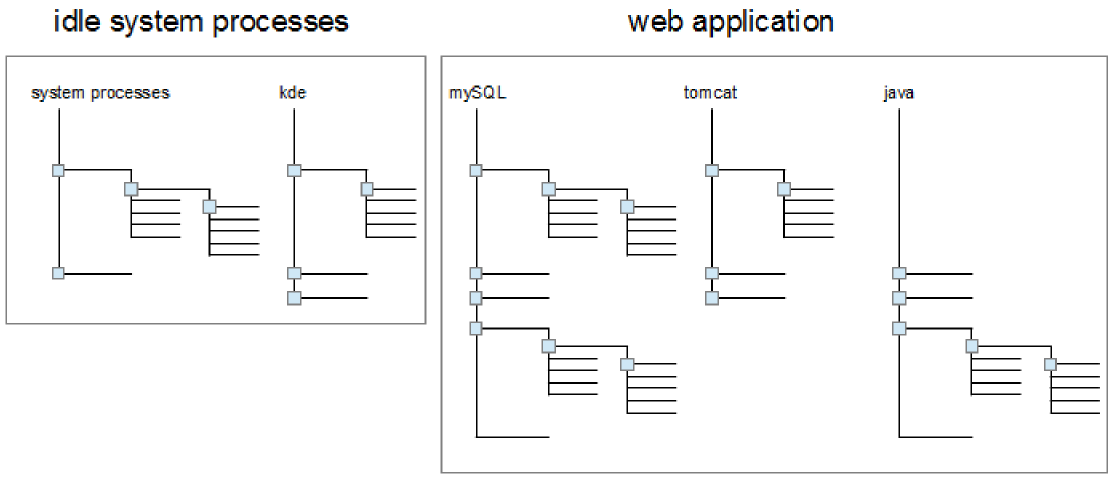


Figure 4.3: Process trees after applying both use cases

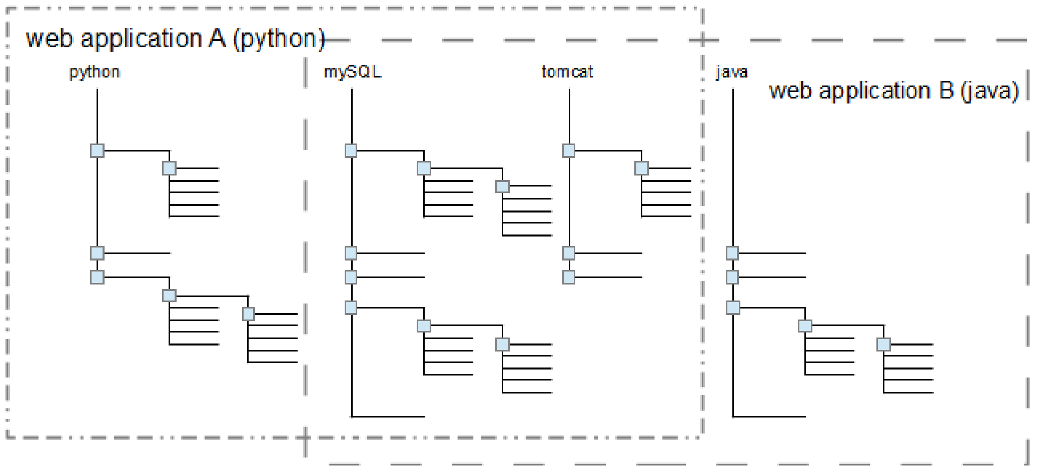


Figure 4.4: Application A uses two of the same process trees as application B

mate an average power consumption for both applications. Being identified by exact the same tree constellation results in sharing operational costs as well. By taking some additional workload information into account approximated resource usage could identify the applications in detail. These are just some ideas of what could be interesting for the future.

4.1.5 Summary

Different hosting technologies require different monitoring approaches. Migrating applications itself and tracking their power consumption is seen to be difficult as outlined above. Cloud providers therefore host applications in virtual machine environments. Each application is hosted in a single virtual machine, which is much easier to migrate, but still is difficult to monitor. Papillon provides the possibility to gather low level operating information which can be used to retrieve the power profile. Once a valid power profile is created, virtual power metering can be applied. Within the following chapters, virtual application hosting will be compared to physical hosting regarding power effectiveness. Therefore a reference application will be used in order to baseline both approaches.

4.2 Server power analysis on physical and virtual machines

As mentioned in Section 2.2, applications are hosted within virtual machines to be portable in an easy manner. Underlying hypervisor software handles dynamic migration processes. [Loeffler, 2009] Not every application might perform better if it is outsourced into a portable virtual machine. Depending on the application type, a physical server setup still might lead to a better performance.
Different types of applications are different in their power efficiency. Some of them are more power hungry than the others. Additionally even the server type has an influence on their power consumption. There is no general guideline which applications run most efficient on which server types. Database applications might use less power on Oracle servers than on HP-plates. By developing a tool, displaying both power characteristics, an application can be categorized.
Initial costs of a physical server setup might be much less than those of a powerful machine, hosting multiple virtual servers. In contrast, certain applications might have much higher operational costs on a physical machine than in a virtual environment. Especially cloud providers might save costs and energy by choosing the right hosting technology based on the type of applications. The following section outlines the methodology, developed within this thesis to characterize isolated applications. In order to validate an applications power performance in a certain working environment, a possible approach will be discussed. By developing a server power performance analysis tool, based on the Papillon API, power profiles of physical and virtual setups can be compared. The advantage of getting an idea which application type runs more efficiently in a virtualized environment than in a physical one is seen to be an advancement in application hosting and charge-back models.

4.2.1 Methodology

The monitoring approach is based on the virtual power metering provided by Papillon. Each server type uses a different amount of electricity depending on the application being hosted. Within the scope of this thesis, web-based applications being hosted in a cloud environment are taken into account. Therefore a reference sample application provided by Sizing Servers is taken for detailed evaluations. Virtual as well as physical servers run the same application which is stressed by the vApus test-bench. Stressing the installed sample applications creates a certain amount of workload which is close to realistic loads.

During these stress tests, the server power is measured and displayed in the developed *Server Power Analysis Tool*.
The average power consumption of every single machine outlines the most efficient server setup for this type of application. The following Fig. 4.5 shows the overall methodology of how to characterize application types.

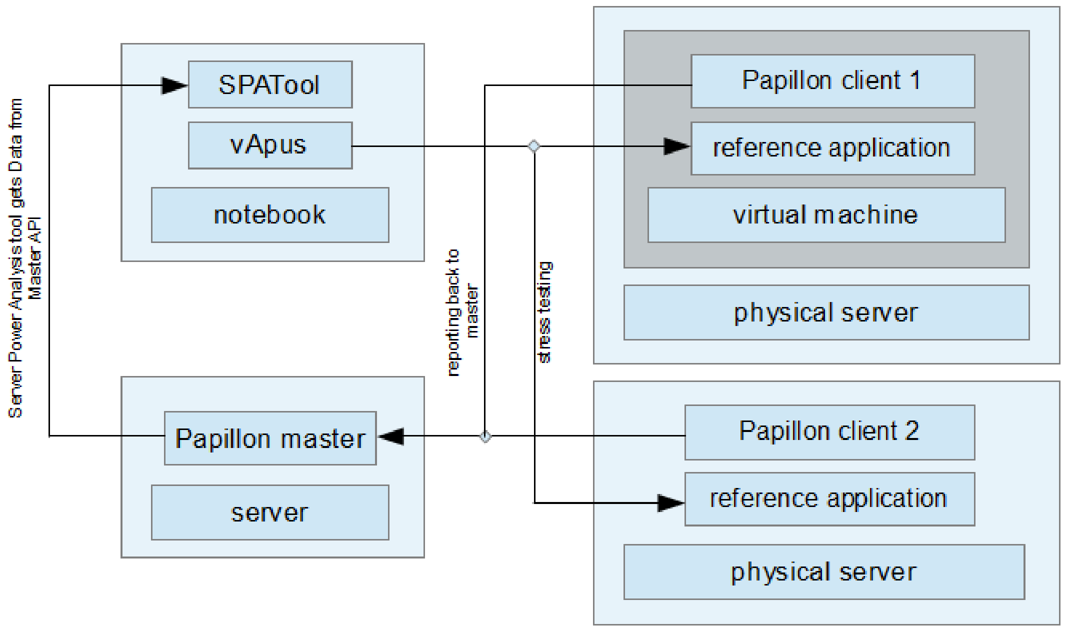


Figure 4.5: Methodology to monitor physical and virtual servers running a reference application

In the overview two different server types are used. Papillon client 1 monitors a virtual machine which runs the reference application and client 2 gathers power information from a physical machine. Both reference applications are being stressed by the vApus Testbench which puts a realistic workload on the server environments. The developed SPA (Server Power Analysis) Tool collects corresponding electricity power values from the Papillon master REST API and displays them in a browser based dashboard. By observing the reference application under a certain workload over a given time, the average power consumption can be calculated. This gives an idea of the cost effectiveness for specific server setups. Based on those observations business models can be better planned and calculated.
Offering custom application hosting solutions which run more efficiently, reduces costs for the customer itself. The next section contains the documentation of the SPA tool implementation which is seen to be a prototype for above mentioned monitoring duties.

4.3 Server Power Analysis tool

4.3.1 Requirements analysis

Papillon has an extensive API to access a large database with collected server power and application data. By using REST query requests in JSON format, third party applications can use this data to make strategic decisions in real time. Numeric raw data from a large database, for example memory usage and power values, are difficult to interpret. Therefore an overlaying presentation layer is required. Providing a dashboard and making power usage visible is seen to be crucial for investigation in server efficiency.
The application is required to be web based in order to support all types of platforms and run without installation just on demand. The Papillon Server with its REST API additionally hosts the web-application and the corresponding MySQL database.The possibility of monitoring several different servers at the same time is seen to be essential. Therefore connected servers should be selected from the database in order to create a corresponding power monitor. The application has to support two different monitoring modes. By picking a start as well as an end date a timespan should be displayed. In contrast real time monitoring is required as well. The real time monitor has to be updated whenever a new power value is available in the database. Additionaly ervery monitor has to calculate the mean value of power consumption over the displayed period. The Papillon API offers detailed information on the top three processes using 80% of the processing power on the server. Those processes should be exposed and shown in a piechart. For every data point in the monitoring graph this piechart should provide additional information.

4.3.2 Architecture

The Server Power Analysis Tool has to be platform independent and therefore has a web application architecture. The hosting is done within an appropriate web server environment. Apache Tomcat is already required to run Papillon and its corresponding REST API. In addition the web based server power analysis tool is hosted on the same instance of Apache Tomcat to keep the required infrastructure as low as possible. Fig. 4.6 gives an overview of the components being required on the server- and client-side.

The client only requires a web browser with an installed JavaScript plug-in to display jQuery UI elements. Every state of the art browser has JavaScript support and so no additional installation process is required for the server power analysis tool. Due to the installed Papillon master software which is hosted within the Apache Tomcat environment also on the server side no additional installation has to be done. The above mentioned advantages makes a web application platform independent and powerful. By designing the user interface in a smart manner also mobile devices are able to use this tool. The Fig. 4.7 below outlines the run-time environment on the server side which is already set up in the installation process of the Papillon master.The server setup includes in addition a MySQL server to host the Papillon database. Detailed information on the setup can be

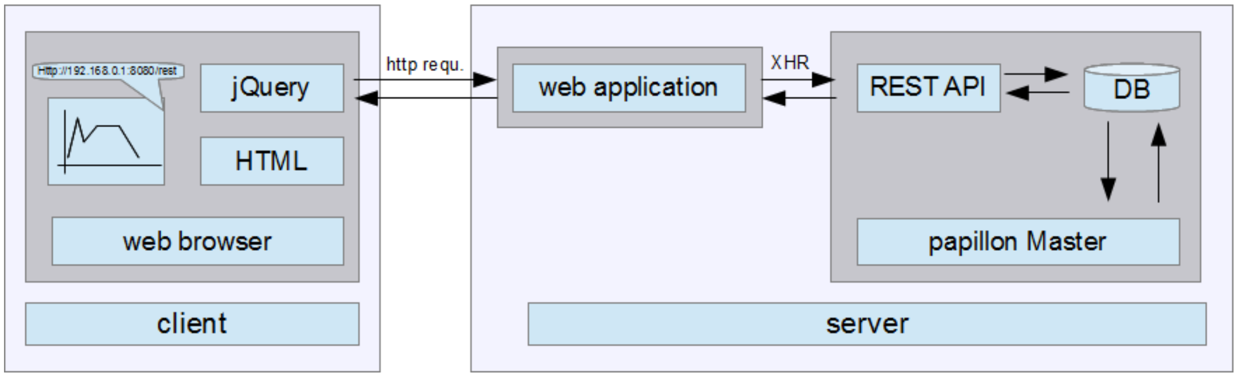


Figure 4.6: Server Power Analysis tool architecture

found in section 3.2.1.

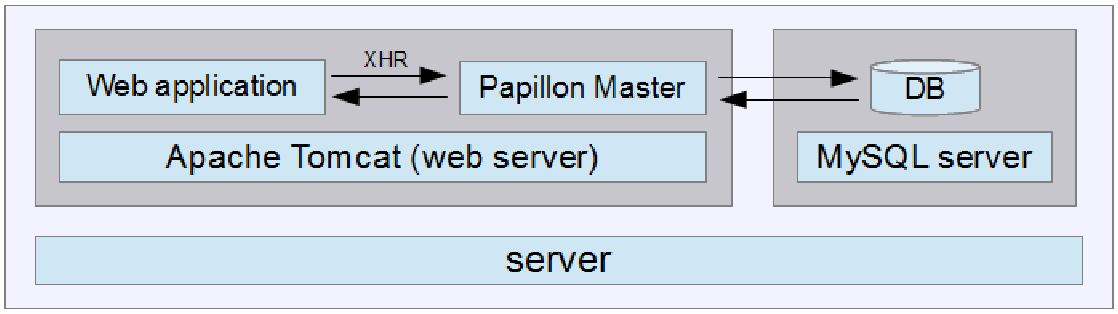


Figure 4.7: Run-time environment for Papillon being used by the SPA tool

Due to the required platform independence, client-side as well as server side development technologies are possible. With respect to complexity and development costs an open approach was chosen. HTML, JavaScript and jQuery provides enough functionality to develop a powerful web-application. Eclipse in addition offers a good development environment and is seen to be approved.
The web-application itself follows a *Model View Controller* design pattern to split the user interface and the application logic. Using such a software design approach offers modularity, better code visibility and good maintenance opportunities. Especially HTML and JavaScript needs to be seperated in order to keep the application modular.

4.3.2.1 Model View Controller

A MVC pattern consists of three different components. As outlined in the title, the components are the model, the view and a corresponding controller. In the listing below component responsibilities are declared.

The *model* manages an applications state and has the following responsibilities.

- it holds application data and keeps consistence to background database
- it includes application logic with corresponding methods
- external service (REST) interaction

The so called *view* is seen as presentation layer to enable user interaction and additionally,

- retrieves information from the model and presents them to the user
- handles user interaction
- supports client-side data validation

The *controller* delegates user input to the model and vice versa as well as,

- invokes methods of the model due to certain user inputs
- invokes GUI elements of the view to display data out of the model
- is responsible for the control flow

The *Model View Controller* pattern often includes a so called *Observer* pattern as well. The controller observes the view component and the model. User interactions in the view notify the controller in order to store or retrieve data from the model. In addition, the model notifies the controller as well if data is ready to be displayed or was correctly stored in the database. Fig. 4.8 gives an overview of the *Model View Controller* pattern. [Grove, 2011]

By separating the presentation layer from the data processing level the application becomes modular and clear. Having a modular software architecture comes with several advantages. Integrating additional web programming frameworks to develop nice looking graphic user interfaces is a major benefit to name only one. The variety of commercial and non commercial JavaScript frameworks is large. Some of them are well known in the web developer community. With respect to choose an open development approach,

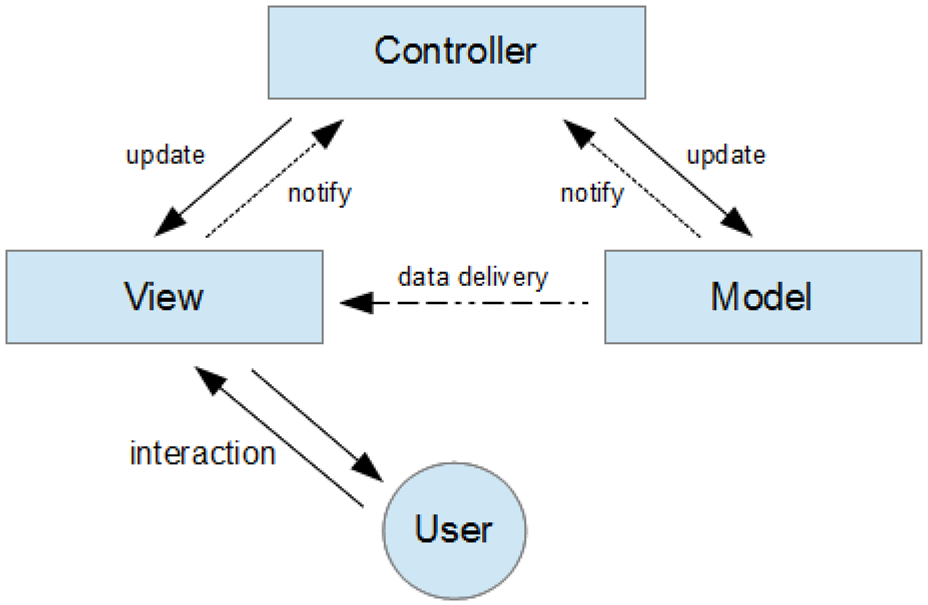


Figure 4.8: Model View Controller

jQuery was taken into account.
The well known jQuery framework has a strong core functionality and additionally offers a wide range of great user interfaces. The open source approach is the reason for lots of useful extension libraries being available. In order to handle charting, the so called *flot* JavaScript plugin can be integrated within the jQuery core library as well. *Flot* supports different charts and statistic functionality to create powerful web-based dashboards.
JavaScript is known to be a client side scripting language approach which provides additional dynamic functionality.

4.3.2.2 Papillon REST API

The REST API provided by Papillon follows a certain architecture. By using certain URL constraints, data can be received either in JSON or XML format. REST offers a number of typical methods. POST, PUT, GET and delete are the main methods being supported by the API. Within this application only GET is used to retrieve data from the master database. A REST request consists of concatenated URL snippets. Due to the fact, that the Papillon master is hosted within a Tomcat environment, the API can be accessed with the following base URL.

Base URL: http://localhost:8080/PapillonServer/rest/

Additional snippets are added depending on the type of data being delivered to the web application. Following components are available via the Papillon API structure.

- Datacenter
- Floor
- Rack
- Server

Each Data Center consists of different floors, which have a certain amount of racks. Within each rack, servers are hosted. The following snippet shows a relative URI for a GET request retrieving a list of servers in a certain rack.

Relative URI	*GET* /datacenters/DataCentreId/floors/FloorId/racks/RackId/hosts
Description	Retrieves a rack´s servers

Table 4.1: Relative sample URI

A sample URI implemented in the application returning all servers being hosted in a Data Center with id 1, floor 1 and rack 1, looks like the following.

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

Invoking REST requests with jQuery can be done in a easy manner. The library provides the option to execute Ajax calls with predefined parameters. Ajax is is an acronym for Asynchronous JavaScript and XML. Ajax uses XHR (XMLHttpRequest) to invoke methods like GET or PUT on the server side API. The asynchronous method offers the possibility to load data in the background while the user still interacts with the GUI. The method is also called non blocking. Especially loading data from a database takes its time. Therefore the user interface should be still click able and respond to user interactions. As mentioned within the *Model View Controller* description, the *model* handles all external API calls. The code snippet 4.1 gives the basis of an jQuery Ajax call. Additional parameters can be specified to select the type, data type, or timeout of the request.

The content type defines the encoding and is set to utf-8. The general type of the call is seen to be a GET request with a corresponding URL assigned. Due to the fact, that the Papillon API as well as the web application is hosted within the Tomcat environment, the URL includes *localhost* with port 8080. The data type is chosen to be JSON in order to retrieve structured data. An Ajax call includes two callback functions. Due to the asynchronous behavior, callback functions are required to process the data once they are retrieved. On success, the data, being delivered in JSON format are stored in the *model*. The observing *controller* gets notified by the model to update the *view* with new data.

```
\$.ajax({
            contentType: ”application/json; charset=utf−8”,
                type: ”GET”,
                url: ”http://localhost:8080/PapillonServer/rest/
                    datacenters/1/floors/1/racks/1/hosts/”,
                dataType: ”json”,
                data : { load : true },
                timeout: 10000,
                error: function(err){
                  ...
                },
                success: function(data){
                  that.notifyHostsLoaded();
                  ...
                }
        });
```

Listing 4.1: Ajax request snippet

4.3.3 Implementation

4.3.3.1 Development Environment

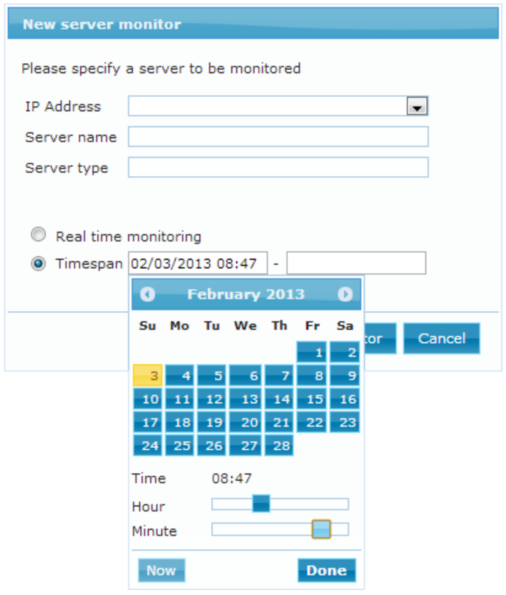
Web development offers a large variety of different development environments. Each developer prefers different development tools. Within this project eclipse was taken as development environment. Due to the open source approach, eclipse comes with a good diversity of additional plug-ins.
The web developer tools provide good syntax highlighting for HTML and JavaScript. Syntax highlighting is state of the art in almost every editor, but Eclipse offer some nice features in addition. Especially developers,being already familiar with Eclipse take advantage of setting up a web project. The local Apache Tomcat for example can be easily integrated within eclipse. Deploying the web application is then straight forward and takes less time than hosting it externally. Another nice feature is the SVN repository plug-in which can be installed in addition as well.

In order to test the developed web application, Firefox or Google Chrome are used. Both browsers offer the opportunity to extend their standard functionality with developer tools.
Firefox therefore provides the so called Firebug plug-in which comes with various debug features. HTML, CSS, Scripts and much more can be viewed in detail. This tool is seen as really helpful in order to detect bugs due to asynchronous application calls. Especially REST requests often cause troubles due to asynchronous callbacks. Those issues can be detected pretty good with Firebug.

4.3.3.2 Functionality

The main functionality of the SPA tool is analyzing the electricity power consumption of a server being monitored by Papillon. The major additional task is to compare two or more servers regarding their power profile while they run a defined sample application. Displaying the power consumption for a certain time-span in order to compare physical with virtual machines is seen to be crucial as well.

The web application can be accessed by opening the corresponding URL path in a web-browser. On start up the application opens up a configuration form shown in Fig. 4.9.


New server monitor

Please specify a server to be monitored

IP Address	
Server name	
Server type	

Real time monitoring [✓ Timespan] 02/03/2013 08:47 -

February 2013

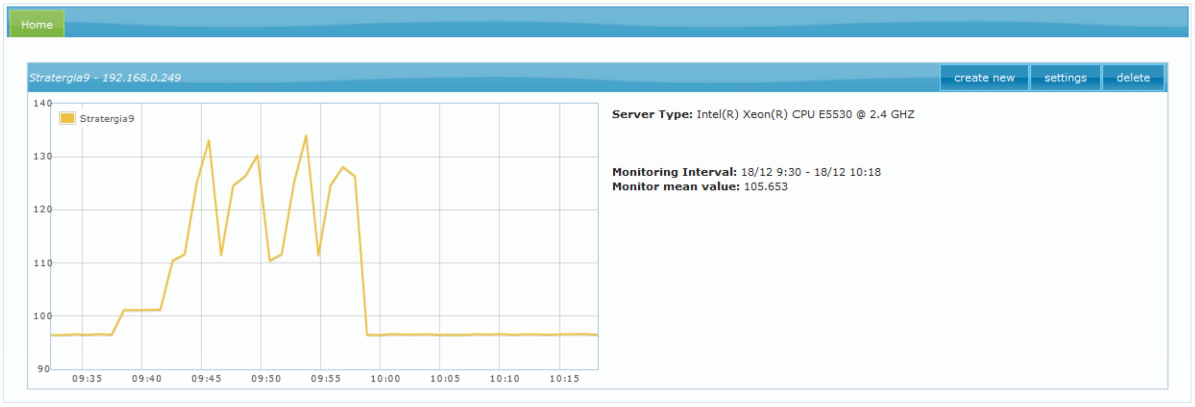
Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28		

Time 08:47

Figure 4.9: Configuration dialog on start up

This configuration form offers the possibility to select a certain server for which a power monitor will be created. The drop down box contains a full range of available servers loaded from the Papillon master database. By selecting a certain server, additional information is auto-completed. The radio button section allows to choose between two different monitoring modes. By selecting *Real time monitoring*, the power monitor will show a certain time-frame which is being updated by new incoming power values. In addition a defined time-span can be chosen as well. Therefore a defined start- end end-date can be chosen which should be displayed in the power monitor. This feature is seen to be very important if test runs were processed in the past and still need to be investigated. By analyzing specific power peaks during a defined time-span important assumptions can be made.

The major part of the application itself consists of the power monitors. After selecting an appropriate server, the corresponding power monitor is created. Fig. 4.10 gives a sample of a physical server being monitored with the SPA tool.

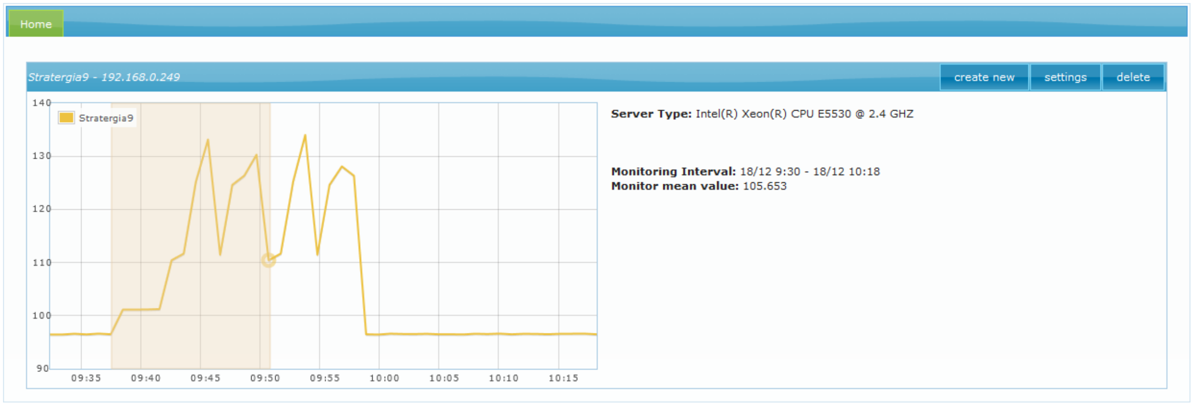


Server Type: Intel(R) Xeon(R) CPU E5530 @ 2.4 GHZ

Monitoring Interval: 18/12 9:30 - 18/12 10:18
Monitor mean value: 105.653

Figure 4.10: Power monitor for chosen server

The graph shows a continuous course of the application power consumption over a given period. The monitor mean value is calculated over the selected time-span and depends on the workload being processed on the server. Detailed investigations can be made just by interpreting the graphical view. The idle consumption for example can be observed by choosing a large time-span. Certain workloads create unique power patterns which can be determined by viewing peaks in detail. The detail view can be accessed by selecting a defined time-span within the plot. The selection of a detailed section is shown in the following Fig. 4.11.

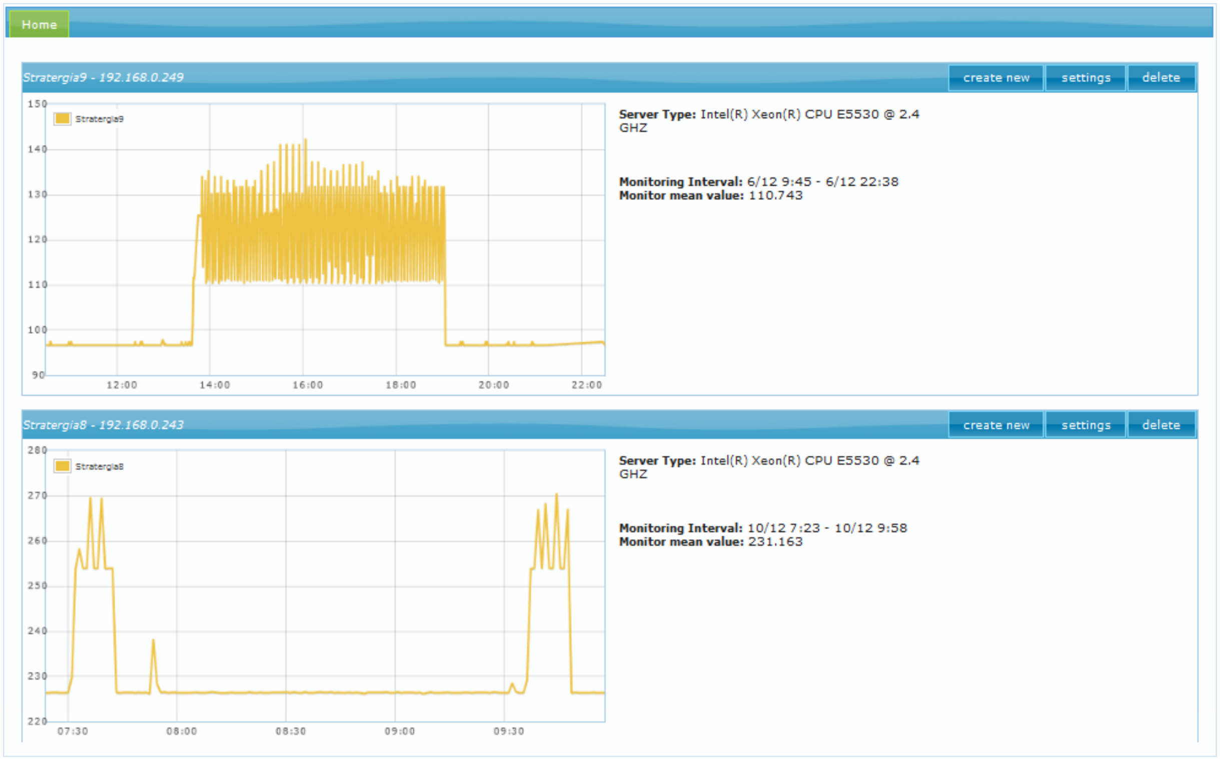


Server Type: Intel(R) Xeon(R) CPU E5530 @ 2.4 GHZ

Monitoring Interval: 18/12 9:30 - 18/12 10:18
Monitor mean value: 105.653

Figure 4.11: Detailed view on plot selection

Three menu buttons are available within the graphic user interface. For example switching between the two monitoring modes (real-time or time-span monitoring) can be applied in the settings section. Deleting the actual monitor or creating an additional one is available as well in the menu bar.On creation of a new monitor a new configuration dialog form pops up. The user has the opportunity to select another server from the Papillon master database with a corresponding custom time-span being viewed. The additional monitor is added to the *home* tab below the existing one. Depending on the server type and workload, a different application power profile will be shown. As shown in Fig. 4.10, two different server types run at different workloads created by benchmarks. Server 9 is seen to be more efficient in general due to his low idle baseline. Server 8 in contrast states an idle power consumption that doubles the one seen on server 9. This results in high costs over a long term period. In this example server 9 is a fairly new server that runs much more efficient in comparison to the older power hungry server setup on machine 8.



Server Type: Intel(R) Xeon(R) CPU E5530 @ 2.4 GHZ

Monitoring Interval: 6/12 9:45 - 6/12 22:38
Monitor mean value: 110.743

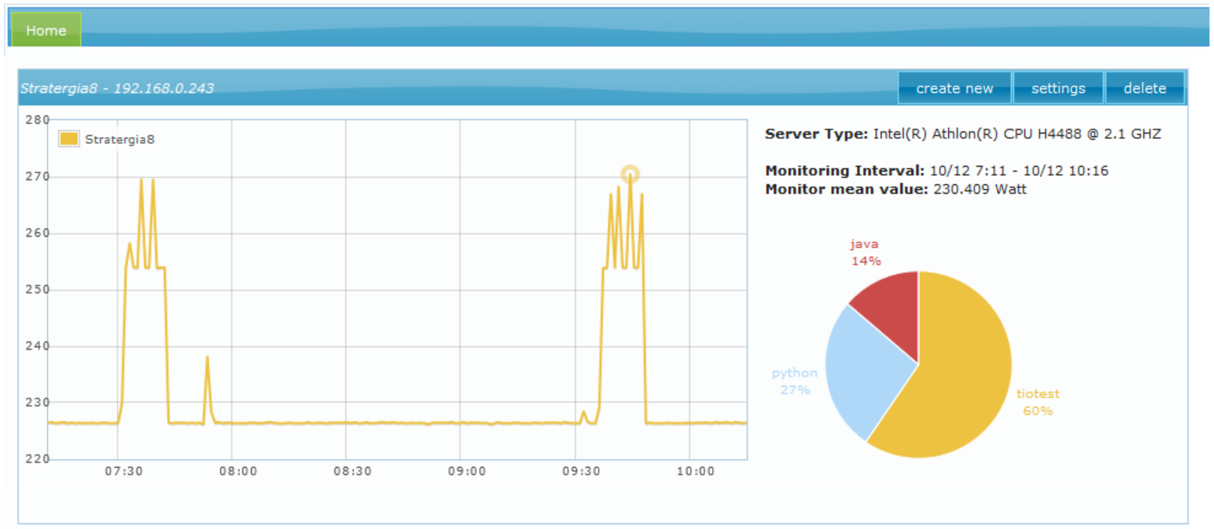
Server Type: Intel(R) Xeon(R) CPU E5530 @ 2.4 GHZ

Monitoring Interval: 10/12 7:23 - 10/12 9:58
Monitor mean value: 231.163

Figure 4.12: Two monitors with two different server types

Instead of a physical server, virtual machines can be viewed as well. The SPA tool does not differ between physical or virtual server setups. If a virtual server delivers power values in a correct manner to the Papillon master database, a corresponding power monitor can be created. The visualization of the server power, based on a graphical plot, offers valuable information in case of power effectiveness investigations.
Beside the overall server power consumption, Papillon delivers the corresponding three topmost processes leading to the power values. Every data point in the plot is clickable and contains additional information on the three top processes. By clicking the data points, a pie chart is shown which gives the process distribution for the selected power value.

This information can be very helpful in determining which application can be processed on the server. Data base intense applications might show MySQL or any other database related process in the top three. In the Fig. 4.13, the additional pie chart, displayed by clicking onto data points is shown.



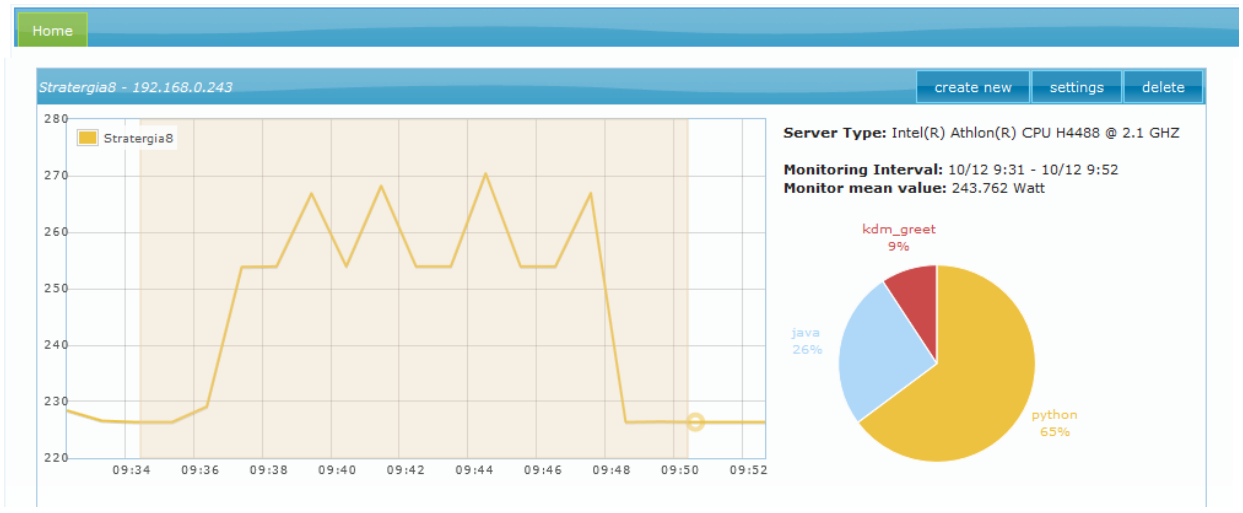
Server Type: Intel(R) Athlon(R) CPU H4488 @ 2.1 GHZ

Monitoring Interval: 10/12 7:11 - 10/12 10:16
Monitor mean value: 230.409 Watt

Figure 4.13: Pie charts are shown on clicking a data point

4.3.3.3 Metrics

The monitor mean value is the average of all data points being shown in the given time span. Depending on the period, the idle power has to be taken into account as well. In order to get the exact average power of a reference application being stressed over a period of time, the monitor should be exactly adjusted for the time frame. Once the time-frame is right, the mean value and corresponding top three processes give a detailed view on the processed application. In Fig. 4.14, the selection of a benchmark test being executed over a defined time-span is shown.



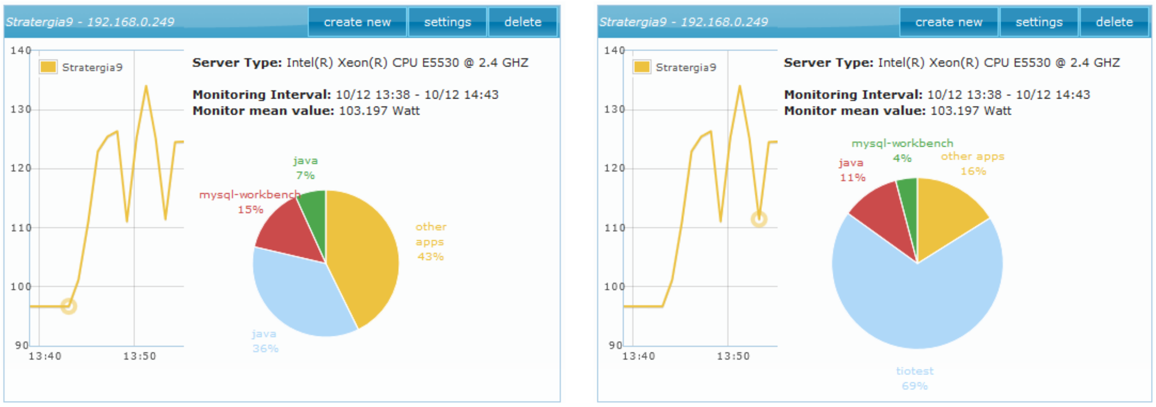
Server Type: Intel(R) Athlon(R) CPU H4488 @ 2.1 GHZ

Monitoring Interval: 10/12 9:31 - 10/12 9:52
Monitor mean value: 243.762 Watt

Figure 4.14: Selecting reference application to get the mean value

With respect to observations of the server in idle mode, a certain idle power can be determined. In Fig. 4.14, the server shows an idle power consumption of approximately 225 Watt. The mean value for the exact processing period is given with 252 Watt. Assumptions can be made, that the benchmark application, which took 10 minutes execution time, used 27 Watts additional power overhead. This detailed investigations enable analysts a totally new perspective to characterize applications on different server setups.

In addition the top three processes, taking approximately 80% of the CPU time of a certain data-point are displayed in a pie chart right beside the power monitor. The remaining percentage goes to additional processes on the server. Depending on the server utilization, the top three are always seen to have a much higher processing time on the server. In case of the idle mode, also the remaining tasks might have the same size and therefore the pie chart shows 50% other applications. The pie chart always takes the utilization of the server into account. In Fig. 4.15 two different data-points outline the different calculation within the pie chart.


Server Type: Intel(R) Xeon(R) CPU E5530 @ 2.4 GHZ

Monitoring Interval: 10/12 13:38 - 10/12 14:43
Monitor mean value: 103.197 Watt

Server Type: Intel(R) Xeon(R) CPU E5530 @ 2.4 GHZ

Monitoring Interval: 10/12 13:38 - 10/12 14:43
Monitor mean value: 103.197 Watt

Figure 4.15: Process utilization depending on the server workload

In the first part of Fig. 4.15, the pie chart shows the process details for an idle data point. Within idle, most of the processes are rated equally and so the top three processes use less than 80% of the CPU time. In this case, the *other apps* amount is seen to be very high. In contrast the pie chart at the right hand side is taken during program execution. The CPU time therefore is used to approximately 80% by the top three processes and the other apps state the remaining CPU time. In this example, *tiotest* uses almost the whole CPU time and so the peaks in the power monitor results from this application.

Chapter 5

Evaluation

The Evaluation uses different approaches and developed methods collectively to investigate physical and virtual server power characteristics. Software based power metering itself is seen to be a milestone in the field of Data Center technologies. Using stress test software which creates real world based workloads is an interesting and new approach as well. Using both together collectively with an additional power analysis dashboard offers the possibility to evaluate the different power profiles of physical and virtual machines. This approach gives insights on whether certain workloads run more efficient on virtual machines than on physical ones. The visualization of certain workloads is seen to be an interesting step in the field of green IT and might contribute in the area of efficient cloud computing.

5.1 Reference application

Stress testing servers with real world applications is seen to be much more valuable than just running number crunching benchmarks. Chapter 3.3 mentions a stress testing software called vApus which offers the possibility to simulate virtual users, interacting with a certain given sample application.

By using connection proxies, vApus is able to connect to every application on the server side and simulate a set of users to create certain workloads. vApus stress tests given servers and investigates certain metrics like throughput, response times and much more. The benchmark approach is used to put a defined workload on a physical machine and compare its power characteristics against virtual server environments. The focus is set on the creation of a reference workload which can be used across different server architectures. Only if the workload is a reference, does the evaluation of different power characteristics make sense.

Sizing Servers Lab provides a minimal CentOS 6 64bit installation image which comes with some predefined settings in order to install the reference application. The big advantage, having a .iso image is seen in the flexibility to install it also in a virtual environment. Once the operating system is installed, a given installation manual helps to be guided through the installation process of the sample application. The application itself is an ordinary php-based web-application stating a forum which is hosted on the system. A forum is seen to be a real world application which is hosted on servers in real data centers as well. A web forum is an application which is dependent on its user activity. Depending on the user concurrency and traffic, the host machine has to handle huge amounts of http-requests. Concurrent user entries and activities stress the server to a certain extend and provide a realistic workload for an evaluation.

In contrast a crunching number algorithm would stress the CPU of a server, but nothing more. A web application uses more than only the processing power of a CPU and therefore sufficient stress test scenarios have to be found. vApus offers a unique stress test approach which gives a real world test environment if it is used with a corresponding sample application in a sufficient way. Analysing the power characteristics of physical and virtual servers by only using a software metering approach in combination with real world application workloads is seen to be unique in the field of research. The following sections will outline the evaluation process in detail and give the overall results of this thesis.

5.2 Physical Evaluation environment

In order to evaluate the different power characteristics of physical and virtual machines, a corresponding test environment is affordable. The given stress test software vApus only operates on a 64 bit Windows machine. The following system overview outlines the evaluation Environment and gives an idea of the distributed monitoring approach.

- **Physical Server**
  Operating Sytem: Ubuntu 12.04 LTS 64-Bit
  Processor: Intel Xeon(R) CPU E5530 @ 2.40 GHz x 8
  Memory: 15.7 GiB
  Disk: 481.8 GiB

The requirements for a given server are shown in Fig. 5.1. A physical or virtual server is set up with minimal CentOS in a 64 bit version. In addition the reference application provided by Sizing servers Lab is installed on the machine. The sample application, a web based forum, requires also a MySql server to be installed. In order to measure the server power, the thin Papillon client needs to be installed on the minimal Cent OS as well. Each machine has its own power profile which has to be determined before every

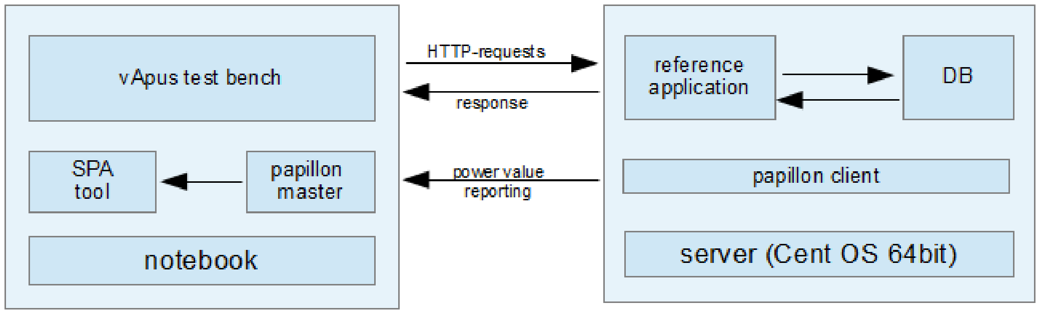


Figure 5.1: Evaluation architecture for server power analysis

evaluation process. Once the Papillon client is installed and the corresponding power model is stored on the Papillon master side, power values from the server are transmitted every minute. The sample application is stressed with a custom created test-bench which creates a defined workload on the system. The test-bench runs on a Windows based 64 bit machine and fires HTTP-requests to the opposite reference application. By simulating various amounts of concurrent user interactions with vApus, huge amounts of HTTP-requests create defined workloads on the server. The notebook also runs a Tomcat web-server in order to host the Papillon master system as well as the server power analysis tool. The server power analysis tool is seen to be the visual dashboard which provides visibility to the power consumed by the server during different workload tests.

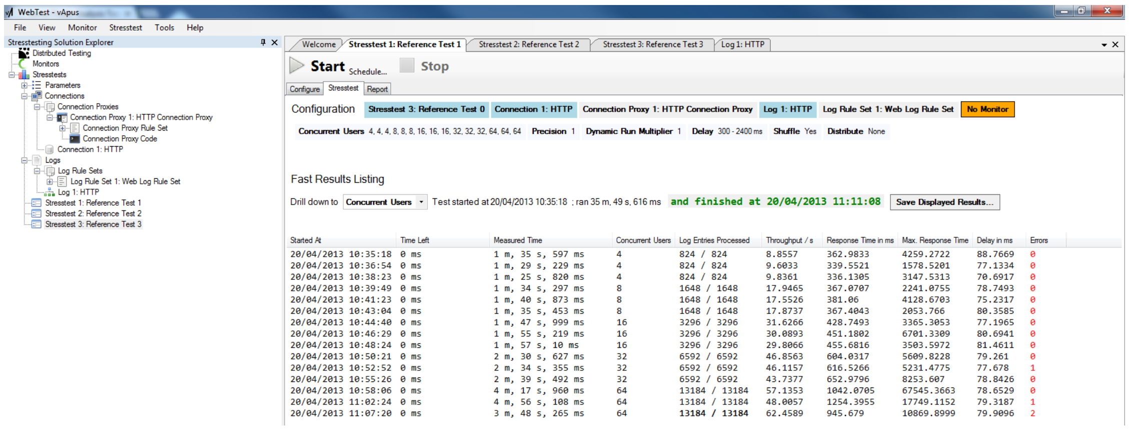
The whole evaluation is conducted in a remote manner and shows the non-intrusive power monitoring approach. The physical server is located at the School of Computer Science in Dublin whereas the Papillon master and the Server power analysis tool run on a laptop based in Austria. Only the network gateway has to be established in order to receive power values collected by the Papillon client.

5.3 Physical server monitoring

Hosting approaches are seen to be different in every data center and depend on the type of application being processed as well. Certain Data Centers have a higher range of physical machines than others. Virtualization can save processing power but also comes with a set of disadvantages as well. A physical machine for example does not have to share its hardware with different other machines which increases the portion of task being processed in a certain time span. Whereas virtual machines are using physical hardware in a scheduled manner, physical machines might complete tasks a bit faster. In contrast high idle phases are the reason for inefficient power usage in data centers and can be addressed by virtualization. In the following section different workloads are processed on a physical server and corresponding observations are discussed.

5.3.1 Reference Workload 1

The physical server runs CentOS in order to host the sample web-application which consists of a simple php forum. The vApus stress test tool, which is hosted on a remote laptop in Austria offers the possibility to create a project file containing the overall test configuration. Fig. 5.2 gives an idea of the setup used within the stress test tool.


WebTest - vApus
File View Monitor Stresstest Tools Help
Stresstesting Solution Explorer
Distributed Testing
Monitors
Stresstests
Parameters
Connections
Connection Proxies
Connection Proxy 1: HTTP Connection Proxy
Connection Proxy Rule Set
Connection Proxy Code
Connection 1: HTTP
Logs
Log Rule Sets
Log Rule Set 1: Web Log Rule Set
Log 1: HTTP
Stresstest 1: Reference Test 1
Stresstest 2: Reference Test 2
Stresstest 3: Reference Test 3
Welcome
Stresstest 1: Reference Test 1
Stresstest 2: Reference Test 2
Stresstest 3: Reference Test 3
Log 1: HTTP
Start Schedule...
Stop
Configure Stresstest Report

Configuration: Stresstest 3: Reference Test 0 | Connection 1: HTTP | Connection Proxy 1: HTTP Connection Proxy | Log 1: HTTP | Log Rule Set 1: Web Log Rule Set | No Monitor

Concurrent Users 4, 4, 4, 8, 8, 8, 16, 16, 16, 32, 32, 32, 64, 64, 64 Precision 1 Dynamic Run Multiplier 1 Delay 300 - 2400 ms Shuffle Yes Distribute None

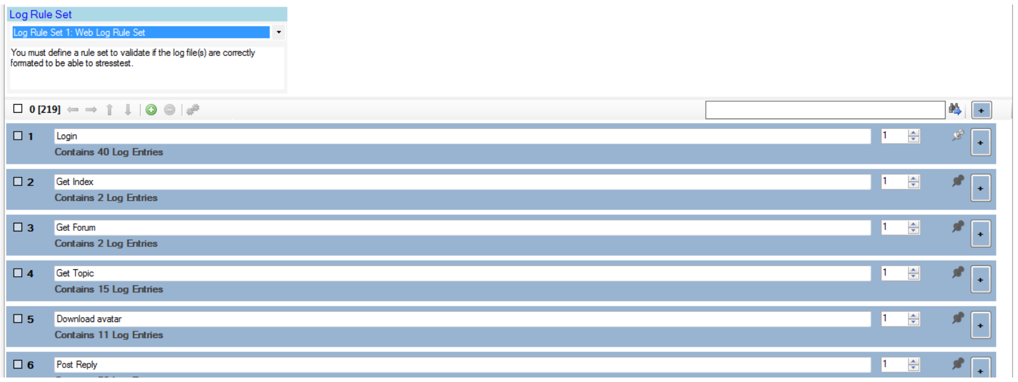
Fast Results Listing

Drill down to Concurrent Users Test started at 20/04/2013 10:35:18 ; ran 35 m, 49 s, 616 ms and finished at 20/04/2013 11:11:08 Save Displayed Results...

Started At	Time Left	Measured Time	Concurrent Users	Log Entries Processed	Throughput / s	Response Time in ms	Max. Response Time	Delay in ms	Errors
20/04/2013 10:35:18	0 ms	1 m, 35 s, 597 ms	4	824 / 824	8.8557	362.9833	4259.2722	88.7669	0
20/04/2013 10:36:54	0 ms	1 m, 29 s, 229 ms	4	824 / 824	9.6033	339.5521	1578.5201	77.1334	0
20/04/2013 10:38:23	0 ms	1 m, 25 s, 820 ms	4	824 / 824	9.8361	336.1305	3147.5313	70.6917	0
20/04/2013 10:39:49	0 ms	1 m, 34 s, 297 ms	8	1648 / 1648	17.9465	367.0707	2241.0755	78.7493	0
20/04/2013 10:41:23	0 ms	1 m, 40 s, 873 ms	8	1648 / 1648	17.5526	381.06	4128.6703	75.2317	0
20/04/2013 10:43:04	0 ms	1 m, 35 s, 453 ms	8	1648 / 1648	17.8737	367.4043	2053.766	80.3585	0
20/04/2013 10:44:40	0 ms	1 m, 47 s, 999 ms	16	3296 / 3296	31.6266	428.7493	3365.3053	77.1965	0
20/04/2013 10:46:29	0 ms	1 m, 55 s, 219 ms	16	3296 / 3296	30.0893	451.1802	6701.3309	80.6941	0
20/04/2013 10:48:24	0 ms	1 m, 57 s, 10 ms	16	3296 / 3296	29.8066	455.6816	3503.5972	81.4611	0
20/04/2013 10:50:21	0 ms	2 m, 30 s, 627 ms	32	6592 / 6592	46.8563	604.0317	5609.8228	79.261	0
20/04/2013 10:52:52	0 ms	2 m, 34 s, 355 ms	32	6592 / 6592	46.1157	616.5266	5231.4775	77.678	1
20/04/2013 10:55:26	0 ms	2 m, 39 s, 492 ms	32	6592 / 6592	43.7377	652.9796	8253.607	78.8426	0
20/04/2013 10:58:06	0 ms	4 m, 17 s, 960 ms	64	13184 / 13184	57.1353	1042.0705	67545.3663	78.6529	0
20/04/2013 11:02:24	0 ms	4 m, 56 s, 108 ms	64	13184 / 13184	48.0057	1254.3955	17749.1152	79.3187	1
20/04/2013 11:07:20	0 ms	3 m, 48 s, 265 ms	64	13184 / 13184	62.4589	945.679	10869.8999	79.9096	2

Figure 5.2: vApus project file with reference workload 1

As shown in Fig. 5.2, a chronological order of concurrent users is processed. Each cycle consists of a number of log entries being processed on the server side. The amount of log entries is calculated out of the concurrent users multiplied by the fixed set of log files for each user. A set of log files consists of certain actions which where recorded during the test bench creation. Fig. 5.3 shows a set of log entries which each concurrent user sends to the stress tested server. The set consists of thirteen different actions which are sent to the server. For example the first action is called Login, which has 40 different sub entries.



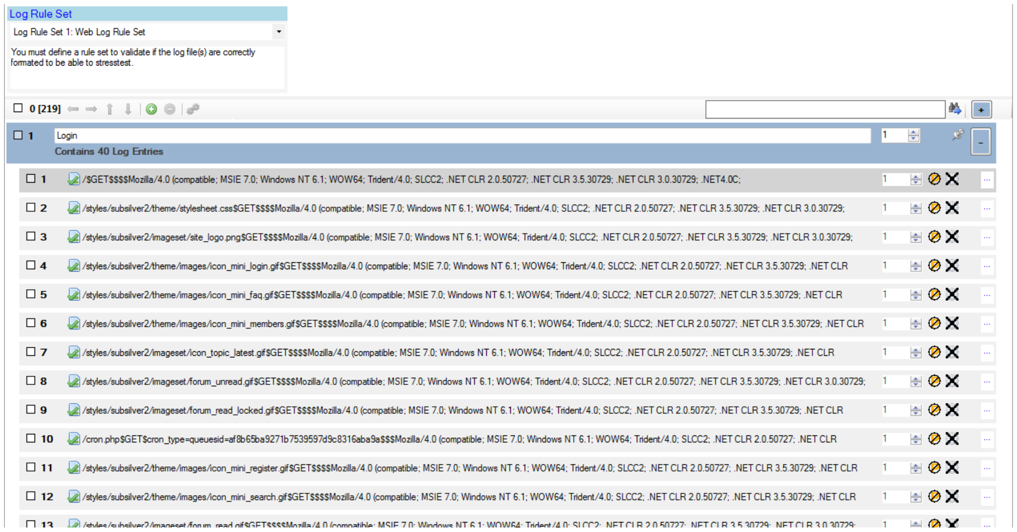
Log Rule Set 1: Web Log Rule Set

You must define a rule set to validate if the log file(s) are correctly formated to be able to stresstest.

1	Login Contains 40 Log Entries	1
2	Get Index Contains 2 Log Entries	1
3	Get Forum Contains 2 Log Entries	1
4	Get Topic Contains 15 Log Entries	1
5	Download avatar Contains 11 Log Entries	1
6	Post Reply	1

Figure 5.3: Set of log rules in vApus project file

Sub entries are the single steps a user has to take in order to execute the given action. The test software vApus provides a record tool which created these sub entries automatically during a login process. The Fig. 5.4 shows the first 13 entries out of 40 which has to be executed on the server side to complete a login on the php forum.



Log Rule Set

Log Rule Set 1: Web Log Rule Set

You must define a rule set to validate if the log file(s) are correctly formated to be able to stresstest.

0 [219]

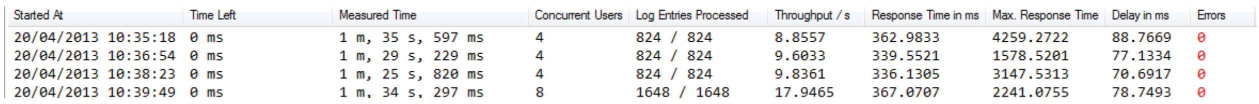
1 Login 1
Contains 40 Log Entries

```
1   /$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; .NET4.0C;   1
2   /styles/subsilver2/theme/stylesheet.css$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;   1
3   /styles/subsilver2/imageset/site_logo.png$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;   1
4   /styles/subsilver2/theme/images/icon_mini_login.gif$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR   1
5   /styles/subsilver2/theme/images/icon_mini_faq.gif$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR   1
6   /styles/subsilver2/theme/images/icon_mini_members.gif$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR   1
7   /styles/subsilver2/imageset/icon_topic_latest.gif$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR   1
8   /styles/subsilver2/imageset/forum_unread.gif$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;   1
9   /styles/subsilver2/imageset/forum_read_locked.gif$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR   1
10  /cron.php$GET$cron_type=queuesid=af8b65ba9271b7539597d9c8316aba9a$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR   1
11  /styles/subsilver2/theme/images/icon_mini_register.gif$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR   1
12  /styles/subsilver2/theme/images/icon_mini_search.gif$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR   1
13  /styles/subsilver2/imageset/forum_read.gif$GET$$$$Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;   1
```

Figure 5.4: Sub entries in the login section

As mentioned, the log rule set has got 40 sub entries. All in all the 13 log rule sets create 206 single requests. Those requests are multiplied by 4 concurrent users which result in 824 entries being sent to the server. Each virtual user sends his 209 entries with a certain delay in between the single entries. A single cycle of 4 concurrent users is handled within 1 minute and 35 seconds as shown in Fig. 5.2.

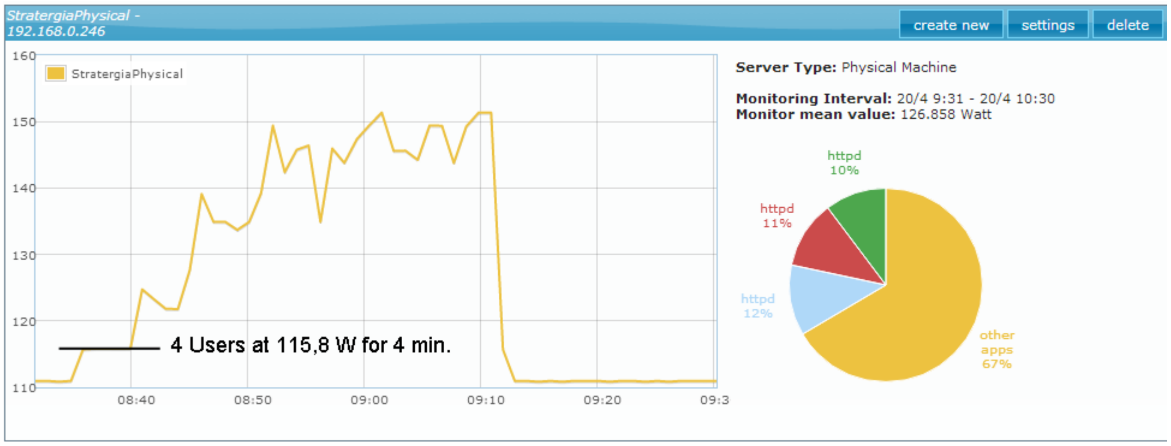
This outlines that the physical server processed 824 instructions within this time frame. This outlines that every 115 ms a single entries was processed in order to handle the workload.
With respect to certain inconsistencies this test case was repeated three times. The following graphic 5.5 shows the three test cycles.



Started At	Time Left	Measured Time	Concurrent Users	Log Entries Processed	Throughput / s	Response Time in ms	Max. Response Time	Delay in ms	Errors
20/04/2013 10:35:18	0 ms	1 m, 35 s, 597 ms	4	824 / 824	8.8557	362.9833	4259.2722	88.7669	0
20/04/2013 10:36:54	0 ms	1 m, 29 s, 229 ms	4	824 / 824	9.6033	339.5521	1578.5201	77.1334	0
20/04/2013 10:38:23	0 ms	1 m, 25 s, 820 ms	4	824 / 824	9.8361	336.1305	3147.5313	70.6917	0
20/04/2013 10:39:49	0 ms	1 m, 34 s, 297 ms	8	1648 / 1648	17.9465	367.0707	2241.0755	78.7493	0

Figure 5.5: Three test cycles with 4 concurrent users each

The response time as well as the delay is irrelevant in this stress test, because we do not want to measure the performance of the network constellation but the server power of the machine itself. Almost all of the three tests with four concurrent users took the same timespan to be processed. This minor workload caused the physical server to increase its power consumption. Fig. 5.6 outlines the power consumption caused by only 4 concurrent users on the system.



Server Type: Physical Machine

Monitoring Interval: 20/4 9:31 - 20/4 10:30
Monitor mean value: 126.858 Watt

Figure 5.6: 4 concurrent users with 3 cycles using 115,8 Watts

The idle consumption of 110 Watts shows that four users cause the power to rise by 5,8 Watts. The physical server takes 115,8 Watts over 4 minutes in order to handle three cycles with 4 concurrent users each. Within those 4 minutes 824 x 3 log entries are executed. This sample shows the exact approach of how a reference workload is being analysed by the developed Server power analysis tool. In order to extend the analysis the exact processes causing the extra 5 Watts are shown in the pie chart on the side. Due to the fact, that the sample application uses a simple php forum, all the processes are of type httpd.

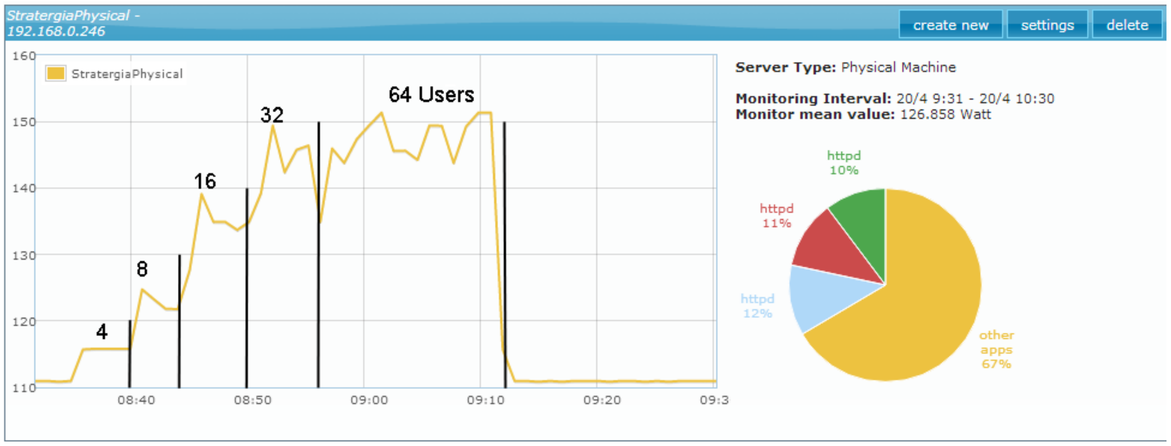
The overall results are shown in Table 5.1. Fig. 5.7 is a capture of the overall server power analysis for a given reference workload 1.

Users	Power usage / Watts	Timespan / min	overall Timeframe
4	115,8	4	5
8	122,8	4	5
16	135,6	5	6
32	145,5	4	5
64	148,3	14	18

Table 5.1: Power consumption for different amount of concurrent users

The overall test took 37 minutes to be completed. Based on the obvious power profile, visible in Fig. 5.7, every single workload step can be analysed. The detailed investigation outlines that up to 32 concurrent users the power is increased in a linear manner. By doubling the users, the server power rises by approximately 10 Watts. The processing time seems to be at a constant level of 5 min for each case.

An interesting phenomenon is seen to be in the 64 users section. Whereas the power just increases by 3 Watts, the processing time triples. This is obviously a sign that the



Server Type: Physical Machine

Monitoring Interval: 20/4 9:31 - 20/4 10:30
Monitor mean value: 126.858 Watt

Figure 5.7: Server Power Analysis tool with the overall power profile of reference workload 1

physical machine reaches out to its saturation. This concludes that everything over 64 Users just results in sequential processing time which increases the costs per user.

As mentioned in Chapter 3.3, vApus uses threads in order to create the concurrent user base. This approach is called to be a pseudo-parallel method and therefore is only a limited real world scenario. The server power more or less depends on the actions requested by vApus. The following two samples in table 5.2 outline the difference between the processing of 64 users by using two different vApus stress testing approaches.

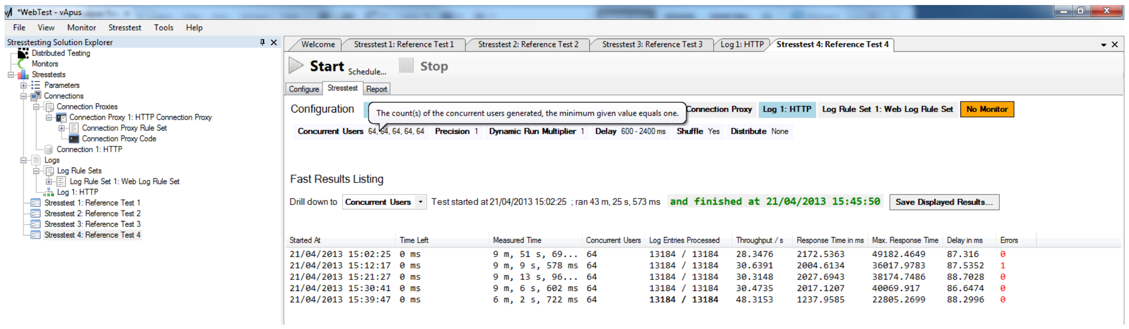
Users	vApus Instances	Power usage / Watts	Duration / min
64	1	146,5	5
32	2	141,0	6

Table 5.2: 64 concurrent users created with two different approaches

A single instance of vApus is used to create 64 concurrent user actions. This test took five minutes to its completion. The average power usage is 146,5 Watts over 5 minutes. In contrast, two separate instances of vApus where used with 32 users each. The test was started simultaneously to create 64 parallel users on the server. This test run just took 141 Watts over 6 min, which all in all is less energy than with only a single instance.
The reason for different results is seen in the pseudo-parallel threading method used by vApus. Whereas a single instance stacks the threads, the second instance hits the server parallel. The server is able to handle parallel user actions more power efficient than sequential requests.

5.3.2 Reference Workload 2

Whereas reference test 1 continuously rises the workload during the test in order to reach the saturation tipping point, the reference 2 test straight away sets the server in saturation regarding his power consumption. The difference between both reference tests is seen in the test setup. Reference Test 2 uses two different vApus instances in order to create parallel requests. As shown in the screen shot 5.8 below, a single vApus instance created several cycles with 64 concurrent users each.


Welcome
Stresstest 1: Reference Test 1
Stresstest 2: Reference Test 2
Stresstest 3: Reference Test 3
Log 1: HTTP
Stresstest 4: Reference Test 4
Start
Schedule...
Stop
Configure
Stresstest
Report

Configuration

The count(s) of the concurrent users generated, the minimum given value equals one.

Connection Proxy | Log 1: HTTP | Log Rule Set 1: Web Log Rule Set | No Monitor

Concurrent Users 64, 64, 64, 64, 64 Precision 1 Dynamic Run Multiplier 1 Delay 600 - 2400 ms Shuffle Yes Distribute None

Fast Results Listing

Drill down to Concurrent Users Test started at 21/04/2013 15:02:25 ; ran 43 m, 25 s, 573 ms and finished at 21/04/2013 15:45:50 Save Displayed Results...

Started At	Time Left	Measured Time	Concurrent Users	Log Entries Processed	Throughput / s	Response Time in ms	Max. Response Time	Delay in ms	Errors
21/04/2013 15:02:25	0 ms	9 m, 51 s, 69...	64	13184 / 13184	28.3476	2172.5363	49182.4649	87.316	0
21/04/2013 15:12:17	0 ms	9 m, 9 s, 578 ms	64	13184 / 13184	30.6391	2004.6134	36017.9783	87.5352	1
21/04/2013 15:21:27	0 ms	9 m, 13 s, 96...	64	13184 / 13184	30.3148	2027.6943	38174.7486	88.7028	0
21/04/2013 15:30:41	0 ms	9 m, 6 s, 602 ms	64	13184 / 13184	30.4735	2017.1207	40069.917	86.6474	0
21/04/2013 15:39:47	0 ms	6 m, 2 s, 722 ms	64	13184 / 13184	48.3153	1237.9585	22805.2699	88.2996	0

Figure 5.8: vApus configuration for 5 times 64 concurrent users

All in all a single vApus instance runs 5 cycles with 64 users each. A second vApus instance is running exactly the same amount of users. Both are started at the same time to create parallel requests on the physical server. Fig. 5.9 gives the overall power characteristic monitored with the Server Power Analysis tool.

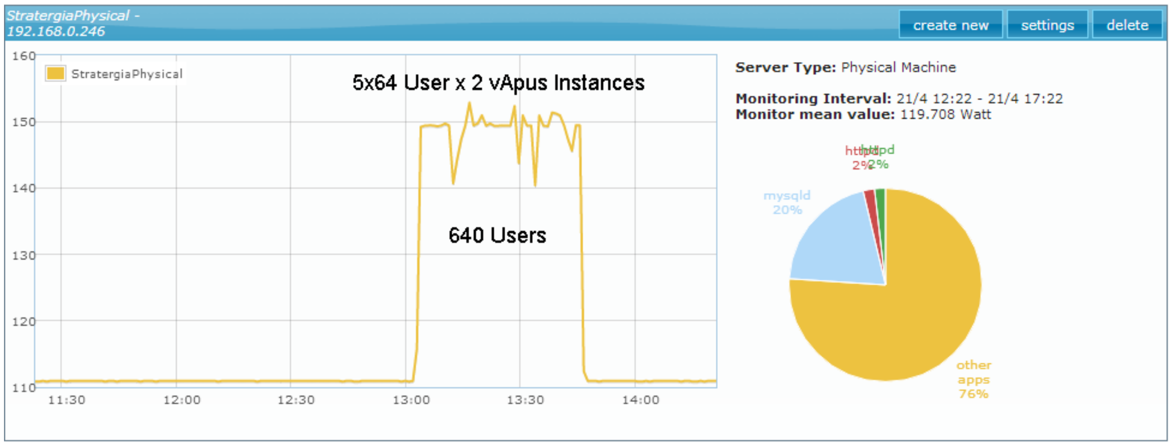


Figure 5.9: Server Power Analysis tool with the overall power profile of reference workload 2

The significant rise in power usage indicates the saturation behaviour. This does not necessary include, that the server is working inefficient. This just concludes, that any more concurrent users just affect the processing time rather than the power level. In this Test five times 64 Users multiplied by two different vApus instances result in 640 Users during the test. The test took 42 min at an average load of 149 Watts.

A very interesting observation can be seen in Fig. 5.9. The pie chart gives insights on every single process being executed at certain power levels. In contrast to the reference 1 workload tests, the mysqld process is using a huge part of the pie. The two separate vApus instances obviously put much more stress on the mysql server than a single one.

5.3.3 Electricity costs

The overall question to be answered is, what are the electricity costs for certain average workloads. The Papillon system in combination with the developed Server Power Analysis tool is a powerful system to calculate the overall electricity costs for certain workloads and processing periods.

Table 5.3 lines up reference workload 1 and workload 2 in order to compare the consumed electricity and corresponding costs. The corresponding costs are calculated with an average european electricity unit price of € 0.2 per kWh which equals € 0.00333 per minute.

$$0.135kW * 0.0033 * 37min = 0.0165Euro \tag{5.1}$$

	Users	avg. Power usage / Watts	Duration / min	Costs / euro
Ref. Load 1	372	135	37	0.0165
Ref. Load 2	640	149	42	0.0206

Table 5.3: Reference workload 1 costs vs. workload 2

The possibility to calculate the electricity costs for a defined workload is seen to be a new approach in the field of server power analysis. Depending on the user base, average electricity costs per user can be calculated as well. With an expected amount of users on certain hosted web applications a detailed electricity forecast with corresponding costs could be created. Such forecasts might be a milestone in the field of cost chargeback models for cloud computing services.

Different workloads show different power characteristics on physical machines, depending on certain hardware configurations. The possibility to monitor an applications power during execution offers great opportunities to create smart hosting models. Depending on the application characteristics, the hardware can be chosen. Only a drop by 1 Watt of average power usage results in a tremendous amount of saved energy over the year.

5.4 Virtual server monitoring

A virtual server offers the possibility to host several virtual machines running multiple different operating systems. In order to organize the hardware sharing, different supervisor applications are available on the market. For this evaluation VMware Workstation is being installed on a desktop server.

- **Desktop Server**
  Operating Sytem: Ubuntu 12.04 LTS 64-Bit
  Processor: Intel(R) Core 2 CPU 6300 @ 1.86GHz x 2
  Memory: 3.3 GiB
  Disk: 156,5 GiB

As outlined in the drawing 5.10 below, each virtual machine runs a full functional operating system. As described in chapter 5.2, CentOS is required to provide the runtime environment for the reference web-application. Each virtual machine has its own SQL server running in order to create the reference workload during the load tests. In addition, a Papillon client is installed to report power values back to the central master on the notebook. The major difference in the Papillon reporting architecture in this virtual environment are the different types of clients. Each client normally has its underlying hardware power profile which in this case is the same for all three clients. Only for VM1 and VM2 a certain offset has to be calculated due to hardware sharing during runtime.

5.4.1 Known issues

The virtual server system is much more complex to characterize than a physical server. While on a physical server every rise in electricity consumption is direct related to certain workloads triggered by applications, the power characteristics on virtual machines depends on the set of concurrent hosted machines sharing the same hardware resources. The power modelling of a virtual machine requires detailed investigation in hardware resource sharing and depends on the workloads being processed on other machines. Within this thesis only the physical Papillon client (3) with its corresponding matching power-model is used to monitor the server´s workload. Papillon client (1) as well as (2) would not retrieve valuable power data due to the missing virtual power profiles.

The physical Papillon client, with its matching power model, gathers the overall power consumption of the physical server which consists of the two virtual machines and their supervisor system. As long as only a single VM is hosted, the power profile is easy to determine. The base lined physical server characteristic can be subtracted from the monitored power consumption during the reference test. In case of running two virtual machines, the investigation is seen to be more difficult.

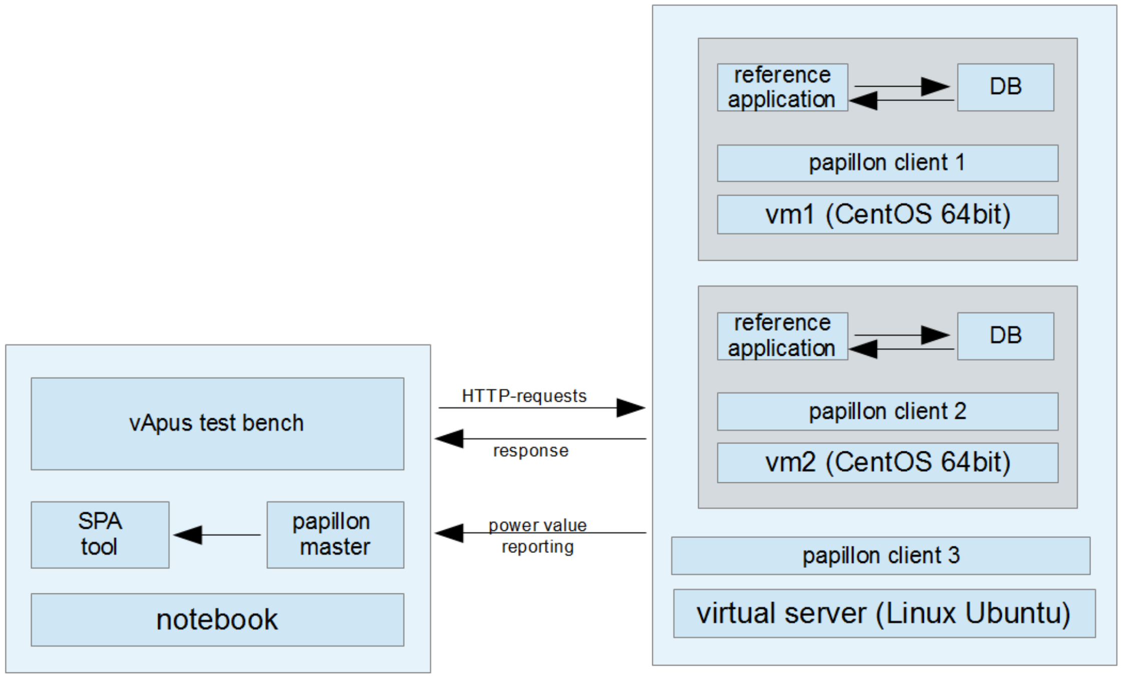


Figure 5.10: Evaluation architecture for the virtual environment

Due to the required supervisor software, a certain overhead in electricity consumption is expected. If a virtual machine would have its own accurate power model, this overhead could be determined with ease. The overall question, if the reference workload runs more efficient in a virtual environment therefore can not be proven within the scope of this work. In order to get as much information as possible, a second virtual machine will be loaded with the same reference application. This should be efficient enough to give a rough statement on the power effectiveness of web-applications in virtual environments.

5.4.2 Reference Workload 1

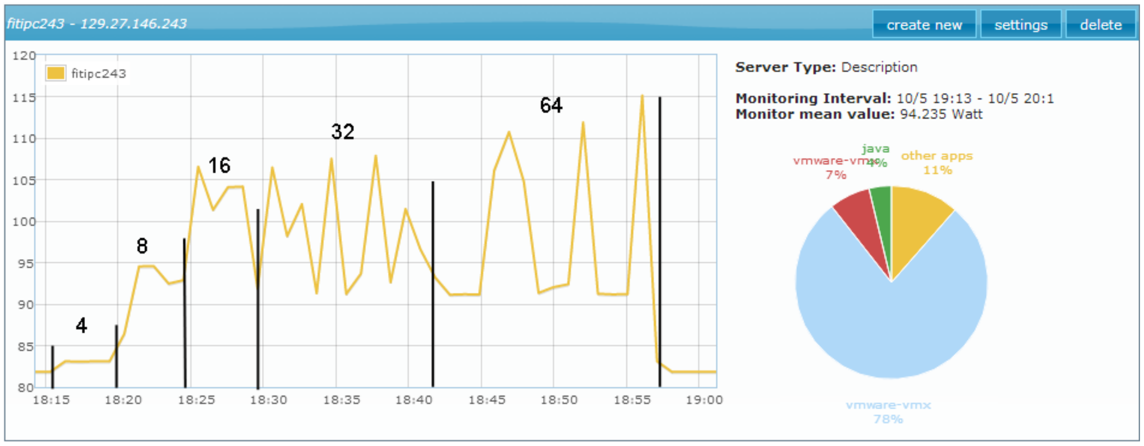
The initial idle power consumption of the physical server is measured at 81.8 Watts. Compared to the physical machine in section 5.3 this is approximately 30 Watts less. With respect to the hardware components, the physical server from section 5.3 is seen to be an industrial server with proper processing power which automatically results in a higher idle power consumption. In comparison, the server, which is being used for the hosting of the virtual environment, has less processing power due to limited project resources.
In a real world cloud computing environment, host servers of virtual machines are expected to have more processing power in order to handle multiple operating systems simultaneously.
Neither the less, interesting observations during the processing of referential workloads arose. Against the assumptions in section 5.4.1, the hyper-visor software does not have any valuable affect on the overall power consumption. As outlined in the section *Related*

Work, [Marcu and Tudor, 2011] observed exactly the same behaviour. They measured the exact same power consumption on a physical system than on a virtual one, but claimed that the VMs have a lower throughput. [Marcu and Tudor, 2011]

The reference workload 1 exists of a set of concurrent users, interacting with the php-forum. Each set is processed three times before the next set starts. Three times 4 Users are processed in the first set before the workload is increased to 8 users. The overall processing queue looks like the following. [4,4,4] [8,8,8] [16,16,16] [32,32,32] [64,64,64]

As shown in Fig.5.11, the virtual machine shows a similar saturation curve than the physical server in the drawing 5.7 from section 5.3.1. 4, 8 and 16 users are processed within the same timespan (5min) than on a physical machine. Similar to the physical machine, the power consumption seems to reach a saturation point. Instead of observing a significant rise in power consumption, just a minimal overhead is seen, but the processing time doubles at least. The jumping peaks seem to be caused by the workload itself. On each cycle, the amount of test users have to be created on Cent-OS which might result in a peak. After creation, less power is required to start the user actions in the php-forum.

In comparison to the physical server mentioned in section 5.3, the virtual machine seems be less robust concerning high workloads and CPU driven processes. As discussed, 4, 8 and 16 users are being processed as efficient as on the physical machine.



Server Type: Description

Monitoring Interval: 10/5 19:13 - 10/5 20:1
Monitor mean value: 94.235 Watt

Figure 5.11: Server Power Analysis tool with the overall power profile of reference workload 1

An interesting observation arises concerning the overall efficiency. Due to the low idle power consumption of the hosting server the overall costs of the reference 1 workload test are less than on the initial tested physical server in section 5.3.

Table 5.4 compares the physical machine with the virtual one concerning the costs of the referential workload test 1.The average workload power consumption was calculated by cutting off the idle power of the server.

	idle usage / Watts	avg. usage / Watts	avg.-idle / Watts	duration / min	costs / euro
physical	111	135	24	37	0.00293
virtual	82	96	14	42	0.00194

Table 5.4: Reference workload 1 costs on physical and virtual server

As outlined, both servers show proximately the same processing time as well as a similar power graph for the referential load.
The physical machine differs in the overall power effectiveness. Instead of 82 Watts idle consumption used by the virtual server, the physical requires 111 Watts. Even by cutting off this idle, the average power usage during the stress test is seen to be 10 Watts higher than on the virtual machine. The interesting part here is also, that the virtual environment has less processing power and still handles the workload with the same efficiency.
This showcase gives an idea of how many applications in cloud computing environments might be hosted at higher hardware resource costs than actually afforded.

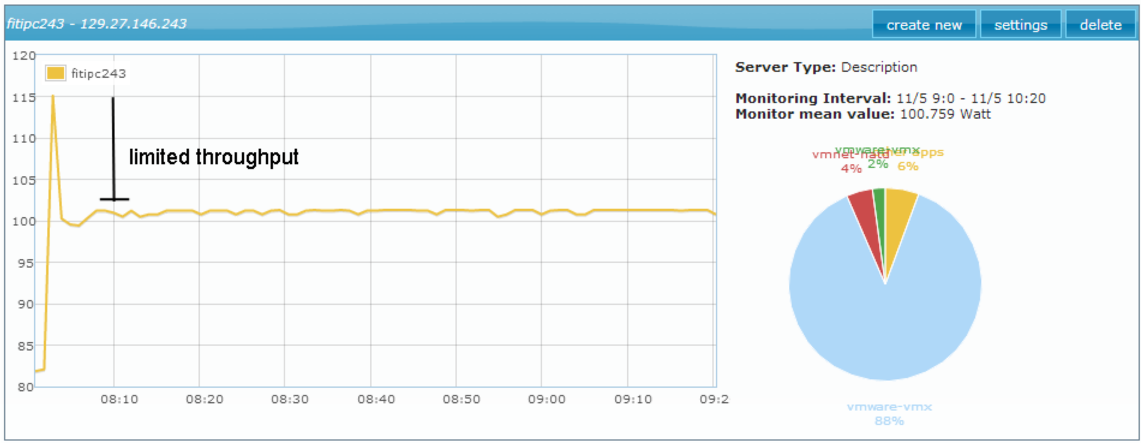
As mentioned before, not every type of workload can be processed on a virtual system more efficiently than on a physical one. Within the next section advantages and disadvantages of virtual machines arise by using a reference workload 2.

5.4.3 Reference Workload 2

In section 5.3.2, the workload 2 was used in order to force the physical server into its saturation. The workload consists of a test, using two separate vApus instances, which start a set with 5 cycles creating 64 concurrent users each. This results in 640 concurrent users to be processed in the virtual machine. Fig.5.12 shows the power monitor during the stress test.

The initial peak results from the creation of those concurrent users which is followed by a massive drop to a constant power level at 102 Watts. The test was expected to run two times (separate vApus Instances) by 5 cycles with 64 users each. Fig.5.12 shows a snippet of the first cycle. The test was stopped after this observation because the message is clear enough.
The hyper-visor seems to regulate the power consumption to a certain level which results in adding sequential processing time. The overall timespan to finish this workload was expected to be more than 5 hours and so the test was stopped after 90 min. As [Marcu and Tudor, 2011] outlined in their research, the virtual machine is limited in its throughput.

Table 5.5 shows the comparison between the physical machine and the VM.



Server Type: Description

Monitoring Interval: 11/5 9:0 - 11/5 10:20
Monitor mean value: 100.759 Watt

Figure 5.12: Server Power Analysis tool with the overall power profile of reference workload 2

	avg. Power usage / Watts	Duration / min
physical	149	42
virtual	102	undef.

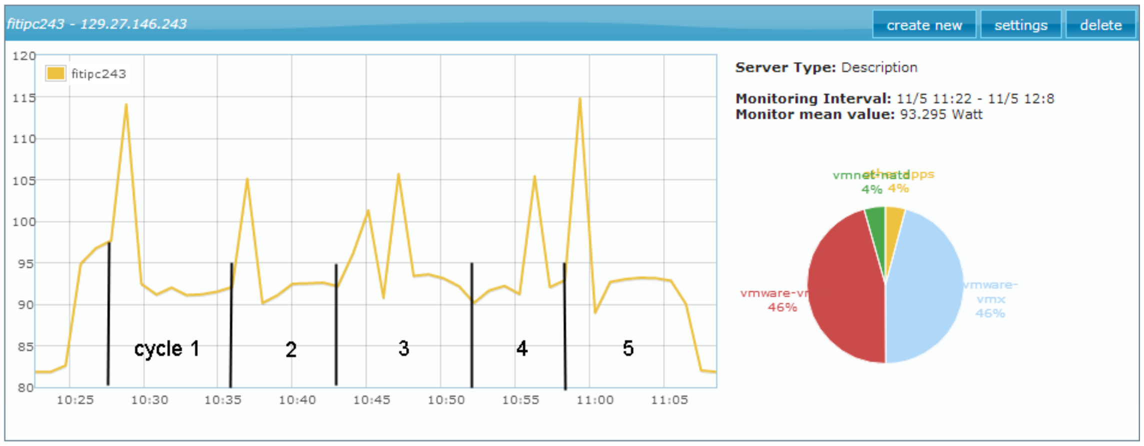
Table 5.5: Physical vs. virtual machine processing workload 2

This show case concludes, that the physical environment is able to handle resource intense workloads way better than a limited virtual machine. This result might arise because the server, which hosts the virtual machines, has less processing power than the physical one and therefore runs pretty early into saturation.
Within this section it is quite clear, that each server has its advantages and disadvantages. While a weak virtual server requires less energy, a strong physical machine offers a much better throughput on large workloads.

5.4.3.1 Using two virtual machines

In order to handle huge workloads on a weak server, virtualization helps. By using two different virtual machines, processing half of the workload each, parallel computing is achieved. Due to the hyper-visors limitation approach, a single machine is less efficient than multiple are. The following drawing 5.13 gives an idea how virtualization helps in this case.

The workload is still the same with two times (separate vApus instances) 5 cycles of 64 concurrent users. The only difference is seen in the processing approach. Whereas both vApus instances formerly stressed a single VM, now each instance puts the load on a separate VM. The overall power consumption of the server is still monitored with a single Papillon client running on the physical machine.
By analysing Fig.5.13, the pie chart shows the balanced workload, controlled by the hyper-visor. Each VM processes exactly the same workload.



Server Type: Description

Monitoring Interval: 11/5 11:22 - 11/5 12:8
Monitor mean value: 93.295 Watt

Figure 5.13: Two virtual machines processing workload 2

Table 5.6 shows the comparison between the physical server from section 5.3 and the virtual approach with the parallel processed workloads.

	idle usage / Watts	avg. usage / Watts	avg.-idle / Watts	duration / min	costs / euro
physical	111	149	38	42	0.0053
virtual	82	94	12	43	0.0017

Table 5.6: Reference workload 1 costs on physical and virtual server

Although the workload was high enough to force a strong physical server into its saturation, the weak virtual system is more efficient. Splitting the workload in order to do parallel processing is seen to be the right approach for this type of workloads.

The main reason, why the weak virtual server has a better energy efficiency is seen in the reference application itself. The provided php-forum might be limited in its throughput and so the physical server is forced to wait until the requests are processed.
In contrast the two virtual machines each have a php-forum installed which allows parallel processing. This fundamental observations help to detect weak points in application architectures based on fundamental power analysis.

5.5 Summary

In the evaluation part, two different environmental approaches have been compared. A strong equipped physical machine is compared with a weak virtual environment. In order to validate the observations from section 5.4, both environments should have exactly the same operating conditions. Due to limited resources, this could not be afforded. Although the physical server has got a better hardware than the virtual one, the virtual system works much more efficient with the given reference application. This confirms the suspicion, that the energy efficiency is based on the application architecture.

Using two different virtual machines and hosting an instance of the reference application each on one of them, is seen to be different than only hosting it on a physical machine. In order to establish the exact same application, both virtual machines would have to synchronize their database. This synchronization was not included in this evaluation and would use energy as well. But with respect to the tremendous amount of saved energy compared to the physical environment, this synchronization still might not consume a lot more energy. This suspicion could not be proofed in detail within this work, but states a field of interest for the future as well.

Chapter 6

Conclusion

Energy related issues are some of the most challenging aspects of cloud computing. This thesis focuses on developing a technique for determining the power consumption of applications, running in a virtualized environment under conditions that can be emulated and which are representative of the applications normal or typical workload.
The technique combines two fundamental technologies. Papillon, a software based power monitoring approach is used in order to measure the energy consumption of a server. vApus offers the possibility to emulate typical workloads on applications. Both used in conjunction form a unique test environment to determine an applications power consumption. By using a self developed server power analysis tool upon the Papillon REST API, valuable insights are presented in Chapter 5 of this work. Using real world workloads in order to simulate a cloud based type of application, brings the required credibility to this field of research. The comparison of the power consumption between a physical and a virtual system offers key observations for future studies.

The evaluated reference application from chapter 5 matches the requirements of a typical web-application. A web-service which processes a certain load of concurrent users is the average use case.
The main objective for virtualization is consolidation of servers. Studies have shown that 30% of servers are used 3% or less. In this low productive mode, servers are consuming 50-60% of their maximum power consumption. By taking the loads from several inactive servers and consolidating them onto one highly active server (e.g the load of 6 servers at 10% activity can be transferred onto one server at 60% activity) there is the possibility of decommissioning them to zero power and only marginally increasing the power of the single server. Tthis observation is from high value for future research projects. By analysing the power consumption of cloud application workloads, *smart hosting* could contribute to the overall approach of saving energy.

Due to the ongoing research in the field of green IT, interesting challenges are still to be done in order to make monitoring approaches even more accurate. By evaluating a wider range of server environments, with a bigger variety of different types of workloads, a better classification could be achieved. The following section outlines challenges and approaches to be taken in the future.

6.1 Future work

The field of server power analysis in data centers is a very active area. Some topics for consideration in the context of this thesis are the following.

6.1.1 Application identification

As mentioned in chapter 4.1, application identification has a huge potential for future research. Therefore the Papillon monitoring approach has to provide much more details on processes. Currently, only the top three processes are retrieved every minute which is not sufficient enough in order to identify an application. While certain workloads stress the memory of a server, others mainly use the network I/O for communication.
Once an application is identified by its process tree, the exact power consumption can be determined. The major advantage is seen in the improvement of smart hosting. *Smart Hosting* takes the type of application and their average power consumption into account and puts only those together on a server which use the server components most efficiently.

6.1.2 Power modelling a virtual server

The power modelling of a physical server is solved and requires a physical hardware meter to be connected to the machine itself. By stress testing the server with certain benchmarks, accurate power models are created for certain types of servers. Modelling a virtual machine was seen to be much more complex than expected. Depending on the number of started virtual machines, the power-model might be influenced by other VMs that are also resident on the server. Within this scope of the thesis only the physical machine itself delivers valuable power information due to a valid generated power model.
The huge drawback with not having matching power models for each single virtual machine itself is seen in the investigation in TCO and ROI. For the future, a detailed power model for a virtual machine will be crucial in order to form a powerful monitoring solution with Papillon.

6.1.3 Power Model accuracy

The accuracy of the Papillon power monitoring approach depends on the underlying hardware power models. Depending on the variety of benchmarks executed during the power model generation, the system is more or less accurate. In certain cases, the benchmarks do not reach out to a servers performance maximum. Those edge cases might cause a

certain percentage of inaccuracy. Benchmarks often create less complex workloads than real world applications do. As a result, particular edge cases can not be monitored to its full accuracy.
In the future, the power modelling approach could be adapted in a manner that real world applications are used in addition to the benchmarks. The power model might include a self learning algorithm which increases the quality over and over. Rather than just create a single model initially at the beginning of the Papillon system installation, the models could be updated regularly.

6.1.4 Server Power monitoring

Within the scope of this thesis, a sample reference application was chosen in order to create a reference workload for the evaluation part. The reference workload only covers a few types of real world applications. Future studies could put the focus on the evaluation of different types of sample applications.
In addition the diversity of virtual supervisor platforms offers a large field of research as well. Whereas certain supervisors support CPU dominant applications, others might handle network dominant ones more efficiently. Once the best runtime environment for certain application types can be determined, new opportunities in the field of DCIM (Data Centre Infrastructure Management) would arise.

Bibliography

1e [2011]. *Capacity Planning for NightWatchman Management Center*. http://www.1e.com/.

2006, VMware [2006]. *Reducing Server Total Cost of Ownership with VMware Virtualization Software*.

An, T. S. [2011]. *GETTING STARTED WITH DATA CENTER INFRASTRUCTURE MANAGEMENT (DCIM) Conclusion and Parting Advice, (Dcim).* http://www.nlyte.com/german/cat_view/95-englishdownloads/101-white-papers.

Belady, Microsoft [2010a]. *(CUE): A Green Grid Data Center Sustainability Metric*. http://www.thegreengrid.org/~/media/WhitePapers/CarbonUsageEffectivenessWhitePaper20101202.ashx.

Belady, Microsoft [2010b]. *Green Grid Data Center Power Efficiency Metrics: PUE and DCIE*. http://www.eni.com/green-data-center/it_IT/static/pdf/Green_Grid_DC.pdf.

Blackburn, 1E [2010]. *The Green Grid Data Center Compute Efficiency Metric: DCcE*. http://www.thegreengrid.org/~/media/WhitePapers/DCcE_White_Paper_Final.pdf?lang=en.

Carr, Nicholas [2009]. *The Big Switch: Rewiring the World, from Edison to Google*. W.W. Norton & Company. ISBN 0393333949, 9780393333947.

Chuan, Chen, Fan Ying, and Liu Lina [2012]. *A new live virtual machine migration strategy*. pages 173–176.

Cisco [2010]. *Managing the Real Cost of On-Demand Enterprise Cloud Services with Chargeback Models*. pages 1–10.

Cook, Greenpeace International [2012]. *How clean is Your Cloud?* http://www.greenpeace.org/international/Global/international/publications/climate/2012/iCoal/HowCleanisYourCloud.pdf.

Cook, Jodie Van Horn [2011]. *How dirty is your data? A Look at the Energy Choices That Power Cloud Computing*. http://www.greenpeace.org/international/Global/international/publications/climate/2011/Cool%20IT/dirty-data-report-greenpeace.pdf.

Da Costa, Georges and Helmut Hlavacs [2010]. *Methodology of measurement for energy consumption of applications. 2010 11th IEEE/ACM International Conference on Grid Computing*, pages 290–297. doi:10.1109/GRID.2010.5697987. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5697987.

DeGelas, Johan [2010a]. *Quad Xeon 7500, the Best Virtualized Datacenter Building Block?* http://www.anandtech.com/print/3846.

DeGelas, Johan [2010b]. *vApus Fos installation guide*. http://www.sizingservers.be/en.

Erek, Pröhl, Drenkelfort [2013]. *IKT-Performance Measurement Systeme*. http://opus.kobv.de/tuberlin/volltexte/2013/3927/pdf/9783798325210_content.pdf.

Gmach, D., J. Rolia, and L. Cherkasova [2011]. *Chargeback model for resource pools in the cloud*. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pages 622 –625. doi:10.1109/INM.2011.5990586.

Google [2013]. *Big Picture*. http://www.google.com/green/bigpicture/.

Group, The Climate [2008]. *SMART 2020: Enabling the low carbon economy in the information age*. http://www.smart2020.org/_assets/files/02_Smart2020Report.pdf.

Grove, Ralph F [2011]. *THE MVC-WEB DESIGN PATTERN*. http://grove.cs.jmu.edu/groverf/papers/WEBIST2011.pdf.

Jin, Yichao, Yonggang Wen, and Qinghua Chen [2012]. *Energy efficiency and server virtualization in data centers: An empirical investigation*. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 133 –138. doi:10.1109/INFCOMW.2012.6193474.

JouleX [2012]. http://www.joulex.net/.

Keilen, Steve [2010]. *Improving Data Center Energy Efficiency*. http://www.viridity.com/improving-data-center-energy-efficiency/.

KENT, SU [2011]. *Agents or Agentless?* http://www.1e.com/it-efficiency/software/nightwatchman-enterprise-pc-power-management/.

Kumar, Rakesh [2012]. *The Six Triggers for Using Data Center Infrastructure Management Tools*. http://www.gartner.com/id=1937116.

Liu, Lu, Osama Masfary, and Jianxin Li [2011]. *Evaluation of server virtualization technologies for Green IT*. *Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE)*, (Sose), pages 79–84. doi:10.1109/SOSE.2011.6139095. `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6139095`.

Loeffler, Chris [2009]. *Is your data center ready for virtualization?* `http://www.techrepublic.com/whitepapers/is-your-data-center-ready-for-virtualization/958395`.

M., D. [2011]. *The nlyte data centre infrastructure management maturity model*. `http://www.nlyte.com/analyst-reports/cat_view/129-campaigns/163-emea`.

Marcu, M. and D. Tudor [2011]. *Power consumption measurements of virtual machines*. *2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 445–449. doi:10.1109/SACI.2011.5873044. `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5873044`.

Oracle [2010]. *Strategies for Solving the Datacenter Space , Power , and Cooling Crunch : Sun Server and Storage Optimization Techniques*. (March).

Patterson, Intel [2011]. *Water Usage Effectiveness (WUE)*. `http://www.thegreengrid.org/~/media/WhitePapers/WUE`.

Polimeni, Mario Giampietro Blake Alcott, Kozo Mayumi [2008]. *The Jevons Paradox and the Myth of Resource Efficiency improvements*. Earthscan.

Saunders, H.D. [1992]. *The Khazzoom-Brookes postulate and neoclassical growth*. The Energy Journal.

Sun, Yifeng, Yingwei Luo, Xiaolin Wang, Zhenlin Wang, Binbin Zhang, Haogang Chen, and Xiaoming Li [2009]. *Fast Live Cloning of Virtual Machine Based on Xen*. *2009 11th IEEE International Conference on High Performance Computing and Communications*, pages 392–399. doi:10.1109/HPCC.2009.97. `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5167019`.

Vandroemme, Dieter [2010]. *vApus V2*. `http://www.sizingservers.be/en`.

Wikipedia [2011]. *Sigar library*. `http://en.wikipedia.org/wiki/Sigar_(software)`.