Maynooth University
National University of Ireland Maynooth

Topics in Power Usage in Network Services

A dissertation submitted for the degree of
Doctor of Philosophy

By:

Karl James O'Dwyer

Under the supervision of:

Dr. David Malone

Hamilton Institute
National University of Ireland Maynooth
Ollscoil na hÉireann, Má Nuad

February 2017

In memory of

Denis O'Dwyer
&
Carmel Carroll

Abstract

The rapid advance of computing technology has created a world powered by millions of computers. Often these computers are idly consuming energy unnecessarily in spite of all the efforts of hardware manufacturers. This thesis examines proposals to determine when to power down computers without negatively impacting on the service they are used to deliver, compares and contrasts the efficiency of virtualisation with containerisation, and investigates the energy efficiency of the popular cryptocurrency Bitcoin.

We begin by examining the current corpus of literature and defining the key terms we need to proceed.

Then we propose a technique for improving the energy consumption of servers by moving them into a sleep state and employing a low powered device to act as a proxy in its place.

After this we move on to investigate the energy efficiency of virtualisation and compare the energy efficiency of two of the most common means used to do this.

Moving on from this we look at the cryptocurrency Bitcoin. We consider the energy consumption of bitcoin mining and if this compared with the value of bitcoin makes this profitable.

Finally we conclude by summarising the results and findings of this thesis.

This work increases our understanding of some of the challenges of energy efficient computation as well as proposing novel mechanisms to save energy.

Contents

List of Figures

List of Tables

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisor Dr. David Malone for the continuous support of my PhD studies and related research, for his patience, motivation, and immense knowledge. His guidance helped me immensely research and writing of this thesis. I could not have imagined having a better advisor and mentor.

Besides my advisor, I would like to thank my thesis committee: Prof. Christine Griffin, Dr. Sasitharan Balasubramaniam, and Dr. Charles Markham, for their insightful comments and examination of this thesis.

My sincere thanks also goes to Dr. Oliver Mason, Prof. Douglas Leith, and Prof. Ken Duffy, for the support, guidance and for the many stimulating debates over lunch.

I would also like to thank all of my colleagues from the Hamilton Institute, it was a privilege to work with so many wonderful people on a daily basis. I would especially like to thank Rosemary Hunt and Kate Moriarty for all of their help, guidance and for helping me to cope with the administrative/bureaucratic workload.

From the bottom of my heart I must also thank my family for their support throughout all of my crazy endeavours, and for their often misplaced belief in me. Similarly I must also thank all of my friends who supported me through this, it wasn't easy, so I am especially grateful for your patience and understanding throughout this.

Finally there are a few people who requested special acknowledgements: I'd like to acknowledge the contribution of Jennifer Keane, without whom this thesis might have been several pages longer.

. . . and Dr. Davy Kavanagh is cool too I guess.

Declaration

I hereby declare that I have produced this manuscript without the prohibited assistance of any third parties and without making use of aids other than those specified.

The thesis work was conducted from January 2012 to February 2017 under the supervision of Dr. David Malone in the Hamilton Institute, National University of Ireland Maynooth. This work was supported by TGI and funded by the Higher Educational Authority. The programme for research in Third level Institutions Cycle 5 is co-funded by the Irish Government and the European Union under Ireland's EU Structural Funds Programme 2007-2013.

Maynooth, Ireland,

28^{th} of February 2017

CHAPTER 1

Introduction

In this chapter, we discuss the motivations behind the work of this thesis and provide an overview of the material presented in the following chapters.

1.1 Motivation

The computer was, for the longest time, a person equipped with some sort of writing tool. They spent days performing calculations for all manner of tasks such as calculating how much tax was owed and when to observe Easter. This concept began to change in the 19^{th} and 20^{th} centuries. The creation of a new branch of mathematics along with advances in materials science and electronics created a new type of computer. The piles of paper, pencils, abacuses and slide rules slowly faded into obscurity and the digital computer leaped into sharp focus.

By today's standards these machines would not be considered impressive – they were huge – unwieldy to operate and often application specific, but they were relentlessly hard workers. Over the years they were improved upon and generalised to the point where computers are now ubiquitous. It's hard to find one aspect of our lives that isn't somehow influenced in some way by the output of a computer program.

But this isn't without a downside, the march towards computerisation has traditionally neglected the efficient use of energy. Until recently this wasn't a

serious concern. In 2005 Information Technology alone used 7.8% of all the electricity consumed in the EU, and this in turn was responsible for 1.9% of all Carbon Dioxide (CO_2) emissions [145]. In response to this growing concern a new field has emerged: Green IT.

Green IT is focused on making information technology that is environmentally sound as well as trying to reduce the negative impact of existing technologies. This includes building computers out of less hazardous materials, making computers have longer usable lifespans, and making them more energy efficient to better suit that longer lifetime. These are huge challenges which many researchers in both academia and industry are working hard on.

In this thesis we will look at topics concerning the energy efficiency of network services and the software that enables these services.

1.2 Overview

This thesis is organised as follows. In chapter 2 we present a literature review covering many key areas, topics and problems in Green IT.

Chapter 3 defines the key principals of electricity and how to measure it, as well as looking at different tools we used to measure electrical power consumption for this thesis.

In chapter 4 we explore techniques for energy saving that use low powered devices as a proxy to allow the proxied service to be scaled down or shut off completely while still presenting as available. Our proposed solution demonstrated an energy saving of up to 20% with only a minor impact on end users.

In chapter 5 we compare the energy efficiency of running applications in a Virtual Machine, a Linux Container and on a physical host. We discovered that this isn't as straightforward a proposition as first imagined, but this opens the door to further research around the energy efficiency of virtualisation.

In chapter 6 we look at the emerging phenomena of Bitcoin and investigate its relationship with power consumption. We discover that the profitability of bitcoin mining is restricted by the energy efficiency of the hardware.

We conclude with chapter 7 which briefly presents the conclusions and the final

thoughts of this thesis. In general we will introduce technical terms before they are used but we also provide a glossary of terms.

1.3 Publications

The research for this thesis has resulted in the following papers, two of which have appeared at international conferences and another two which are under consideration for publication.

Karl J. O'Dwyer, Eoin Creedon, Mark Purcell, and David Malone. Power saving for web servers using proxies. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2013*, pages 1–5, October 2013. This paper is a preliminary version of the work presented in chapter 4.

Karl J. O'Dwyer and David Malone. A comparison of the energy efficiency of virtual machines and linux containers. This paper is yet to be published but appears in chapter 5.

Karl J. O'Dwyer and David Malone. Improved power saving for web servers using proxies. This paper is yet to be published but appears in chapter 4.

Karl J. O'Dwyer and David Malone. Bitcoin mining and its energy footprint. In *Irish Signals Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014). 25th IET*, pages 280–285, June 2014. The contents of this paper appear in chapter 6 albeit with some minor corrections and updates.

CHAPTER 2

Literature Review

In this chapter we present the current state of Green IT. We look at the high level motivations before looking briefly at current state of the area. We then look at pressing issues in various areas and we conclude by showing how the work presented in this thesis contributes to the area.

2.1 Introduction

In an effort to increase economic efficiency and reduce the emission of green house gases, the European Commission has said that [53]:

> Information and Communication Technologies have an important role to play in reducing the energy intensity and increasing the energy efficiency of the economy, in other words, in reducing emissions and contributing to sustainable growth.

Here the Commission is referring to the potential transformative power of Information Technology to improve the efficiency of the economy. But it has long been known that IT also is a contributor to green house gas emissions. In 2005, it was estimated that 7.8% of the electricity used in the EU was used to power IT equipment in all its forms and that this consumption was responsible for 1.9% of the EU's total CO_2 emissions [145]. The same report has projected

that without improvement, IT would consume 10.5% of all electricity generated in the EU by 2020 and would be responsible for 4.2% of CO_2 emissions.

The energy consumption of computers and their efficient use of this energy has become an increasingly important topic in both academia and industry. The impact of IT has been well explored [159], but many competing approaches still exist to reduce its negative environmental impact. Traditionally this has been the concern of hardware manufacturers but increasingly software's impact on energy efficiency is being explored [86]. The nature of open source software has allowed researchers to explore what optimisations if any, developers are using in their code for energy efficiency [115].

It also has been stated that focusing on improving energy efficiency doesn't lead to a reduction in green house gas emissions and it would be more useful to focus directly on the latter [62]. The 19^{th} century economist William Stanley Jevons may have agreed with this after he noticed that the improved energy efficiency of Watt's steam engine lead to huge increase in coal consumption [131]. This paradox is observable in many areas and render simple gains in energy efficiency ultimately self-defeating unless paired with suitable conservation policies as demonstrated by Freire-González et al [72].

The rest of this chapter will focus on topics in this area and will devote a section to each.

2.2 Green IT

There are many definitions of what constitutes Green IT, but for the purposes of this thesis we define it as: any attempt to mitigate or offset the impact of IT on the environment. This includes, but is not limited to, reducing energy consumption, better recycling of electronic waste and better planning of required computing capacity.

There are many innovative examples of Green IT demonstrated in the literature. One example of this is environmentally opportunistic computing, where waste heat from a data centre is recycled to heat a building, or in the case of [165], it is a green house which is next to the data centre.

There have been two major waves in Green IT according to [81]. The first wave focused on the data centre whereas the second wave seems to shifted

focus onto providing sustainable services to business. These broad ideas are now being taught to graduate students as part of planning green IT strategies in organisations [166]. These practises are already starting to be employed in several institutes of higher education in the UK and are starting to achieve critical evaluation [44].

In the subsections that follow we will look at the role of end users in section 2.2.1 and in section 2.2.2 we explore the idea of making the Operating System more responsible for power management.

2.2.1 Green IT and End Users

End users' behaviour plays a part in energy usage. How to address this is an interesting question. Getting end users to think about the environmental impact of their hardware and their choice of Internet based service is hard, [137] looked at the behaviours of students in a Malaysian university and showed that while green services are popular, more traditional power management is not very well understood by their students. A longitudinal study of 83 users over 48 weeks [174], found that giving users measurement based feedback on their energy consumption is not enough and shows that a reward system might be able to help encourage users to improve their behaviour.

Another approach to this might be to take care of this on the users' behalf. In busy office environments it is often not possible to put users PC's to sleep as their computer still needs to maintain some network presence or perform some background tasks. Replacing their desktop machine with a virtual machine and consolidating when they aren't in use is an interesting proposal [32]. Unfortunately migrating multiple virtual machines can lead to network congestion. This work proposes a unique approach of partial virtual machine migration, which only migrates the working set, metadata and required pages of these idle virtual machines, placing the desktop PC into a low power mode until it needs to access new pages in its memory. These migrate back to the desktop when the user resumes their activities.

2.2.2 Green OS level Improvements

Making the efficient use of energy a priority in the operating system is another approach which has some merit for saving energy. Technologies to make

hardware more energy efficient have been created and are increasingly more commonplace in consumer grade computer systems. But these technologies don't come with guidelines for best practice or effective usage.

Dynamic Voltage and Frequency Scaling (DVFS) is a technology found in modern Central Processing Units (CPUs) which regulates the voltage of the CPU and its operating frequency. Picking the appropriate CPU frequency is usually controlled by part of the operating system. In [143] the authors developed a CPU frequency governor which is more tightly coupled with the operating system scheduler and alters the CPU frequency in line with scheduled events. They implement this idea fully in the Linux kernel and report power savings of up to 10%.

Of course to better use DVFS it is important to understand the relationship between the operating temperature of the CPU and DVFS [79]. This has been extended by [27] which uses a grey box approach, where they have some limited knowledge of the behaviour to identify a suitable thermal model for multi-core CPUs. Understanding the relationship between load and temperature can also be used for load balancing in such a way to lessen the need for data centre cooling [65]. It should also be noted that using DVFS can use more energy and reduce throughput in CPU bound tasks, but in I/O bound tasks it can be beneficial [58, 38].

Wake-on-LAN (WoL) is an industry standard for remotely powering on computers, but it can use be used to trigger computers to wake from suspend modes. The existence of these two technologies together present an opportunity to create services which can manage their own power consumption by powering off when they are no longer needed. An excellent example of this idea is PowerNap, an approach to reducing server power utilisation. In this scheme servers switch quickly between a low power idle state and a full power mode to serve requests. [111] A theoretical approach outlined in [87] proposes a sleeping scheme which employs a queue for requests and proves that an optimal sleeping policy has a simple structure in terms of hysteretics.

To make these type of decisions it is important to have actionable information, [83] looked at improving power management in data centres used for research and development. They develop a data acquisition framework to enable algorithms to determine when to power machines down without impacting

ongoing work.

2.3 Green Clouds

Cloud Computing is an emerging field with a somewhat loose definition, for our purposes we will try to stick to the definitions and characteristics outlined by [76, 112], which define cloud computing as:

> a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

The authors also define the models of cloud computing service as Software as a service, Platform as a service and Infrastructure as a service.

The Cloud poses an interesting set of problems and opportunities. The computing infrastructure required to both compute and store data as well as the infrastructure required to support these operations (air conditioning, power distribution, emergency electricity generators, etc) present opportunities to gain efficiencies. Many areas are singled out as a potential area for improvement [126], but ensuring that these resources are used effectively and efficiently is a great challenge.

To this end, the better use and scheduling and the allocation of resources is a popular topic in the literature. Virtual Machines in the cloud are singled out for a lot of attention. Likewise, data centres themselves are also the subject of considerable interest. Each of these topics will be discussed in the following subsections.

2.3.1 Resource Allocation

Provisioning resources is an important topic for the providers of cloud services. Most approaches to provisioning focus on resource efficiency or cost efficiency and do not explicitly focus on energy-efficiency.

Designing software stacks that manage resources isn't necessarily a popular topic of research, but some authors are putting in the effort and developing new

ideas and systems. For example *sCloud* is a system for workload placement and migration across data centres in different geographic locations to exploit better green energy availability [48]. There is also the *GreenCloud* Architecture, which is aimed at reducing power consumption in data centres that host virtual machines. The proposed architecture enables live monitoring, virtual machine migration and optimised virtual machine placement [104]. In addition [167] proposes *Green Master* which is a system that aims to strike a balance between load and power management, [102] considers extending this work by adding network bandwidth control to it.

In [52] the authors work towards a green resource manager for the Infrastructure as a service (IaaS) platform OpenStack. They develop a monitoring tool to enable the operator to view the energy consumption of their infrastructure. This allowed them to do an analysis of live migration strategies and identified how to use these in a load manager. A contrasting approach in [113] used workload predictions from web traffic and the authors put forward a scheme which dynamically alters the resources allocated to a service in a data centre. This takes Service Level Agreements (SLAs) into consideration and finds, as might be expected, that the more stringent the requirements of the SLA, the less scope to save energy.

An interesting approach is demonstrated in [107] which implements a system called flex to aid distributed schedulers by deploying customised storage systems for a big data computing job before it is scheduled and to consolidate these jobs to allow for better resource utilisation.

Energy-Proportional Computing is when a computers total power consumption is a linear function of its utilisation [24]. One effort to achieve Energy-Proportional Computing is to use groups of small, low powered computing devices. These are called Small or Wimpy Nodes. This was first proposed by [142], but was explored in depth by [105]. In this paper big data workloads on clusters of wimpy and non-wimpy nodes, while I/O intensive workloads were more energy efficient on the non-wimpy nodes, processing database queries were more energy-efficient on the wimpy-nodes but with a slightly lower throughput.

2.3.2 Virtual Machines and the Cloud

Virtual Machines are a topic of considerable interest. They trace their roots back to the 1960s and the days of time sharing systems, but have evolved over time to suit new ideas and demands. Modern virtual machines come in many different flavours, the most popular are used to provide a layer of software which emulate physical hardware and allows for the installation of an operating system to run a users programs. This has many advantages.

The ability to move a running virtual machine from one physical host to another known as live migration opens a world of possibilities. Live migration is commonly employed as a means to redistribute workloads across a data centre, or even between data centres, based demand or the availability of resources.

Often the energy cost of using live migration is either not considered at all or is given a fixed cost. This has been shown to be insufficient, this was first modelled by [103] and later demonstrated empirically [6, 141]. The energy consumed by a live migration is non-trivial and the type of workload of the virtual machine has an impact on this cost.

Another study which looked at virtual machine migration in Kernel-based Virtual Machine (KVM) [141]. It found that the workload of the virtual machine has no bearing on the source server but it does impact the energy consumption of the destination server and that based on the size of the virtual machine and the network bandwidth available, migration uses a non-negligible energy overhead and migration latency on both source and destination servers. This work was further extended by [152] to produce a lightweight model to estimate the cost of a virtual machine migration.

Assigning virtual machines to host machines is a problem of increasing importance in IaaS environments. There have been many proposals as to how to best achieve this. Some have put forward techniques like evolutionary algorithms and multi-objective optimisation [61], stochastic bin packing [37], using look-ahead control [96] or modelling the problem as an exercise in profit maximisation [146]. This problem may not have an optimal solution, but novel algorithms are still being proposed which offer new insights. For example [140] investigates whether improving the application running on a virtual machine

is better than migrating to a different machine. It is important to note that while they don't combine these ideas, they demonstrate that both approaches are useful in different scenarios.

The task assignment problem is related to the virtual machine assignment problem, where work is assigned to running virtual machines with a goal of best using current resources. This problem also has garnered some attention [18].

We'll review works considering the task assignment problem, tuning the resources that each virtual machine requires next. The idea being that virtual machines which are given their optimal resources will preform their tasks as well as allowing for better overall resource usage.

Typically, proposed solutions to this involve monitoring the workload of virtual machines and their resource utilisation. One such system called *TARGO* is aimed at users of Infrastructure as a service systems and optimises virtual machines based on user supplied rules, observation of utilisation and workload [70]. *GreenCloud* [104] is a similar proposal aimed at reducing power consumption in data centres that host virtual machines. Another variation on this theme is called *GARL* and employs a genetic algorithm and reinforcement learning to achieve similar goals [77].

An interesting approach is demonstrated in [29] which casts the operators desired profit, the cost of running the data centres infrastructure, and the tasks to execute, as a mathematical programming problem and uses a solver to derive a near optimal schedule. Unknown values in the model, such as a task level SLA are learnt through machine learning. This enables a scheduler to allocate tasks to hosts and strike a balance between cost of energy, Quality of Service (QoS) and operator revenue.

A common application hosted in the cloud are Relational Database Management Systems (RDMSs) like MySQL or PostgreSQL. The authors of [123] propose an optimal solution for resource allocation in a specific RDMS use case. In a similar vain, main memory databases are an emerging technology but are normally optimised for bare metal and as such do not run very efficiently in a virtual environment, [144] proposes an approach where the Database Management System (DBMS) can communicate its resource requirements.

This allows a global controller to better manage resources and live migrate the VM running the DBMS to improve its resource efficiency.

One problem with hosting multiple virtual machines on a single host is the hidden contention for resources between virtual machines, [176] proposes a system to manage virtual machines which takes this into account, therefore allowing for higher SLAs and more efficient resource usage.

Another solution might be to pin guest operating systems to CPUs. The authors of [130] looked at how pinning workloads to specific CPUs and using partially loaded CPUs impacts on performance and energy efficiency. They found systems that have light background loads could benefit from per-thread (guest operating systems share CPU cores) pinning leading to better resource utilisation, whereas heavily loaded systems could benefit from per-chip (one guest per NUMA node) pinning which leads to better CPU resource isolation, which in turn offers more stable CPU performance. The authors also found that in co-location scenarios with partial CPU utilisation a well picked CPU pinning configuration can improve the energy efficiency.

There are a number of articles looking at data centre networks and how three new technologies that have emerged around virtual machines have impacted on these networks. For example [26] looks at virtual network bridging, live virtual machine migration and multipath forwarding protocols. In particular, they are interested in how this impacts on the common network optimisation goals of traffic engineering and energy efficiency. This isn't the only concern about virtual machines and networking; [169] looks at how the energy consumption varies according to traffic patterns and CPU affinity strategies in the virtualised cloud environment. This paper also shows that virtual machines generate more load on the CPU for networking related tasks than when running on real hardware.

Better understanding the energy usage of virtual machines is important. Studies which measure, model and evaluate the power consumption of virtual machines show promise [127, 100]. This was taken further by [161] which outlines a methodology for modelling a virtualised data centre running a business process. The model enables the collection of simulated data about the process to enable reasoning about QoS and energy efficiency.

Request batching is a technique which can be used to save energy, but it is difficult to extend to virtual machines. [162] proposes a technique for batching requests for virtual machines. Combining request batching and DVFS to reduce energy consumption while still meeting targets set by an SLA, [47] designs a two stage expert fuzzy logic controller to scale CPU frequency and tune request batching. It may be possible to combine these two approaches.

2.3.3 A Closer Look the Data Centre

The data centre itself is a key part of the equation. There is often a variety of competing problems found in this domain.

The use of electricity in the data centre is often a major concern. Operators are billed for two items: their overall consumption of electricity and their peak demand for electricity. Reducing this peak demand is seen as a means to gain efficiency.

Many schemes to reduce peak demand rely on some characterisation of the power demands of data centres, but data about this appears to be scarce. The authors of [91], try to address this, they measured the power consumption of six Microsoft data centres over six months and presents some in-depth analysis. The authors of [171] took this further by studying the contracts between electrical utility companies and data centre operators, they find that peak demand charge is a major component of the total cost of running a data centre, more importantly they then study partial execution as a mechanism to reduce the peak power demand thus reducing the peak demand charge and the overall energy cost of the data centre.

With these more recent works it is possible to better evaluate approaches like [45] which puts forward the MUSE framework for modelling data centres and uses it to implement a resource management system. In a similar vain [170] proposes an energy efficient framework for resource allocation in the cloud called Anchor. It is also possible to take a different approach like using look-ahead control to reduce energy usage [96].

To better meet the myriad of demands placed on the data centre, Green SLAs are proposed to help operators reduce their environmental impact while meeting the concerns and needs of their costumers [80]. This idea has been expanded on and [16] looks at ways operators can meet their targets.

One interesting approach to reduce the carbon footprint of a provider is to have multiple data centres located to take advantage of sources of renewable energy. Using virtualisation it's possible to migrate virtual machines and their workloads between data centres to best take advantage of available renewable energy. This approach is often called *"follow-the-renewables"*, and was employed in the GreenStar network in Canada [99]. The practicalities of building data centres for follow-the-renewables schemes were explored by [30], they propose a framework, an optimisation problem and an approach for a solution to the problem of where to locate green data centres as well as implement a system to migrate virtual machines to follow the availability of renewable energy.

2.4 General Application Energy Modelling

To build more energy efficient software it's important to be able to model the energy consumption of applications, both to eliminate the prohibitive costs of attaching measuring equipment to every software developers machine, but also to enable developers to reason abstractly about the impact their code is having on the energy usage of the system.

In the case where the developer has energy measurement instrumentation, [120] develops a toolkit and an Application Programming Interface (API) to estimate the energy usage of an application. In the case where the developer doesn't have access to this specialist hardware, [5] demonstrates tools which are based on an neural network which can estimate the energy cost of an application. The model is trained and validated against two real servers for a selection of workloads and models power for four subsystems (CPU, Memory, Disk and Network interfaces).

The CPU itself is the subject of particular interest in the literature, for example [164] designs a tool which takes accurate power measurements and characterises the power consumption of the CPUs instruction set. This tool is designed to be easily extended to include new architectures, enabling more efficient designs in the future. Other methodologies for power modelling CPUs include stochastic models [59]. Models of this type tend not to consider multi-core CPUs or their power saving mechanisms, but [25] proposes a new model which takes resource sharing and power saving into consideration.

Embedded devices with CPU–Graphics Processing Unit (GPU) architectures present many interesting opportunities and challenges. [49] looks at using a device equipped with an ARM CPU and a NVIDIA GPU and measures its energy efficiency and performance for a number of database benchmarks. The CPU did better in terms of performance and lower energy consumption when doing simple operations like selection and aggregation but the GPU performed better for sort and hash join operations in both performance and energy consumed.

Considering the possibility of future energy crises or natural disasters, [136] looks at the issues facing an architecture for the Internet which can operate in places or in a possible future with intermittent energy supply.

2.5 Mobile Devices and Energy Issues

The proliferation of smart phone applications has created a power management problem for users where some applications can create a significant load on the system and consume precious battery power. A working prototype of a system which logs system calls and records the power demands of system components is presented in [91]. The largest study of this kind which measured the energy usage of 1520 smartphones "in the wild"; from this data they put forward a new power model which combines utilisation and finite state machine models as well as a detailed analysis of where energy and CPU time are spent [46].

Mobile phone applications are becoming of increasing interest; one study on periodic transfers by mobile phone applications. This was the first large scale investigation on these transfers and tried to characterise the reasons for their use. The authors then investigate network strategies to mitigate their impact [134]. Another study looked at the energy cost of instant messaging using different mobile phone applications. They evaluate the cost of the notification that the other user is typing and put forward a message bundling scheme which reduces the energy consumption [160]. A similar study which looked at a number of different mobile applications which advertise the users' presence and finds that having these type of applications running on an otherwise idle device can drain the battery up to nine times faster. To reduce energy consumption and network costs they consider a low frequency and low volume two way push notification system [19]. Another study looked at data from 20 users over 5

months and found that the majority of radio energy is consumed when the screen is turned off and proposes that this be considered as part of a traffic optimisation scheme [82]. Applications that run while the screen is off are typically more delay tolerant as the user isn't interacting with the application. For example streaming a radio program using HTTP Live Streaming means that the file can be buffered well in advance and the device only needs to download enough at a time to stop the playback being interrupted.

2.6 Green Networks

If we ever hope to achieve truly sustainable computing then it is important to consider the networks which our computing devices use to communicate. A review of the current research in sustainable communications infrastructures [109], albeit focusing on the cloud, shows many interesting trends. We will explore some of these themes and more in this section.

A network management system which incorporates real-time assessments of energy efficiency is demonstrated in [54], network performance and network availability are also assessed in order to allow energy saving policies to be enforced which comply with high level business decisions to deal with the trade off between energy savings and network performance. Knowing the complete state of all links in a network is often infeasible in large or complicated network structures. So distributed network management is sometimes preferable, [31] outlines a set of distributed algorithms which set the sleep state of links based only on knowledge of the link's current load. This work has a similar goal to [43], which proposes a dynamic network management scheme to optimise networks which use Open Shortest Path First. They find that on average they can put 20% of nodes and 40% of links to sleep and not compromise network stability or performance.

Another question that appears in the literature is about the energy efficiency of network protocols, Transmission Control Protocol (TCP) in particular is the subject of a large amount of attention. The first in-depth investigation into this was [177], which looked at the energy efficiency of TCP for bulk data transfer in an environment where channel errors are correlated (in other words, a model of a wireless network). Other work in this general area looks at the power consumption of TCP [34], using TCP congestion control to improve

energy efficiency [39] & the authors of [125] investigate if there is a relationship between capacity scaling and TCP congestion control and if this interaction impacts on energy savings and QoS.

Comparisons between different implementations of TCP and different protocols are also made in some literature. [147] looks at the energy consumption of the TCP congestion avoidance algorithms Reno, New Reno and SACK in multi-hop wireless networks. The results show that one TCP version isn't necessarily better than the others for all use cases they explored. A similar study looks at the energy consumed by TCP, TCP Reno and Stream Control Transmission Protocol (SCTP) in a simulated cloud computing environment, using the green-cloud simulator and found that some energy consumption differences between TCP implementations are present, but aren't significant [94]. SCTP performs slightly better, but isn't a drop in replacement for TCP. Results consistent with these were presented in [35], which demonstrated experimentally that the software implementation is only a minor factor, but that the number of transitions between idle and active states in the CPU were a bigger factor and that this was dependent on the workload. The energy efficiency of WiMAX for long duration TCP connections under different WiMAX network parameters and using different scheduling algorithms was considered by [157]. This was examined by simulation and shows that using a best effort scheduler for long running TCP connections results in a lower energy efficiency per bit.

2.6.1 Green Mobile Networks

Mobile devices are also targeted in the literature, but these must be taken with a grain of salt as [128] demonstrates the pitfalls of energy efficiency studies. Particularly when the device draws energy from a battery. Many ideas and applications have been tested such as: using indoor base stations to compliment wireless access networks [155, 175]; examining the energy efficiency of a mobile device that is streaming video over WiFi under various conditions [156]; a new transmission protocol to stream video from mobile devices, using different transmission laws for different channel behaviour [106].

Modelling the power consumption of a network interface is especially important in mobile devices, [67] proposes a new power consumption model for 4G wireless networks. With such models in mind, [168] proposes an energy efficient Medium

Access Control (MAC) protocol for IEEE 802.11 networks, their simulations show better energy usage and throughput than the previous MAC.

An important subject is how the application layer impacts the energy usage and performance. This was examined in WiFi on the 5GHz ISM band. This experimental evaluation used varying packet sizes and transmission rates and evaluates energy consumed [153].

Cellular connections can be expensive in terms of energy, [118] investigates sharing connections them with nearby devices. This is shown to reduce the overall energy consumption of a group of devices. Poor signal strength is often suspected of increasing energy consumption in mobile devices, but its impact is poorly understood. [64] seeks to better understand this by analysing data collected from 3785 smartphone users and quantifies the extra energy consumed while the device is experiencing poor signal strength. The authors go on to model this and explore the idea of opportunistically delaying network traffic to save energy.

The structure of mobile networks also impacts the energy efficiency of end users. While studying the impact of middle boxes in mobile networks, it has been found that certain Network Address Translation (NAT) policies impact on the energy used by end users [163]. Reducing the round trip time is one way to reduce the usage of radio interfaces in wireless devices; [148] proposes to save energy in mobile devices by using a proxy to speed up delivery of web pages.

Radio bundling (using multiple interfaces to transfer data simultaneously) might be useful to reduce overall energy usage [119]. The authors evaluated different approaches and found that there is a need for an energy-aware bundling protocol to achieve an acceptable trade-off between performance and power consumption. To this end the authors of [93] put forward an energy model for radio bundling which incorporates variation in packet intervals, packet length and channel quality on a per packet basis.

Multipath Transmission Control Protocol (MPTCP) offers an interesting opportunity to build more resilient networks and could use multiple network interfaces on devices which offer them. This unfortunately is in itself not very energy efficient, [101] creates an energy model for MPTCP uses these findings

to design and implement an energy efficient MPTCP. MPTCP may also pose problems for congestion control, [97] proposes a congestion control algorithm for MPTCP which is energy aware.

2.6.1.1 Wireless Sensor Networks

Wireless Sensor Networks (WSN) are required to be energy efficient due to the constraints of their hardware. In fact a large portion of the research in that field is dedicated to it [172]. Rather than go in depth in this subject it may be helpful to look at some of the research which is also applicable in other areas.

A common technique used in WSN is sleep mode, used to extend the lifetime of the sensor network. [51] looks at how this impacts the lifetime of the network hardware.

Another related trade off is between maximising network lifetime by minimising the load on each device or balancing the load evenly across all nodes. A new approach being put forward is a distributed protocol which dynamically reorganises the network to reduce load on network bottlenecks which in turn should slow the energy depletion of these nodes [60].

2.7 Conclusion

It could be argued that the small percent of energy used to power the world's information technology is being used better than in other sectors and that efforts to improve this are not worthy of the amount of attention that it has received. But these efforts are not in vain, as it can be argued that information technology has changed many aspects of modern life and reduced our reliance on technologies which are less friendly for the environment.

But most important is the power that IT technology brings to help us make more informed decisions about our environment. If we are to use IT to help sustain the environment, we must ensure that our use of IT produces the smallest negative impact possible.

As measurement is an important precursor in almost everything we've seen in this chapter, it is vital that before we consider anything we first must look at how we measure power consumption and what tools are available to us to do this. Chapter 3 addresses this requirement.

Inspired by the work of others using low powered or *wimpy* devices as a substitute for larger more powerful systems and innovative use of sleep strategies in WSN we researched using low power devices as proxies that eventually became chapter 4.

Seeing the vast amount of work that is dedicated to virtual machines, we decided to investigate an emerging technology which is threatening the position of virtual machines in some fields, Containers. Chapter 5 looks at these with a view to understanding their energy efficiency.

It is also important to investigate new and emerging technologies carefully. As these may become the new standard of tomorrow or the legacy systems of the day after. With this in mind, in chapter 6 we turned our attention to Bitcoin, a peer to peer currency which employs a computationally heavy proof of work scheme to verify transactions.

CHAPTER 3

Measuring Power Consumption

Much of the work in this thesis required us to measure the power consumption of hardware. This chapter outlines how to measure electrical power consumption and explains what tools we used in our experiments.

3.1 Introduction

To understand how to measure power consumption we must first understand what is power, what is energy and how they are different. Our aim is to give a clear and concise definition while not turning this thesis into a physics text book.

Getting to grips with energy and power means we must first define work, work is exerting force over a distance. This can be rolling a boulder up a hill, pulling a plough across a field, or pushing electrons through a strand of copper wire. Energy is what it takes to do this work. If we know the amount of force we need to exert to move something and how far we need to move it, we can work out the energy that we require.

Energy though doesn't tell us how fast this work can be done. Power is energy per unit of time.

In terms of electricity, or indeed any source of energy, the basic unit of energy is the Joule. This isn't the entire story as we need to account for time too

and how much energy we use over time as this is the unit of power, the Watt, which is one Joule per second. First allow me to introduce the other units.

Volts (V) are the amount of work in Joules (J) per Coulomb (C) of charge, V is $\frac{\mathrm{J}}{\mathrm{C}}$. An Ampere (A) is the number of Coulombs per Second, A is $\frac{\mathrm{C}}{\mathrm{s}}$. When writing these symbols it is important to keep in mind the two symbols related to this, I which is used when dealing with quantities of charge and A for when discussing units of current.

Power is Volts multiplied by the current (usually in Amperes), so power is $\frac{\mathrm{J}}{\mathrm{C}}\frac{\mathrm{C}}{s} = \frac{\mathrm{J}}{s} = \mathrm{W}$. The unit name for $\frac{\mathrm{J}}{s}$ is Watts (W). Or in a more general sense $\mathrm{VI} = \mathrm{P}$.

As we've seen $\mathrm{VI} = \mathrm{P}$, so if we understand how V and I change over time we can use equation (3.1) to get the energy consumed between time t_0 and t_1.

$$\int_{t_0}^{t_1} \mathrm{V}(t)\mathrm{I}(t)dt = \int_{t_0}^{t_1} \mathrm{P}(t)dt \tag{3.1}$$

Unfortunately electronic devices are discrete circuits which can only sample the voltage and current. So in this case we can use equation (3.2) to obtain the approximate power consumed.

$$\sum_{n=1}^{\frac{t_2-t_1}{\Delta t}} \mathrm{V}(t_1 + n\Delta t)\mathrm{I}(t_1 + n\Delta t)\Delta t \tag{3.2}$$

Where Δt is the length of a finite sampling interval.

Thus far we've managed to avoid mentioning resistance, but it is one of the last pieces of information that we need to understand power measurement. Resistance is the measure of the reduction in electrical current across a device or a conductor. Simply put resistance is the measure of how much a material resists a current flowing across it. The unit for resistance is Ohms (Ω). The relationship between voltage, current and resistance in simple conductors is defined by Ohm's law. Ohm's law shows that current is proportional to voltage and inversely proportional to resistance, assuming that resistance is constant. This can also be defined in terms V and I as $\mathrm{R} = \frac{\mathrm{V}}{\mathrm{I}}$.

In order to measure power usage, we need to measure the voltage and current of the circuit. This is normally done by using a voltmeter connected in parallel to the component of the circuit that you wish to measure and connecting an

	Unit	Symbol
Potential Difference	Volt (V)	V
Charge	Coulomb (C)	Q
Current	Amperes (A)	I
Energy	Joules (J)	J
Power	Watts (W)	P

Table 3.1: List of electrical properties, units, and symbols

Ammeter in series with the circuit. But in a discrete circuit we must create these by much simpler means. To measure the drop in voltage across a circuit you connect an Analogue-to-Digital Converter (ADC) to it in parallel. ADCs quantise a continuously varying voltage into a binary representation.

Then to measure the current we connect a resistor (of known resistance) in series to the circuit and using an ADC we measure the difference in voltage across this resistor. This gives us a voltage and a known resistance so we can rearrange Ohm's law and get $I = \frac{V}{R}$, which gives us the current. So we know the voltage and current at one sample, if we keep repeating the readings we can employ equation (3.2).

The final unit we should mention is the Kilowatt-hour (kWh). It is commonly used to measure power consumption and as the billable unit of energy. It is equivalent to one thousand Watts for one hour or to put it simply 3.6×10^6 J.

Further reading on this subject is presented in [158] which has a good deal more information than needed for our purposes, especially on matters of electrical supply to data centres.

3.2 Power Measuring Hardware

In this section we shall look at some of the hardware options we considered using for the research that makes up this thesis.

3.2.1 Power Meter Plug

We started out by using a 2000MU-UK power meter by Prodigit Electronics Co Ltd from Maplin^{TM}[133]. It is a small device which plugs in between a mains socket and the device you want to monitor. It provides measurements of voltage (180 V to 250 V), current (0 A to 15 A), power consumption(0 W to

3750 W) & AC frequency (47.0 Hz to 63.0 Hz). The specifications say that it is accurate down to 1% max for voltage measurements but for voltages from 190 to 250 it is accurate down to 0.2%. For measuring current, the manufacturer claims that it is accurate down to 1% max, but it is typically 0.3% accurate for currents from 0.2 A to 15 A.

It has proved useful for preliminary investigations and getting general information. But it provides no way to log data for later analysis.

3.2.2 Current Cost EnviR

The Current Cost EnviR [56] is a consumer grade device to allow users to measure the energy usage of their home with a wireless current clamp unit which attaches to the line at the meter. This uses the magnetic field created by the current as a proxy to measure the current. Which has the advantage of not physically altering the circuit we wish to measure. It also can measure the energy used by individual appliances by using an Individual Appliance Monitor [57]. The Individual Appliance Monitor is similar to the power plug meter seen in section 3.2.1, but it a lacks a display of its own and instead it transmits power consumption data over a wireless interface.

This system displays power used on a simple unit with a liquid crystal display and allows for data collection over a Universal Serial Bus (USB) interface [55]. The USB interface is a serial to USB device and capturing and logging the data is a relatively straightforward task. To help us do this we wrote a Unix daemon in Python to log the output of each sensor.

The frequency of data reported is low, about once every 7 seconds, and not of the highest possible accuracy. Although it proved useful in our experiments.

The availability of easily modifiable software and the ability to monitor multiple devices made this a good choice in studies where we wished to monitor the aggregate power usage of multiple systems. For example, in chapters 4 and 5 the Current Cost EnviR system was used to collect data.

3.2.3 Energino

The Energino is a shield which can be added to an Arduino micro-controller and provides a low cost way to read the power consumed by a device connected

to it [75, 21]. It was first designed to gather data about the power consumption of wireless networked devices, with the goal of using this data to better model and understand the energy consumption of these devices. As the device has no means to display information by itself it must be connected to a host computer over USB or using a Ethernet add-on board to log its data output. As the design is distributed under an open source licence it is possible to modify or improve the device to better suit the task at hand. This unit was originally designed with a Sparkfun low current sensor breakout board [150] based on a ACS712 integrated circuit [8]. It can measure up to 5A, but the breakout board adds an operational amplifier which enables it to measure very small currents (at a sensitivity from 66 mV/A to 185 mV/A). The downside of this extra sensitivity is a tricky calibration stage which can be difficult to master. This can be avoided if instead you opt for a different sensor or breakout board. We decided that for the uses we had in mind we did not require this sensitivity and found a different breakout board [138]. Our Energinos were built using a ACS714 sensor [7] which can measure from -5 A to 5 A and at a sensitivity of 185 mV/A. This also had the advantage of having a simplified calibration.

3.2.4 Monsoon Power Monitor

The Monsoon Solutions Inc. Power Monitor [116] is a piece of laboratory equipment designed to monitor the power consumption of small electronic devices powered by lithium batteries such as mobile phones. It can also be used to monitor the power consumed by USB powered devices. It can measure from 2.01 V to 4.55 V in increments of 0.01V and a maximum of 3A and it can sample these at 5kHz. The Monsoon monitor offers highly accurate, high frequency measurements, but is better suited to observing power events at very small time scales.

3.2.5 Power Distribution Unit

Power Distribution Units (PDUs) are used to connect the power supply units of servers to an electrical supply. Following the general trend of devices becoming more intelligent and more connected, some newer models of PDU support monitoring the power consumption of the devices they supply power to.

We had access to one such device made by APC, a rack mounted PDU model number AP7900 and found that its readings were not consistent when lightly

loaded compared with the Power Meter Plug and Current Cost meters from sections 3.2.1 & 3.2.2. But more reasonable readings could be achieved if it was under more considerable load.

3.2.6 Intelligent Platform Management Interface Devices

The Intelligent Platform Management Interface (IPMI) is a set of specifications for out-of-band management interfaces. These interfaces are typically used for servers and usually include a suite of sensors to allow System Administrators to monitor the current status of the system (for example system temperature). Some implementations of IPMI interfaces such as version 6 of Dell's™Integrated Dell Remote Access Controller (iDRAC6) supports measuring the power consumption of the host system.

3.3 Discussion

As we have seen there are many options to consider when wishing to measure power consumption. This really is a question of frequency as well as a question of accuracy. This question has a different answer depending on what you want to measure and for how long. Measurements of power consumption over shorter time scales (seconds to minutes) might demand a measurement system with higher reporting frequency. But this runs the risk of being noisy as many tools will employ windowed averaging to overcome transient abnormal readings.

Measurements of power consumption over a longer time scale (hours to days and weeks) don't benefit from these higher reporting frequencies which at best create the problem of having a large amount of data. At worst these can result in having vast amounts of noisy data which requires a significant amount of additional processing. Another consideration is what you intend to use this data for. Some post processing is almost always required, but the amount required may make some applications prohibitive.

For our experiments detailed in chapters 4 and 5, which required longer term measurements, the Current Cost EnviR was used, as it is reasonably accurate and it is possible to automate collecting data from it.

For smaller shorter term investigations we used an Energino because it offered reasonably high accuracy and sampling frequency as well as the ability to log the collected data.

CHAPTER 4

Power Saving for Web Servers Using Proxies

Electricity is a major cost in running a data centre, and servers are responsible for a significant percentage of the power consumption. Given the widespread use of Hyper Text Transfer Protocol (HTTP), both as a service and a component of other services, it is a worthwhile goal to attempt to reduce the power consumption of web servers. In this chapter we consider how reverse proxies, commonly used to improve the performance of web servers, might be used to improve energy efficiency. We suggest that when demand on a server is low, it may be possible to switch off servers. In their absence, an embedded system with a small energy footprint could act as a reverse proxy serving commonly-requested content. When new content is required the reverse proxy can power on the servers to meet this new load. Our results indicate that even with a modest server, we can get a 25% power saving while maintaining acceptable performance.

4.1 Introduction

New server technologies promise to reduce power consumption when resources are idle. Servers are specified to meet or exceed current peak demands; however common power-saving techniques are unable to power off the server while idle, as the server must remain responsive to new requests. Empirical tests show

that while powered off or sleeping, servers consume significantly less power than in the lowest-power idle states [73].

Mechanisms for temporarily turning servers on and off do exist. Recent server hardware supports the Advanced Configuration and Power Interface (ACPI) power-saving modes, which were previously only available on laptop and desktop computers. These include methods to Suspend to Disk (also known as hibernation) and Suspend to RAM, which are low power modes with quick recovery to normal operation. The Wake-on-LAN (WoL) standard for remote activation of devices offers a convenient method to initiate remote recovery.

Thus, if we can overcome the adverse effect on service availability resulting from putting a server to sleep, then it may be practical to save energy. The use of reverse proxies for website acceleration or load balancing is relatively commonplace. In this chapter we consider using a low-power device acting as a reverse proxy. It will have the additional abilities to shut down the server when demand is low, serve cached content while the server is sleeping and wake the server if new content is required. In contrast to previous works, which have considered how to reduce power usage when a pool of servers provide a service (e.g. [73, 45]), we are targeting services typically provided by a single server which may have low-demand periods (e.g. in-house servers at night, low-demand hosted web servers, ...).

We explore this approach by developing a small testbed that allows us to replay web access patterns, estimate energy savings, etc. In section 4.2, we describe the web access patterns that we observe on a campus web server and their implications for designing a power-saving scheme. In section 4.3, we describe the relevant power-management features and measure their power usage. We then discuss the limits of possible power savings without introducing additional delays in section 4.4. This leads us to the design of a power saving scheme in section 4.5 and our evaluation of this scheme in section 4.6. In section 4.7 we discuss our results and conclude in section 4.8.

4.2 Investigation of Traffic

Our aim is to exploit patterns in web traffic in order to turn off web servers when they are not required. There has been considerable work to characterise

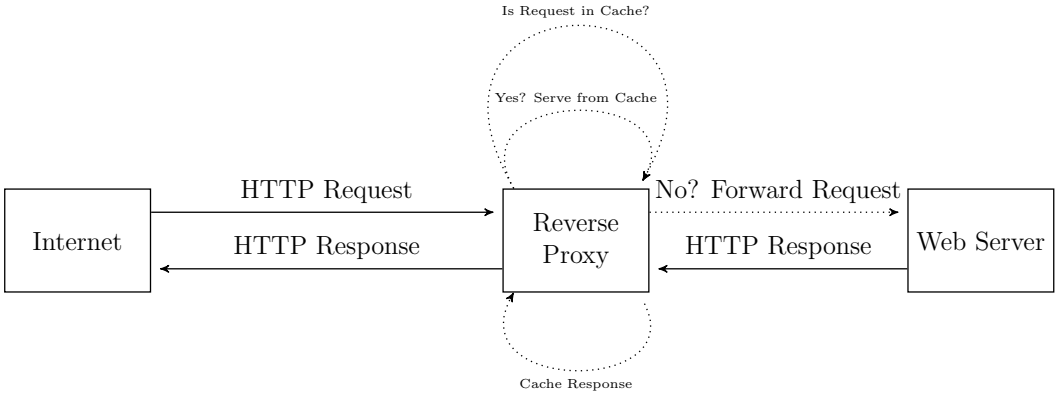


Figure 4.1: A simplified diagram of the operation of a reverse proxy

web traffic (e.g. [36, 129, 22, 28, 42]), and it is known that web access patterns are bursty.

A reverse proxy is a web server which accepts requests from clients and forwards them to a *back end* server, which stores or generates all the website's available content. The reverse proxy caches the content as it is served, and uses the cached content to answer requests where possible. As websites often have 'hot' static content, or content that is expensive to generate but can be cached once generated, reverse proxies can often result in performance improvements. Figure 4.1 shows a simplified diagram explaining the operation of a reverse proxy. We use Varnish Cache [88] as a reverse proxy in our research. Varnish is written in C and is easily extendible by adding C code to the cache configuration files. This allows us to alter the behaviour of the cache as well as add new features as we need them. The reverse proxy's ability to cache content, and so to save power, will depend on the details of the accesses. Consequently, we will design and assess our scheme using a log file of actual requests from a campus web server.

This web server hosts the websites of around 400 student clubs, societies and individual students. It has a variety of content and is accessed frequently by those on and off the campus, and exhibits a mix of web content and access patterns. In total, it amounts to over 77GB of content in 400,000 files excluding content stored in databases. Frequently-accessed content represents a considerably smaller subset of this total. The server is not busy, typically serving around 30,000 requests per day, but the volume of content should make

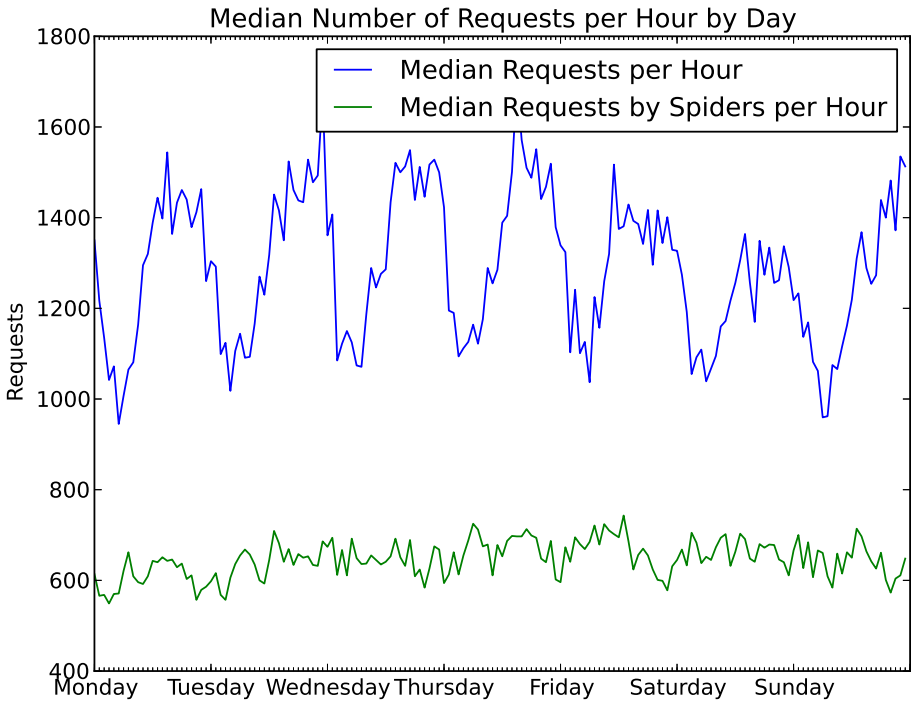


Figure 4.2: Median number requests per hour by day, taken from sample web logs

`Googlebot`	`Slurp`	`Baiduspider`
`bingbot`	`urlresolver`	`Speedy Spider`
`Sosospider`	`Sogou web spider`	`Gigabot`

Table 4.1: Strings used to identify common spiders

caching more challenging.

We looked at the median and mean number of requests per hour over 270 days worth of log data and grouped them by a specific hour during a week. The median is shown in figure 4.2, with the noisier mean values in figure 4.3. We do not see periodic activity, but a diurnal pattern emerges, showing significant variation throughout the day. Figure 4.3 also shows some large spikes which we determined where an attempt to index all the content on the server.

Further inspection of the data reveals that a significant number of requests are

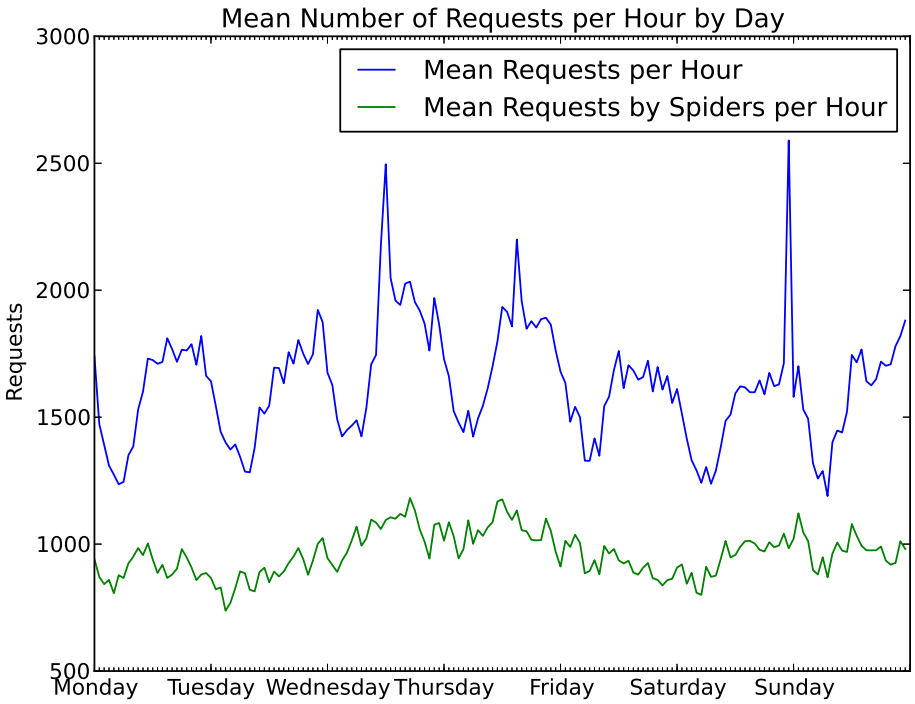


Figure 4.3: Mean number requests per hour by day, taken from sample web logs

from web spiders [95] [1]. Based on a manual inspection of the User Agent field of the log file, and consulting lists of common spiders, we found that matching the list in Table 4.1 allowed us to identify the majority of requests made by spiders to this site. We randomly sampled requests with User Agent fields that do not match this list, and inspected them manually. Only $\approx 5\%$ of the remaining requests come from suspected spiders.

We calculate the median and mean request rate from spiders in this list and also show this median and mean in figure 4.2 and figure 4.3 respectively. We see that a significant number of requests are actually from these spiders and that this traffic does not exhibit the same diurnal pattern. This suggests that it may be useful to handle web traffic from spiders as a special case.

If we want a server to sleep between requests, then an important factor to consider is the period between requests. We estimate this using the length of

[1] A web spider is an automated system that loads web pages [122], typically to gather information e.g. search-engines' crawlers.

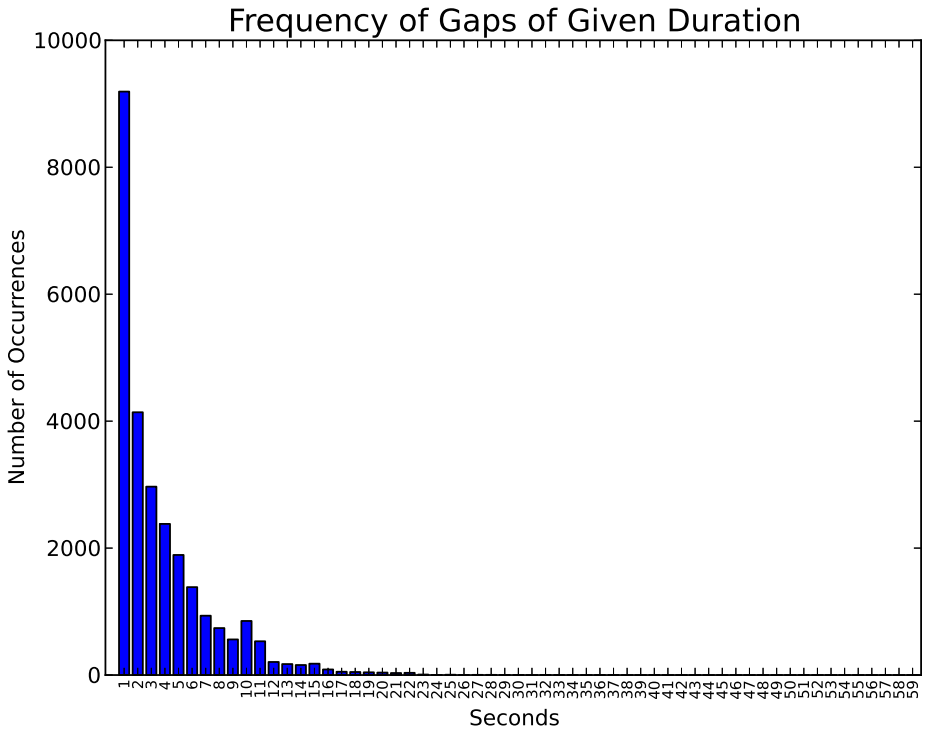


Figure 4.4: Number of gaps of a particular duration between requests. We omit gaps of duration less than one second

gaps between logged requests (logged at a resolution of 1 second in Combined Logfile Format). The length of these gaps will indicate if it may be possible to switch off the web server between requests, or if such opportunities are limited.

Using a subset of our data, amounting to 40305 requests over 28 hours, we look at the distribution of the gaps. Figure 4.4 shows the frequency of the gap of a particular duration. As we expect, gaps are typically quite short, which limits our chances to turn a server on and off without impacting on web traffic.

To consider the impact of caching on the gaps between requests to the (back end) web server we replayed the requests to the campus server using Varnish [88] as a reverse proxy to cache the content [2]. The resulting distribution of gap lengths is shown in figure 4.5. We see an increase in the number of long duration gaps, representing an increase in opportunities to put the server to sleep.

[2]The cache starts empty. We use the default Varnish configuration.

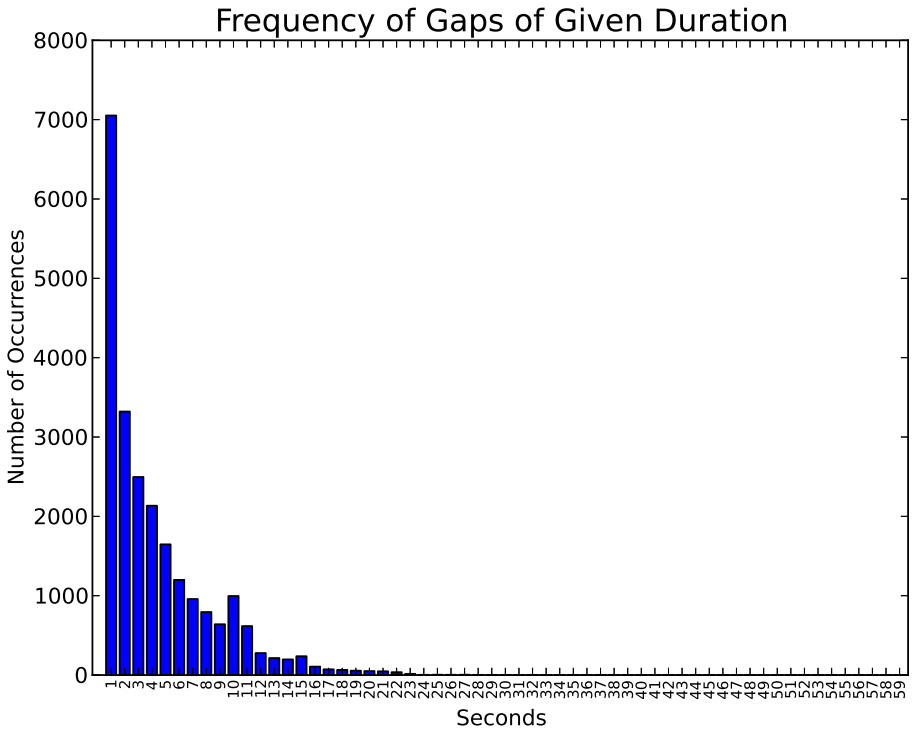


Figure 4.5: Number of gaps of a particular duration between requests to back end omitting requests served by reverse proxy. We omit gaps of duration less than one second

For comparison, we also consider the distribution of gaps between requests when we omit requests from spiders. We do this on the basis that a spider does not usually need the content immediately, and indeed, some spiders can be told to make their requests later using a HTTP 503 response with a Retry-After header [71, 85]. The results are shown in figure 4.6. We now see that the tail of the distribution of gap durations has thickened, which suggests a better chance of powering the server off without impacting on user requests. Comparing these results with figure 4.5, it appears that for this log file, smart handling of spiders might have a bigger impact than caching of commonly-accessed content.

4.3 Power States and Recovery

Various systems are available for controlling the power state of servers. In particular, we will make use of an interface for putting the system into a low power state (ACPI) and then waking it at some later point (WoL).

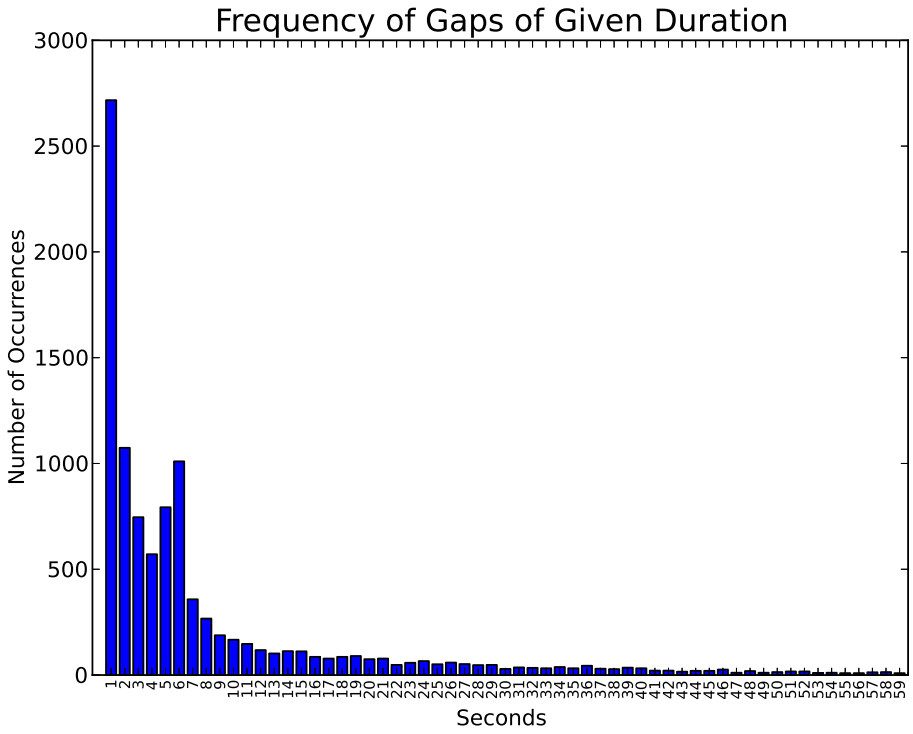


Figure 4.6: Number of gaps of a particular duration between non-spider requests. We omit gaps of duration less than one second

ACPI is a standard that aims to consolidate all power management and configuration standards [1]. The ACPI Standard defines a number of *power states*, from G0 (active) to G3 (mechanical off). The sleeping state, G1, is subdivided into four sub-states (S1-S4). For us, S3 and S4 are interesting as they define Suspend to RAM and Suspend to Disk respectively. As we will see in these states, servers consume almost as little power as when powered off. With operating system support, the power state of a server can be changed.

WoL is an industry standard for remotely powering on computers. It requires a compatible network interface card which remains powered on after the computer is powered off. A number of different signals can be used to wake a computer, based on hardware support, including PHY activity, Address Resolution Protocol (ARP) and broadcast, unicast and multicast messages. The most commonly supported signal used to wake up a computer is called the *Magic Packet* [84], a broadcast frame with a payload that is 6 B set to 255 followed by the target computer's MAC address repeated 16 times. WoL can

	On Power Usage (W)	Off Power Usage (W)	Switch Cost (J)	Time to Sleep (s)	Time to Recover (s)
MSI Server	43.3	1.6	170.7	3.7–4.3	0.1–0.2
Soekris net 5501	5				

Table 4.2: Measured power profiles for devices, the power consumed while off is mainly being used to support WoL

usually be configured via the BIOS/firmware or operating system.

To illustrate these power management features we investigate the power consumption of the device we use as a server. For this work we investigated a number of different methods for measuring power consumption, this can be seen in chapter 3. We decided to use a Current Cost EnviR as described in section 3.2.2. Below we describe how we model the energy use of our servers. For these systems, we also measure the time needed to put the system to sleep and the time to wake after a WoL message. This will influence the responsiveness of our scheme. We call the sum of these the *threshold*. We also characterise the *threshold* below.

We characterise our hardware in table 4.2. For comparison, we also show the power consumption of a Soekris net5501. We will use this as a reverse web proxy during quiet periods.

MSI Server The server used was a custom-built server based on a MSI "Military class" motherboard, an Intel i7 2700k processor and 16GB of Random Access Memory (RAM) running Ubuntu 12.04 LTS. We used the on-board 1Gbit Ethernet for a network connection. We tested several other servers of this generation, and older generations. We found that this system had good support for both WoL and use of ACPI power states. The power supply is rated at several hundred Watts, though in practice it uses considerably less when operating as a web server.

Soekris net5501 The Soekris net5501 is a single board PC based around the AMD Geode^{TM}processor, using a maximum of 20W, but typically much lower as we see in table 4.2. We run an older version of Ubuntu, 8.04.4 LTS, with smaller resource requirements on this device and which also still supports the i686 processor architecture. We do not use or report on

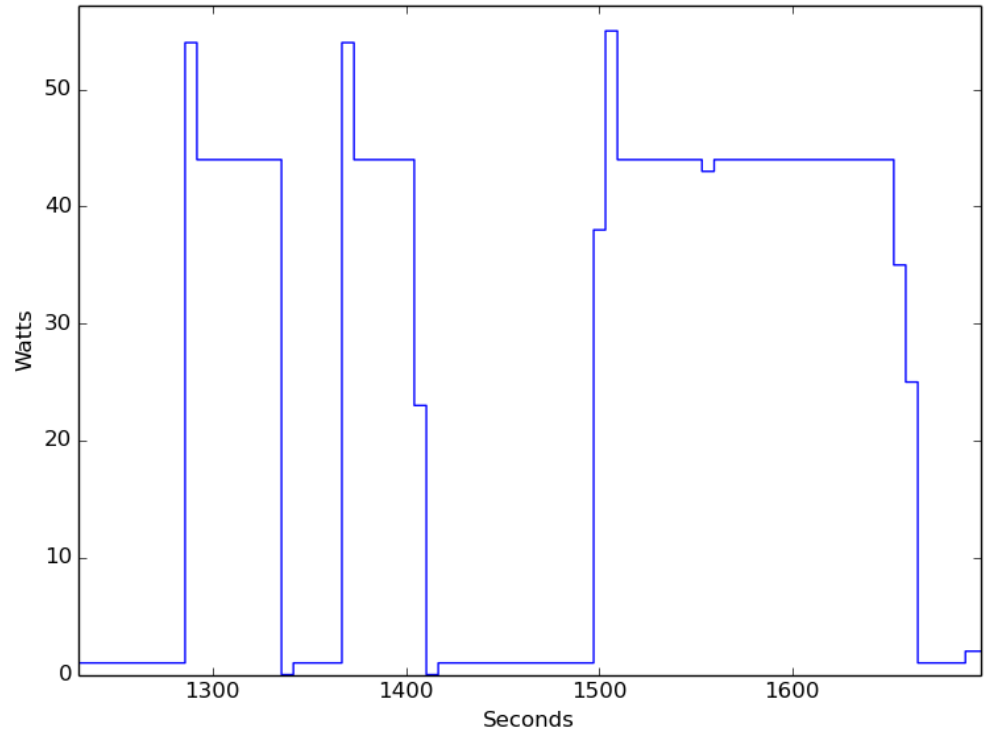


Figure 4.7: Power consumption for a server when sleeping, waking, & powered on

the sleep/wake features of the net5501, as we intend to use this device as the reverse proxy.

In reviewing older server hardware, we found that WoL and ACPI support was less consistent. For example, after upgrading the firmware on a Dell® PowerEdge™1800 we found WoL and ACPI were available. However as only suspend to disk was possible, wake times were over one minute. The Dell® Optiplex™755 desktop system supported suspend to RAM, with much better wake times, however we found that WoL did not always wake the system and though capable of running a web server, was not typical of server-class hardware.

4.3.1 Modelling Power and Energy Usage

Figure 4.7 shows the power usage for our server over a number of sleep/wake/powered-on cycles. We can see that there is a pattern, which suggests roughly constant power usage when sleeping or powered on/off. There is a brief spike in power

usage when the machine wakes. Based on this, we model the server as having a power usage P_{on} while on, a power usage P_{off} when off and there is a cost for each switch-on event of E_{switch}. From figure 4.7 we see that these quantities are not fixed, but in fact random.

We do this by measuring the power used by the test bed while it performs a series of sleeps and wakes of variable length in an experiment labelled i. For each experiment we record four variables: the total amount of time awake T_{on_i}, the total amount of time sleeping as T_{off_i}, the number of times it switches on or off N_{switch_i} and the total energy used E_{total_i}. Our model suggests that in expectation we have

$$T_{\text{on}_i}\mathbb{E}[P_{\text{on}}] + T_{\text{off}_i}\mathbb{E}[P_{\text{off}}] + N_{\text{switch}_i}\mathbb{E}[E_{\text{switch}}] = \mathbb{E}[E_{\text{total}_i}] \tag{4.1}$$

We then want to estimate the Power used while the server is on and off and the energy overhead for a switch between these two states. Over a number of runs we can write this as

$$\begin{pmatrix} T_{\text{on}_i} & T_{\text{off}_i} & N_{\text{switch}_i} \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \mathbb{E}[P_{\text{on}}] \\ \mathbb{E}[P_{\text{off}}] \\ \mathbb{E}[E_{\text{switch}}] \end{pmatrix} \approx \begin{pmatrix} E_{total_i} \\ \vdots \end{pmatrix}. \tag{4.2}$$

We can then use the least squares estimator for $\mathbb{E}[P_{\text{on}}]$, $\mathbb{E}[P_{\text{off}}]$ and $\mathbb{E}[E_{\text{switch}}]$ given E_{total_i},

$$\begin{pmatrix} \mathbb{E}[P_{\text{on}}] \\ \mathbb{E}[P_{\text{off}}] \\ \mathbb{E}[E_{\text{switch}}] \end{pmatrix} \approx (A^T A)^{-1} A^T \begin{pmatrix} E_{total_i} \\ \vdots \end{pmatrix}, \tag{4.3}$$

where A is the matrix of on/off times and switch counts. The results of performing these estimates for our server are shown in table 4.2.

4.3.2 Server Wake/Sleep Times

The length of time required to put a server to sleep and to wake it again is also important to our scheme. These determine the minimum amount of time we can sleep for, and also how long it will take to answer an uncached request that arrives while the server is asleep. We describe how we measure both times below using ICMP ping packets, to determine if the Internet Protocol (IP) stack of the server is operational.

To measure the time that it takes the server to be put asleep, we issue the sleep command to the server via Secure Shell (SSH). At the same time, we ping the server once every 100ms until we receive no response. The number of pings between issuing the SSH command and the last received ping gives us the time to sleep in tenths of a second. Note, that there is some timing overhead associated with the SSH connection, which would not be present if a more lightweight way of issuing the sleep command was used. To estimate this, we also timed how long it took SSH to run a null command.

To estimate the wake-up time, we also send a ping to the server every 100ms and then send a WoL magic packet. We then observe the amount of time between issuing the magic packet and when the server starts to respond. In order to avoid issues with ARP entries timing out, we manually set the ARP entries for the server. Again, this gives us an estimate of the time to wake the system to the nearest tenth of a second.

The results of estimating both the wake and sleep times for our server are shown in table 4.2.

4.4 Idealised Power Saving Model

In this section, we show how to estimate the possible power savings for a web server, given information about power usage, gaps between requests and how quickly it can be turned on and off. Consider an idealised situation, where the server could somehow determine when the next request will arrive. After serving each request, it could see if the time to the next request is greater than its turn-around-time. If so, the server goes immediately to sleep and schedules a wake up just in time to serve the next request. This would allow the server to serve *all* requests without introducing *any* delays waking the server, while sleeping for the maximum time possible. However, as we would only like to consider sleeping if we get a resulting power saving, and as we will see in section 4.5.3, this can also be converted into a minimum threshold on the sleep time.

We can calculate the power saving given by this idealised scheme. Let $(t_i)_{i=1..N}$ be the sequence of gaps between requests. We can find T_o and T_s, the time

that the server spends on or sleeping respectively. For this scheme,

$$T_o = \sum_{i=1, t_i < t_{\text{thr}}}^{N} t_i$$

and

$$T_s = \sum_{i=1, t_i \geq t_{\text{thr}}}^{N} t_i,$$

where t_{thr} is the threshold given by the minimum turn around time and the requirement for power saving. Total energy usage, in kWh, will then be

$$\frac{T_o P_{\text{on}} + T_s P_{\text{off}} + N E_{\text{switch}}}{3.6 \times 10^6},$$

where N is the number switches, given by the number times $t_i < t_{\text{thr}}$.

Using this model, we can assess the power saving possible without introducing extra delays. Figure 4.8 shows the total energy consumption possible as a function of the threshold time, t_{thr}, assessed for the original sequence of requests served by the web server. We see that for the original log file, for a threshold of 20s or more, there are few opportunities to sleep, and the total power usage is similar to having the server always on. However, threshold times of 2 s to 5 s actually result in significant savings.

Following our observations from section 4.2, we consider the impact of spiders and caching. First, figure 4.8 also shows the results if we are willing to ignore requests from spiders. Here the situation is much more promising. With our idealised scheme, a threshold of 20s allows a reduction of energy consumption to around one third of the consumption for a server that is always on. The nearly linear increase seen in the "Original without Spiders" line indicates that when the threshold is smaller than most of the gaps, the power usage is roughly proportional to the theshold size.

Second, we consider the impact of caching in figure 4.9. Here, we use the idealised power-saving model on requests that are passed by a varnish cache to the the back end server. This eliminates the need for the back end to serve the content cached by the reverse proxy. As expected based on our discussion in section 4.2, we see a saving over answering all requests; however the improvement is not as large as the saving for ignoring requests from spiders. We also see that combining caching with the special handling of requests from spiders results in useful gains.

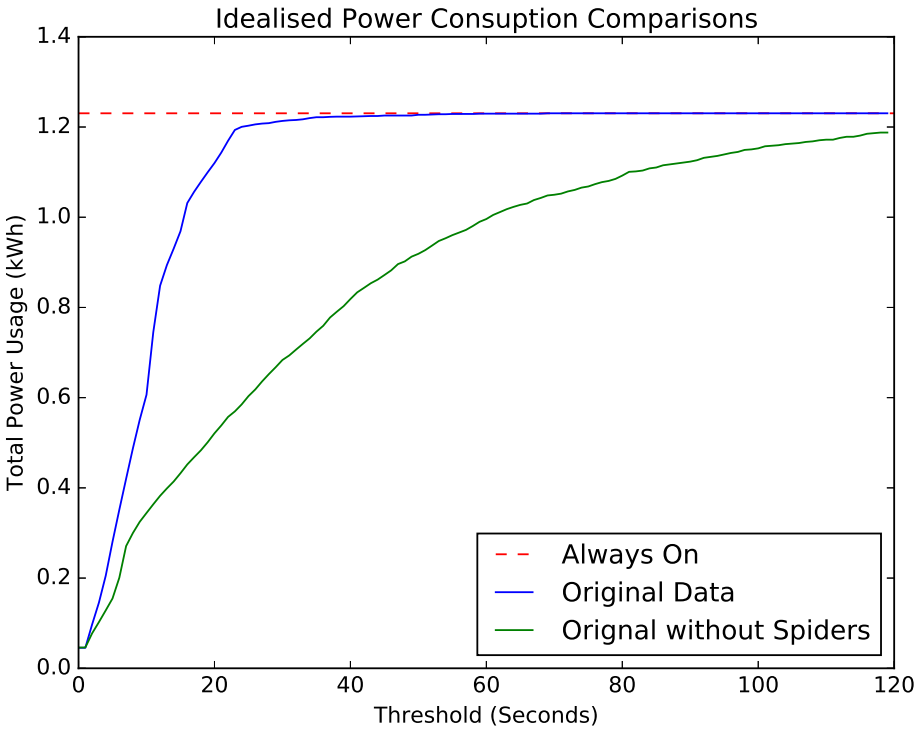


Figure 4.8: Power saving for the idealised scheme that results in no delay, as a function of the threshold

4.5 Design of a Practical Power Saving Strategy

In this section we will consider how to design a practical scheme to allow web servers to sleep. We consider when to power the server down, when to power it up and, as the scheme does not have advance knowledge of the requests, how to handle requests that arrive when the server is powered down.

4.5.1 Architecture

The system we consider is a high-power web server with a low-power reverse proxy (though other configurations are discussed in section 4.7). The reverse proxy manages the power state of the web server via ACPI and WoL. We assume that the low-power proxy does not cause performance problems during periods of high load. In practice, this issue can be overcome by switching to directly serving requests (there are a number of means to achieve this for example DNS, ARP/IP mapping or using software defined networks) or

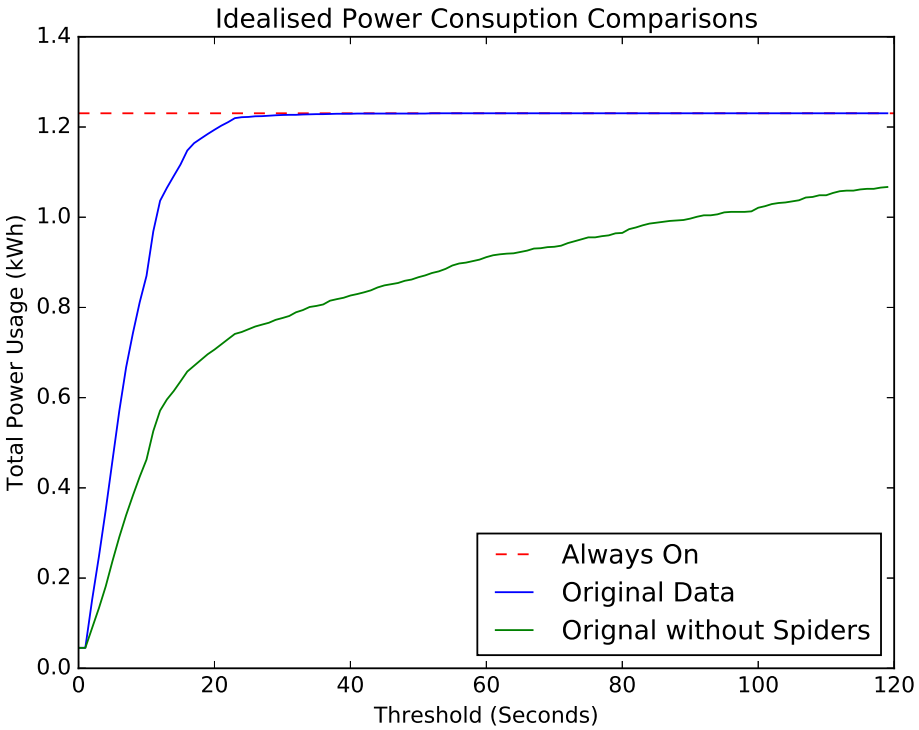


Figure 4.9: Power saving for idealised scheme, answering requests that cannot be responded to by reverse proxy as a function of the threshold

switching in a higher-power proxy at higher loads. Similarly, we assume that caching is managed in the usual way by Varnish, with parameters set as described in section 4.6.1.

4.5.2 Power-On Decisions

Power-on decisions must be made by the reverse proxy, as the server is not always available to make decisions itself. Thus we describe the power-on decisions in terms of the operation of the reverse proxy. Of course, if the web server is on, we configure the reverse proxy to operate as usual, responding to queries from the cache or making requests to the back end.

If the web server is off, we trigger power-on requests based on the arrival of server requests. Naturally, if the request can be served from the cache, then there is no need to wake the back end. Based on the discussion in section 4.2, we believe there may be an advantage to handling requests from spiders in a special way. If the back end is off and the request is from a spider, we also

choose not to wake the back end and instead issue a HTTP 503 response, indicating that the service is currently unavailable and that the spider should return later. We suggest that this time could be chosen to be a known-busy hour of the day or a period specially chosen to facilitate website crawling by spiders. The logic is summarised in algorithm 1.

Algorithm 1 Decision process to wake sleeping back ends

```
if request in cache then
  serve request from cache
else if back end server available then
  serve request from back end server
else if request from spider then
  issue HTTP 503 to request spider
else
  wake back end
end if
```

4.5.3 Power-Off Decisions

In practice, the power-off decision could be made by either the back end web server or the reverse proxy. We considered a number of possible strategies for making this decision, including forecasting demand based on previous weekly/diurnal patterns, monitoring the hit-rate in the reverse proxy's cache, or using a learning algorithm. We will consider a scheme that aims to achieve power savings on average while achieving a target delay for serving requests.

One possible improvement that applies to most schemes is to ensure server sleep time isn't interrupted by requests from web crawling agents used to index websites for search engines. This can be achieved by the ether responding to all requests from such agents either with cached-content, no update or a HTTP 503 response (this should mitigate any negative effect on search ranking). Consideration should also be given to client side caching for supported clients as a means to reduce the number of possible requests to the server.

4.5.3.1 Expectation of Power Saving

An important consideration for switching off is, *will turning off save power?* Ideally we would like the power used to turn off and then power back on again to be less than the power used for turning on, i.e. $TP_{\text{off}} + E_{\text{switch}} < TP_{\text{on}}$, where T is time between requests, and P_{off}, P_{on} and E_{switch} are the energy

usage parameters described in section 4.3.1. Since these quantities, particularly T, are random quantities, we can instead ask to save power on average, so we take the expectation,

$$\mathbb{E}[TP_{\text{off}} + E_{\text{switch}}] < \mathbb{E}[TP_{\text{on}}] \tag{4.4}$$

$$\mathbb{E}[T]\mathbb{E}[P_{\text{off}}] + \mathbb{E}[E_{\text{switch}}] < \mathbb{E}[T]\mathbb{E}[P_{\text{on}}] \tag{4.5}$$

This can be further simplified to give us this lower bound, below which we should not send the server to sleep:

$$\mathbb{E}[T] > \frac{\mathbb{E}[E_{\text{switch}}]}{\mathbb{E}[P_{\text{on}}] - \mathbb{E}[P_{\text{off}}]} \tag{4.6}$$

Provided that $\mathbb{E}[P_{\text{on}}] - \mathbb{E}[P_{\text{off}}]$ is positive, which we expect for realistic on/off power values. We also assume that T is independant of P_{off} and P_{on}

To implement this condition, we estimate the power usage parameters, as described in section 4.3.1, and a window of ten requests to estimate the current spacing to the next request T.

4.5.3.2 Managing Delay

If we are not concerned about delaying requests, then putting the server to sleep for extended periods can save a lot of power. However, we suppose that we have an SLA that requires that no more than a fraction f_{SLA} of the requests can be delayed by more than a delay δ over some window.

Suppose that at some time, the number of requests that will arrive in a threshold period is $\mathcal{K}$. If $\mathcal{D}$ is the number of requests that have exceeded the delay threshold δ and $\mathcal{T}$ is the total number of requests seen in the current window, then to achieve the SLA, we require the server to maintain a counter of the total number of requests $\mathcal{T}$ and a counter of delayed requests $\mathcal{D}$. Estimating the expected number of requests in the threshold $\mathcal{K}$ is relatively simple, by maintaining a request rate over a window. Note, that we expect that maintaining this condition will increase power usage, which is in line with other studies considering power usage and SLAs [113].

4.5.3.3 Power Cycling Cost and De-bounce

Our scheme, in practice, limits the amount of switching between on and off states, however a server might want to limit the amount of hardware stress due

to power cycling or bouncing between on and off states because of conflicting on and off conditions. We implement two simple conditions, first we will not power off the server for 60s after a power-on, to avoid excessive power cycling. Second, we wait at least 5s after the most recent request before powering the server down, to allow slow clients to request all the content for a page load.

4.5.4 Mitigating Against Mistakes

We note that a power-saving scheme does not know when requests will arrive so it will ultimately make some mistakes. If requests from users are delayed while waiting for a server to awaken, this could have a negative impact on their experience of using the service, or might result in a connection time-out. In the case that the reverse proxy detects that is must wake the server, we propose a method to combat this. If the content is delayed, alternative web pages could be presented by the proxy, such as a splash screen or a loading bar. Alternatively, an advertisement or suggestion that the user take a short survey might be offered.

Note that we must only stall the user for approximately five seconds in order to wake a server from the Suspend-to-RAM state, when the user can be redirected to their originally requested content. We have implemented this last method as an addition to Varnish. If a request requires a server to power on, the end user is served a page asking them to take part in a survey and giving them a countdown until they will be redirected to their requested content. We call this method 'Survey'.

4.5.5 Other Considerations

Reverse proxies are generally used to increase throughput or performance. To decrease power usage, most of our tests are conducted with a low-power device, which may actually decrease performance. We evaluate the impact of this in section 4.6 by studying actual request delays. However, we also assess using a high-performance embedded device in section 4.6.4.

4.6 Results

In this section we outline our test bed, the parameters of our experiments and the results we obtained.

4.6.1 Test System

Our prototype test system consists of three components: a client, a reverse proxy, and a server. The client makes requests to the reverse proxy or server using a group of Python scripts that replay a given Apache access log in real time (i.e., requests are spaced as in the log file). These scripts record the status of the request responses and the delay in serving the content.

For a reverse proxy, we use a Soekris net5501 embedded computer, with power usage detailed in table 4.2. The system runs a version of Varnish Cache (version 3.0.1), which has been modified to allow the operation described in section 4.5.1. Varnish has been configured with a 128MB cache space. The connection time out to the back end is set to 10s. The server runs Apache version 2.2.2 and is a MSI-based system whose power requirements are detailed in table 4.2.

To implement the power-saving scheme from section 4.5.1, we send User Datagram Protocol (UDP) packets between the reverse proxy and the server. We found that all the WoL implementations we tested sometimes entered a state where wake requests were ignored. This is most likely due to a bug in the operating system or firmware. To work around this, we initially tried using an optocoupler attached to the Soekris net5501 General-Purpose Input/Output (GPIO) pins to "press" the power button on the server. While this mechanism was more reliable than using a WoL packet, we still found the servers occasionally in a state where they would not wake. Our conclusion is that either the Linux hibernation code or the ACPI implementation on the servers we tested is not dependable enough to run our scheme.

Consequently, for our experiments to assess the performance of the scheme, instead of putting the server to sleep, we set up a firewall rule that blocking port 80, which is used for requests, when we need to simulate the server being unavailable. These rules are removed, after a delay to simulate the wake time, when the system is woken. We use 5s as the delay, to overestimate the full wake/sleep cycle from table 4.2. We then estimate the power usage using the log of sleep/wake events and the power model from section 4.3.1. One advantage of this method is that we can vary the threshold and power model to estimate performance of other systems.

4.6.2 Evaluation

To evaluate the performance and power usage of our scheme, we replay the 28 hour section of log file described in section 4.2. We consider five different configurations of the test system, chosen to offer some insight into the performance trade-offs involved in powering the system down. In each case the cache starts empty.

No Varnish The server is always powered on and requests are sent directly to Apache;

Varnish Requests are routed to Varnish running with default parameters. Varnish contacts the always-on back end Apache server as necessary;

Varnish (Aggressive Caching) Requests are routed to Varnish running with grace period set to one hour and Varnish contacts the always-on back end Apache server as necessary;

Varnish (Sleep and Bot Redirection) Requests are routed to Varnish running with aggressive cache settings. Varnish runs the power-control scheme in section 4.5.3 and section 4.5.2. Wake and sleep times are set to 5s and 5s respectively, after Varnish makes the request. Bots are redirected if the back end is asleep when a request arrives;

Varnish (Sleep, Bot Redirection, and Survey) Requests are routed to Varnish running with aggressive settings, running the power-control scheme described above. Wake and sleep times are also as above, with bot redirection active. Request from users are redirected to a survey while the server is waking (as described in section 4.5.5).

In the last two configurations, the algorithm works with a delay budget of 50ms, for 90% of requests and uses 5s as an estimate for the threshold.

4.6.3 Results

First let us see if our technique can save power in practice. Figure 4.10 shows the power usage for the five different runs described above. The run with Apache alone, and no Varnish gives us a baseline of close to 45W. The second and third experiments do not attempt to power the server on and off, and so

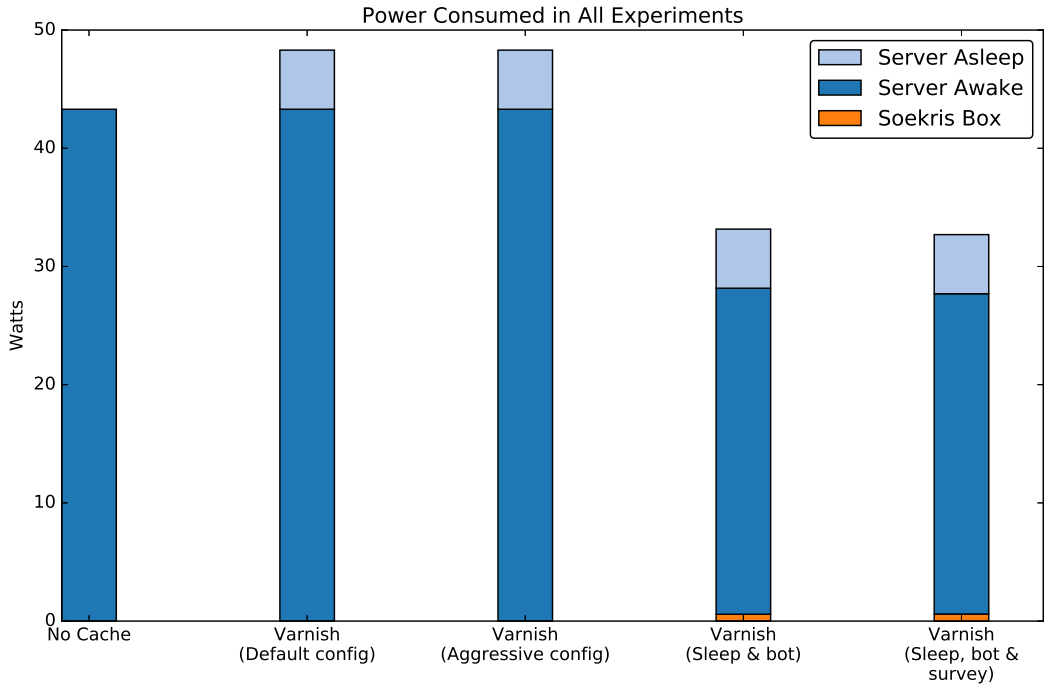


Figure 4.10: Comparison of the results of various experiments in terms of energy used

the power usage is simply the sum of power usage of the server and Soekris box. The final two experiments do produce an actual power saving; the extra power used by keeping the Soekris box on is more than matched by the power saved by letting the server sleep. The final bars show that it is possible to save power of approximately 25% even accounting for this overhead. Note that the server we used has quite a modest power usage, and greater power savings would be seen with more power-hungry servers.

Now, we need to see the impact of this power saving on performance. Table 4.3 shows the breakdown of how requests are served. The first column shows the results when Apache fields all requests. While most requests are successful, a number are for files that do not exist, and these are listed under the *HTTP 404* heading, for File Not Found. We also record the delays involved in serving the files. We see that the mean response time is around a millisecond, and the maximum delay is around 0.4s.

The second column shows the impact of introducing Varnish to the system. Observe that a significant amount of the content, around 7000 requests or 17%, can be cached and served by Varnish. In terms of performance, we see that

the low-power reverse proxy cannot serve content as quickly as the Apache server. The low power device only increases average delay by around 5ms, even though all requests are directed through it. There are a small number of requests with larger delays and we can look at the cumulative distribution of these in figure 4.11. We see that the fraction of requests with a greater than 50ms delay is small.

The third column shows the impact of adjusting Varnish's settings to allow more aggressive caching of content. We see that this is quite effective in increasing the number of requests served by Varnish, raising the number to around 12000, or about 31%. The mean delays for the requests served by both Varnish and Back end requests does not substantially change, but overall mean delay decreases because the fraction of requests served by Varnish has increased.

The fourth column of the table shows the results when the server is powered on and off, introducing a 5s delay. The most significant difference we observe here is that requests from spiders while the server is off now receive a Retry After response, indicated by the *Bot Redirection* row. There are additional delays in responding to back end requests, due to some requests now having to wait for the server to wake. However, they are still small on average coming to less than 60ms. Consulting figure 4.11, we see that we easily meet the target of 90% of requests served in under 50ms, however there are a cluster of requests that take several seconds to serve corresponding to the time taken to wake the server.

The fifth column of table 4.3 shows results where if a request arrives while the back end is sleeping, we suggest the browser take a survey and wake the back end. The request is then replayed after 15 seconds. This results in a higher number of overall requests in this scenario, and we show the results of these extra requests separately in the table. We see introducing the survey reduces the average time to serve requests to less than 10ms, so it is quite an effective technique. Figure 4.11 shows the increased number of requests, and that the cluster of requests is now gone, with just a handful of requests taking more than one second to serve, which is broadly in line with the number when we use Varnish but do not put the server to sleep.

Experiment Label	No Varnish	Varnish (Default config)	Varnish (Aggressive config)	Varnish (Sleep & bot)	Varnish (Sleep, bot, & survey)
Requests	40305	40305	40305	40305	41696
Malformed Requests	7	7	7	7	7
Served By Apache	40298	33288	27673	20280	20245
Served By Varnish		7010	12625	20018	21444
HTTP 200	36846	36837	36834	29879	29922
HTTP 404	3459	3459	3459	2674	2695
HTTP 503		9	12	241	163
Bot Redirection				7511	7525
Served Survey					1391
Replayed – HTTP 200					1041
Replayed – HTTP 404					205
Replayed – HTTP 503					14
Max Delay – Apache (s)	0.04	11.23	8.01	12.53	8.05
Max Delay – Varnish		0.20	0.12	3.0	1.98
Mean Delay – All Requests	0.001	0.006	0.005	0.058	0.009
Mean Delay – Apache	0.001	0.007	0.006	0.111	0.012
Mean Delay – Varnish		0.001	0.001	0.005	0.005

Table 4.3: Statistics showing results of HTTP requests for each configuration

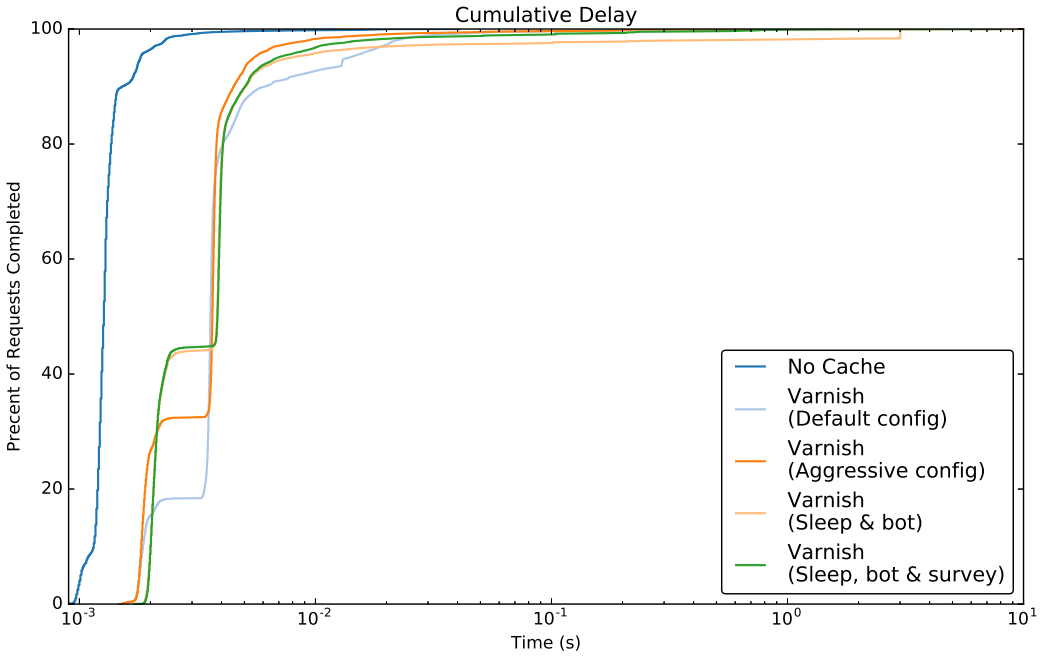


Figure 4.11: Number of delays and their duration across a number of experiments

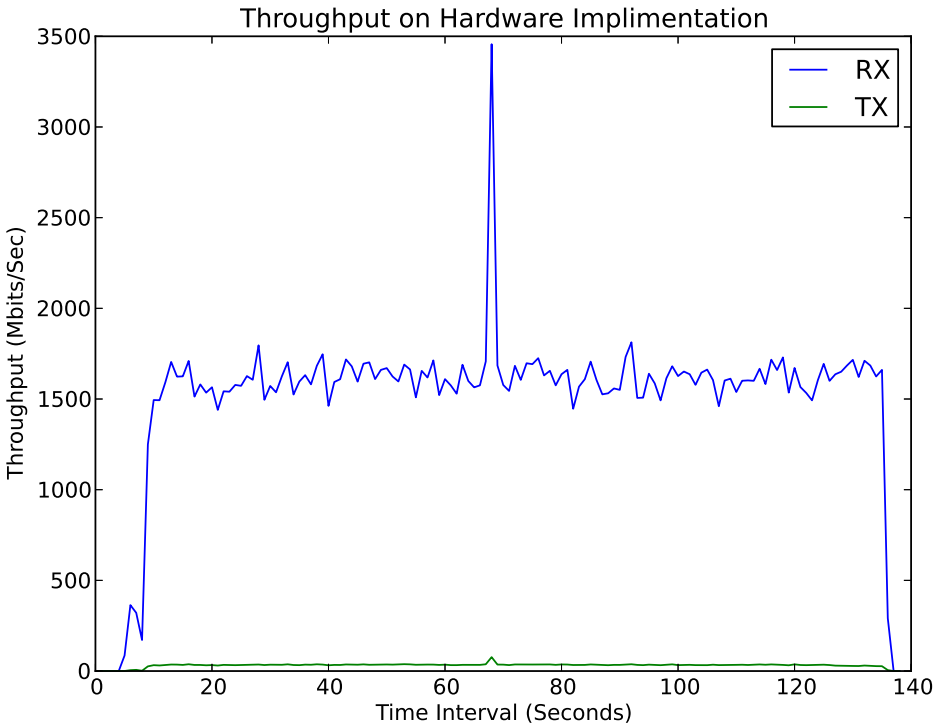


Figure 4.12: Performance of PowerEN™-based reverse proxy

4.6.4 High Performance Proxy

In previous sections we reported the results of using a power saving reverse proxy on a Soekris net5501, a low-power single-board computer. This raises the possibility that even though content is being successfully cached during busy periods, the net5501 may not be able to serve content as quickly as a fully-featured server system.

However, the reverse proxy could instead be run on an embedded device designed for high performance networking, such as a IBM® PowerEN™board [74]. These devices are designed to offer high-performance networking at a lower power consumption than the equivalent PC-based system. In this subsection we demonstrate that using such a device, it is possible to maintain high network throughput. As the device we test is a prototype, we cannot report accurate power usage statistics.

In our test, we ran varnish on the PowerEN™ board, which proxied content from a server. The content to be proxied was hundreds of files varying from

tens of bytes to hundreds of megabytes. Apache JMeter [78] was configured to use multiple threads to request content, with a short ramp-up period. The resulting throughput of the JMeter host is shown in figure 4.12. We see that an embedded device designed for high network performance can maintain a throughput of over 1Gbps, and seems to be able to burst at a higher rate when a number of requests for large files arrive concurrently. This raises the possibility of using an embedded device, even within a server or router, as a reverse proxy to reduce power usage while maintaining high performance.

4.7 Discussion

In section 4.6 we saw that in practice we could make power savings of 25%. This is a reasonable saving, however it is small compared to the idealised saving for around of 60–70% for a 5s threshold, (when the power usage of the proxy is factored in). This suggests there may be some scope for improvement, however it is important to consider that the idealised version we described uses the arrival time of future requests, which must be estimated by a practical scheme.

Our practical power savings do have a performance cost associated with them. For example, the increased baseline delay of a couple of milliseconds that is visible in figure 4.11. Some of this is the inevitable result of introducing an extra proxy system between the client and server, but some of these values relate our experimental setup. For example, we began our experiments with an empty cache with a modest size (128MB). With a larger cache and a longer test run, it is possible that caching of static content could be more effective again, resulting in fewer wake ups and fewer delays in serving content.

We do not know if our method of introducing a survey to stall users while the back end is being woken will be irritating for users, however it has been reported that negative feelings caused by waiting for content can be alleviated by explaining the reason for these delays [139]. Conducting user validation studies would allow us to see if the added delays or the occasional stalled request may negatively impact on the end user. These studies would require adapting our testbed to serve a selection of different types of web content and will have users rate their experience of accessing this content while the testbed is performing a variety of scenarios, such as simulated heavy request loads and simulated light loads with request stalling. Such a user study would let us

gauge our impact on users, but is beyond the scope of the current study.

Taking a more realistic view of the users, requests and web sites might also allow us to improve our power savings. Tracking the behaviour of users could allow us to predict their next likely request. This might be achieved by tracking IP addresses or sessions where active users are monitored. A long period of inactivity or explicit user log out could provide better indications of when to sleep. This could be based on the statistics of the dwell times of typical users, or could be customised for particular website content, possibly anticipating wake-ups from patterns of accesses to cached requests.

Similarly, we could extend the techniques that we use to handle spiders. Site maps are commonly employed by websites to inform web crawlers, both of the structure of a website, and the frequency of updates to its content. These could be used to help reduce the number of requests by spiders to see if page content has been updated or prevent requests for resources that do not need to be indexed. Another commonly-requested resource is `robots.txt`, a text file defining which well behaved spiders are allowed to crawl the page and what URLs they are allowed or not allowed to follow. Both the `robots.txt` and site map files could be used to mediate the requests from spiders [95].

Another important question is how to generalise our results to other web traffic or server configurations. Based on a search for sites with on-line statistics publicly available, we find many sides outside the Alexa top 1,000,000 have a lower load than the server we consider here. Other aspects of a site's traffic may have patterns that can be exploited. If request patterns are highly predictable, or such that delays in responses of a few seconds are not a concern, then this could lead to improved power savings. Alternatively, if content is highly dynamic or has tight constraints on how quickly it must be served, then this will make power saving more challenging.

We have considered a situation with a single server hosted on physical hardware that can be powered down. A similar configuration with a higher-powered server should result in the scheme achieving larger savings. The principle of the scheme also generalises to other configurations. Multiple servers hosted on multiple physical systems could be cached by a single reverse proxy, spreading the cost of a higher-performance proxy over several web servers.

Hosting of web servers on virtual hardware is also common. One way to apply the power-saving scheme would be to suspend and resume the virtual hardware. This may result in indirect power savings because of lower resource usage on the physical host, though we expect the savings would be smaller unless enough virtual hosts were suspended to allow a physical machine to sleep. This might be facilitated by allowing migration of suspended virtual machines to reduce the number of physical hosts currently required.

4.8 Conclusion

We have explored the use of a low-powered reverse proxy to save power by powering off a web server leaving the reverse proxy on. We have considered features of web traffic that facilitate this, including traffic patterns and prevalence of spiders. We demonstrated that if a server is not too busy and can recover from a low-power state quickly (say 5 s to 10 s) then it is possible to save power by putting a high-power web server to sleep during low activity periods. In our testbed, we were able to save 25% energy while keeping performance at an acceptable level.

CHAPTER 5

Energy Efficiency in Virtual Machines, Linux Containers and on Physical Hosts

Increasingly the world of computing is becoming more disconnected from what our ideas of what a computer is, we are now seeing a world where a computer can be an abstraction made of software that runs in software on computers that host many other software computers. This is called virtualisation and this holds promise for a whole new world of possibilities. In this chapter we look closer at two technologies used in this area and see what impacts, if any, these have on the efficient use of energy.

5.1 Introduction

A virtual machine is a software layer which emulates physical hardware. This virtual hardware can have a full operating system installed on it and a piece of software called a hypervisor manages this environment and the interactions between the virtual environment and the computer that it is hosted on. The hypervisor also allows for multiple virtual machines to be hosted together on one physical host machine. Virtual machines are useful for a huge number of applications, including testing applications on many different versions of operating systems, emulating obsolete hardware to run legacy code, developing

software for new processor architectures that aren't yet available, isolating processes when hosting multiple applications on a small number of physical machines or hosting applications in the cloud. A diagram outlining the general structure of virtual machines is shown in figure 5.1.

In recent years virtualisation has become an increasingly integral part of the modern computing landscape. Loosening the coupling between the physical hardware and the software that runs on it has enabled operators to improve resource utilisation by hosting many virtual machines on one physical host as well as offering virtual machine hosting as part of their Infrastructure as a service products.

Operating-system-level virtualisation is a type of virtualisation which is becoming popular for some server side applications [173, 149, 124, 89, 132]. Instead of a hypervisor creating a virtual hardware platform and installing an operating system to manage it, operating-system-level virtualisation creates isolated userspace instances which share the host kernel and operating system. These can co-exist with other virtualisation solutions if required.

One version of these userspace instances are called Linux containers. These are used to run applications and contain all the applications dependencies, making this technology well suited for packaging applications for many deployment scenarios that are common for cloud applications. Similar technologies have existed before such as chroots [50], also known as *root jails*, available in many flavours of Unix, but chroots only restrict the applications contained inside to a sub-tree of the host machines file system, whereas Linux containers allow more fine grained control over resources (CPU, block I/O, memory, networking, etc).

Creating, managing and generally orchestrating Linux containers are facilitated by the use of some software tools to automate these processes. One of the more popular solutions for this is called *Docker* [114]. We give a general outline of the software architecture of containers in figure 5.2.

While virtual machines and Linux containers have some things in common, they are inherently different beasts, but they can be used to achieve the same goals (such as process isolation and improved hardware utilisation) in some cases. This begs the question: *if the situation allows for the use of either a*

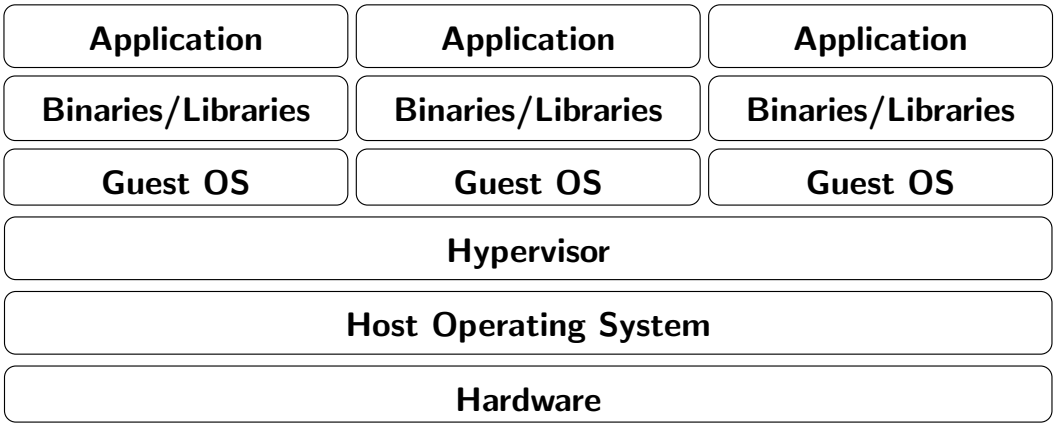


Figure 5.1: A simplified diagram of the software architecture of virtual machines

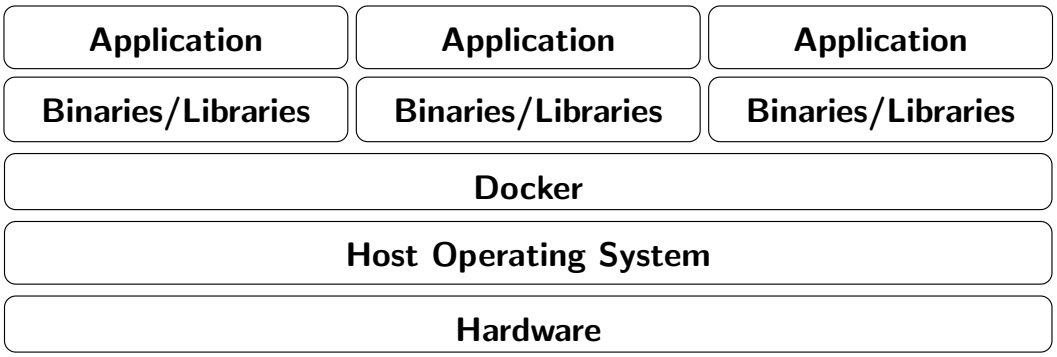


Figure 5.2: A simplified diagram of the software architecture of containers

virtual machine or a Linux container, which is more energy efficient?

This chapter outlines our attempts to answer this question.

5.2 Experiments

To make managing Linux containers easier, a number of completing and complimentary tools have been developed. Docker is what we have used for our investigation [114].

Virtual machines are a mature technology and as we have seen in chapter 2, they have been the topic of a considerable amount of research. For our experiments we will use Xen, a virtualisation hypervisor which is well supported, actively developed and used across industry & academia [23]. Relational Database Management Systems (RDMSs) are a commonly used and mature technology, these are used across a wide number of areas and even though they can provide some access control to the data they store in some cases it might make sense

to run separate instances in isolation. RDMSs often include tools to measure or benchmark the performance of the service. We used PostgreSQL and the benchmarking tool pgbench for these experiments [151].

The parameters used in the experiments are outlined in table 5.1. With experiments 0, 1 & 2, looking at the impact of the size of the database. Experiments 3 and 4 look at the same database configuration as experiment 1, only changing the type of the query from the set of selection and update queries seen in listing 5.1 to using only selection and in the case of experiment 4 using PostgreSQLs prepared statements, which are queries written in Structured Query Language (SQL) which are parsed and optimised ahead of when they are executed and are expected to be called multiple times.

We think that experiment 0 should provide a simple baseline, the small size of the database should mean that it should be able to fit into memory comfortably and shouldn't be adversely impacted by differences in the performance of disc I/O. Experiments 1 & 2 increase the scaling size and therefore the size of the database considerably. This should increase the stress on disk I/O and reveal any direct performance bottle necks owing to this.

Experiments 3 & 4 alter the type of query used, with 3 only reading rows from the database and not updating any values, which should indicate the performance of these systems to read from the disc. Experiment 4 makes use of the prepared statement feature of PostgreSQL and should lead to better overall performance.

The "Scaling Factor" column in table 5.1 refers to the size of the database. The total size can be attained by multiplying the scale by 16MB.

```
\set nbranches :scale
\set ntellers 10 * :scale
\set naccounts 100000 * :scale
\setrandom aid 1 :naccounts
\setrandom bid 1 :nbranches
\setrandom tid 1 :ntellers
\setrandom delta -5000 5000
BEGIN;
UPDATE pgbench_accounts SET abalance = abalance + :delta
```

```
    WHERE aid = :aid;
SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
UPDATE pgbench_tellers SET tbalance = tbalance + :delta
    WHERE tid = :tid;
UPDATE pgbench_branches SET bbalance = bbalance + :delta
    WHERE bid = :bid;
INSERT INTO pgbench_history (tid, bid, aid, delta, mtime
    ) VALUES (:tid, :bid, :aid, :delta, CURRENT_TIMESTAMP
    );
END;
```

Listing 5.1: The SQL used in experiments 0-2

Experiment	Size	Scaling Factor	Query Type
0	240MB	15	Simple
1	1.12GB	70	Simple
2	9.6GB	600	Simple
3	1.12GB	70	Selection Only
4	1.12GB	70	Prepared Statements

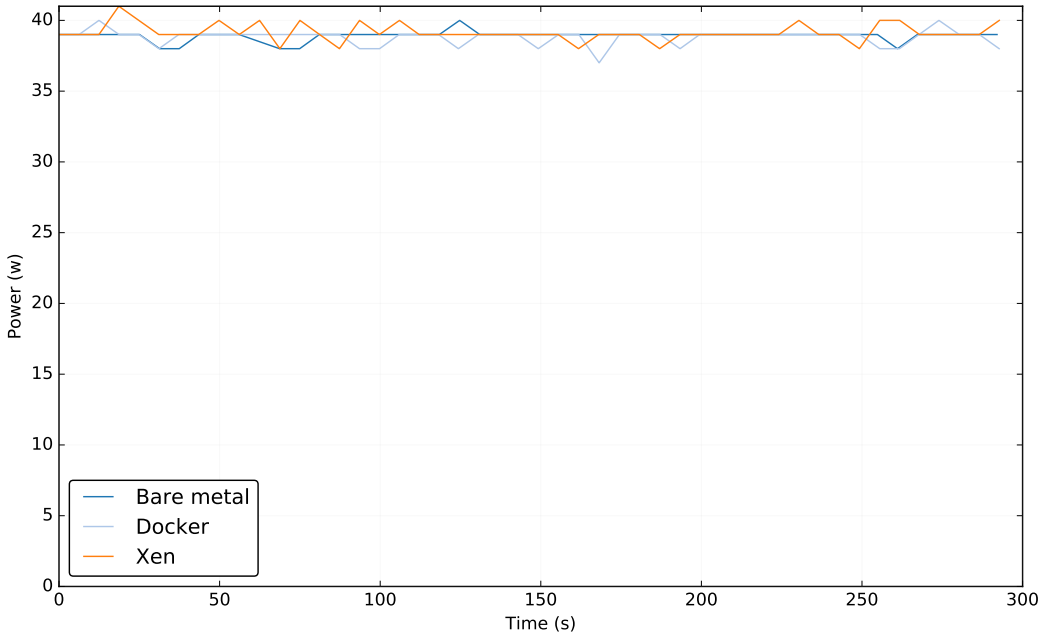
Table 5.1: Description of experiment parameters

We ran each experiment for 300 seconds and measured the power consumed by using the Current Cost EnviR described in section 3.2.2. The physical machine used was a Dell PowerEdge R320 II with an Intel Pentium 1403 CPU (which has two CPU cores and 5MB of cache and a clock speed of 2.6GHz) and 4GB of RAM, running Ubuntu 14.04.4 with version 3.19.0-59 of the Linux Kernel, Xen version 4.4.2 and Docker 1.11.1.

The Xen virtual machine was running Ubuntu 14.04.4 as the operating system with one virtual CPU and was given 512MB of RAM. The container was configured using Docker and was allowed a maximum of 512MB of RAM and was pinned to run on the second physical CPU core to allow for a more fair comparison with the virtual machine.

The final platform we tested on was the physical machine itself which we refer to as bare metal.

Each of the platforms tested ran PostgreSQL version 9.1 and the pgbench tool



5.3.1: Power consumed over time for experiment 0

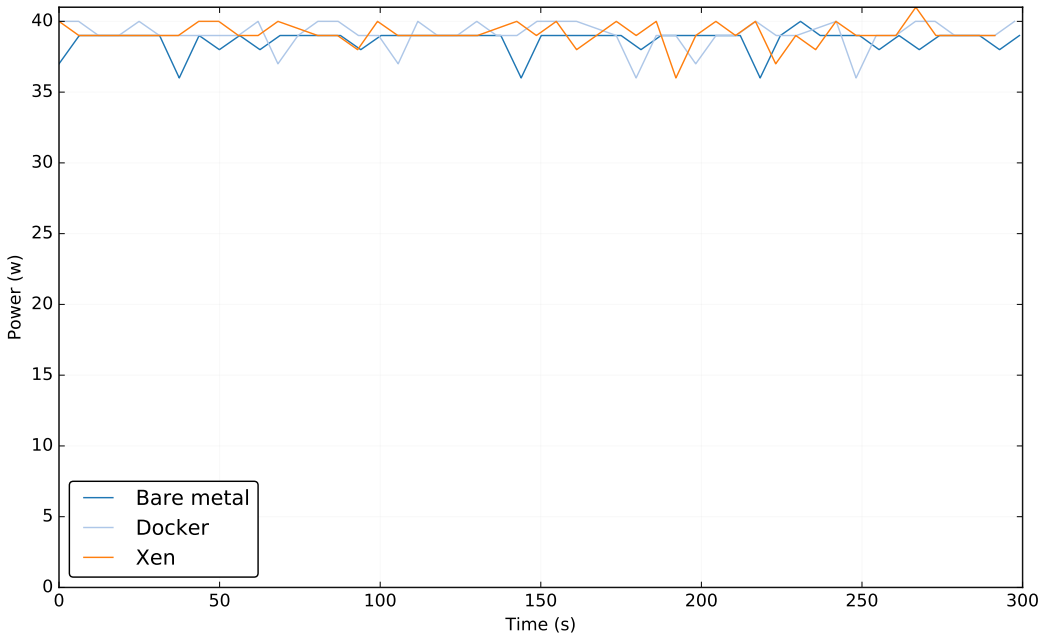
was run on a separate computer networked with the physical machine using Ethernet over a dedicated 100Mbitps LAN. With pgbench configured to run for 300s and simulate two clients using a single thread.

For each experiment we measured the power consumed over time, sampled at 7s intervals using the Current Cost EnviR seen in section 3.2.2, and the performance of PostgreSQL in transactions per second, from this we were able to get four sets of graphs, power usage over time, total power consumption, how many joules of energy were used per transaction and how many transactions were completed per second.

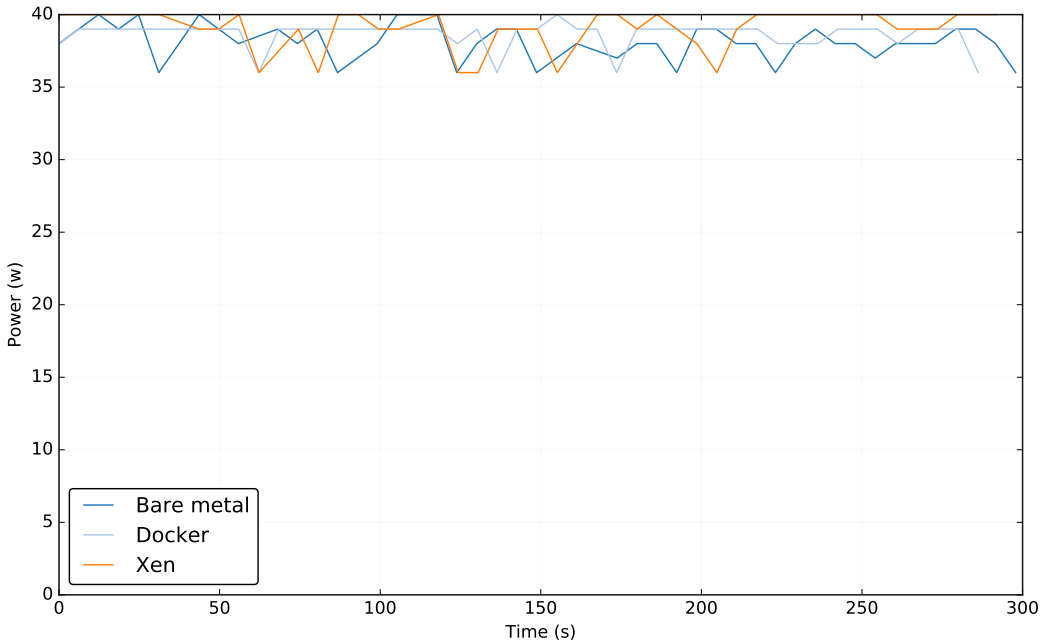
5.3 Initial Results

As we see in figures 5.3.1 to 5.3.5, which show the power consumption over time, the power consumption was relatively flat across all the experiments, with variations on the scale of approximately 10%, since these variations over time are small, it is reasonable to summarise the results with an average.

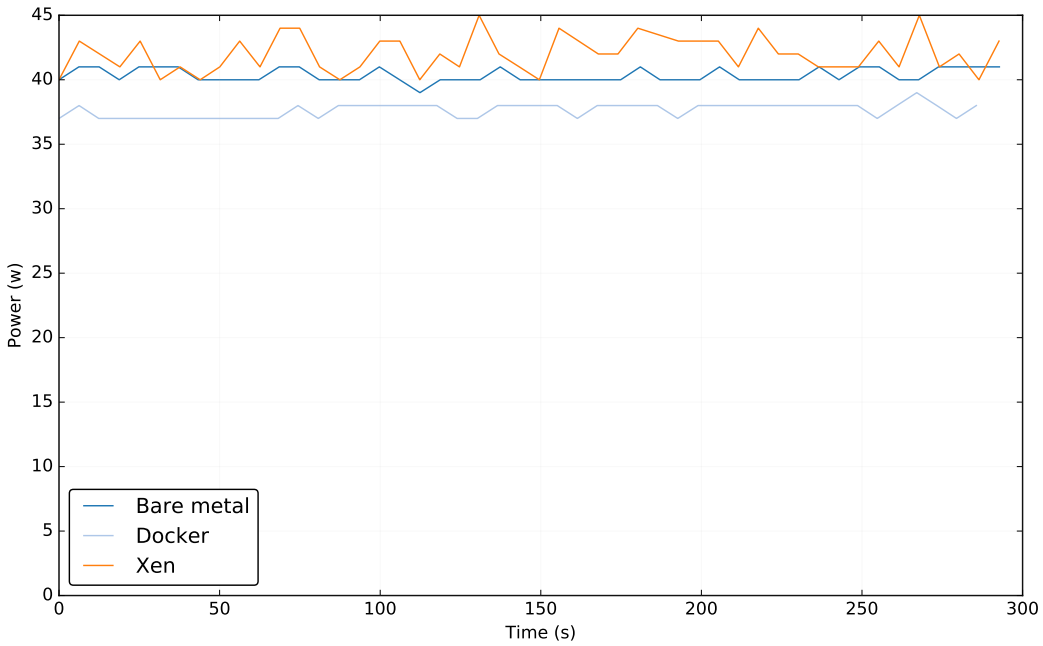
In figures 5.4.1 to 5.4.5 we can see the total power used in each experiment in Joules. These figures make the comparison between Bare Metal, Docker & Xen more straightforward. Figure 5.4.1 shows a flat total power usage whereas



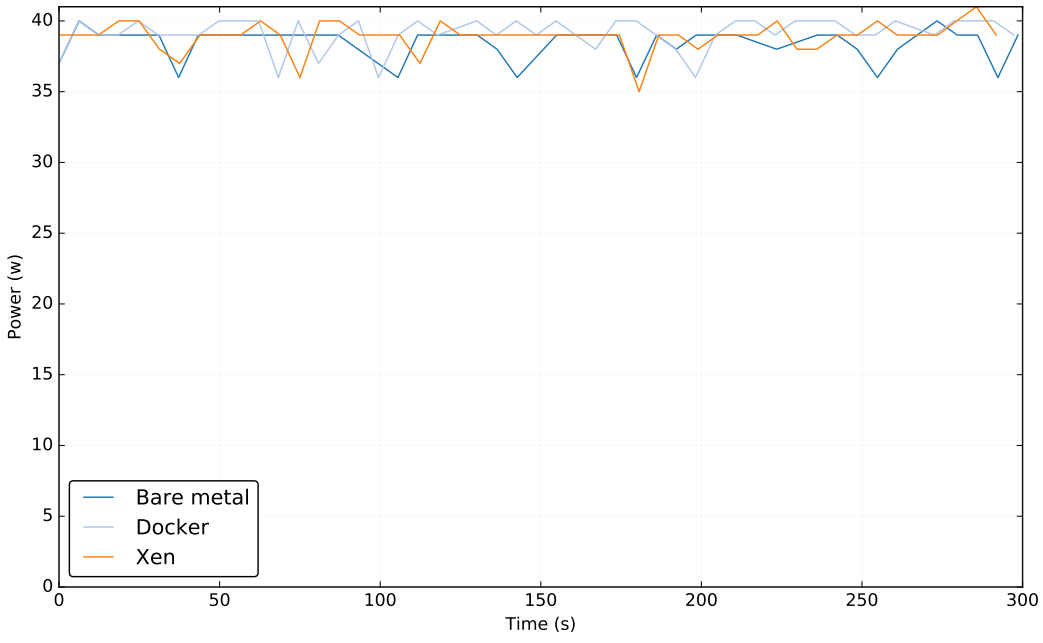
5.3.2: Power consumed over time for experiment 1



5.3.3: Power consumed over time for experiment 2

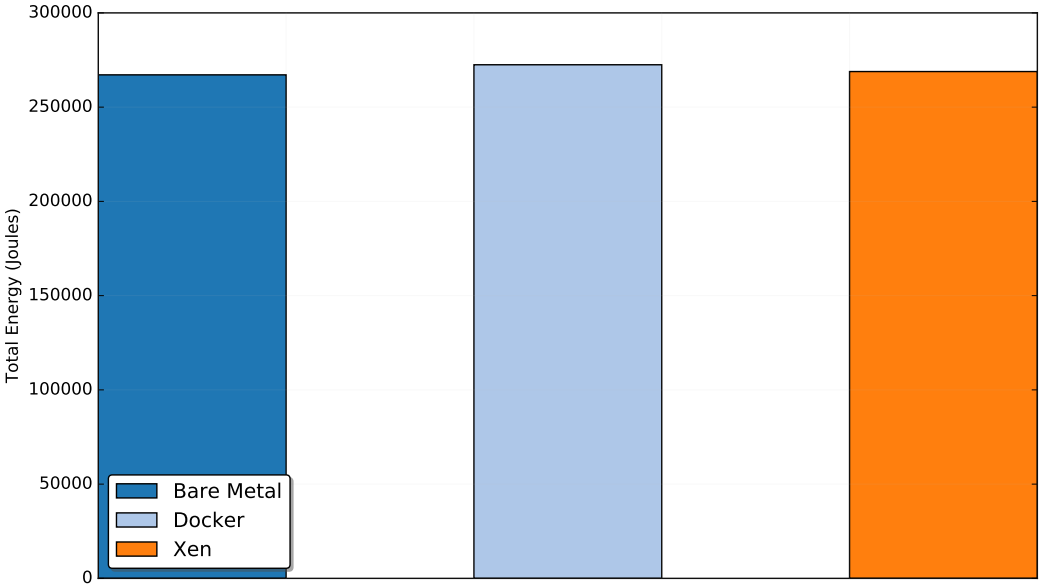


5.3.4: Power consumed over time for experiment 3

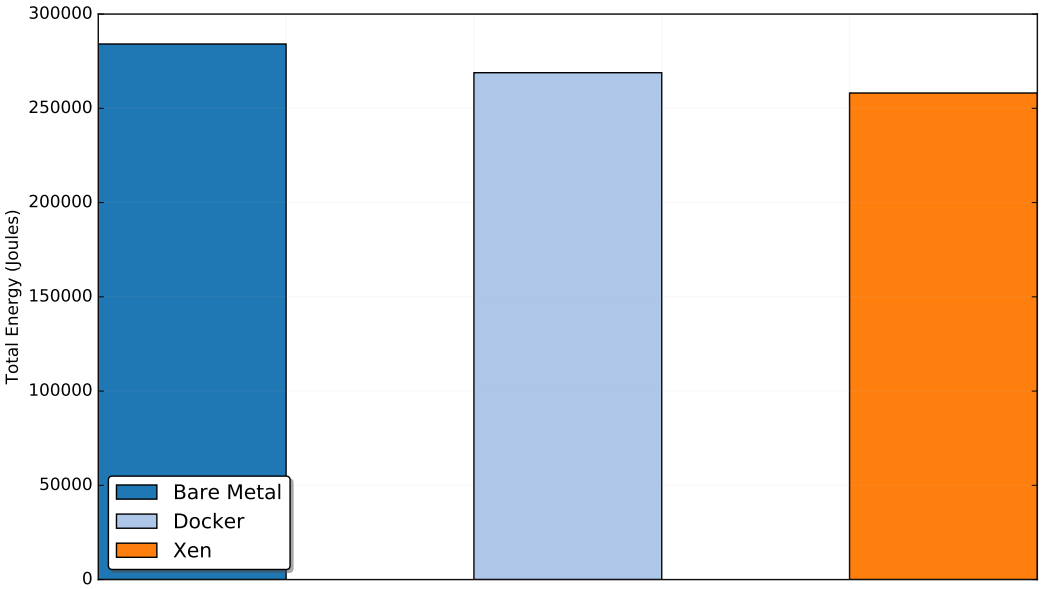


5.3.5: Power consumed over time for experiment 4

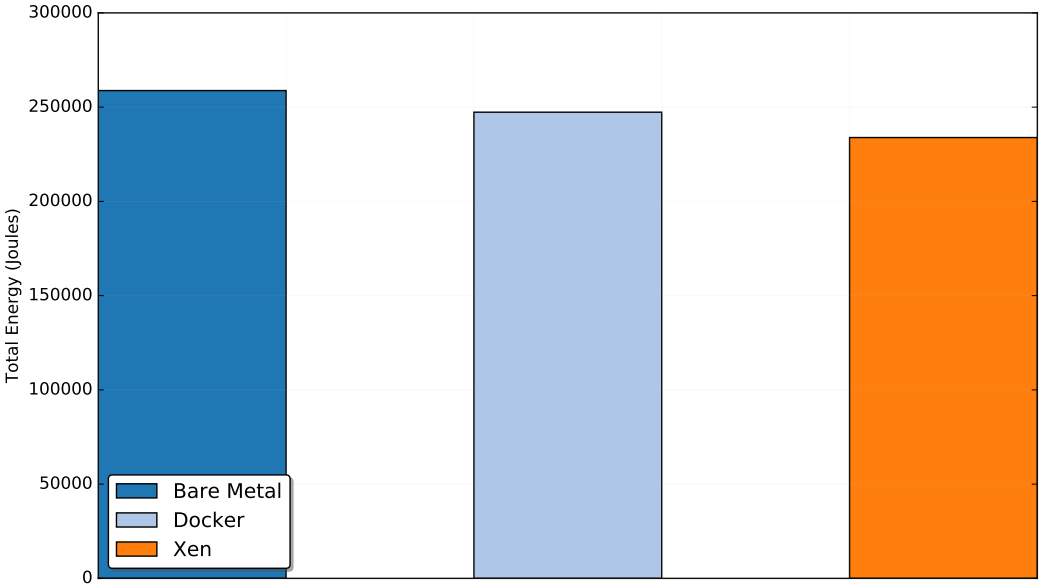
Figure 5.3: Power consumption over time for experiments 0 to 4



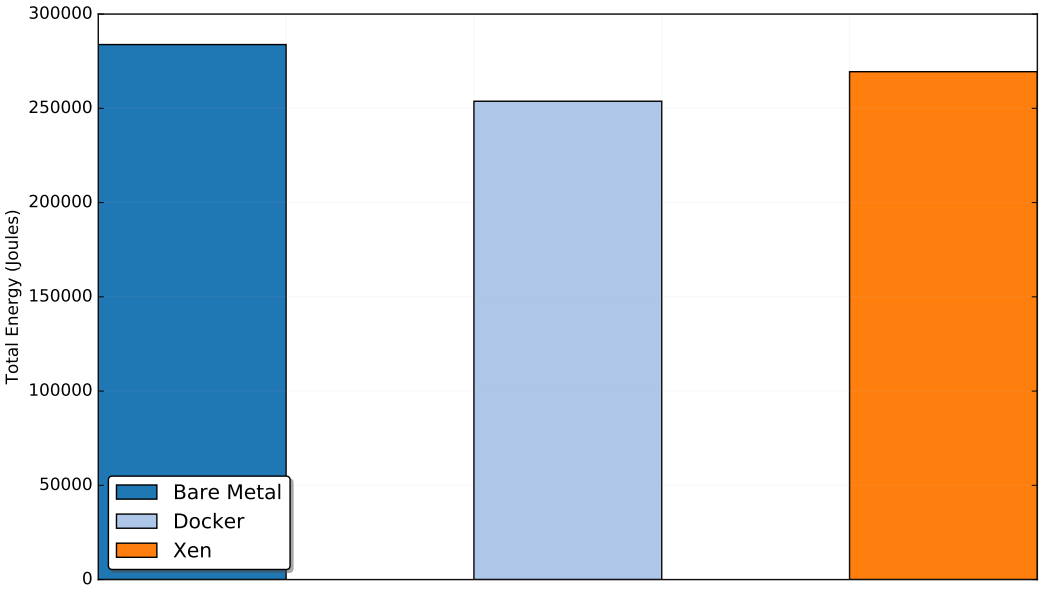
5.4.1: Total energy consumed in experiment 0



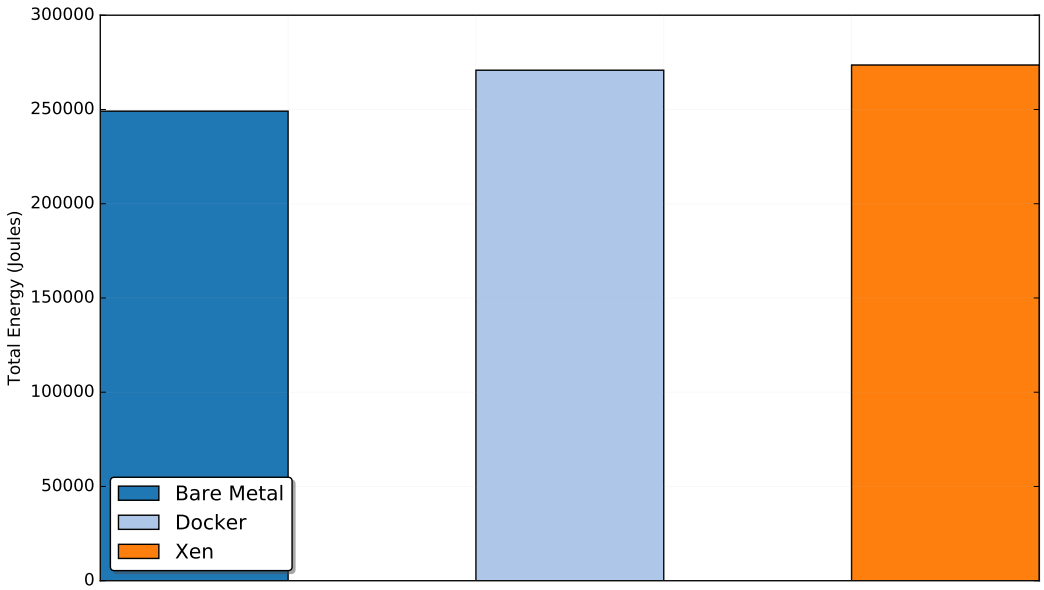
5.4.2: Total energy consumed in experiment 1



5.4.3: Total energy consumed in experiment 2

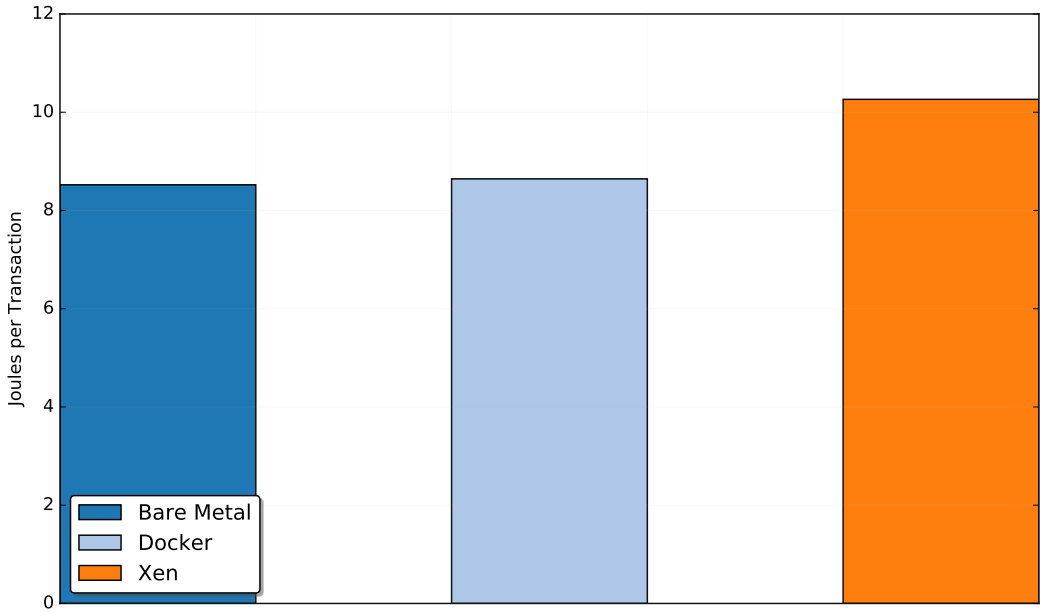


5.4.4: Total energy consumed in experiment 3

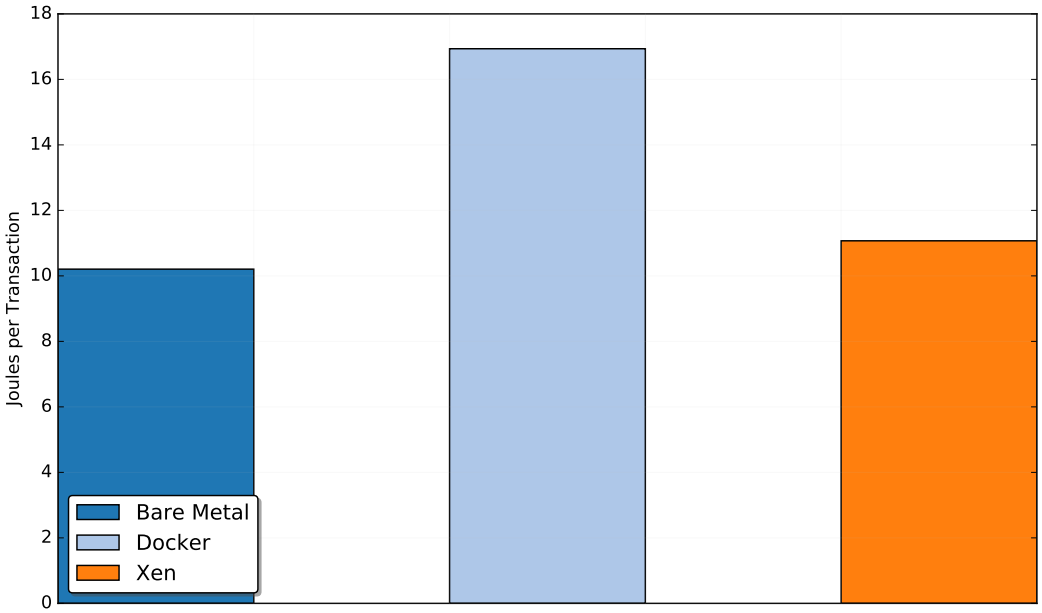


5.4.5: Total energy consumed in experiment 4

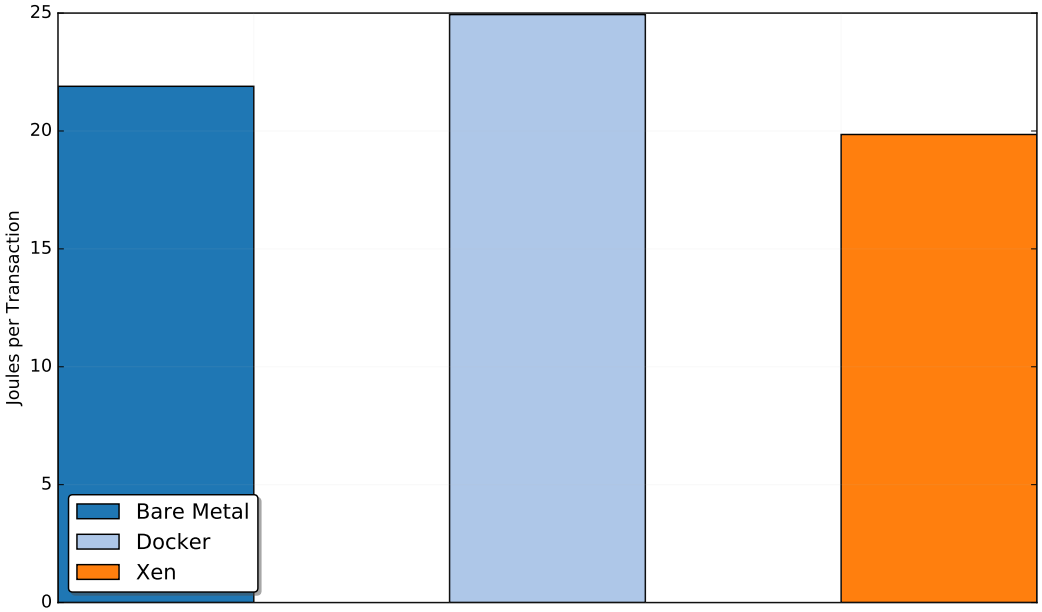
Figure 5.4: Total energy consumption for experiments 0 to 4



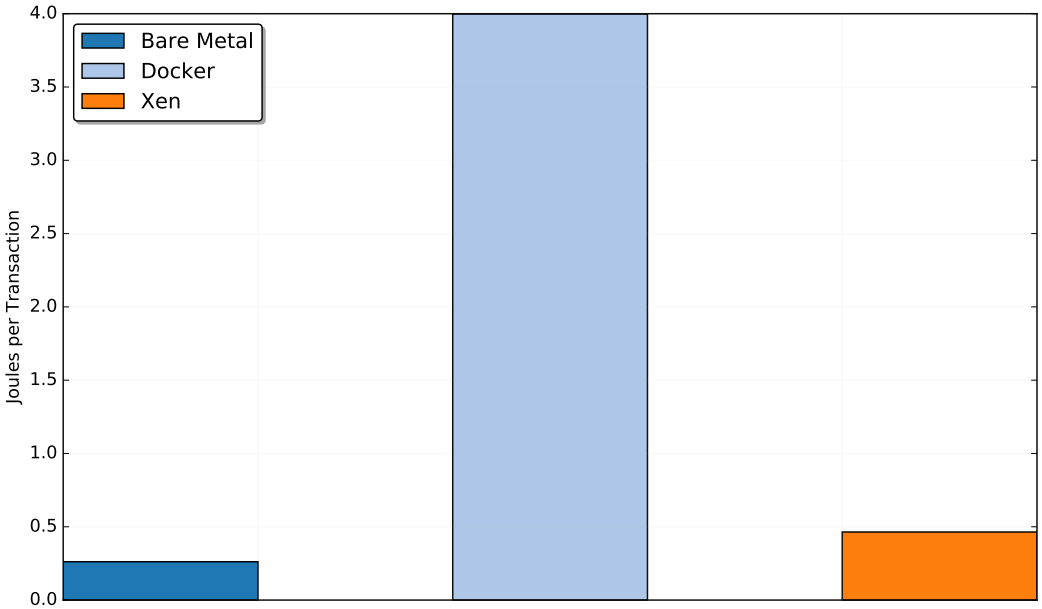
5.5.1: Joules per transaction in experiment 0



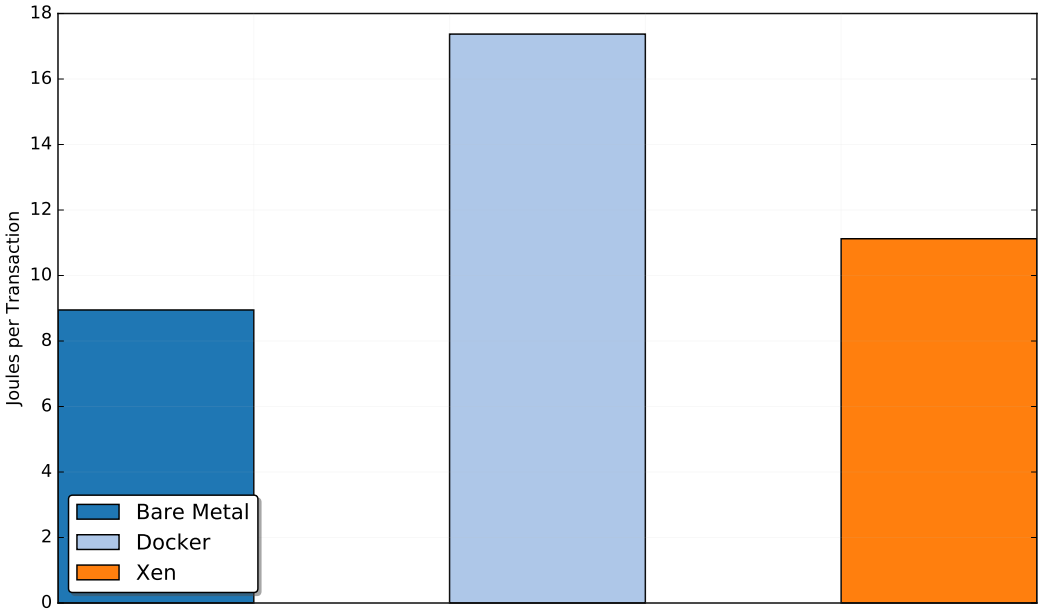
5.5.2: Joules per transaction in experiment 1



5.5.3: Joules per transaction in experiment 2

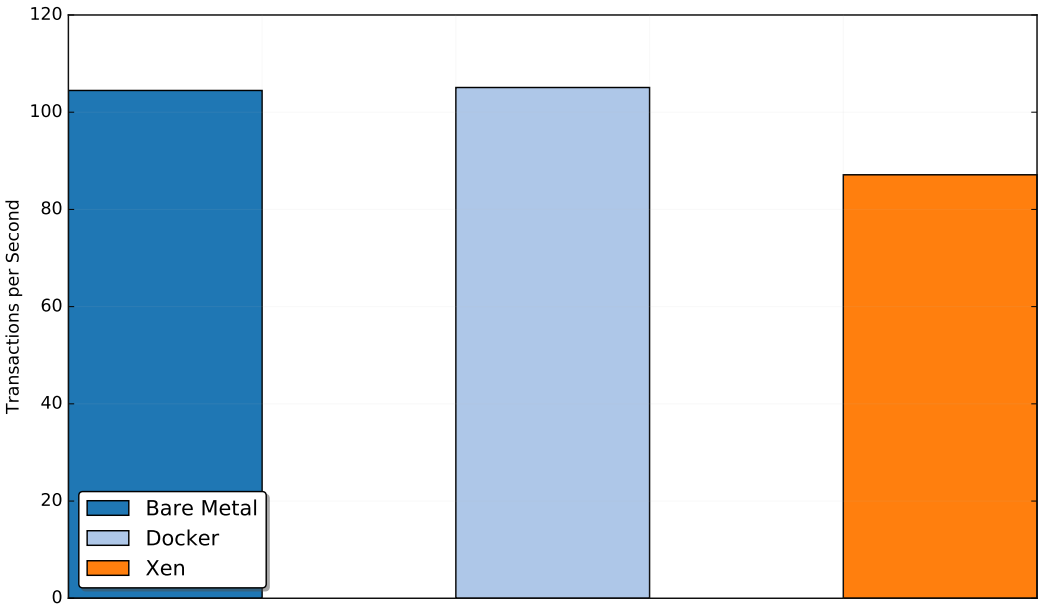


5.5.4: Joules per transaction in experiment 3

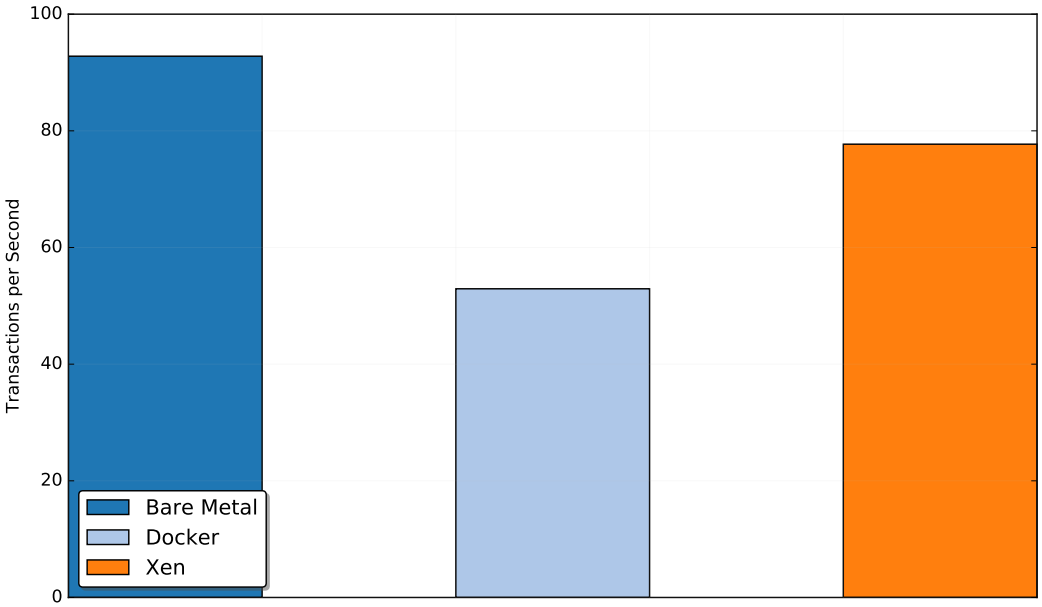


5.5.5: Joules per transaction in experiment 4

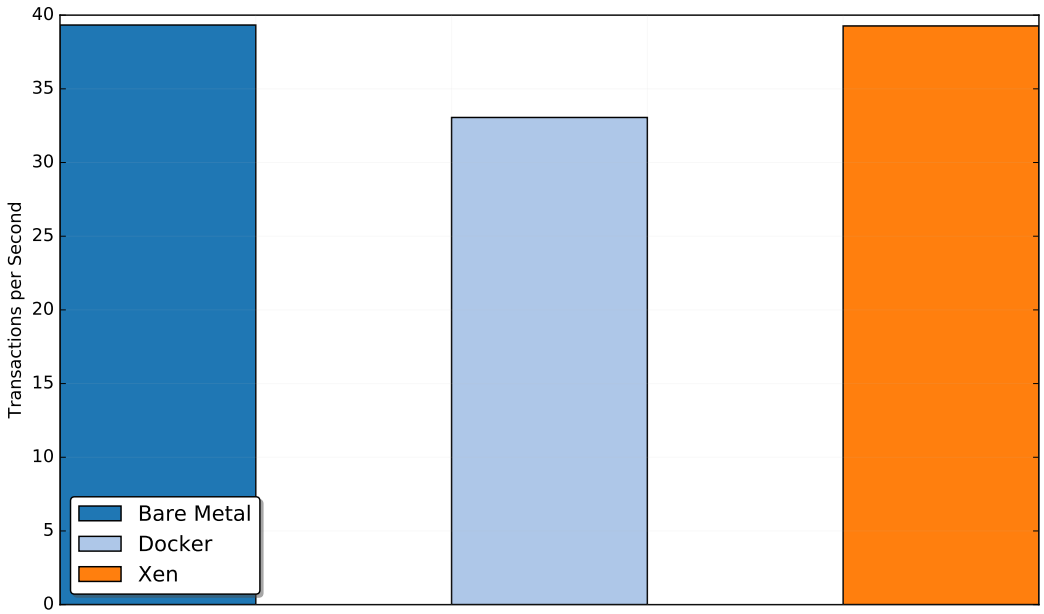
Figure 5.5: Average number of Joules consumed per transaction in experiments 0 to 4



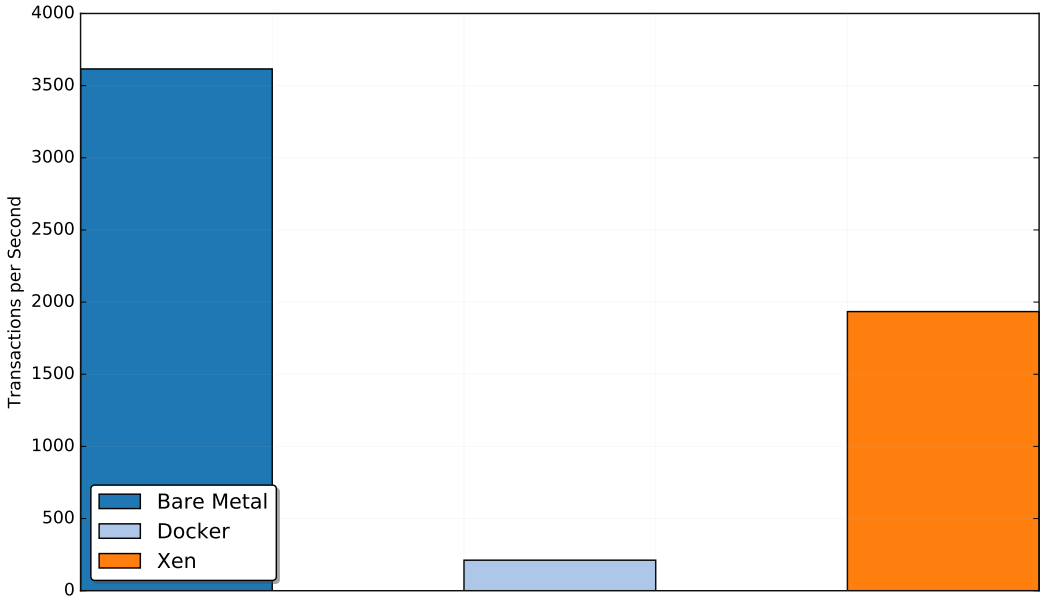
5.6.1: Transactions per second in experiment 0



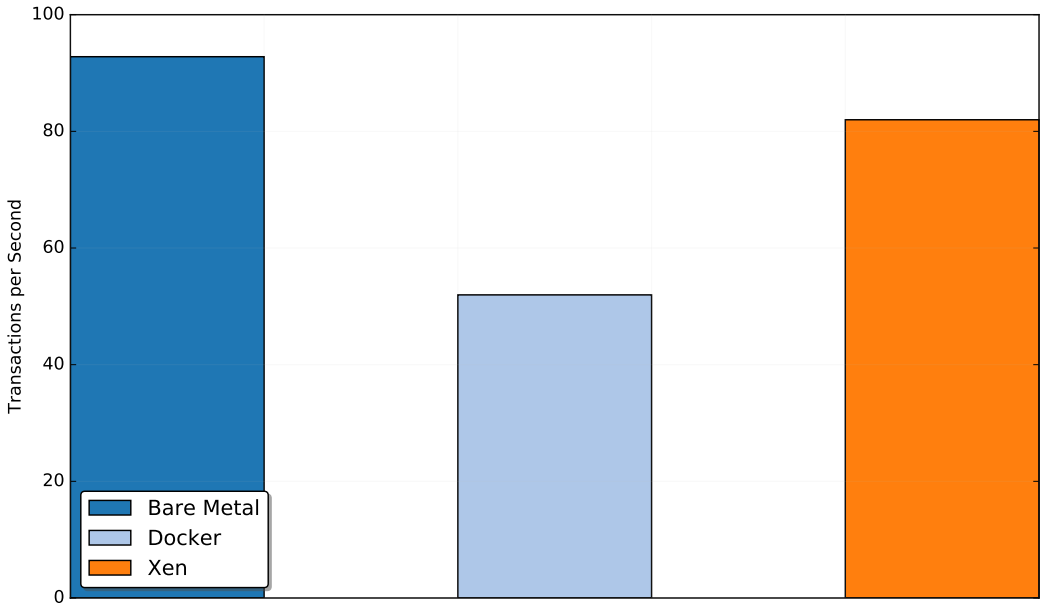
5.6.2: Transactions per second in experiment 1



5.6.3: Transactions per second in experiment 2



5.6.4: Transactions per second in experiment 3



5.6.5: Transactions per second in experiment 4

Figure 5.6: Mean number of transactions per second in experiments 0 to 4

figures 5.4.2 and 5.4.3 show only small difference in total energy consumption between all three platforms. In figure 5.4.4 there is a similar difference but the order is reversed with Docker using the least amount of energy in total. In figure 5.4.5 we see while there is the same small gap in energy consumption, Bare Metal consumed the least and Docker and Xen are close to being even.

However, total power usage does not tell the full story.

Figures 5.5.1 to 5.5.5 show Joules per transaction across the experiments. Bare metal uses the least joules per transaction in each experiment bar figure 5.5.3. This is expected because we believe virtualisation to introduce overhead in terms of power usage. Xen uses more energy for each database transaction than bare metal except in figure 5.5.3. Docker seems to use more energy than Bare Metal and Xen except in figure 5.5.1, which appears at first to be an anomaly.

To understand these results, Figures 5.6.1 to 5.6.5 show the number of transactions per second of each experiment, these figures include the connections to the database. Bare Metal performs best by this measure, with Docker coming in last in all experiments but figure 5.6.1. These figures confirm that the dif-

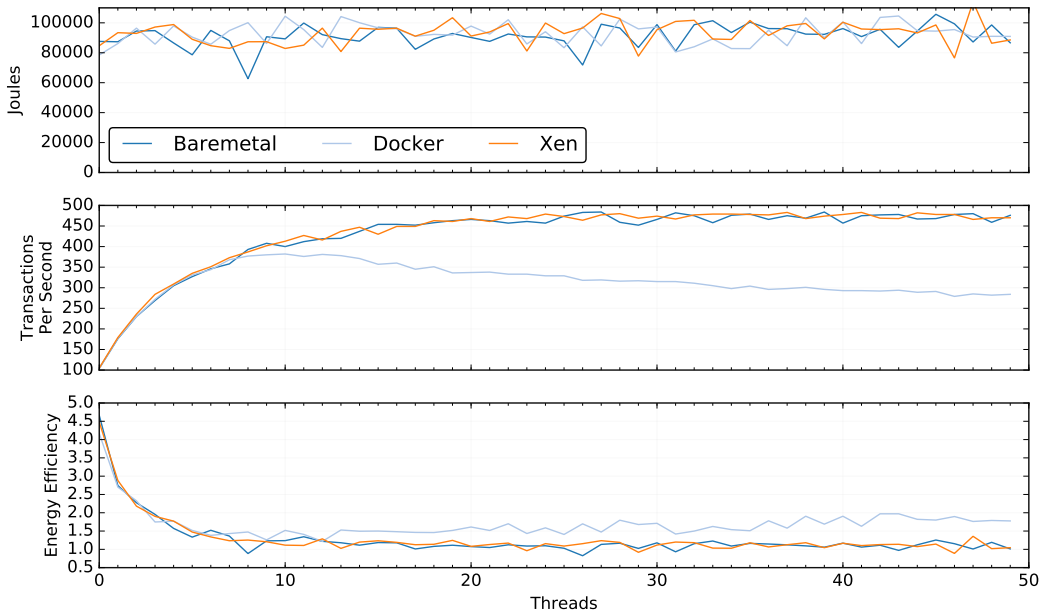


Figure 5.7: Experiment 0 repeated with an increasing number of threads

ference is energy efficiency per transaction which arises from the performance of PostgreSQL under the different virtualisation systems.

5.4 Scaling up the Experiments

Building on the first set of experiments we wondered if the amount of concurrent requests changed the results and if so, how? By scaling up these experiments they might become bound by a different resource (for example CPU access or I/O) and would test other aspects of virtualisation. The experiments were repeated but the number of clients and threads that pgbench used were increased with each repetition from 1 thread simulating 2 clients until pgbench used 50 simultaneous threads of execution to simulate 100 clients.

Figures 5.7 to 5.11 show the total Joules consumed, the number of transactions per second and the energy efficiency in Joules per transaction.

As we can see in figure 5.7 up until 6 threads, there is no real difference between bare metal, Xen and docker, but the number of transactions per second decline for Docker. Figure 5.8 shows that Docker's performance increases slowly with the number of threads, peeks early and slowly declines, whereas Xen and Bare Metal reach their peak performance around ten threads and stay around 300 to 350 transactions per second. A similar result is seen in figures 5.9 and 5.11.

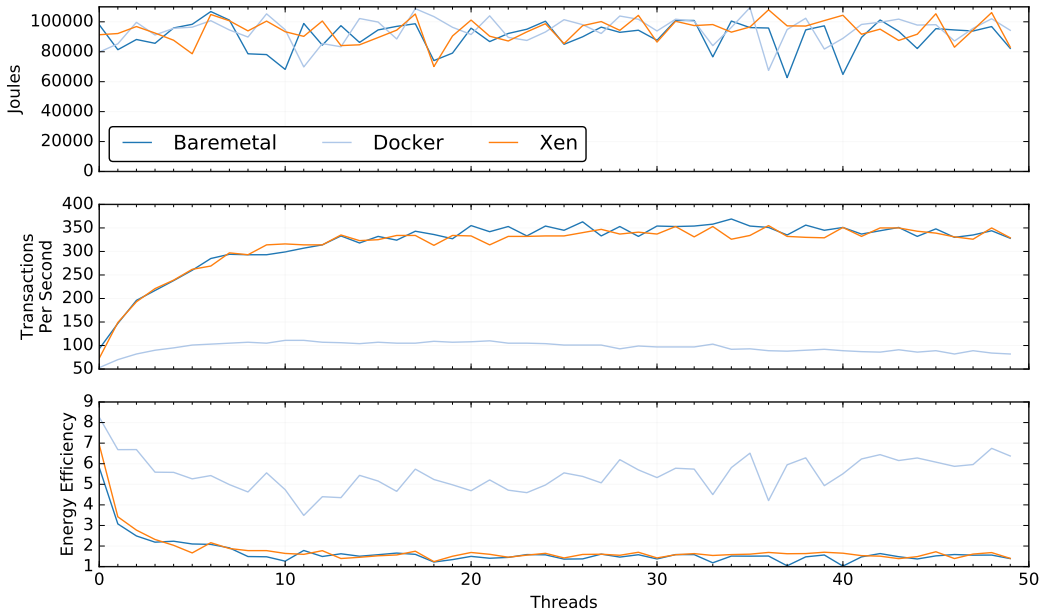


Figure 5.8: Experiment 1 repeated with an increasing number of threads

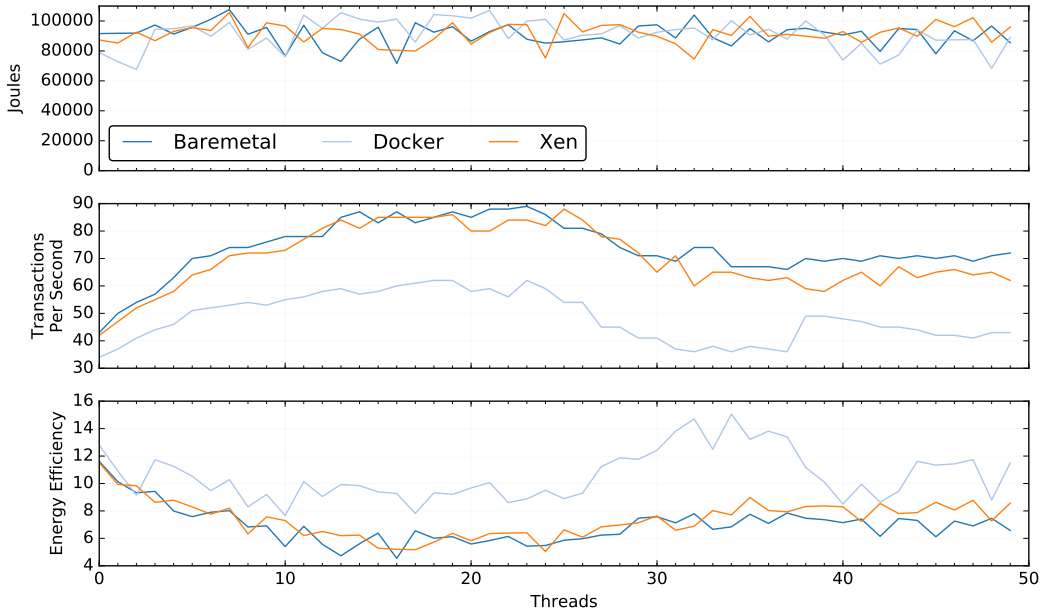


Figure 5.9: Experiment 2 repeated with an increasing number of threads

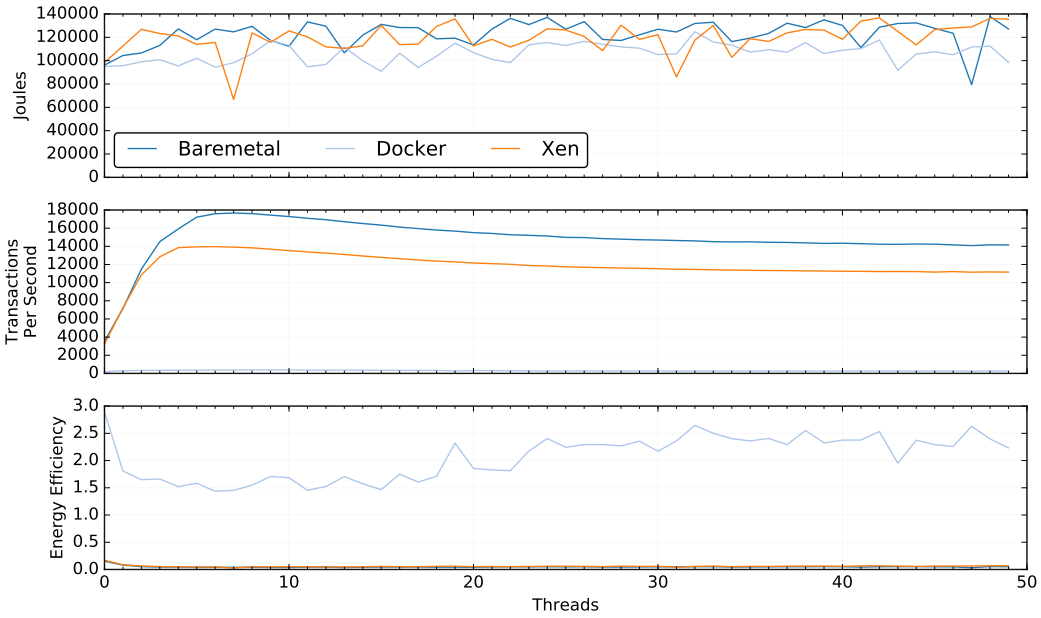


Figure 5.10: Experiment 3 repeated with an increasing number of threads

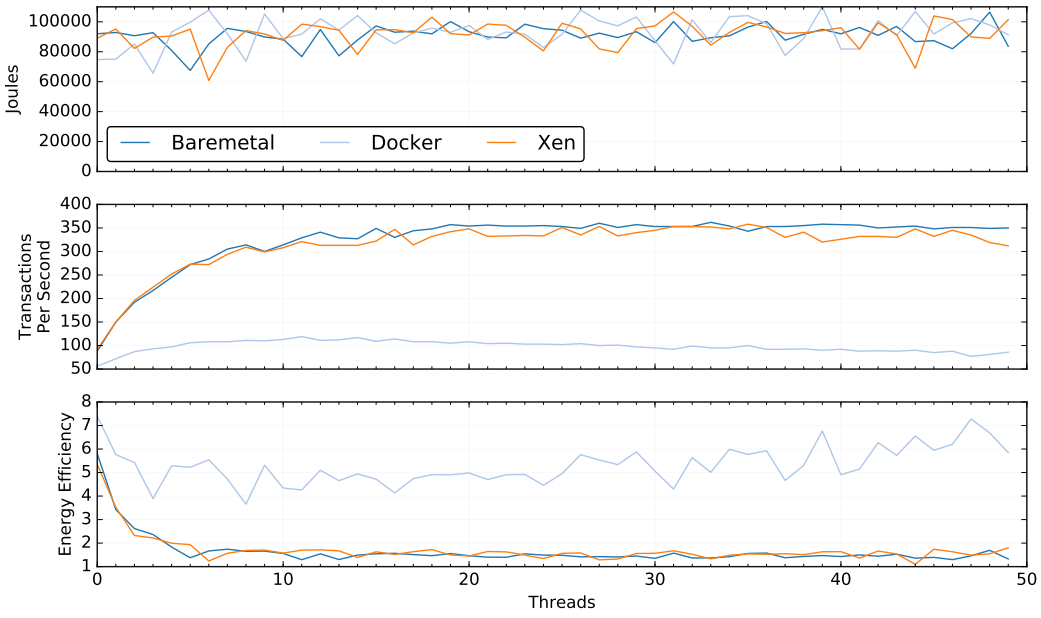


Figure 5.11: Experiment 4 repeated with an increasing number of threads

Figure 5.10 isn't very surprising when compared with figure 5.6.4 which shows that Docker didn't perform favourably when compared with Xen and Bare Metal, but when scaled up with larger number of threads we see that Xen and Bare Metals transactions per second diverge and then seem to follow each other by some offset of a few thousand transactions per second.

This seems odd that the performance of the Docker container is measurably worse than the virtual machine and bare metal in the majority of our experiments. The Docker system is relatively simpler by comparison to the virtual machine, which would make us believe that without the work of the hypervisor and the virtual hardware layer Docker would have fewer constraints and at least achieve a comparable performance across all the benchmarks. We can see this if we look at figures 5.1 and 5.2, which show an outline of the respective software architectures of containers and virtual machines. These gaps in performance warrant further investigation, which is what we will do in the following sections.

5.5 Understanding the Results

These results were interesting but slightly unexpected. We made every effort to reduce the possibility of misconfiguration, but perhaps this was caused by a misconfiguration or maybe it is something else we failed to consider. To better understand the implications of these tests we will look at the performance of the CPU, RAM, Disk I/O and Networking under Docker, Xen and on the host. We keep the configuration of Docker and Xen the same in order to better understand the results shown in sections 5.3 and 5.4

5.5.1 CPU

The sharing CPU schemes used by Docker and Xen are similar and when we measure CPU performance using SysBench version 0.4.2 [4], it offers a straightforward means of testing CPU performance by calculating all the prime numbers up to a given number, in this case 20000 and uses a single thread [1]. We can see the results of this in table 5.2.

We see only a very small difference between the three. But it is not enough to explain the differences observed in the previous section.

[1] There are 2262 prime numbers between 0 and 20000

	Bare Metal	Xen	Docker
Total time taken (s)	33.05	33.05	32.50
Per-request statistics (ms)			
Minimum	3.25	3.28	3.24
Average	3.30	3.31	3.25
Maximum	7.36	6.99	21.22
95^{th} Percentile	3.31	3.31	3.25

Table 5.2: CPU benchmark experiments

	Bare Metal	Xen	Docker
Throughput (MBps)	631.02	373.18	363.79

Table 5.3: RAM benchmark experiments

5.5.2 RAM

Access to RAM is important in all of the scenarios we tested. To understand this better we used the SysBench tool and used its memory benchmark, this test allocates a buffer and with each operation it writes or reads to this memory in a sequential or random manner using a specified number of threads.

For this first experiment we use a buffer size of 1kB and just write to this buffer sequentially, using 4 threads, each experiment wrote 102400MB into RAM.

The results of this first experiment are shown in table 5.3. As we can see running the test on the Bare Metal performs considerably better than the Xen and Docker instances, which in part is explained by the host machine having access to all the available RAM but also having access to both CPU cores.

5.5.3 Disk IO

Reading and writing to a hard disk can often be the source of many bottlenecks. We wished to see what difference the means of abstracting access to the disk has on the performance of the Xen virtual machine and on the application in the Linux container created and managed by Docker. If we look at figures 5.1 and 5.2 we can see that there are different layers of abstraction between the application and the hardware which is executing it. Understanding the impact of these differences may explain the performance disparity we observed.

	Bare Metal	Xen	Docker
Disk Seek Rate			
Requests Completed	914	28 700	905
I/O Operations per Second	305	10 000	302
Throughput (MBps)	1.2	39.0	1.18
Request Time Statistics			
Minimum (ms)	0.114	0.093	0.115
Average	3.3	0.100	3.31
Maximum	0.434	17.3	18.3
95^{th} Percentile	3.6	0.008	3.56
Sequential Read Speed			
Requests Completed	785	9200	814
I/O Operations per Second	265	3200	275
Throughput (MBps)	66.4	809.5	68.8
Request Time Statistics			
Minimum (ms)	1.2	0.264	1.15
Average	3.8	0.308	3.63
Maximum	29.2	3.2	19.8
95^{th} Percentile	2.3	0.047	1.72

Table 5.4: I/O latency benchmark experiments

We begin by looking at I/O latency, table 5.4 shows the results of using the ioping tool [2] to measure disk seek rate, which tries to measure how many I/O operations per second the hard disk can support, and sequential read speed, which measures the transfer rate for reading large contiguous blocks of data from the disk.

Looking first at disk seek rate, the important results are the I/O operations per second and the average time taken for each I/O operation. As we can see in table 5.4, Bare Metal and Docker have very similar results. But the Xen VM out performs both by quite a substantial margin.

Looking at sequential read speed the important results are the measurements for the throughput. Again we see Bare Metal and Docker have very similar results. But the Xen virtual machine is clearly outperforming the others.

It's worth keeping in mind that all of these are using the exact same physical hard disk drive. The close performance numbers between Docker and Bare Metal would be expected when you consider these are just two processes

accessing the disk using the same kernel, the main difference being that the Linux kernels cgroups feature might be adding small amount of overhead to the processes running in the Docker container.

Meanwhile for the Xen virtual machine, it is a complete operating system contained by the hypervisor. The guest operating system is using a virtual disk which is hosted in a file on the host operating system disk. The hypervisor may be providing some buffering between the guest operating system and the hard disk drive of the host operating system.

To look a bit further at this, we used the SysBench tool to measure the performance of disk I/O. We look at measuring the performance of reading and writing to the disk both in a sequential and random fashion. In our tests we used a total file size of 10GB, synchronous disk access mode and a maximum test time of 300 seconds. The total file size was chosen as it was close to the size of the largest database generated in the previous experiments and is considerably larger than the RAM available to any of the platforms tested, so this will force physical disk accesses. The test scenarios we decided to examine were: sequential reads from the disk, sequential writes to the disk, random writes, and combined random read & writes.

In terms of disk I/O sequential reads and writes refers to accessing locations on the disk in a contiguous manner, whereas random refers to accessing locations on the disk in a non-contiguous manner.

Table 5.5 shows the performance of Bare Metal, Xen and Docker at doing sequential read operations from the disk. As we can see, Bare Metal and Docker have comparable performance, where as Xen completes the task with fewer read operations and these operations executed faster on average.

Table 5.6 shows the performance of sequential disk writes. In this test the performance gap between Xen and the other two platforms appears to narrow a bit. It is still performing with a higher number of requests executed per second, but the average length of request times are closer.

Table 5.7 shows the performance of the three platforms at carrying out random reads and writes. In this case the number of requests executed per second was lower for Xen.

	Bare Metal	Xen	Docker
Sequential Read			
Operations Performed			
Read	655 360	453 729	655 360
Write	0	0	0
Other	0	0	0
Requests/sec Executed	6615.72	8001.36	7143.86
Request Time Statistics			
Minimum (ms)	0.01	0.00	0.01
Average	0.15	0.12	0.14
Maximum	116.98	78.21	105.30
95^{th} Percentile	0.75	0.62	0.80

Table 5.5: Sequential disk reading benchmark experiments

	Bare Metal	Xen	Docker
Sequential Write			
Operations Performed			
Read	0	0	0
Write	655 360	453 729	655 360
Other	128	0	128
Requests/sec Executed	6266.59	8233.05	7039.38
Request Time Statistics			
Minimum (ms)	0.01	0.01	0.01
Average	0.15	0.12	0.13
Maximum	5025.61	4438.04	4080.87
95^{th} Percentile	0.03	0.02	0.05

Table 5.6: Sequential write disk benchmarking experiments

Table 5.8 shows the performance of random writes. In contrast to tables 5.5 and 5.6 which shows the performance of doing disk operations sequentially, we see that performing random disk operations is better performed on Bare Metal, while Xen and Docker showed lower performance.

It is interesting to note that Xen appears to be performing fewer operations but achieving similar amount of output, this suggests that Xen's virtual hard disk may use a different block size from the others. This may explain some of the differences in efficiency we have seen but this does not explain the disparity in performance between Docker and the other platforms.

	Bare Metal	Xen	Docker
Combined Random Read/Write			
Operations Performed			
Read	26 004	22 955	23 514
Write	17 336	15 303	15 676
Other	55 424	48 896	50 048
Requests/sec Executed	144.47	127.52	130.63
Request Time Statistics			
Minimum (ms)	0.01	0.01	0.01
Average	3.32	3.99	4.41
Maximum	69.00	76.71	28.54
95^{th} Percentile	11.16	11.77	11.24

Table 5.7: Combined random read/write to the disk

	Bare Metal	Xen	Docker
Random Write			
Operations Performed			
Read	0	0	0
Write	39 500	31 300	42 600
Other	50 534	40 006	54 448
Requests/sec Executed	131.67	104.33	141.99
Request Time Statistics			
Minimum (ms)	0.02	0.01	0.01
Average	0.03	0.03	0.03
Maximum	0.25	0.69	0.29
95^{th} Percentile	0.03	0.03	0.07

Table 5.8: Random writes to the disk

5.5.4 Networking

As much as we wanted to control for as many variables as possible, Docker and Xen employ radically different schemes for network access. Xen uses a virtual network bridge and each virtual machine can be manually configured or can get an IP address via Dynamic Host Configuration Protocol (DHCP), whereas Docker makes use of the host machine's instance of iptables and remaps a port on the host to the port the application is listening on. If we consult figures 5.1 and 5.2 we can see that there are different layers of abstraction between the application and the hardware which is executing it, these may also impact the networking performance.

	Host	Xen	Docker
Bandwidth (Mbitps)			
Sender	94.0	94.0	94.0
Receiver	93.9	93.9	93.7

Table 5.9: Network benchmark experiment

To measure the performance of the networking we used iperf version 3.0.11 [3]. This tool is commonly used for network throughput testing and creates streams of either TCP or UDP data between two hosts, with one host acting as a client requesting this data transfer, and other acting as the server tasked with serving it.

Each experiment lasted 30 seconds and was a test of network throughput using a TCP connection, as this is the more common way to communicate with a RDMS such as PostgreSQL. The experiment used the environment under test as the server and the machine that ran pgbench in the earlier experiments as the client. The results are summarised in table 5.9. We conclude there is little difference between the platforms tested in terms of the performance of TCP over IP.

5.6 Conclusions

From what we have demonstrated here, the overall power consumption is very similar for all each setup, however when we consider the performance per transaction there is a very different story, due to the variable performance of Docker, Xen and the physical host.

In section 5.5 we tried to diagnose some of the larger performance discrepancies. These performance issues weren't easy to diagnose, and now we are forced to consider that these seem to arise from the detailed interactions between the guest operating system and the virtualisation hypervisor.

This may indicate that while virtualisation is useful and mainstream, the arising performance issues are still complicated and that complicates effective measurement of power efficiency.

It is possible that Xen being a mature system with numerous active users and developers has been optimised to handle the sort of anomalies that plagued

Docker without large degradations in performance.

5.7 Future Works

The issues surrounding virtual machines and power have been explored in some detail by others, as we saw in section 2.3.2, but current research is limited to looking at single aspects of virtual machines, it would be useful to be able to understand the trade off between the energy efficiency of transactions on a virtual machine and the cost of consolidating multiple running virtual machines to a single host machine.

There are currently attempts to introduce something akin to virtual machine style live migration to Docker and Linux containers in general. This would allow running containers to be moved between hosts like virtual machines, but as we have seen live migration isn't without costs. Further study and comparison is required to understand the benefits and drawbacks to container migration versus live virtual machine migration.

CHAPTER 6

Energy and Cryptocurrencies

Bitcoin is a digital cryptocurrency that has generated considerable public interest, including both booms in value and busts of exchanges dealing in Bitcoins. One of the fundamental concepts of Bitcoin is that work, called mining, must be done in checking all monetary transactions, which in turn creates Bitcoins as a reward. In this chapter we look at the energy consumption of Bitcoin mining. We consider if and when Bitcoin mining has been profitable compared to the energy cost of performing the mining, and conclude that specialist hardware is usually required to make Bitcoin mining profitable. We also show that the power currently used for Bitcoin mining is comparable to Ireland's electricity consumption.

6.1 Introduction

Bitcoin is a peer-to-peer cryptocurrency mainly used for monetary transactions on the Internet [117] and is designed to be similar to Fiat Money and commodities. Bitcoins are intrinsically valueless, their worth is decided by those trading in them. At the time of writing [1], 1 Bitcoin (฿) is worth approximately 370 Euro (€) [2]. Bitcoin has generated a huge amount of interest in the media lately and has sparked a wave of copy-cat-currencies [98, 63] and even a fully working parody currency [108]. It has also generated interest in academic

[1] April 2016

[2] A graph of historical exchange rates between Bitcoin & US Dollars can be seen in figure 6.4

circles due to issues it creates in user privacy e.g. [17], as well as attempts to gain insights into who is behind the transactions e.g. [110] and attempts to better understand its implications as a payment system e.g. [90].

Bitcoin is based on a peer-to-peer network within the Internet. The members of the peer-to-peer network effectively maintain a ledger of Bitcoin transactions which have been accepted by the network. In this ledger, Bitcoins are owned by *Bitcoin addresses*, which are public keys from a key pair. In order to assign Bitcoins, or some fraction thereof, to a new owner, the current owner must sign the transaction with the private key of the key pair using an Elliptic Curve Digital Signature Algorithm (ECDSA) scheme. Before a transaction is accepted by the network, the transaction is checked for validity, including the presence of these signatures.

Bitcoins are not issued or governed by a central authority but, instead are created in a process called *mining*. Mining is one of the key concepts behind the Bitcoin protocol, in which valid transactions are collected into *blocks* and are added to the ledger by linking it to the previously accepted blocks. The network forms a common view, called the *blockchain*, of which transactions have taken place, preventing users from reusing Bitcoins and attempting to spend them more than once.

To add a block to the blockchain, a signature must be found linking the transactions in the block to the previous blocks. This requires finding a *nonce* value which satisfies a particular equation involving the SHA256 cryptographic hash function [121]. This is a computationally expensive task; however, a member of the peer-to-peer network who finds a suitable value is rewarded by being able to assign newly mined Bitcoins to an address of their choosing.

In this chapter we consider the energy cost of Bitcoin mining. Solving of the computational problem requires energy. We consider how this energy can be calculated and the impact of using different types of hardware for this computation. Using historical information from the Bitcoin network and Bitcoin exchanges, we compare the monetary cost of the energy to the reward for calculating a Bitcoin block. We also consider the likely power consumption of the whole Bitcoin mining operation, and show that it is comparable to Ireland's average electricity consumption.

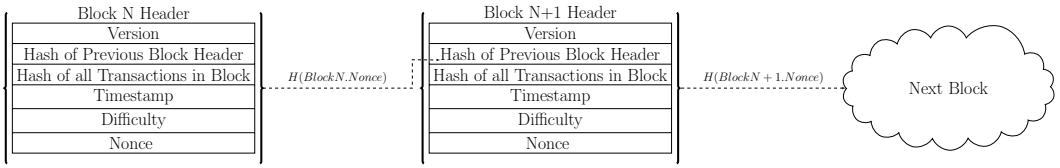


Figure 6.1: Basic operation of Bitcoin mining, including the information stored in the header of each block

6.2 Bitcoin Mining

As we mentioned, a Bitcoin miner is part of Bitcoin's peer-to-peer network that collects recent transactions and aims to complete a proof of work scheme, based on the ideas of Hashcash [20]. In this scheme, there is a current target value T, which is periodically recalculated by the network (see section 6.2.1). The miner's aim is to find a nonce value so that

$$H(B.N) < T \tag{6.1}$$

where B is the string representing the recent transactions, N is the nonce value, '.' is the concatenation operator and H is the Bitcoin hash function, in this case

$$H(S) := \text{SHA256}(\text{SHA256}(S)).$$

The proof of work can be achieved by choosing values for N randomly or systematically until equation (6.1) is satisfied. When an N is found, the resulting block can be sent to the Bitcoin network and added to the Bitcoin blockchain. Finding a block results in a reward of extra Bitcoins for the block's finder. Thus, the process of finding a suitable N value is referred to as *Bitcoin mining*.

Figure 6.1 gives us an outline of the bitcoin mining process, this shows how hash values are included in the blocks that form Bitcoin's blockchain.

6.2.1 Difficulty

The rate at which Bitcoins can be discovered can be controlled by the Bitcoin Network's choice of the value of the target, T, in equation (6.1). However, the target depends on the current number and speed of miners in the Bitcoin network, and is normally quoted in terms of the *difficulty*, D. The relationship between the difficulty and the target T is

$$D = \frac{T_{\text{max}}}{T}$$

where the largest possible value of the target $T_{\max}$ is $(2^{16}-1)2^{208} \approx 2^{224}$.

The hash function H for Bitcoin has been chosen so that it behaves approximately as a uniformly random value between 0 and $2^{256}-1$. Thus, for any given nonce value, the probability of it satisfying equation (6.1) is

$$p = \frac{T}{2^{256}} = \frac{T_{\max}}{D2^{256}} \approx \frac{1}{D2^{32}}.$$

Each nonce value tested should behave like an independent trial, so the number of trials until a block is successfully completed will be geometrically distributed, therefore the the expected number of hashes to find a block is $D2^{32}$. If we have a system calculating hashes at a rate R, the expected time to find a block is

$$\mathbb{E}[t] = \frac{1}{Rp} \approx \frac{D2^{32}}{R}. \tag{6.2}$$

For example, if you can calculate a Bitcoin hash 1 million times a second, and the difficulty [3] is $166,851,513,283$, then $\mathbb{E}[t] \approx 7.16621792838594592768 \times 10^{14}$seconds [4].

6.2.2 Change in Difficulty

The difficulty D, is recalculated every 2016 blocks, with the aim of keeping the average time to discover a new block near 10 minutes. At this ideal speed, 2016 blocks will be discovered every two weeks. To calculate the new difficulty, the length of time that it took to calculate the the last 2016 blocks is used to estimate the hash rate of the entire Bitcoin network. The new difficulty is selected so that if the same average hash rate is maintained, it will take two weeks to calculate the next 2016 blocks. If the resulting difficulty is more than four times harder (or four times easier) than the current difficulty, then the result is capped to four times harder (or easier). Restrictions on the range of acceptable difficulties/targets are also applied. The historical values of difficulty to date are shown in figure 6.2. The increasing trend in difficulty has been caused by an increase in the resources dedicated to calculating hashes in the Bitcoin network.

6.2.3 Change in Reward

There are two sources of reward for calculating a new block. First, the block is formed from Bitcoin transactions, and a transaction may choose to include a

[3] Current as of April 1st 2016.

[4] approximately 22.72 million years

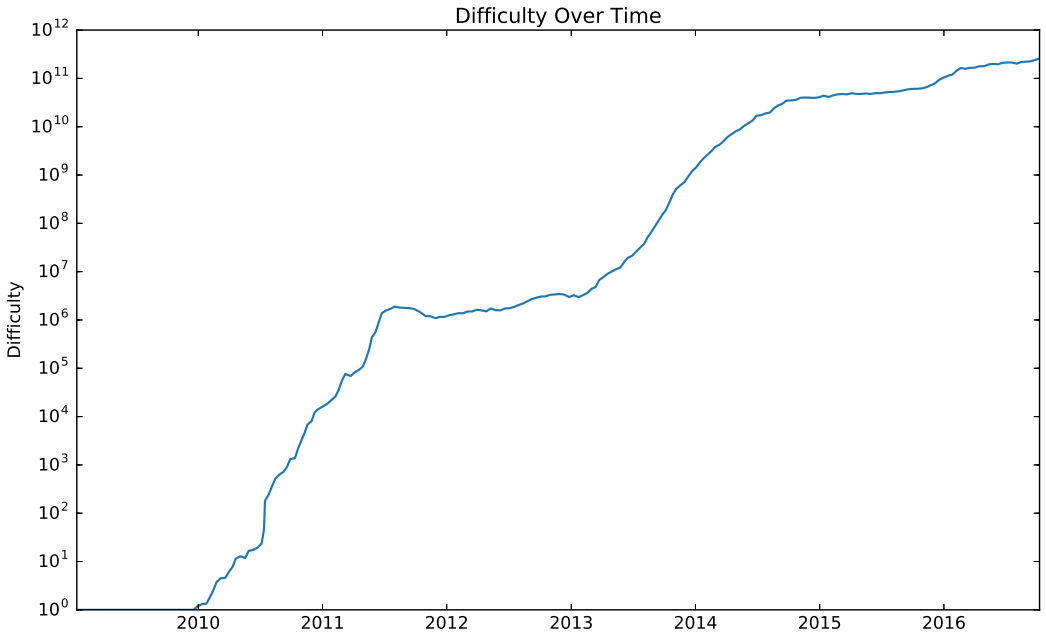


Figure 6.2: Change of the difficulty to generate a Bitcoin over time, based on information from the block chain [40].

transaction fee, to be paid to whoever finds a block containing this transaction. Second, a standard reward is provided depending on how many blocks have been successfully calculated. This reward started at ฿50 per block and is halved every 210,000 blocks. It halved to ฿25 on the 28th of November 2012. As of mid-October 2016, the reward is ฿12.5, having halved on the 9th of June 2016.

The reward will eventually reach ฿0; after such time it is imagined that the network of miners will continue mining but will do so in order to gain processing fees. This means that there is a limit on the number of Bitcoins which will be mined, but each Bitcoin is divisible up to 8 decimal places.

The mean value of the transaction fee over a day is plotted for a range of days in figure 6.3. As we can see the current standard reward of ฿12.5, is considerably larger than the current or historical average transaction fees. This may change in the future, as the standard reward continues to halve.

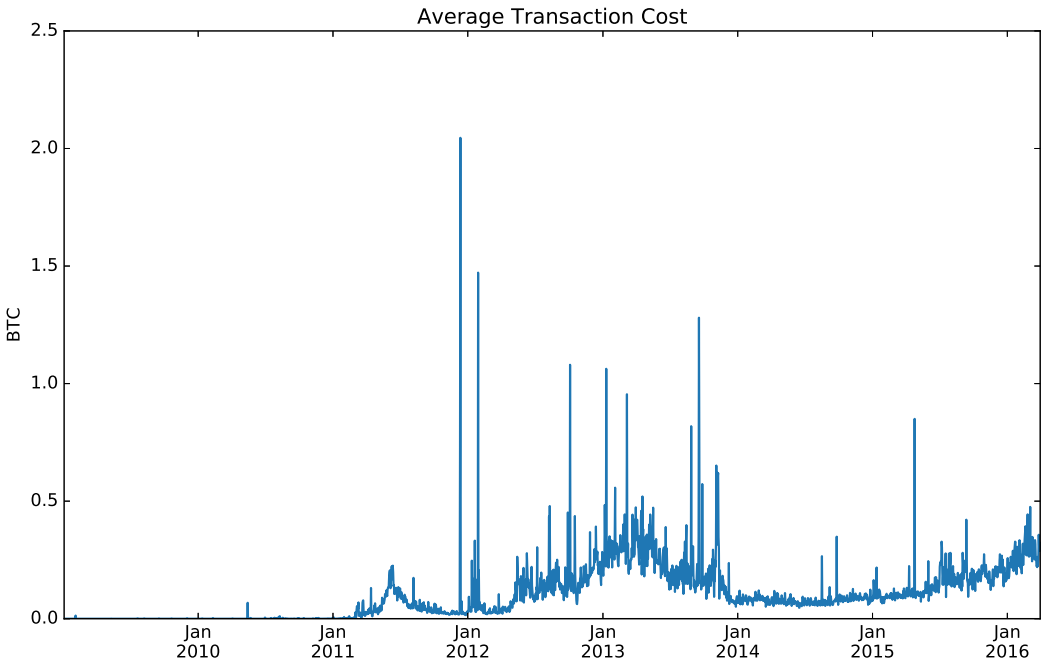


Figure 6.3: Average transaction fee per block per day. Data derived from `http://blockchain.info/charts`.

6.3 Hardware Arms Race

The major limiting factors in Bitcoin mining are the hash rate of hardware and the cost of running this hardware. The hash rate R, is typically measured in millions of hashes per second or Mega-hashes (Mhash/s). This is combined with the power usage P, of the hardware to get the energy efficiency of the hardware $\mathcal{E} = R/P$ (Mhash/J) which serves as a helpful statistic to compare hardware. Statistics are shown for a selection of hardware in table 6.1.

Initially mining took place on normal computers [5]. As Bitcoin gained popularity, there was something akin to an arms race as miners attempted to increase their hash rate. GPUs which can perform many parallel calculations are well-adapted to Bitcoin mining. Standard programming interfaces, such as Open Computing Language or CUDA, made GPUs popular among Bitcoin miners. Their higher hash rate compared with their lower energy footprint made them better suited to mining than normal CPUs.

As the use of GPUs became more widespread, people were forced to look for

[5] Where 'normal' is defined as a general purpose computer, such as an IBM PC type computer with an x86 CPU.

Name	Type	Hash Rate R (Mhash/s)	Power Use P (W)	Energy Efficiency $\mathcal{E}$ (Mhash/J)	Cost ($)	Reference
Core i7 950	CPU	18.9	150	0.126	350	[12, 33]
Atom N450	CPU	1.6	6.5	0.31	169	[11, 33]
Sony Playstation 3	CELL	21.0	60	0.35	296	[15, 33]
ATI 4850	GPU	101.0	110	0.918	45	[14, 33]
ATI 5770	GPU	214.5	108	1.95	80	[10, 33]
Digilent Nexys 2 500K	FPGA	5.0	5	1	189	[13, 33]
Monarch BPU 600 C	ASIC	600000.0	350	1714	2196	[41, 33]
Block Erupter Sapphire	ASIC	333.0	2.55	130	34.99	[9, 33]

Table 6.1: Examples of bitcoin-mining devices

alternatives to keep ahead of the crowd. Field Programmable Gate Arrays (FPGAs) came into vogue for a brief period before Application Specific Integrated Circuits (ASICs) came onto the scene. ASICs can perform the Bitcoin hash at higher rates but with a much smaller energy requirement. The evolution of hardware for Bitcoin mining is described in detail in [154].

6.4 Energy Cost/Reward Trade Off

Bitcoin is similar to other currencies, in that the exchange rate between Bitcoin and other currencies fluctuates over time. This in turn impacts on the viability of Bitcoin mining: if the value of a Bitcoin is less than the cost of the energy required to generated it then there is a disincentive to continue mining. The exchange rate with the US dollar is shown in figure 6.4.

On the other hand, as the number of people mining Bitcoin increases and the difficulty of mining follows suit, so the likelihood of discovering a valid block decreases. To overcome this, more powerful hardware is required to achieve the same success rate. However, since the cost of energy is a limiting factor, newer hardware will have to have a higher hash rate and a lower energy footprint.

Thus, there is a trade off between two time varying factors: first, the energy cost of discovering a block,

$$C_e = \mathbb{E}[t]PU \approx \frac{D2^{32}PU}{R} = \frac{D2^{32}U}{\mathcal{E}}$$

where U is the unit cost for a Joule of energy; second is the cash reward for discovering the block, which is simply the reward for the block, in ฿, times the current exchange rate for a Bitcoin. Alternatively, we may normalise this

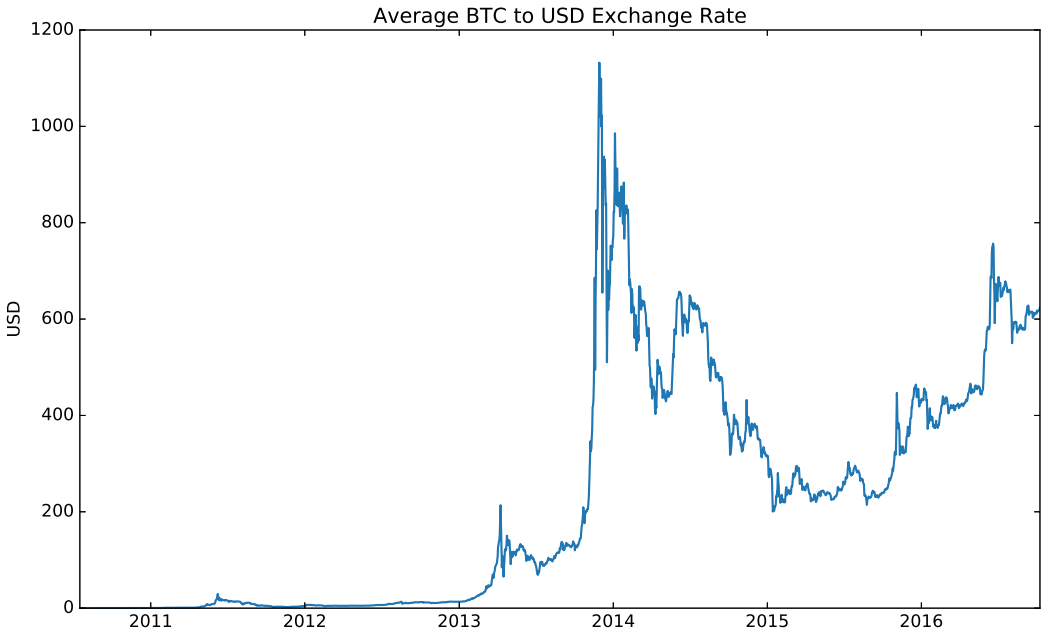


Figure 6.4: Exchange rate between bitcoin and dollars, based on aggregate statistics [40, 135].

per Bitcoin. Figure 6.5 shows the energy cost and the value for generating a Bitcoin for various hardware from table 6.1. We use a dashed line for hardware before its release and a grey line to show the exchange rate between USD and Bitcoin. As long as the lines for each mining platform remain under the grey line, they are profitable as the cost to generate a bitcoin is less than its value.

To allow easy comparison with the Bitcoin exchange rate, we use a cost of 0.10 US dollars per kWh. This is the lowest cost of electricity in Eurostat's 2013 statistics [68]; for Industrial rates in Finland. As typical consumer prices are twice this or more, this should provide a lower bound for the energy cost of mining Bitcoins in Europe. When calculating the value of each block, we have used the standard reward and not included transaction fees, as we have seen that the transaction fees are uncertain and currently a small fraction of the total reward.

For the period for which exchange rate data is available, we see that it has never been profitable to use a generic Core i7 CPU, and it appears that it may only have been briefly been profitable to use a Playstation 3. Using FPGAs or GPUs appears to have been close to profitable until mid-2013, when the increase in difficulty outpaced the increase in Bitcoin value. The yet-to-be-available ASIC

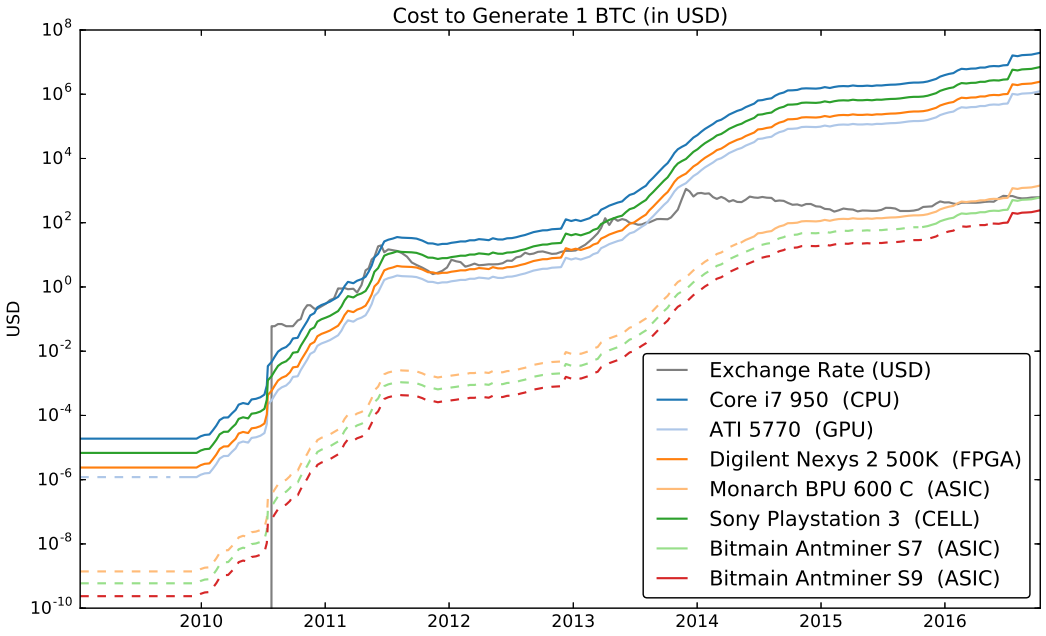


Figure 6.5: Cost of generating a bitcoin and the value of the resulting reward.

hardware could be profitable, though the gap is closing.

6.5 Network Power Usage

As we know that the Bitcoin network aims for an aggregate block discovery rate of one every 10 minutes, we can use equation (6.2) to estimate the hash rate of the entire network if we know the difficulty:

$$R_{\text{net}} \approx \frac{D2^{32}}{600\text{s}}.$$

Combining this with the efficiency $\mathcal{E}$ for different hardware, we can estimate the network's power usage as $P_{\text{net}} = R_{\text{net}}/\mathcal{E}$. For commodity hardware (CPUs/GPUs), efficiency values above 2 Mhash/J are unlikely [33]. For FPGAs, values around ten times this are possible. For ASICs values of 100–1000 times are possible.

Figure 6.6 shows conservative estimates for the total power used for Bitcoin mining, assuming that it consists of either efficient commodity hardware ($\mathcal{E} =$ 2 Mhash/J) or efficient specialist hardware ($\mathcal{E} = 2000$ Mhash/J). The actual network will be a mix of hardware of types at different levels of efficiency, so we expect that the actual efficiency will be between the two. This suggests that the total power used for Bitcoin mining is around 0.1 GW to 10 GW. Average

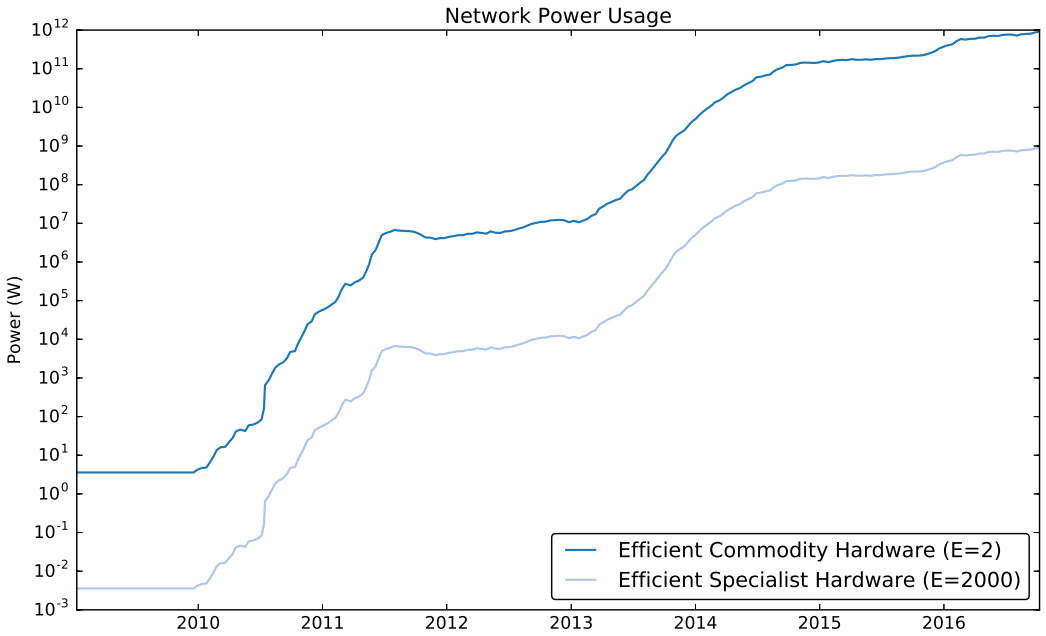


Figure 6.6: Estimated power consumption of the bitcoin mining network

Irish electrical energy demand and production is estimated at around 3 GW [69, 66], so it is plausible that the energy used by Bitcoin mining is comparable to Irish national energy consumption.

6.6 Conclusion

In this chapter, we have described aspects of Bitcoin relevant to Bitcoin mining and its energy consumption. Even though the value of Bitcoin is decided by those who trade in them, it is also related in some way to the value of electricity. We have seen that the cost of Bitcoin mining on commodity hardware now exceeds the value of the rewards. Thus, the competition created in mining for Bitcoin has led to a situation where in order to be financially viable the hardware has to become faster and more energy efficient.

In this chapter we looked at the energy issues around Bitcoin mining and its profitability. We also estimated under reasonable assumptions, that currently the entire Bitcoin mining network is on par with Ireland for electricity consumption.

CHAPTER 7

Conclusions

In this chapter, we review and summarise the work presented in this manuscript and give recommendations for possible extensions to it.

7.1 Summary

The work presented in this thesis looked at network services and the energy efficiency of these services.

Chapter 2 reviewed the field and what is in the current literature. Chapter 3 looked at measuring electrical energy usage, as well as the important theoretical concepts in electricity measurement and considered different devices available to do this. The contribution of these two chapters is to set the stage for the research described in the chapters that follow them.

Chapter 4 focused on saving power by moving under-utilised servers to a low power state. This required us to investigate the web traffic of a typical web server and design a strategy to transition the server to a low power mode and a mechanism to return to full power around this.

We examined this mechanism but due to the lack of dependable hardware we had to model the power consumed by our experimental test bed and used this model to estimate the potential savings. Despite this we were able to

demonstrate the possibility to save a significant amount of energy using a practical algorithm.

Chapter 5 compared the Docker Linux container system and Xen, the virtual machine hypervisor system, both promise and deliver a lot of useful functionality. Both can be used to host applications, for providing a complete working environment to a developer and as a way to ensure isolation of applications that share physical hardware. All things considered, we aimed to determine which performs better if energy efficiency is your primary concern. The energy usage was similar but there was a serious difference in the performance we observed. After a substantial amount of investigation we could not isolate the reasons why Docker performed so poorly. This did not lead us to a direct answer to our question but fuels us on to investigate the many interesting aspects of virtualisation and operating-system-level virtualisation.

Chapter 6 explored bitcoin, an emerging cryptocurrency with interesting technological implications. In this chapter we explored the key concepts of bitcoin and the fast paced competition to gain more energy efficient and effective mining. Our main contribution was to estimate the energy efficiency of the bitcoin network. We then compared the cost of generating bitcoin and the value of bitcoin and found that only the very latest mining hardware offers a profitable means to mine bitcoin.

7.2 Future Works

There are many possible additions or improvements which could be build on this body of work.

As we saw in chapter 3 there are many ways to measure energy consumption, and none of these are universally applicable. A useful addition would be a comprehensive set of benchmarking tests to allow for better comparison of different energy measurement devices. This would allow researchers to make more informed choices about the measurement equipment they use.

Chapter 4 proposed a mechanism to reduce power consumption in HTTP servers without a serious impact on the availability of this service. This could be further built on by generalising this to other services that use a stateless protocol like HTTP. Other interesting improvements could be found by adding

machine learning techniques to identify when servers are best suited to be put to sleep or to predict what items should be cached before sleeping to maximise possible sleep time.

What we didn't consider was the case where the HTTP servers are hosted on virtual machines, and technologies like live migration enable consolidation of a group of running virtual machines to run on a single physical host. This could then leverage our scheme in order to further reduce the number of running virtual machines on the host.

The work described in chapter 5 looked at virtual machines and Linux containers, this work could be extended further by investigating the efficiency versus performance trade-offs around the interactions between the guest operating system and the virtualisation hypervisor. For example the hypervisor has the ability to control the CPU frequency by changing the voltage of the CPU, this can be controlled in accordance with a number of different algorithms. The impact of using such algorithms could be a slight reduction in performance but an improvement in overall energy efficiency.

Containers and virtual machines aren't completely opposed, as we stated it is possible for them to coexist on the same physical host or to host containers inside a virtual machine. It might be interesting to rerun the main experiments from chapter 5 against an instance of PostgreSQL running inside a container which in turn is running inside a virtual machine. This would almost certainly impact the performance and the efficiency of the tasks in the experiment, but it is unclear the direction or magnitude of this change.

Chapter 6 looked at energy and the cryptocurrency bitcoin, it would be interesting to examine the energy efficiency of other cryptocurrencies such as Litecoin [98] which use a different proof of work scheme to reduce the advantage of using specialist hardware, or Peercoin [92] which introduces a modification to the Bitcoin proof of work to improve the energy efficiency of mining.

In the time since the publication of the original work we've seen an explosion in the efficiency of mining hardware as well as many fluctuations in the value of bitcoin. This area may continue to be of considerable interest to those of us who are looking into issues of efficiency. Just as the monetary value of bitcoin has accelerated the demand for more efficient hardware, it is possible that this

phenomenon could occur again in a different field.

7.3 Final Thoughts

I would like to to conclude with some final personal observations that I have made while conducting my research.

- From the work presented in this thesis it is evident that energy efficiency does not have to come at the sacrifice of computational power or of flexibility, but it does require some forethought in order to gain sufficient benefit.
- This work is at odds with the accepted conventional wisdom at many points. This was not our intention but it should serve as a reminder to researchers that challenging the accepted wisdom is probably not going to lead to earth shattering results but can often lead you to more interesting results.
- The growth in the cloud, widespread acceptance of software as a service and emergence of private cloud infrastructure has made continued research into energy efficiency essential. But to understand this field better, more research is required into how to measure power consumption of hardware as well as how to gauge the efficiency of the software that runs on it.
- While planning any experiment it is vital to consider what you hope to measure and how you plan to measure it. But it is equally important to consider the third question *"is my means of measurement sufficient for what I hope to achieve?"*. With this in mind I would recommend that anyone working this field consider this question early in the design of their experiments.
- In the course of this thesis we have seen a group of seemingly unrelated topics which have actually been related by concerns of their efficient use of energy. This work is certainly not the first of its kind, but it is not going to be the last, as new technologies are always emerging. All new technologies should give us the opportunity to pause and to consider their impact on the environment.

Glossary

ACPI

Advanced Configuration and Power Interface.

ADC

Analogue-to-Digital Converter.

API

Application Programming Interface.

ARP

Address Resolution Protocol.

ASIC

Application Specific Integrated Circuit.

Bare Metal

A colloquial term referring to an application running directly on hardware without an operating system or to an operating system which runs *on the metal* of the computer, not inside a virtual machine.

Big Data

Big data refers to large and often complex data sets and the challenges they present in terms of analysis, curating, data privacy, searching, storing, sharing and sorting.

CO_2

Carbon Dioxide.

CPU

Central Processing Unit.

Cryptocurrency

A digital currency which uses cryptographic techniques to verify the transfer of money as well as the creation of new currency units. Typically these currencies operate without a central bank or an equivalent authority.

CUDA

An API allowing developers to execute code for general purpose processing on certain GPUs.

DBMS

Database Management System.

DHCP

Dynamic Host Configuration Protocol.

Diurnal

Mainly active during the day time.

DVFS

Dynamic Voltage and Frequency Scaling.

ECDSA

Elliptic Curve Digital Signature Algorithm.

Fiat Money

Money issued by a Government which is declared to be legal tender and not convertible to any other thing nor does it have a fixed value by an objective standard.

FPGA

Field Programmable Gate Array.

GPIO

General-Purpose Input/Output.

GPU

Graphics Processing Unit.

HTTP

Hyper Text Transfer Protocol.

HTTP Live Streaming

A standard developed by Apple for streaming live or on demand video or audio over HTTP.

HTTP Status Code 200

OK, the request was successful.

HTTP Status Code 404

Not Found, the requested resource was not found.

HTTP Status Code 503

Service Unavailable, the service is currently unavailable due to planned maintenance or is overloaded.

Hysteretics

Relating to hysteresis, the lag in response of a system to changes acting on it.

I/O

Input/Output.

IaaS

Infrastructure as a service.

Infrastructure as a service

A cloud computing service model which abstracts the user from the detail of the infrastructure like physical computers and security.

IP

Internet Protocol.

IPMI

Intelligent Platform Management Interface.

ISM

Industrial, Scientific, and Medical radio band.

KVM

Kernel-based Virtual Machine.

Linux Container

A system which creates isolated userspace instances which share a kernel, similar to FreeBSD's jails.

MAC

Medium Access Control.

MPTCP

Multipath Transmission Control Protocol.

NAT

Network Address Translation.

Open Computing Language

A framework and language for writing programs that execute on different computing platforms including CPUs, FPGAs and GPUs.

Operating System

Software which manages all the hardware resources of a computer and provides an environment to execute programs by exposing software interfaces to access the hardware.

Operating-System-Level Virtualisation

A vitualisation technique which enables running multiple isolated user space instances on a single kernel.

PDU

Power Distribution Unit.

Platform as a service

A cloud computing service model where where consumers can run, develop and manage applications without the need to build or maintain the underlying physical infrastructure.

PostgreSQL

PostgreSQL, also known as Postgres, is a popular relational database server.

QoS

Quality of Service.

RAM

Random Access Memory.

RDMS

Relational Database Management System.

SCTP

Stream Control Transmission Protocol.

SHA256

Secure Hash Algorithm 2 using a 256-bit digest size.

SLA

Service Level Agreement.

Software as a service

A cloud computing service model where software is licensed and hosted from a vendor.

SQL

Structured Query Language.

SSH

Secure Shell.

TCP

Transmission Control Protocol.

UDP

User Datagram Protocol.

USB

Universal Serial Bus.

Virtual Machine

An emulation of a computer system where a physical computer is replaced by software, specialist hardware or a combination of the two.

WoL

Wake-on-LAN.

WSN

Wireless Sensor Networks.

Bibliography

[1] Advanced Configuration and Power Interface. http://www.acpi.info, 2011. 35

[2] ioping: simple disk I/0 latency measuring tool. `https://github.com/koct9i/ioping`, Accessed 2017. 76

[3] iPerf - the TCP, UDP and SCTP network bandwidth measurement tool. `https://iperf.fr/`, Accessed 2017. 80

[4] SysBench. `https://github.com/akopytov/sysbench`, Accessed 2017. 74

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

[6] David Aikema, Andrey Mirtchovski, Cameron Kiddle, and Rob Simmonds. Green cloud VM migration: Power use analysis. In *Green Computing Conference (IGCC), 2012 International*, pages 1–6, June 2012. 10

[7] Allegro MicroSystems. *Automotive Grade, Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor*, 2013. Rev. 9. 25

[8] Allegro MicroSystems. *Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor*, 2013. Rev. 7. 25

[9] Amazon.com. ASICMiner Block Erupter USB 330MH/s sapphire miner. http://www.amazon.com/ASICMiner-Block-Erupter-USB-Sapphire/dp/B00CUJT7TO, 2014. [Online; accessed 19-March-2014]. 88

[10] Amazon.com. ATI Radeon HD 5770. http://www.amazon.com/ATI-Radeon-DisplayPort-Dual-DVI-7120184001G/dp/B005EDG750, 2014. [Online; accessed 19-March-2014]. 88

[11] Amazon.com. Intel Atom N450 processor. http://www.amazon.com/CPU-Central-Processing-1-66GHz-FCBGA559/dp/B00HKIFV5O/, 2014. [Online; accessed 19-March-2014]. 88

[12] Amazon.com. Intel Core i7-950 3.06 GHz 8 MB Cache Socket LGA1366 processor. http://www.amazon.com/Intel-i7-950-Socket-LGA1366-Processor/dp/B002A6G3V2, 2014. [Online; accessed 19-March-2014]. 88

[13] Amazon.com. Nexys2 500K Xilinx Spartan-3E FPGA development kit. http://www.amazon.com/Nexys2-500K-Xilinx-Spartan-3E-Development/dp/B001D9EN6E/, 2014. [Online; accessed 19-March-2014]. 88

[14] Amazon.com. Sapphire Radeon HD4850. http://www.amazon.com/Sapphire-Radeon-HD4850-PCI-Express-100245HDMI/dp/B001XW4BKO, 2014. [Online; accessed 19-March-2014]. 88

[15] Amazon.com. Sony Playstation 3. http://www.amazon.com/Playstation-3-250GB-System-Slim-Redesign/dp/B00AEX81SG/, 2014. [Online; accessed 19-March-2014]. 88

[16] Ahmed Amokrane, Mohamed Faten Zhani, Qi Zhang, Rami Langar, Raouf Boutaba, and Guy Pujolle. On satisfying green SLAs in distributed clouds. In *Network and Service Management (CNSM), 2014 10th International Conference on*, pages 64–72, November 2014. 13

[17] Elli Androulaki, Ghassan O. Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security*, volume 7859 of *Lecture Notes in Computer Science*, pages 34–51. Springer Berlin Heidelberg, 2013. 83

[18] Jordi Arjona Aroca and Antonio Fernandez Anta. Empirical comparison of power-efficient virtual machine assignment algorithms. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2015*, pages 1–8, April 2015. 11

[19] Andrius Aucinas, Narseo Vallina-Rodriguez, Yan Grunenberger, Vijay Erramilli, Konstantina Papagiannaki, Jon Crowcroft, and David Wetherall. Staying online while mobile: The hidden costs. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, pages 315–320, New York, NY, USA, 2013. ACM. 15

[20] Adam Back. Hashcash — a denial of service counter-measure. `ftp://sunsite.icm.edu.pl/site/replay.old/programs/hashcash/hashcash.pdf`, 2002. 84

[21] Massimo Banzi. *Getting Started with Arduino*. Make Books - Imprint of: O'Reilly Media, Sebastopol, CA, ill edition, 2008. 25

[22] Paul Barford, Azer Bestavros, Adam Bradley, and Marl Crovella. Changes in web client access patterns — characteristics and caching implications. *World Wide Web*, 2:15–28, 1999. 30

[23] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Operating Systems Review*, 37(5):164–177, October 2003. 57

[24] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007. 9

[25] Robert Basmadjian and Hermann de Meer. Evaluating and modeling power consumption of multi-core processors. In *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, pages 1–10, May 2012. 14

[26] Dallal Belabed, Stefano Secci, Guy Pujolle, and Deep Medhi. Striking a balance between traffic engineering and energy efficiency in virtual machine placement. *Network and Service Management, IEEE Transactions on*, 12(2):202–216, June 2015. 12

[27] Francesco Beneventi, Andrea Bartolini, Andrea Tilli, and Luca Benini. An effective gray-box identification procedure for multicore thermal modeling. *Computers, IEEE Transactions on*, 63(5):1097–1110, May 2014. 7

[28] Leeann Bent, Michael Rabinovich, Geoffrey M. Voelker, and Zhen Xiao. Characterization of a large web site population with implications for content delivery. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 522–533, New York, NY, USA, 2004. ACM. 30

[29] Josep L. Berral, Ricard Gavalda, and Jordi Torres. Adaptive scheduling on power-aware managed data-centers using machine learning. In *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, GRID '11, pages 66–73, Washington, DC, USA, 2011. IEEE Computer Society. 11

[30] Josep L. Berral, Íñigo Goiri, Thu D.. Nguyen, Ricard Gavalda, Jordi Torres, and Ricardo Bianchini. Building green cloud services at low cost. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, pages 449–460, June 2014. 14

[31] Aruna Prem Bianzino, Luca Chiaraviglio, and Marco Mellia. Distributed algorithms for green IP networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 121–126, March 2012. 16

[32] Nilton Bila, Eric J. Wright, Eyal De Lara, Kaustubh Joshi, H. Andrés Lagar-Cavilla, Eunbyung Park, Ashvin Goel, Matti Hiltunen, and Mahadev Satyanarayanan. Energy-oriented partial desktop virtual machine migration. *ACM Trans. Comput. Syst.*, 33(1):2:1–2:51, March 2015. 6

[33] bitcoin.it. Mining hardware comparison. `https://en.bitcoin.it/wiki/Mining_hardware_comparison`, 2014. [Online; accessed 19-March-2014]. 88, 90

[34] Raffaele Bolla, Roberto Bruschi, Olga Maria Jaramillo Ortiz, and Paolo Lago. The energy consumption of TCP. In *Proceedings of the Fourth*

International Conference on Future Energy Systems, e-Energy '13, pages 203–212, New York, NY, USA, 2013. ACM. 16

[35] Raffaele Bolla, Roberto Bruschi, Olga Maria Jaramillo Ortiz, and Paolo Lago. An experimental evaluation of the TCP energy consumption. *Selected Areas in Communications, IEEE Journal on*, 33(12):2761–2773, December 2015. 17

[36] Hans-Werner Braun and Kimberley C. Claffy. Web traffic characterization: an assessment of the impact of caching documents from the NCSA's web server. In *Second International World Wide Web (WWW) Conference '94*, Chicago, IL, October 1994. 30

[37] David Breitgand and Amir Epstein. Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds. In *INFOCOM, 2012 Proceedings IEEE*, pages 2861–2865, March 2012. 10

[38] Alaa Brihi and Waltenegus Dargie. Dynamic voltage and frequency scaling in multimedia servers. In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pages 374–380, March 2013. 7

[39] Roberto Bruschi, Alfio Lombardo, Carla Panarello, Fabio Podda, Enrico Santagati, and Giovanni Schembra. Active window management: Reducing energy consumption of TCP congestion control. In *Communications (ICC), 2013 IEEE International Conference on*, pages 4154–4158, June 2013. 17

[40] btc.blockr.io. Blockr block reader. `https://btc.blockr.io/documentation/api`, 2016. [Online; accessed 2-April-2016]. 86, 89

[41] Butterflylabs. The Monarch BPU 600 C. `https://products.butterflylabs.com/600-gh-bitcoin-mining-card.html`, 2014. [Online; accessed 19-March-2014]. 88

[42] Jin Cao, William S. Cleveland, Yuan Gao, Kevin Jeffay, F. Donelson Smith, and Michele Weigle. Stochastic models for generating synthetic

HTTP source traffic. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1546 –1557 vol.3, March 2004. 30

[43] Antonio Capone, Carmelo Cascone, Luca G. Gianoli, and Brunilde Sansò. OSPF optimization via dynamic network management for green IP networks. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2013*, pages 1–9, October 2013. 16

[44] Supaporn Chai-Arayalert and Keiichi Nakata. The evolution of green ICT practice: UK higher education institutions case study. In *Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference on*, pages 220–225, August 2011. 6

[45] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, Amin M. Vahdat, and Ronald P. Doyle. Managing energy and server resources in hosting centers. *SIGOPS Operating Systems Review*, 35(5):103–116, October 2001. 13, 29

[46] Xiaomeng Chen, Ning Ding, Abhilash Jindal, Yu Charlie Hu, Maruti Gupta, and Rath Vannithamby. Smartphone energy drain in the wild: Analysis and implications. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '15, pages 151–164, New York, NY, USA, 2015. ACM. 15

[47] Dazhao Cheng, Yanfei Guo, and Xiaobo Zhou. Self-tuning batching with DVFS for improving performance and energy efficiency in servers. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*, pages 40–49, August 2013. 13

[48] Dazhao Cheng, ChangJun Jiang, and Xiaobo Zhou. Heterogeneity-aware workload placement and migration in distributed sustainable datacenters. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 307–316, May 2014. 9

[49] Xuntao Cheng, Bingsheng He, and Chiew Tong Lau. Energy-efficient query processing on embedded CPU-GPU architectures. In *Proceedings of*

the 11th International Workshop on Data Management on New Hardware, DaMoN'15, pages 10:1–10:7, New York, NY, USA, 2015. ACM. 15

[50] Bill Cheswick. An evening with Berferd in which a cracker is lured, endured, and studied. 56

[51] Luca Chiaraviglio, Syed Fakhar Abbas, and William Liu. To sleep or not to sleep: Understanding the social behavior of lifetime-aware networks. In *Computational Social Networks*, pages 262–272. Springer, 2015. 19

[52] Vojtech Cima, Bruno Grazioli, Seán Murphy, and Thomas Michael Bohnert. Adding energy efficiency to Openstack. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2015*, pages 1–8, April 2015. 9

[53] European Union: European Commission. Communication from the commission to the european parliament, the council, the european economic and social committee and the committee of the regions addressing the challenge of energy efficiency through information and communication technologies, May 2008. 4

[54] Carlos H.A. Costa, Marcelo C. Amaral, Guilherme C. Januario, Tereza C.M.B. Carvalho, and Catalin Meirosu. SustNMS: Towards service oriented policy-based network management for energy-efficiency. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2012*, pages 1–5, October 2012. 16

[55] currentcost.com. Current cost data cable. `http://www.currentcost.com/product-datacable.html`, 2016. [Online; accessed 22-June-2016]. 24

[56] currentcost.com. Current cost EnviR. `http://www.currentcost.com/product-envir.html`, 2016. [Online; accessed 22-June-2016]. 24

[57] currentcost.com. Current cost individual appliance monitor. `http://www.currentcost.com/product-iams.html`, 2016. [Online; accessed 22-June-2016]. 24

[58] Waltenegus Dargie. Analysis of the power consumption of a multimedia server under different DVFS policies. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 779–785, June 2012. 7

[59] Waltenegus Dargie. A stochastic model for estimating the power consumption of a processor. *Computers, IEEE Transactions on*, 64(5):1311–1322, May 2015. 14

[60] Debraj De and Sajal K. Das. SREE-Tree: self-reorganizing energy-efficient tree topology management in sensor networks. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2015*, pages 1–8, April 2015. 19

[61] Albert P.M. De La Fuente Vigliotti and Daniel Macedo Batista. A green network-aware VMs placement mechanism. In *Global Communications Conference (GLOBECOM), 2014 IEEE*, pages 2530–2535, December 2014. 10

[62] Charles Despins, Bill St.Arnaud, Richard Labelle, and Mohamed Chériet. Green ICT: The rationale for a focus on curbing greenhouse gas emissions. In *Wireless Communications and Signal Processing (WCSP), 2010 International Conference on*, pages 1–6, October 2010. 5

[63] Gaelcoin Developers. Gaelcoin Ireland's cryptocurrency. `http://www.gaelcoin.org/`, 2016. [Online; accessed 1-April-2016]. 82

[64] Ning Ding, Daniel Wagner, Xiaomeng Chen, Abhinav Pathak, Y. Charlie Hu, and Andrew Rice. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. *SIGMETRICS Perform. Eval. Rev.*, 41(1):29–40, June 2013. 18

[65] Joseph Doyle, Florian Knorn, Donal O'Mahony, and Robert Shorten. Brief paper - distributed thermal aware load balancing for cooling of modular data centres. *IET Control Theory Applications*, 7(4):612–622, March 2013. 7

[66] Eirgrid. System demand. `http://www.eirgrid.com/operations/systemperformancedata/systemdemand/`, 2014. [Online; accessed 19-March-2014]. 91

[67] Opeoluwa Tosin Eluwole and Mahboubeh Lohi. Coordinated multi-point power consumption modeling for energy efficiency assessment in LTE/LTE-advanced cellular networks. In *Telecommunications (ICT), 2012 19th International Conference on*, pages 1–6, April 2012. 17

[68] Eurostat. Electricity and natural gas price statistics. `http://epp.eurostat.ec.europa.eu/statistics_explained/index.php/Electricity_and_natural_gas_price_statistics`, 2014. [Online; accessed 27-March-2014]. 89

[69] Eurostat. Electricity production, consumption and market overview. `http://epp.eurostat.ec.europa.eu/statistics_explained/index.php/Electricity_production,_consumption_and_market_overview`, 2014. [Online; accessed 27-March-2014]. 91

[70] Daren Fang, Xiaodong Liu, Lin Liu, and Hongji Yang. TARGO: Transition and reallocation based green optimization for cloud VMs. In *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pages 215–223, August 2013. 11

[71] Roy Fielding, James Gettys, Jeff Mogul, Hemrik Frystyk Nielsen, Larry Masinter, P. Leach, and Tim Berners-Lee. RFC 2616: Hypertext transfer protocol — HTTP/1.1, 1999. 34

[72] Jaume Freire-González and Ignasi Puig-Ventosa. Energy efficiency policies and the jevons paradox. *International Journal of Energy Economics and Policy*, 5(1):69–79, 2015. 5

[73] Anshul Gandhi, Mor Harchol-Balter, and Michael A Kozuch. The case for sleep states in servers. In *Proceedings of the 4th Workshop on Power-Aware Computing and Systems*, page 2. ACM, 2011. 29

[74] Amit Golander, Nancy Greco, Jimi Xenidis, Maria Hyland, Brian Purcell, and David Bernstein. IBM's PowerEN developer cloud: Fertile ground for academic research. In *IEEE 26th Convention of Electrical and Electronics Engineers in Israel (IEEEi)*, pages 803–807, 2010. 51

[75] Karina Gomez, Roberto Riggio, Tinku Rasheed, Daniele Miorandi, and Fabrizio Granelli. Energino: A hardware and software solution for energy consumption monitoring. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2012 10th International Symposium on*, pages 311–317, May 2012. 25

[76] Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen, and Zhenghu Gong. The characteristics of cloud computing. In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pages 275–279, September 2010. 8

[77] Yanfei Guo and Xiaobo Zhou. Coordinated VM resizing and server tuning: Throughput, power efficiency and scalability. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on*, pages 289–297, August 2012. 11

[78] Emily Halili. *Apache JMeter: A Practical Beginner's Guide to Automated Testing and Performance Measurement for Your Websites*. Packt Pub Limited, 2008. 52

[79] Heather Hanson, Stephen W. Keckler, Soraya Ghiasi, Karthick Rajamani, Freeman Rawson, and Juan Rubio. Thermal response to DVFS: analysis with an Intel Pentium M. In *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, pages 219–224, August 2007. 7

[80] Md. E. Haque, Kien Le, Íñigo Goiri, Ricardo Bianchini, and Thu D. Nguyen. Providing green SLAs in high performance computing clouds. In *Green Computing Conference (IGCC), 2013 International*, pages 1–11, June 2013. 13

[81] Robert R. Harmon and Nora Auseklis. Sustainable IT services: Assessing the impact of green computing practices. In *PICMET '09 - 2009 Portland International Conference on Management of Engineering Technology*, pages 1707–1717, August 2009. 5

[82] Junxian Huang, Feng Qian, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. Screen-off traffic characterization and optimization in 3G/4G networks. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC '12, pages 357–364, New York, NY, USA, 2012. ACM. 16

[83] Chris Hyser, Daniel Gmach, Umesh Ml, Yuan Chen, and Vijay Suryanarayana. Improving server power management in research and development data centers. In *Proceedings of the Fourth Annual ACM Bangalore*

Conference, COMPUTE '11, pages 6:1–6:6, New York, NY, USA, 2011. ACM. 7

[84] Advanced Micro Devices Inc. Magic packet technology. Technical Report 20213, Advanced Micro Devices Inc., November 1995. 35

[85] John I. Jerkovic. *SEO Warrior: Essential techniques for Increasing Web Visibility*. O'Reilly Media, 2009. 34

[86] Melanie Kambadur and Martha A. Kim. An experimental survey of energy management across the stack. In *Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications*, OOPSLA '14, pages 329–344, New York, NY, USA, 2014. ACM. 5

[87] Ioannis Kamitsos, Lachlan Andrew, Hongseok Kim, and Mung Chiang. Optimal sleep patterns for serving delay-tolerant jobs. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 31–40, New York, NY, USA, 2010. ACM. 7

[88] Poul-Henning Kamp. Varnish cache, 2013. [Online; accessed 11-January-2013, Version: varnish-3.0.1 revision 6152bf7]. 30, 33

[89] Poul-Henning Kamp and Robert NM Watson. Jails: Confining the omnipotent root. In *Proceedings of the 2nd International SANE Conference*, volume 43, page 116, 2000. 56

[90] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 906–917, New York, NY, USA, 2012. ACM. 83

[91] Julia Kindelsberger, Felix Willnecker, and Helmut Krcmar. Long-term power demand recording of running mobile applications. In *Global Software Engineering Workshops (ICGSEW), 2015 IEEE 10th International Conference on*, pages 18–22, July 2015. 13, 15

[92] Sunny King and Scott Nadal. PPCoin: Peer-to-peer crypto-currency with proof-of-stake. `https://peercoin.net/assets/paper/peercoin-paper.pdf`, August 2012. 94

[93] Jonghoe Koo, Wonbo Lee, Sunghyun Choi, and Yongseok Park. PIMM: Packet interval-based power modeling of multiple network interface-activated smartphones. In *Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems*, e-Energy '15, pages 111–120, New York, NY, USA, 2015. ACM. 18

[94] Nawel Kortas and Wafa Kammoun. Energy consumption TCP, TCP-Reno and SCTP within cloud computing. In *Computer Applications Research (WSCAR), 2014 World Symposium on*, pages 1–6, January 2014. 17

[95] Martijn Koster. About /robots.txt. `http://www.robotstxt.org/robotstxt.html`. 32, 53

[96] Dara Kusic, Jeffrey O. Kephart, James E. Hanson, Nagarajan Kandasamy, and Guofei Jiang. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, 12(1):1–15, 2009. 10, 13

[97] Tuan Anh Le, Choong Seon Hong, M.A. Razzaque, Sungwon Lee, and Heeyoung Jung. ecMTCP: An energy-aware congestion control algorithm for multipath TCP. *Communications Letters, IEEE*, 16(2):275–277, February 2012. 19

[98] Charles Lee. Litecoin - open source P2P digital currency. `http://litecoin.org/`, 2016. [Online; accessed 1-April-2016]. 82, 94

[99] Mathieu Lemay, Kim-Khoa Nguyen, Bill St. Arnaud, and Mohamed Cheriet. Toward a zero-carbon network: Converging cloud computing and network virtualization. *Internet Computing, IEEE*, 16(6):51–59, November 2012. 14

[100] Ricardo Lent. Evaluating the performance and power consumption of systems with virtual machines. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 778–783, November 2011. 12

[101] Yeon-sup Lim, Yung-Chih Chen, Erich M. Nahum, Don Towsley, and Richard J. Gibbens. How green is multipath TCP for mobile devices?

In *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, &; Challenges*, AllThingsCellular '14, pages 3–8, New York, NY, USA, 2014. ACM. 18

[102] Yu-Chang Lin, Wei-Tsong Lee, and Jing-Yue Qiu. Reducing the power consumption of servers with bandwidth consideration. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2014 Tenth International Conference on*, pages 650–653, August 2014. 9

[103] Haikun Liu, Hai Jin, Cheng-Zhong Xu, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. *Cluster Computing*, 16(2):249–264, 2013. 10

[104] Liang Liu, Hao Wang, Xue Liu, Xing Jin, Wen Bo He, Qing Bo Wang, and Ying Chen. Greencloud: A new architecture for green data center. In *Proceedings of the 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session*, ICAC-INDST '09, pages 29–38, New York, NY, USA, 2009. ACM. 9, 11

[105] Dumitrel Loghin, Bogdan Marius Tudor, Hao Zhang, Beng Chin Ooi, and Yong Meng Teo. A performance study of big data on small nodes. *Proc. VLDB Endow.*, 8(7):762–773, February 2015. 9

[106] Alfio Lombardo, Carla Panarello, and Giovanni Schembra. A model-assisted cross-layer design of an energy-efficient mobile video cloud. *Multimedia, IEEE Transactions on*, 16(8):2307–2322, December 2014. 17

[107] Daokuan Ma, Yongwei Wu, Kang Chen, and Weimin Zheng. Flex: Flexible and energy efficient scheduling for big data storage. In *Parallel Processing Workshops (ICPPW), 2015 44th International Conference on*, pages 213–220, September 2015. 9

[108] Billy Markus. Dogecoin. `http://dogecoin.com/`, 2016. [Online; accessed 1-April-2016]. 82

[109] Conor McBay, Gerard Parr, and Sally McClean. Sustainable communications infrastructures in the cloud - supporting the information economy. In *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, pages 986–991, December 2014. 16

[110] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: Characterizing payments among men with no names. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 127–140, New York, NY, USA, 2013. ACM. 83

[111] David Meisner, Brian T. Gold, and Thomas F. Wenisch. Powernap: Eliminating server idle power. *SIGARCH Comput. Archit. News*, 37(1):205–216, March 2009. 7

[112] Peter Mell and Tim Grance. The NIST definition of cloud computing. 2011. 8

[113] Massimiliano Menarini, Filippo Seracini, Xiang Zhang, Tajana Rosing, and Ingolf Krüger. Green web services: Improving energy efficiency in data centers via workload predictions. In *Proceedings of the 2nd International Workshop on Green and Sustainable Software*, GREENS '13, pages 8–15, Piscataway, NJ, USA, 2013. IEEE Press. 9, 44

[114] Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), March 2014. 56, 57

[115] Irineu Moura, Gustavo Pinto, Felipe Ebert, and Fernando Castor. Mining energy-aware commits. In *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*, pages 56–67, May 2015. 5

[116] msoon.com. Monsoon Solutions Inc. Power Monitor. `https://www.msoon.com/LabEquipment/PowerMonitor/`, 2016. [Online; accessed 23-June-2016]. 25

[117] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. `http://fastbull.dl.sourceforge.net/project/bitcoin/Design%20Paper/bitcoin.pdf/bitcoin.pdf`. 82

[118] Cătălin Nicutar, Dragoş Niculescu, and Costin Raiciu. Using cooperation for low power low latency cellular connectivity. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, pages 337–348, New York, NY, USA, 2014. ACM. 18

[119] Ana Nika, Yibo Zhu, Ning Ding, Abhilash Jindal, Y. Charlie Hu, Xia Zhou, Ben Y. Zhao, and Haitao Zheng. Energy and performance of smartphone radio bundling in outdoor environments. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 809–819, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee. 18

[120] Kenneth O'Brien, Alexey Lastovetsky, Ilia Pietri, and Rizos Sakellariou. Towards application energy measurement and modelling tool support. In Victor Malyshkin, editor, *Parallel Computing Technologies*, volume 9251 of *Lecture Notes in Computer Science*, pages 91–101. Springer International Publishing, 2015. 14

[121] U.S. Department of Commerce, National Institute of Standards, and Technology. FIPS PUB 180-2, secure hash standard (SHS), 2002. 83

[122] Christopher Olston and Marc Najork. Web crawling. *Foundations and Trends in Information Retrieval*, 4(3):175–246, 2010. 32

[123] Fatma A. Omara, Sherif M. Khattab, and Radhya Sahal. Optimum resource allocation of database in cloud computing. *Egyptian Informatics Journal*, 15(1):1 – 12, 2014. 11

[124] Steven Osman, Dinesh Subhraveti, Gong Su, and Jason Nieh. The design and implementation of zap: A system for migrating computing environments. *SIGOPS Operating Systems Review*, 36(SI):361–376, December 2002. 56

[125] Carla Panarello, Marco Ajmone Marsan, Alfio Lombardo, Marco Mellia, Michela Meo, and Giovanni Schembra. On the intertwining between capacity scaling and TCP congestion control. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, e-Energy '12, pages 8:1–8:4, New York, NY, USA, 2012. ACM. 17

[126] Yashwant Singh Patel, Neetesh Mehrotra, and Swapnil Soner. Green cloud computing: A review on Green IT areas for cloud computing environment. In *Futuristic Trends on Computational Analysis and Knowledge*

Management (ABLAZE), 2015 International Conference on, pages 327–332, February 2015. 8

[127] Massoud Pedram and Inkwon Hwang. Power and performance modeling in a virtualized server system. In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pages 520–526, September 2010. 12

[128] Kostas Pentikousis. Pitfalls in energy consumption evaluation studies. In *Wireless Communication Systems, 2009. ISWCS 2009. 6th International Symposium on*, pages 368–372, September 2009. 17

[129] James E. Pitkow. Summary of WWW characterizations. *World Wide Web*, 2(1-2):3–13, January 1999. 30

[130] Andrej Podzimek, Lubomír Bulej, Lydia Y. Chen, Walter Binder, and Petr Tuma. Analyzing the impact of CPU pinning and partial CPU loads on performance and energy efficiency. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pages 1–10, May 2015. 12

[131] John M. Polimeni. *The Jevons Paradox and the Myth of Resource Efficiency Improvements*. Earthscan research edition. Taylor & Francis, 2012. 5

[132] Daniel Price and Andrew Tucker. Solaris Zones: Operating system support for consolidating commercial workloads. In *LISA*, volume 4, pages 241–254, 2004. 56

[133] Prodigit Electronics Co. ltd. *Model 2000MU-UK Plug-in Power Monitor Operation Manual*. 23

[134] Feng Qian, Zhaoguang Wang, Yudong Gao, Junxian Huang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. Periodic transfers in mobile applications: Network-wide origin, impact, and optimization. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 51–60, New York, NY, USA, 2012. ACM. 15

[135] quandl.com. Bitcoinaverage's bitcoin price index. `https://www.quandl.com`, 2016. [Online; accessed 2-April-2016]. 89

[136] Barath Raghavan, David Irwin, Jeannie Albrecht, Justin Ma, and Adam Streed. An intermittent energy internet architecture. In *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, pages 1–4, May 2012. 15

[137] Chandra Reka Ramachandiran. Green ICT practices among tertiary students: A case study. In *Business, Engineering and Industrial Applications (ISBEIA), 2012 IEEE Symposium on*, pages 196–201, September 2012. 6

[138] robot-italy.com. ACS714 current sensor carrier - 5A to +5A. `http://www.robot-italy.com/it/1185-acs714-current-sensor-carrier-5-to-5a.html`, 2016. [Online; accessed 27-June-2016]. 25

[139] Gerard Ryan and Mireia Valverde. Waiting in line for online services: a qualitative study of the user's perspective. *Information Systems Journal*, 16:181–211, April 2006. 52

[140] Kateryna Rybina, Waltenegus Dargie, René Schone, and Somayeh Malakuti. Mutual influence of application- and platform-level adaptations on energy-efficient computing. In *Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on*, pages 446–450, March 2015. 10

[141] Kateryna Rybina, Waltenegus Dargie, Anja Strunk, and Alexander Schill. Investigation into the energy cost of live migration of virtual machines. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2013*, pages 1–8, October 2013. 10

[142] Daniel Schall and Theo Härder. Energy-proportional query execution using a cluster of wimpy nodes. In *Proceedings of the Ninth International Workshop on Data Management on New Hardware*, DaMoN '13, pages 1:1–1:6, New York, NY, USA, 2013. ACM. 9

[143] Jan Hendrik Schönherr, Jan Richling, Matthias Werner, and Gero Mühl. Event-driven processor power management. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 61–70, New York, NY, USA, 2010. ACM. 7

[144] Micheal Seibold, Andreas Wolke, Martina Albutiu, Martin Bichler, Alfons Kemper, and Thomas Setzer. Efficient deployment of main-memory DBMS in virtualized data centers. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 311–318, June 2012. 11

[145] Bio Intelligence Service. Impacts of information and communication technologies on energy efficiency, September 2008. 2, 4

[146] Weiming Shi and Bo Hong. Towards profitable virtual machine placement in the data center. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 138–145, December 2011. 10

[147] Harkirat Singh and Suresh Singh. Energy consumption of TCP Reno, Newreno, and SACK in multi-hop wireless networks. In *Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '02, pages 206–216, New York, NY, USA, 2002. ACM. 17

[148] Ashiwan Sivakumar, Shankaranarayanan Puzhavakath Narayanan, Vijay Gopalakrishnan, Seungjoon Lee, Sanjay Rao, and Subhabrata Sen. Parcel: Proxy assisted browsing in cellular networks for energy and latency reduction. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, pages 325–336, New York, NY, USA, 2014. ACM. 18

[149] Stephen Soltesz, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, and Larry Peterson. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. *SIGOPS Operating Systems Review*, 41(3):275–287, March 2007. 56

[150] Sparkfun.com. Sparkfun low current sensor breakout - ACS712. `https://www.sparkfun.com/products/8883`, 2016. [Online; accessed 27-June-2016]. 25

[151] Michael Stonebraker and Lawrence A. Rowe. The design of postgres. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data*, SIGMOD '86, pages 340–355, New York, NY, USA, 1986. ACM. 58

[152] Anja Strunk. A lightweight model for estimating energy cost of live migration of virtual machines. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pages 510–517, June 2013. 10

[153] Markus Tauber, Saleem N. Bhatti, and Yi Yu. Application level energy and performance measurements in a wireless LAN. In *Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference on*, pages 100–109, August 2011. 18

[154] Michael Bedford Taylor. Bitcoin and the age of bespoke silicon. In *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, page 16. IEEE Press, 2013. 88

[155] Sibel Tombaz, Zhihao Zheng, and Jens Zander. Energy efficiency assessment of wireless access networks utilizing indoor base stations. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on*, pages 3105–3110, September 2013. 17

[156] Ramona Trestian, Arghir-Nicolae Moldovan, Olga Ormond, and Gabriel-Miro Muntean. Energy consumption analysis of video streaming to Android mobile devices. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 444–452, April 2012. 17

[157] Shiao-Li Tsao and Shih-Yung Lee. Evaluating the energy efficiency of TCP transmission over a WiMAX network. In *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, pages 1–6, August 2010. 17

[158] Rudi van Drunen. The basics of power. *;LOGIN: The USENIX Magazine*, 34(2), April 2009. 23

[159] Willem Vereecken, Ward Van Heddeghem, Didier Colle, Mario Pickavet, and Piet Demeester. Overall ICT footprint and green communication

technologies. In *Communications, Control and Signal Processing (ISCCSP), 2010 4th International Symposium on*, pages 1–6, March 2010. 5

[160] Ekhiotz Jon Vergara, Simon Andersson, and Simin Nadjm-Tehrani. When mice consume like elephants: Instant messaging applications. In *Proceedings of the 5th International Conference on Future Energy Systems*, e-Energy '14, pages 97–107, New York, NY, USA, 2014. ACM. 15

[161] Monica Vitali, Una-May O'Reilly, and Kalyan Veeramachaneni. Modeling service execution on data centers for energy efficiency and quality of service monitoring. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 103–108, October 2013. 12

[162] Yefu Wang and Xiaorui Wang. Virtual batching: Request batching for server energy conservation in virtualized data centers. *Parallel and Distributed Systems, IEEE Transactions on*, 24(8):1695–1705, August 2013. 13

[163] Zhaoguang Wang, Zhiyun Qian, Qiang Xu, Zhuoqing Mao, and Ming Zhang. An untold story of middleboxes in cellular networks. *SIGCOMM Comput. Commun. Rev.*, 41(4):374–385, 2011. 18

[164] Manuel Wendt, Matthias Grumer, Christian Steger, Reinhold Weiss, Ulrich Neffe, and Andreas Muehlberger. Tool for automated instruction set characterization for software power estimation. *Instrumentation and Measurement, IEEE Transactions on*, 59(1):84–91, January 2010. 14

[165] Michal Witkowski, Paul Brenner, Ryan Jansen, David B. Go, and Eric Ward. Enabling sustainable clouds via environmentally opportunistic computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 587–592, November 2010. 5

[166] Tom Worthington. A green computing professional education course online: Designing and delivering a course in ICT sustainability using internet and eBooks. In *Computer Science Education (ICCSE), 2012 7th International Conference on*, pages 263–266, July 2012. 6

[167] Ming-Zhi Wu, Yu-Chang Lin, Wei-Tsong Lee, Yu-Sun Lin, and Fong-Hao Liu. *Green Master Based on MapReduce Cluster*, pages 557–565. Springer Netherlands, Dordrecht, 2014. 9

[168] Shih-Lin Wu and Pao-Chu Tseng. An energy efficient MAC protocol for ieee 802.11 WLANs. In *Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on*, pages 137–145, May 2004. 17

[169] Chi Xu, Ziyang Zhao, Haiyang Wang, Ryan Shea, and Jiangchuan Liu. Energy efficiency of cloud virtual machines: From traffic pattern and CPU affinity perspectives. *Systems Journal, IEEE*, PP(99):1–11, 2015. 12

[170] Hong Xu and Baochun Li. Anchor: A versatile and efficient framework for resource management in the cloud. *Parallel and Distributed Systems, IEEE Transactions on*, 24(6):1066–1076, June 2013. 13

[171] Hong Xu and Baochun Li. Reducing electricity demand charge for data centers with partial execution. In *Proceedings of the 5th International Conference on Future Energy Systems*, e-Energy '14, pages 51–61, New York, NY, USA, 2014. ACM. 13

[172] Sarika Yadav and Rama Shankar Yadav. A review on energy efficient protocols in wireless sensor networks. *Wireless Networks*, 22(1):335–350, 2016. 19

[173] Yang Yu. *OS-level Virtualization and Its Applications*. PhD thesis, Stony Brook, NY, USA, 2007. AAI3337611. 56

[174] Yi Yu and Saleem N. Bhatti. The cost of virtue: Reward as well as feedback are required to reduce user ICT power consumption. In *Proceedings of the 5th International Conference on Future Energy Systems*, e-Energy '14, pages 157–169, New York, NY, USA, 2014. ACM. 6

[175] Brad Zarikoff and David Malone. A Comparison of RF Exposure in Macro- and Femtocells. *Health Physics*, 105, 2013. 17

[176] Wei Zhang, Mingfa Zhu, Yiduo Mei, Limin Xiao, Li Ruan, Tao Gong, Tao Li, Yunwei Gao, and Yuzhong Sun. LVMCI: Efficient and effective VM

live migration selection scheme in virtualized data centers. In *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*, pages 368–375, December 2012. 12

[177] Michele Zorzi and Ramesh R. Rao. Is TCP energy efficient? In *Mobile Multimedia Communications, 1999. (MoMuC '99) 1999 IEEE International Workshop on*, pages 198–201, 1999. 16