Operating Policies for Energy Efficient Large Scale Computing

Matthew James Forshaw
School of Computing Science
Newcastle University

In Partial Fulfilment of the Requirements for the Degree of
Doctor of Philosophy

Submitted: January 2015

Acknowledgements

First, I wish to express my gratitude to my supervisor Dr Nigel Thomas, who has given me the opportunity to undertake a PhD and has provided me with invaluable support, guidance and crucial insights throughout these four years. Furthermore, I wish to thank Dr Stephen McGough of Durham University, who has been demonstrative and generous in his support throughout the process.

I have had the pleasure of studying at the School of Computing Science at Newcastle University since 2006, through undergraduate and postgraduate degrees, and most recently my PhD. During this time I have had the pleasure of working alongside numerous kind, entertaining and helpful individuals. In particular I wish to thank the following people for their support over the years; Budi Arief, Alex Barfield, Michael Bell, Anirban Bhattacharyya, Matt Collison, John Colquhoun, Marie Devlin, Andrew Dinn, Ryan Emmerson, Paul Ezhilchelvan, Hugo Firth, Carl Gamble, Barry Hodgeson, Phillip Lord, Oonagh McGee, Tudor Miu, Chris Phillips, Ken Pierce, Chris Ritson, Francisco Rocha, Rebecca Simmonds, Gerry Tomlinson, Jennifer Warrender, and Paul Watson. I also wish to acknowledge those with whom I have had the pleasure to collaborate on publications; Ben Allen, Clive Gerrard, Feng Hao, Paul Robinson, Anton Stefanek, Ehsan Toreini, and Stuart Wheater.

Additionally, I wish to thank my external examiner, Professor Stephen Jarvis of Warwick University, my internal examiner Professor Aad van Moorsel, and my independent chair Dr Stephen Riddle, for an enjoyable and engaging viva examination.

Finally, and most importantly, I wish to thank my parents and close friends for their inspiration, patience and support over the years. In particular, Heather and Harry Forshaw, Amy Crimmens, Matthew Marlow, and Michael, Rebecca, Eva and Ashby Ross.

Abstract

Energy costs now dominate IT infrastructure total cost of ownership, with datacentre operators predicted to spend more on energy than hardware infrastructure in the next five years. With Western European datacentre power consumption estimated at 56 TWh/year in 2007 and projected to double by 2020, improvements in energy efficiency of IT operations is imperative. The issue is further compounded by social and political factors and strict environmental legislation governing organisations.

One such example of large IT systems includes high-throughput cycle stealing distributed systems such as HTCondor and BOINC, which allow organisations to leverage spare capacity on existing infrastructure to undertake valuable computation.

As a consequence of increased scrutiny of the energy impact of these systems, aggressive power management policies are often employed to reduce the energy impact of institutional clusters, but in doing so these policies severely restrict the computational resources available for high-throughput systems. These policies are often configured to quickly transition servers and end-user cluster machines into low power states after only short idle periods, further compounding the issue of reliability.

In this thesis, we evaluate operating policies for energy efficiency in large-scale computing environments by means of trace-driven discrete event simulation, leveraging real-world workload traces collected within Newcastle University.

The major contributions of this thesis are as follows:

i) Evaluation of novel energy efficient management policies for a decentralised peer-to-peer (P2P) BitTorrent environment.

ii) Introduce a novel simulation environment for the evaluation of energy efficiency of large scale high-throughput computing systems, and propose a generalisable model of energy consumption in high-throughput computing systems.

iii) Proposal and evaluation of resource allocation strategies for energy consumption in high-throughput computing systems for a real workload.

iv) Proposal and evaluation for a real workload of mechanisms to reduce wasted task execution within high-throughput computing systems to reduce energy consumption.

v) Evaluation of the impact of fault tolerance mechanisms on energy consumption.

Table of Contents

List of figures

List of tables

Nomenclature

List of Acronyms

ACPI Advanced Configuration and Power Interface [103]

CPU Central Processing Unit

DAG Directed Acyclic Graph

DVFS Dynamic Voltage and Frequency Scaling

FCFS First-come, first-served

FGCS Fine-Grained Cycle Sharing Systems

FIFO First In, First Out

HPC High Performance Computing

HTC High Throughput Computing

IaaS Infrastructure as a Service

MPI Message Passing Interface

MTTF Mean time to failure

P2P Peer-to-peer

PUE Power Usage Effectiveness

QoE Quality of Experience

QoS Quality of Service

SJF Shortest Job First

SLA Service Level Agreement

Chapter 1

Introduction

Energy costs now dominate IT infrastructure total cost of ownership, with data centre operators predicted to spend more on energy than hardware infrastructure in the next five years [20]. The U.S. Environmental Protection Agency (EPA) attribute 1.5% of US electricity consumption to data centre computing [43], and Gartner estimate the ICT industry was responsible for 2% of global CO_2 emissions in 2007 [186]. With western european data centre power consumption estimated at 56 TWh/year in 2007 and projected to double by 2020 [32], improving energy efficiency of IT operations is imperative.

In addition to the compelling financial savings sought through reduced power consumption of IT infrastructures, energy saving initiatives are further motivated by legislative pressures. These include the Carbon Reduction Commitment Energy Efficiency Scheme [71] governing private and public sector organisations, UK Government targets of reducing UK carbon emissions by 80% by 2050, and recent calls for a more stringent target of a legally binding 40% energy reduction by 2030 [239].

Finally, there is a social driver for improving energy efficiency in the form of Corporate Social Responsibility , highlighting an organisation's willingness to take the responsibility for the social and environmental impact of their business practices. Responsible business practices can serve as a compelling means of business differentiation in competitive markets, is significant in brand and reputation management, and has been shown to be beneficial in the recruitment and retention of staff [33].

In this thesis we focus on energy consumption in the context of large scale high

throughput computing (HTC) systems. In particular, we consider HTC systems operating on so called *'multi-use'* clusters, where resources are shared with interactive users of the system. As a consequence of increased scrutiny of the energy impact of large scale systems, aggressive power management policies are often employed [213] to reduce the energy impact of these systems. Such policies are often configured to quickly transition servers and end-user cluster machines into low power states after only short idle periods. Consequently, this may result in a negative impact on system performance, further compounding the issue of long-term hardware reliability [121] and lowering the availability of compute resources perceived by applications running in the system.

In this thesis, we evaluate operating policies for energy efficiency and performance in large computing environments by means of trace-driven simulation, leveraging real world workload traces collected within Newcastle University.

1.1 Research Problem

In order to evaluate the energy and performance impact of operating policies within large-scale computing environments, we must investigate the following research problems:

Energy and system modeling for HTC systems Despite a number of previous works considering the energy efficiency of high throughput computing environments, there does not exist a generalisable model for HTC systems and their associated energy consumption.

Trace-driven simulation environment for HTC systems Prior works propose simulation environments capable of modeling grid and Cloud systems; however, these do not explicitely model the operation of HTC systems operating over these compute resources, nor do many offer an adequate model for energy consumption. We must develop a simulation environment, combining our system and energy models such that we may evaluate the impact of operating policies on energy and performance.

Further research problems exist in relation to operational decisions made by HTC systems, and their impact on energy consumption. We investigate the impact of the following operational decisions in this thesis.

Power management In HTC systems capable of the powering on and off of resources, a trade-off exists between powering off idle resources hastily to conserve energy, with the potential for starving the system of required resources and long-term implications for system reliability, and circumspect approaches leading to significant energy waste.

Resource allocation When allocating tasks to a pool of dedicated and non-dedicated resources which are heterogeneous in terms of performance, energy consumption, and reliability, a decision must be made to balance energy consumption while delivering acceptable performance.

Task abandonment When considering a workload comprising a proportion of faulty jobs, as well as unreliable servers, a decision must be made governing when to cease attempting to execute a given job and classify it as being faulty. Doing so too aggressively will lead to tasks being abandoned which would otherwise have completed successfully in a subsequent run, while a conservative approach would impact negatively energy consumption and system load.

Fault tolerance mechanisms The application of fault tolerance mechanisms in HTC systems has the potential for significant improvements in performance, but may incur significant energy consumption. The suitability of applying fault tolerance approaches is dependent on a number of factors including current system load, likelihood of interruption, the composition of resources and characteristics of the offered workload.

1.2 Contributions

The work presented in this thesis makes a number of key contributions:

i) Evaluation of novel energy efficient management policies for a decentralised peer-to-peer (P2P) BitTorrent environment.

ii) Introduce a novel simulation environment for the evaluation of energy efficiency of large scale HTC systems, and propose a generalisable model of energy consumption in HTC systems.

iii) Proposal and evaluation of resource allocation strategies for energy consumption in HTC systems for a real workload.

iv) Proposal and evaluation for a real workload of mechanisms to reduce wasted task execution within HTC systems to reduce energy consumption.

v) Evaluation of the impact of fault tolerance mechanisms on energy consumption.

1.3 Thesis Structure

Chapter 1 describes the motivations behind the work carried out as part of this thesis, and highlights the main contributions of the research. Finally, we describe the related peer-reviewed publications produced throughout the course of the PhD.

Chapter 2 presents technical background material closely related to the work carried out in the chapters of this thesis.

Chapter 3 describes a preliminary investigation into the trade-off between energy and performance, using BitTorrent as an example system. We acknowledge difficulties in enacting energy-efficient operating policies in systems with decentralised decision making.

Chapter 4 outlines the approach we adopt to trace-driven simulation for energy consumption in high-throughput computing systems. We present HTC-Sim for simulating High Throughput Computing systems, and apply this to our case study of the Newcastle University HTCondor pool. We evaluate the impact of running the simulation software both in terms of memory footprint and execution time.

Chapter 5 In Chapter 5 we explore resource allocation and task suspension strategies in high-throughput computing systems in the context of multi-use clusters, and evaluate their impact on energy consumption and performance. We propose resource allocation schemes capable of reducing energy consumption by 55% compared to the polices enacted in the Newcastle University HTC cluster in 2010.

Chapter 6 investigates the issue of *'miscreant'* tasks in HTC systems, and propose a number of mechanisms for curtailing their execution. We show the our approaches to have potential for significant savings in terms of energy consumption.

Chapter 7 evaluates the impact of fault tolerance mechanisms on energy consumption within HTC systems operating within a multi-use cluster environment.

Chapter 8 summarises the conclusions of the work presented in this thesis and motivate future directions for work in the area.

1.4 Related Publications

During the course of my PhD I have contributed to the following peer-reviewed publications.

[88] Matthew Forshaw and Nigel Thomas. A novel approach to energy efficient content distribution with BitTorrent. In *Computer Performance Engineering, Lecture Notes in Computer Science (LNCS) 7587*, pages 188–196. Springer-Verlag Berlin Heidelberg, 2013

This paper introduces an exploratory work into the energy efficiency issues surrounding peer-to-peer (P2P) systems, particularly in the context of BitTorrent. We propose three approaches to promote energy efficient and energy proportional operation of content distribution systems. This paper forms the basis of Chapter 3 of this thesis.

[90] Matthew Forshaw, Nigel Thomas, and A. Stephen McGough. Trace-driven simulation for energy consumption in High Throughput Computing systems. In *Distributed Simulation and Real Time Applications (DS-RT), 2014 IEEE/ACM 18th International Symposium on*, 2014

In this paper we introduce our approach to trace-driven simulation of distributed systems for energy consumption, and perform performance evaluation demonstrating our simulation scales linearly with the modelled workload, for both execution time and memory consumption. This paper forms the basis of Chapter 4.

[161] Andrew Stephen McGough, Matthew Forshaw, Clive Gerrard, Paul Robinson, and Stuart Wheater. Analysis of power-saving techniques over a large multi-use cluster with variable workload. *Concurrency and Computation: Practice and Experience*, 25(18):2501–2522, 2013. ISSN 1532-0634. URL http://dx.doi.org/10.1002/cpe.3082

In this paper we demonstrate the energy and performance impact of resource allocation and task suspension strategies in high-throughput computing systems in the context of multi-use clusters. We demonstrate that these policies could save 55% of the currently used energy for our high-throughput jobs over our current cluster policies without affecting the high-throughput users' experience.

[163] A.S. McGough, M. Forshaw, C. Gerrard, and S. Wheater. Reducing the number of miscreant tasks executions in a multi-use cluster. In *Cloud and Green Computing (CGC), 2012 Second International Conference on*, pages 296–303, 2012. doi: 10.1109/CGC.2012.111

In this paper we explore the impact of so called *'miscreant'* tasks on the energy consumption and overheads of a high-throughput computing system. We investigate techniques to increase the chance of *'good'* tasks completing in a timely manner, while curtailing the execution of *'bad'* tasks. We observe potential reduction in energy consumption of approximately 50 %. This paper forms the basis of Chapter 6 of this thesis.

[89] Matthew Forshaw, A. Stephen McGough, and Nigel Thomas. On energy-efficient checkpointing in high-throughput cycle-stealing distributed systems. In *3rd International Conference on Smart Grids and Green IT Systems (SMARTGREENS)*, 2014

In this short paper we introduce our preliminary investigation into energy-efficient checkpointing in high-throughput systems. We demonstrate through trace-driven simulation the potential of existing checkpointing mechanisms to have a significant negative impact on energy consumption, motivating the need for a class of energy-aware checkpointing strategies. We finally highlight key issues determining whether to employ checkpointing within an HTC environment. This paper contributes in part to Chapter 7 of this thesis.

[91] Matthew Forshaw, A. Stephen McGough, and Nigel Thomas. Energy-efficient checkpointing in high-throughput cycle-stealing distributed systems. *Electronic Notes in Theoretical Computer Science*, 310:65–90, 2015

In this extended paper we expand on ideas introduced in [89], introducing a number of candidate energy-aware checkpointing strategies and conduct a full evaluation of their performance for our HTCondor workload from 2010. We demonstrate the naive application of checkpoint mechanisms to lead to a significant negative impact on energy consumption, and show new approaches energy- and load-aware strategies may lead to significant benefits. This paper and [89] form the basis for Chapter 7 of this thesis.

[41] Jeremy T. Bradley, Matthew Forshaw, Anton Stefanek, and Nigel Thomas. Time-inhomogeneous population models of a cycle-stealing distributed system. In *29th Annual UK Performance Engineering Workshop (UKPEW) 2013*, pages 8–13. Loughborough University, 2013

This paper presents initial application of Hybrid PCTMC (*h*PCTMC) modelling for our HTCondor workloads. The future research directions based on this preliminary work is discussed in detail in Chapter 8 of this thesis.

[159] A. Stephen McGough, Matthew Forshaw, Gerrard Clive, Wheater Stuart, Allen Ben, and Robinson Paul. Reduction of wasted energy in a volunteer computing system through reinforcement learning. *Sustainable Computing, Informatics and Systems*, 2014. doi: 10.1016/j.suscom.2014.08.014

This paper builds upon our previous investigations into resource allocation strategies for HTC systems [161], and present a Reinforcement Learning (RL) [226] approach to resource allocation and task delay decisions. We evaluate the ability of this machine learning approach to adapt to a changing operating environment. We find the approach capable of yielding energy reductions of 30% with no impact on task completion, or up to 53% in situations where a modest overhead increase may be incurred. This paper demonstrates the potential of Reinforcement Learning for parameterless

operating policies, and forms the foundations of ongoing research efforts discussed in detail in Chapter 8.

Cloud-related papers

Furthermore, the following publications were produced during the course of the PhD in the areas of high-throughput computing and Cloud computing. These works consider the energy impact and energy cost on local infrastructure, but we do not currently have reliable information on cloud energy consumption so consider only financial cost of cloud operation. Since we do not explicitly consider energy consumption of cloud resources, these works do not form part of the thesis.

[162] Andrew Stephen McGough, Matthew Forshaw, Clive Gerrard, Stuart Wheater, Ben Allen, and Paul Robinson. Comparison of a cost-effective virtual cloud cluster with an existing campus cluster. *Future Generation Computer Systems*, 2014. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2014.07.002

In this paper we explore the viability of running institutional high-throughput computing workloads on the Cloud. We explore a number of policy decisions governing resource allocation decisions, cloud instance keep-alive, and the delayed deployment of jobs. We evaluate the operation of cloud-based and local clusters in terms of financial cost, with local cost including the proportional share of hardware ownership and energy consumption.

Other Publications

Furthermore, the following journal and conference papers were produced during the course of the PhD on related topics but do not form part of the thesis.

[179] Thai Ha Nguyen, Matthew Forshaw, and Nigel Thomas. Operating policies for energy efficient dynamic server allocation. In *30th Annual UK Performance Engineering Workshop (UKPEW 2014)*, 2014

[204] Kiavash Satvat, Matthew Forshaw, Feng Hao, and Ehsan Toreini. On the privacy of private browsing - a forensic approach. *Journal of Information Security and Applications*, 19(1):88–100, 2014

[203] Kiavash Satvat, Matthew Forshaw, Feng Hao, and Ehsan. Toreini. On the privacy of private browsing - a forensic approach. In *8th DPM International Workshop on Data Privacy Management*. Royal Holloway, University of London, 2013

Chapter 2

Background

Summary

This chapter provides an overview of the relevant background material motivating and underpinning the work conducted in this thesis. Section 2.1 discusses previous works to evaluate and classify the energy consumption of computer systems, from server- to datacentre level. In Section 2.2, mechanisms leveraging these observations are used. Finally, in Section 2.3 we discuss works directly related to each chapter of this thesis.

For a comprehensive summary of approaches to energy-efficient computing, readers are advised to refer to the taxonomy and survey presented by Beloglazov *et al* [28]. Furthermore, Reda *et al* [196] offer a detailed survey of low-level power measurement techniques of computing devices.

2.1 Energy characteristics

In this section we discuss the energy characteristics at server and datacentre level, and detail efforts in the literature to develop predictive models and benchmarks of energy consumption.

2.1.1 Servers

We categorise the literature concerning energy characteristics at the server level as follows. Early works consider the use of low-level system metrics to derive predictive models of server energy consumption (Section 2.1.1). These observations lead to the introduction of a number of industry benchmarks for server energy efficiency 2.1.1. Further works have emphasised the importance of energy proportional system design (Section 2.1.1) and temperature (Section 2.1.1) on the energy characteristics of servers.

Predictive Models

The energy consumption of server and commodity hardware has been studied extensively in the literature. Early works leveraged low-level metrics such as performance counters [25, 212] when developing predictive models of energy consumption, while others aimed to simulate individual [42, 258] or groups of system components [11, 94]. These models tend to require significant architecture knowledge and typically were not generalisable to other hardware, nor scalable to entire computer systems.

Fan *et al* [82] observed that total power consumption of server hardware was strongly correlated with CPU utilisation, and power consumption may be modeled linearly for values between active/idle and peak CPU consumption. The authors also present a linear model as well as an empirical model which includes a parameter which may be obtained through a calibration phase.

Economou *et al* [73] introduce the Mantis model, which extends [82] to also include the energy consumption of memory, storage and network subsystems. The model relies only on readily-obtainable server utilisation metrics, and a single calibration step where resource utilisation is correlated with full-system power consumption. The resulting models benefit from broader applicability to non-CPU-dominant workloads, and for systems whose energy consumption is not dominated by the CPU (e.g. systems with a very large RAM provision).

More recently, Davis *et al* [65] further explore predictive power models using performance measures made available within the Microsoft Windows operating system. However, these models are not applicable to systems running alternative operating systems. Davis *et al* also explore the inter-node variability within homogeneous clusters [64], demonstrating that applying power models obtained from a single node to the rest of the cluster is insufficient in achieving quality predictions. However, this work is limited by using the same OS-specific measures as in [65].

Predictive models of energy consumption typically use the power consumption at peak resource utilisation to represent maximum energy consumption for a server. Meisner *et al* [164] challenge this assumption, demonstrating interactions between server utilisation and the behaviour of switched-mode power supplies, and propose

an operating system-level metric to more accurately predict peak power consumption for a commodity and enterprise-level server.

Benchmarking

SPECpower_ssj2008 [134], released in November 2007, was the first industry-standard benchmark designed to evaluate and provide a means of comparison between measured performance and measured power consumption. SPECpower extends existing SPEC benchmarks incorporate energy meaurement, and is based on an enterprise Java workload. The benchmark exerts graduated levels of load on a given machine, typically evaluating the energy consumption and performance of server hardware between active-idle (0%) and peak (100%) load at 10% graduated load levels. More recently, SPEC released SPECvirt®sc2013 [220], which combines a variety of benchmark workloads (including web server, application server, mail server and CPU-dominant workloads) to evaluate the performance of servers for virtualised environments.

Other energy-aware performance benchmarks include Storage Performance Council (SPC) benchmarks for the energy efficiency of storage systems [224] and TPC-Energy by the Transaction Processing Performance Council (TPC), a benchmark focusing on transactional database systems [236]. A common limitation of many existing energy consumption benchmarking approaches is the dependance on specific workloads, with performance and energy characteristics unpredictable between workloads. Poess *et al* [188] present a survey of energy benchmarks for server systems.

Energy proportionality

Barroso *et al* [19] were first to highlight the need for energy-proportional server designs. An *'energy proportional'* system is defined as one which exhibits a wide dynamic power range (the proportion of system power consumption attributable to the load placed upon the system and its sub-components) and low energy consumption while in an active/idle state (often referred to as the static component of a server's power consumption). They further highlight the impact of traditional server provisioning strategies, leading to typical server CPU consumption of between 10 and 50 percent. This is

true also of the desktop estates we consider as part of this work; with desktop idle time reported in the literature has been shown to be in excess of 75% [174].

Publicly available results from the SPECpower_ssj2008 [134] benchmark have formed the basis of a number of analyses of trends in server energy efficiency [108, 243, 254].

Varsamopoulos *et al* [243] make use of the 139 entries published from 2007 to the time of publication. Two energy proportionality metrics are proposed. First, the *idle-to-peak power ratio* (IPR) metric is defined as

$$\text{IPR} = P_{idle}/P_{peak} \tag{2.1}$$

where P_{idle} and P_{peak} represent a server's energy consumption in idle and peak states respectively. The IPR metric is normalised, allowing direct comparison between servers, with lower IPR values denoting a more energy-proportional system.

Secondly, the *linear deviation ratio* (LDR) metric is expressed as

$$\text{LDR} = max_u^{|\cdot|} \frac{P(u) - ((P_{peak} - P_{idle})u + P_{idle})}{(P_{peak} - P_{idle})u + P_{idle})} \tag{2.2}$$

where $max_u^{|\cdot|}$ is the maximum value, retaining the sign of the maximum value. Lower LDR values signify a more linear energy profile, with negative and positive LDR values denoting sublinear and superlinear energy profiles respectively. The LDR metric is normalised, allowing for direct comparison between systems.

Varsamopoulos *et al* compute these metrics for each of 139 published SPECpower results, and observe clear historical trends in reductions of *idle-to-peak power ratio* (IPR) and increasing *linear deviation ratio* (LDR) values over time, with power profiles becoming more proportional but less linear over time.

Wong *et al* [254] also propose metrics for server energy proportionality, including the deviation between the target machine's energy proportionality curve and that of a perfectly energy proportional machine. Contrary to the work of Varsamopoulos et al [243], this measure does not consider the typical operating conditions of servers, characterised by low system utilisation.

Hsu *et al* [108] analyse a larger corpus of SPECpower results (a total of 177 col-

lected between 2007 and 2010), and challenge the suitability of linear models to represent energy consumption, demonstrate that simple nonlinear functions may be fitted to model the energy profile of servers under aggressive power management schemes. These authors later revisit the area, presenting quadratic models of energy consumption [109].

Temperature

Kazandjieva et al [125] introduce PowerNet, a monitoring infrastructure designed and deployed within Stanford Unviersity across a six month period. PowerNet comprised 85 power meters and collected power consumption and CPU resource utilisation for a group of fifteen workstations and ten servers. A key finding of the study was the impact of ambient temperature on on the power consumption of server hardware. When evaluating a rack of homogeneous 1U servers in a rack, the study observed a server in the highest position in the rack, where temperature will be greater, consumed as much as 20% more energy than that of its neighbours. [73, 198].

Ambient temperature has also been shown to be an important factor in system reliablity, with every 10C temperature increase over 21C shown to decrease the reliability of electronics by 50% [225].

2.1.2 Data centres

Energy consumption in large-scale computing systems originates not only from compute nodes, but also the data centres and clusters in which they are housed. In particular, the cost of powering and cooling within data centres has been shown to dominate data centre costs [231], with cooling costs attributed for as high as 50% of total cost in some cases [205]. The American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) estimate Infrastructure and Energy (I&E) to contribute 75% of total IT energy [23] in 2014. Another significant contributor to energy consumed within data centres are network hardware, which may consume as much as half of the energy consumed by server hardware [210].

Pelley *et al* [184] offer an early attempt at developing an analytical model to rep-

resent the total energy consumption of a data centre. The model comprises a number of components which individually estimate power consumption of servers, power conditioning systems, heat and airflow, as well as cooling equipment. The model has subsequently formed the basis of a number of analytical and simulation studies.

Energy-efficiency metrics

A number of energy-efficiency metrics for data centre operation are proposed in the literature.

The Power Usage Effectiveness (PUE) metric proposed by The Green Grid [21], quantifies the proportion of total facility power consumed by the IT equipment (including servers, network and storage equipment) within the facility, and may be expressed as follows:

$$\text{PUE} = \frac{\text{Total IT Equipment Power}}{\text{Total Facility Power}} \tag{2.3}$$

A PUE value of 2.0 would signify that for every watt of energy used by IT equipment within the datacentre, an additional watt is consumed by the power delivery and cooling systems. Reported PUE values facilities are commonly misinterpreted as an absolute measure of efficiency, with a lower PUE value representing greater efficiency. Rather, PUE is a measure of IT equipment power draw *relative* to total facility power. Two facilities with identical computing capacity and reported PUE values may exhibit very different levels of power consumption, depending on server efficiency and utilisation levels.

Where a PUE value is a ratio, The Green Grid [21] also propose the Data Center infrastructure Efficiency (DCiE) which is defined as follows:

$$DCiE = \frac{1}{PUE} = \frac{\text{IT Equipment Power}}{\text{Total Facility Power}} * 100\% \tag{2.4}$$

Carbon Usage Effectiveness (CUE) [22] is a further metric proposed by The Green Grid which seeks to quantify the operational carbon usage for datacentres, and is de-

fined as follows:

$$CUE = \frac{\text{CO}_2 \text{ emitted (kg CO}_2 \text{ eq)}}{\text{unit of energy (kWh)}} * \frac{\text{Total Data Centre Energy}}{\text{IT Equipment Energy}} \tag{2.5}$$

Finally, the SWaP (Space, Watts and Performance) metric introduces space as a third criteria for comparison, and is defined as:

$$SWaP = \frac{\text{Performance}}{\text{Space x Power Consumption}} \tag{2.6}$$

where performance is measured using industry-standard benchmarks (though guidance as to which is not specified formally), power consumption is measured in Watts, and space is measured as the total count of rack units for the system. SWaP not only has applications in comparison of facilities, but also of servers. While maximising the performance per rack unit is desirable, many facilities are constrained by the power density supported by power distribution and cooling subsystems.

A number of additional metrics have been proposed in the literature. The Rack Cooling Index (RCI) [102] is proposed as a measure of how effectively datacentre racks are cooled and maintained with respect to industry thermal guidelines and standards.

Wang *et al* provide a detailed survey of data centre performance and energy-efficiency metrics [250].

2.2 Energy-efficiency mechanisms

Here we introduce a number of important works which leverage the energy characteristics of computer systems discussed in Section 2.1 to achieve energy savings, enacted at various levels in the system from software-level approaches to those governing the operation and decisions made at network and full-system levels.

2.2.1 Software

Sampson *et al* [202] propose EnerJ, a framework which leverages the observation that high-precision computation is more costly in terms of execution time and energy consumption. EnerJ extends the Java programming language to provide support for

both *approximate* and *precise* data types by way of annotations. In doing so, EnerJ is able to relax the precision of certain computations to reduce energy consumption. The authors port a number of existing open-source Java applications to EnerJ and demonstrate energy savings of 10-50%.

Kansal *et al* [122] concentrate on the potential for energy saving at the software design stage, presenting a tool promoting energy-aware programming which leverages energy profiles and application characteristics to guide software developers' choice of data structures and algorithms.

Koller *et al* [131] acknowledge the significant impact of the target workload on the accuracy of energy predictions [244], introducing WattApp, a framework for application-aware energy prediction within shared datacentres. The authors propose an approach whereby applications are benchmarked on each class of server within the system; however, in our context it is unreasonable to assume wide-spread power instrumentation of our infrastructure, nor longitudinal measurement of power consumption of these resources.

A number of approaches are suggested at the operating system level. Early works in the area, which include ECOsystem [259] and Nemesis OS [178], concentrated primarily on improving energy-efficiency for battery-powered devices, allowing developers and operators of the system to specify target battery lifetimes and quality of service (QoS) requirements for applications. More recently attention has focused to operating system level approaches in the context of servers. Meisner *et al* [165] explore the use of low- and high-power operating states with fast transition times to achieve energy savings and promote energy proportional operation. However, they acknowledge transition times between operating states is not sufficiently fast in current generation hardware.

2.2.2 Virtualisation

Virtualisation is commonly used to reduce energy consumption of large-scale computing, allowing consolidation of workloads onto a smaller number of servers, and the subsequent powering down of idle servers.

Verma *et al* [244] provide one of the earliest works exploring the dynamic allocation of resources in virtualised environments to optimise energy consumption and performance. In doing so they present the pMapper framework, which controls not only the placement of virtual machines across physical hosts, transitions hosts into low-power states and applying DVFS to reduce energy consumption, and consolidates workloads onto fewer hosts using live migration. The authors extend this work in [246], extending pMapper to also consider workload characteristics when allocating applications to VMs.

Beloglazov *et al* [29] present a number of heuristic policies for the energy-aware allocation of resources within Cloud systems, though their approach may be applied to typical virtualised environments. The efficacy of the proposed heuristics are evaluated in terms of energy consumption and Quality of Service (QoS) violations, by simulation using the CloudSim [46] toolkit. The proposed heuristics now form the basis for the resource allocation, VM migration and consolidation strategies in Openstack Neat [27].

2.2.3 Datacentre

The static dynamic capacity provisioning of servers within data centres for energy efficiency has formed the basis of many studies in the literature. Such approaches may assume servers to be homogeneous or heterogeneous, may consider performance and/or thermal issues, and may be analytical or simulation-based. In this section we focus on works closely related to the work carried out within this thesis, and for a detailed taxonomy of these works we direct the reader to [28].

Ranganathan *et al* [195] explore the energy efficient management of groups of servers in a high-density blade enclosure setting, demonstrating - through simulation and prototyping - energy savings while maintaining comparable levels of performance. DVFS is applied to reduce CPU power consumption, while at an ensemble level decisions are made whether to power down particular blades within the enclosure.

Moore *et al* [172] investigate temperature-aware workload placement within data centre environments. A number of heuristics are proposed, including unform job placement throughout the datacentre, favouring servers located in cooler areas of the

datacentre, and extending a previous work [211] to reduce the formation of *'hot spot'* areas within the datacentre. The developed approaches are evaluated in simulation using a computational fluid dynamics (CFD) model representing a datacenter, demonstrating the potential for energy savings.

Adnan *et al* [2] demonstrate the energy savings possible when considering slack scheduling in a MapReduce cluster with user-specified deadlines. In their approach, a minimum active set of servers is maintained to satisfy the offered workload, with tasks dynamically deferred to occupy predicted periods of low utilisation. Simulation and experimental results are presented, demonstrating savings of between 20% and 40% compared to a conventional *"follow the workload"* resource provisioning approach [144].

Tiwari *et al* [233] demonstrate the energy savings possible through Application-aware Dynamic Voltage-Frequency Scaling (DVFS) within an HPC environment. The authors explore the effect of reduced frequency on both power and performance, noting that DVFS has the potential to lead to sub-optimal performance (16% penalty). A system called *Green Queue* is proposed, an application-aware analysis and runtime framework, which enacts CPU clock frequency changes in response to observed load. An inter-node approach reduces the CPU clock frequency for inactive nodes, while the intra-node approach exploits period of application execution dominated by communication, where computational work is stalled while a node awaits required data. Performance evaluation is conducted on a 1024-core Intel Sandy bridge-based supercomputer at the San Diego Supercomputer Center (SDSC), reducing the performance penalty of DVFS from 16% to 2.4%. Energy savings are shown to vary based on offered workload, with mean savings of 10.6% and 17.4% across application runs for the intra-node and inter-node techniques respectively.

Furthermore, a number of studies have sought to mitigate the energy consumption associated with data transfer. For example, Chen *et al* [54] achieve savings by trading off transfer costs and the computational expense of data compression.

Energy proportionality

A number of works have sought to achieve energy-proportional energy characteristics by leveraging low-power sleep states, dynamic provisioning and scheduling of groups of heterogeneous servers.

Krioukov *et al* [133] introduce NapSAC, combining low-power, computationally constrained systems with normal servers, and demonstrate potential energy savings of 63% compared to a cluster provisioned for double of peak load, and 27% for a static cluster right-sized for peak load.

Wong *et al* [254] propose *Knightshift*, an architecture combining a highly performance primary server with a low-power computationally-constrained node (the *Knight*). The authors further offer a discussion of potential implementations, with the Knight node either embedded on the motherboard of the primary server, contained within the rack case (e.g. as a harddrive module), or an 'ensemble' approach where the Knight node is external to the primary server. The authors evaluate their proposed approach for published SPECpower results indicating potential energy savings of up to 75%.

Tolia *et al* [235] take a similar approach to that of [133], but also focus on the energy proportionality of cooling, dynamically adjusting fan control to minimise energy consumption while satisfying thermal constraints. Similar efforts have also been applied to achieving energy-efficient storage subsystems [262].

Data centre operation under power budget constraints

A number of works at the datacentre-level have considered server operation subject to power budget constraints. Approaches discussed here include power capping, power shifting and power routing.

Lefurgy *et al* [138] were first to propose the notion of *'power capping'*. Power capping encapsulates mechanisms which seek to control the peak power consumption of high-density servers, by periodically selecting the operating state whose performance capabilities are greatest, yet still reside within a fixed power budget. The ability to constrain the peak energy consumption of a server is desirable because it allows datacentre

operators to more closely provision power and cooling infrastructures to match typical requirements.

Cochran *et al* [60] combine DVFS and thread packing on multi-core processors to maximise performance subject to budgetary constraints. The authors evaluate the efficacy of their approach for quad-core Intel i7 processors using the PARSEC parallel benchmarks [34] both with and without longitudinal power measurement, meeting power constraints 96% and 82% of the time respectively. However, it is unclear how readily such an approach might be applied to other architectures.

Felter *et al* [86] explores *'power shifting'*, an approach whereby a server's operating adheres to a power budget, and a system power manager divides this budget between the various subsystems (e.g. CPU, memory). This allocation is informed by knowledge of the workload, and seeks to maximise performance while observing to the specified budgetary constraint.

Pelley *et al* [185] adopt a similar approach at the rack level within a datacentre, allocating power budgets to racks subject to the energy requirements and in response to the workload allocated to servers within each rack. This dynamic allocation of power is made possible through a novel topology for power distribution, where secondary power feeds serve multiple PDUs (power distribution units), minimising the need for reserve power capacity.

2.2.4 Network

Much prior literature places an emphasis on energy saving through consolidation and powering down of servers, with network infrastructure often considered to have a fixed energy cost, with idle power of the current generation of network devices is shown to be as high as 95% [52]. Hlavacs *et al* [105] further evaluate consumer and enterprise switch hardware and find the load-dependent component of their power consumption to be insignificant. However, more recently works have sought to augment network operation to achieve energy savings. Bolla *et al* [36] present a survey of recent efforts to promote energy efficient operation of network hardware.

Nedevschi *et al* [176] propose three schemes to reduce the energy consumption

of network hardware. The authors first motivate the need for energy-efficiency sleep modes, such that energy consumption may be minimised during periods of inactivity. Secondly, the paper explores the use of Dynamic Voltage Scaling (DVS) within network hardware to promote closer to energy proportional characteristics under variable load levels. Finally, the authors highlight the need for coordination between network nodes, such that offered workload may be consolidated to fewer network devices, allowing others to gain further benefits from available sleep modes. The work indicates potential savings of 50% for networks whose utilisation is low (10-20%).

Lee *et al* [137] offer a detailed investigation of energy efficiency issues in content dissemination strategies and go further to demonstrate by way of trace-driven simulation the potential benefits of content-centric networking (CCN) [114] for energy efficiency.

The introduction of Software-Defined Networking (SDN) offers interesting possibilities for new works reducing energy consumption. Tu *et al* [237] present the first work in this area, exploring the possibility to reduce energy consumption by controlling the flow path of traffic in an SDN network. Two policies are proposed, a 0-1 Integer programming model and a greedy algorithm, leading to a 30-40% reduction in energy cost.

2.2.5 Federated / multi-site environments

We now discuss works which consider energy-efficient operation of large-scale systems beyond a single datacentre. These approaches typically leverage high-level knowledge of data centres in a heterogeneous multi-site context, distributing work to sites to reduce energy consumption and environmental impact while maintaining satisfactory levels of performance.

Stewart *et al* [223] discuss renewable-aware datacentre management, promoting the use of intermittent renewable power source (e.g. wind turbines and solar power) to power datacentres, with workload distribution mechanisms aware of the availability of renewable power.

Pierson [187] highlights the importance in systems of not simply reducing the raw

energy consumption of large systems, but rather the ecological impact of the energy source (governed by the means of electricity production). We address this requirement within our work by modeling the carbon impact of the energy source for a given cluster (see Chapter 4).

Energy efficiency is of increasing importance as we approach extreme-scale computing. The importance of energy efficiency at exascale is first introduced by Bergman *et al* [30], leading to a number of works in this area [14, 47, 252]. Wilde *et al* [252] emphasise the need for a holistic approach, highlighting the importance of energy efficiency at infrastructure, hardware, software and application levels and introducing specific optimisations which may be made at each of these four levels. Auweter *et al* [14] extend this work, exploring energy aware scheduling on the SuperMUC HPC system, using execution time and energy consumption prediction to inform frequency scaling on a per-application level. Reported energy savings of 6% appear modest, but translate to an annual cost saving of €200,000.

2.3 Directly Related Work

Here we describe works in the literature which are directly related to the work carried out as part of this thesis. Section 2.3.1 presents works related to our investigation into energy efficient content distribution in BitTorrent in Chapter 3. Section 2.3.2 explores various approaches to evaluating energy consumption in large-scale systems, and motivates our choice of trace-driven simulation as the basis of works in Chapters 4, 5, 6 and 7. Section 2.3.3 presents previous works in energy-aware and energy efficient resource allocation in high-throughput and high-performance computing environments, related to our investigation in Chapter 5. Section 2.3.4 outlines previous work in job reallocation and abandonment policies in high-throughput computing environments, forming the basis for our investigation of miscreant task executions in Chapter 6. Finally, Section 2.3.5 explores work directly related to our investigation of energy-efficient checkpointing in Chapter 7.

2.3.1 Energy efficient content distribution with BitTorrent

Early research considering BitTorrent energy efficiency focused primarily on file sharing using devices with limited battery and computational power [126].

Anastasi *et al* [5] propose a scheme allowing multiple peers within a typical LAN environment to delegate the task of downloading to a designated proxy server which takes part in the BitTorrent protocol on their behalf. Meanwhile these peers "behind" the proxy can be switched off without interrupting the download. Upon completion of the download, the requested files are transferred back to the peers.

Blackburn and Christensen [35] introduce a wake-up semantic to the BitTorrent protocol, allowing peers to sleep while remaining active in the system. Centralised control is assumed whereby these peers may be sent a packet and woken up remotely.

Andrew *et al* [8] propose a system to balance the power consumption of servers and peers involved in a peer-to-peer download. This approach assumes centralised control over all peers, enabling these peers to be powered on and off to maximise the download rate of a subset of awake peers.

Chen *et al* [53] explore the impact of two seeding strategies proposed in the BitTorrent specification in the presence of varying levels of freeloading. Mathematical models of both approaches are developed, and validated using a discrete-event simulation.

Hlavacs *et al* [106] consider the application of BitTorrent in a residential setting, extending the analytical model presented in [191] to determine the optimal number of seeders to reduce global energy consumption of the BitTorrent swarm. The results of this analytical model is found to closely follow that of an associated simulation model.

Performance Evaluation

A number of approaches to the evaluation of BitTorrent are proposed in the literature.

Deaconescu *et al* [69] propose a virtualised testing environment for BitTorrent applications, offering greater control over the conditions under which experiments are conducted. However, the need for large computational resources to host these virtualised experiments will be prohibitive to many, does not allow for energy consumption measurement, and prohibits evaluation at scale. Consequently we do not consider the virtualised testing approach in our work.

An alternative is the use of simulation. A number of efforts extend well-established packet-level network simulators to model the behaviour of a BitTorrent network. Katsaros *et al* [123] extend the OMNeT++ [242] discrete event simulation environment to model BitTorrent protocol. Furthermore, Evangelista *et al* [79] also extend OMNeT++ in developing EBitSim. Souza *et al* [217] extend ns-3 [101] in VODSim to add BitTorrent functionality with an emphasis on the evaluation of video-on-demand (VoD) applications. TorrentSim [17] is a Java-based simulation environment for BitTorrent systems, based on Simmcast [18], and due to ease of extension and operating system independence was selected as the basis for our work.

2.3.2 Evaluation of energy consumption in large-scale systems

Throughout our work we employ a trace-driven simulation approach to evaluating the performance and energy consumption of operating policies within high-throughput

computing environments. In this subsection, we discuss approaches to evaluating the energy consumption of large-scale distributed systems, classifying works based on their adopted approaches - namely simulation, experimental testbed and emulation - and provide a survey of simulation approaches applicable to our research area.

Simulation

A number of Grid and Cluster level simulators exist, including SimGrid [139], GridSim [45], and OptorSim [24] though these focus more at the resource selection process both within clusters and between clusters and lack the modelling of energy. More recently Cloud simulators have been proposed which are capable of modelling the trade-off between not only cost and Quality of Service, but also energy consumption. These include CloudSim [46], GreenCloud [129], SiCoGrid [167] and MDCSim [143]. However, these do not allow modelling of multi-use clusters with interactive user workloads, nor do they support checkpointing.

In Table 2.1 we provide an overview of currently available simulation environments for the modelling of grid and cloud systems. We select the following criteria against which we evaluate the capabilities of each simulation environment.

Energy model Does the simulation framework enable the modeling of energy consumption by default?

Performance / SLAs Does the simulation framework support the collection of performance metrics and/or Service Level Agreement violations?

Multi-use / Interactive users Does the simulation framework model compute resources as being dedicated (solely for the purpose of the computational workload running on the system) or does it also support interactive users to these machines?

Can use real workload traces Does the simulation framework support the use of real workload traces (such as those we introduce in Section 4.3.1)?

Fault tolerance / checkpointing Does the simulation framework include in-built modeling of fault tolerance approaches such as checkpointing and migration?

Language Which implementation language(s) is the simulation framework based upon?

On-demand provisioning Does the simulation framework support the on-demand provisioning of compute resources during the execution of a simulation run, or must the simulated environment be statically defined prior to the simulation run commencing?

Virtualisation Does the simulation framework support the modelling of virtualised compute resources, where multiple virtual machines may run on a single physical resource? Such a modeling capability allows for reasoning over virtual machine migration and consolidation decisions.

Heterogeneous resource models Does the simulation framework support the representation of heterogeneous compute resources, both in terms of their performance and energy consumption (where supported)?

Underlying framework Does the simulation framework extend other more general simulation frameworks?

Software license Under which software license is the simulation framework released?

Publicly available Is the source code for the simulation framework currently freely and readily available online?

Latest release (as of 02/09/2014) Used as a measure of activity of ongoing development, the latest release date for the simulation framework at the time of thesis submission.

We observe a number of commonalities between the capabilities of the simulation frameworks we consider. We observe that all simulation frameworks support the modelling of performance/SLAs, with the exception of OptorSim. Similarly, all simulation frameworks except MDCSim support the modelling of heterogeneous compute resources. We observe that the implementation language upon which the simulation frameworks are based is dominated by Java. We favour Java as a choice for implementation language because of the simplified deployment of Java applications across an

HTCondor pool comprising multiple platforms, and due to our own familiarity with the language. Table 2.1 illustrates that our simulation framework HTC-Sim exhibits novelty in its ability to model multi-use cluster and the presence of interactive users, and is one of few which incorporates real workloads and fault tolerance mechanisms.

We find few simulation frameworks support the modelling of virtualised resources, something we also see as an important area of future development for HTC-Sim (as discussed in Section 4.5).

Furthermore, a number of works provide simulation support for fault tolerance mechanisms relevant to Chapter 7 of this thesis.

Zhou *et al* [263] propose an extension to the CloudSim [46] framework to support simulation of fault tolerance mechanisms but its codebase has not been made publicly available.

Vieira *et al* [247] propose ChkSim, a Java-based simulation environment for the evaluation of checkpointing algorithms. The tool focuses on checkpointing approaches for workloads comprising groups of dependent processes communicating with one another across the network, equivalent to an MPI HPC workload. ChkSim focuses on the number of unused checkpoints as its key metric of checkpoint performance; however it does not assess the impact of checkpointing schemes on energy consumption and may not easily be adapted to model a high-throughput environment and interactive user workloads.

Experimental testbeds

Experimental testbeds, comprising a number of physical and virtual machines, are frequently considered for the evaluation of large-scale systems. Practitioners may opt to use one of a number of existing experimental testbeds, or build their own private testbed. A key consideration in selecting a testbed is the trade-off between the capital investment to acquire the required hardware infrastructure and operational expenditure of using an external service. In the context of using testbeds for scientific experimentation, project scale and duration are significant factors. However, when considering the use of testbeds for the evaluation of energy efficiency, the domain is

Table 2.1: Comparison of simulation frameworks

	CloudSim [46]	GreenCloud [130]	HTC-Sim	MDCSim [143]	OptorSim [24]	SiCoGrid [167]	SimGrid [139]	GridSim [45]
Energy model	✓	✓	✓	✓	–	–	–	–
Performance / SLAs	✓	✓	✓	✓	–	✓	✓	✓
Multi-use / Interactive users	–	–	✓	–	–	–	–	–
Can use real workload traces	–	✓	✓	–	–	–	✓	–
Fault tolerance / checkpointing	–	–	✓	✓	✓	–	–	–
Language	Java	C++	Java	Java	Java	Haskell	C	Java
On-demand provisioning	✓	✓	✓	?[a]	✓	✓	✓	✓
Virtualisation	✓	✓	–	–	–	–	–	–
Heterogeneous resource models	✓	✓	✓	?	✓	✓	✓	✓
Underlying framework	SimJava [107] <=2.0, now custom core	NS-2	–	CSIM [209]	–	Parsec [140] and DiskSim [44]	–	–
Software license	LGPL	GPL	–	–	EUDatagrid	–	LGPL	GPL
Publicly available	✓	✓	–[b]	–	✓	–	✓	✓
Latest release (as of 02/09/2014)	02/05/2013	19/12/2013	–	–	24/10/2006	–	02/06/2014	25/11/2010

[a] (?) Information was not obtainable on either of a) the reference (or associated) publications for the simulation tool, nor b) on the homepage for the tool.

[b] A publication for HTC-Sim is currently under preparation for the Journal of Open Research Software, and will be made publicly available following the review process.

dominated by private testbeds, with very few public infrastructures reporting energy metrics. One exception is BonFIRE [37, 124], a scientific testbed distributed across seven sites in Europe. A number of these sites operate managed power distribution units (PDUs) within the data centres and expose end-user energy consumption to its users. A number of frameworks supporting private testbeds emphasise the evaluation of energy consumption, e.g. Enacloud [141] and Openstack Neat [27].

Emulation

A further approach considered in a number of works is the emulation of large-scale systems. In an emulation approach, performance evaluation is carried against the concrete implementation of the system under test, rather than a simulated implementation. Such an approach boasts a number of key benefits, alleviating the need for an abstract model for the system required in simulation or analytical approaches, and allowing the same code used for experimentation to be deployed into a production environment. Naicken *et al* [175] observed significant inconsistencies between results produced by multiple simulation frameworks modelling the same distributed environment, and attribute this variability to inconsistencies between abstract models and implementations, making the ability to tightly couple experimental and production code highly desirable. An emulation approach has been used in the context of peer-to-peer (P2P) [80] systems and networking [104], but few have accounted for the energy consumption of systems in emulation approaches, e.g. [48]. A significant constraint on emulation-based experiments is that of scale, with emulations frequently shown to be capable of evaluating systems with orders of magnitude fewer entities. In our context of large-scale high throughput computing systems, many of the operating decisions and policies we propose may only be evaluated meaningfully at scale so we do not pursue an emulation approach further.

Grid workload traces

When evaluating the impact of operating policies on energy consumption and performance within large-scale computing environments, it is highly desirable to possess trace workloads from production environments. A number of workload traces from

grid systems are available in the literature, most prominently through initiatives such as The Grid Workloads Archive [10, 112] and Parallel Workloads Archive [1, 84]. When considering operating policies dealing with failures and the volatile nature of non-dedicated resources, it is also necessary to acquire traces of machine failures. While a number of initiatives exist to aggregate such datasets, including the USENIX computer failure data repository (CFDR) [206, 207] and Failure Trace Archive [81, 118], all existing trace focus on machine failure traces for HPC and datacentre systems. We acknowledge a gap in the area for a trace of failures for workstations in an office environment.

2.3.3 Resource Allocation

Minartz *et al* [170] proposed switching off nodes within a high-performance cluster to save energy. We go further in this work to show how different policies over how jobs are distributed around a high-throughput heterogeneous cluster can be more energy efficient. Minartz *et al* goes further to model the power consumption of individual components within a system based on the computation performed. This could be adapted to work with our system.

Verma *et al* [245] explore the impact of dynamic consolidation and the use of low-power operating states in the placement of HPC applications within a virtualised environment. Terzopoulos *et al* [228] investigate the use of Dynamic Voltage Scaling techniques to reduce energy consumption in a heterogeneous cluster to conform to power budgets imposed by infrastructure.

Niemi *et al* [180] demonstrated that running multiple jobs on the same node within a high-performance cluster was more energy efficient. We expect such to be the same here for our work. Though at present we lack the knowledge about execution load for our workload to determine if this would work well.

Ponciano *et al* [189] evaluate strategies for energy-aware resource provisioning and job allocation within opportunistic grids, transitioning worker nodes into energy-saving sleep modes during idle periods. Zikos *et al* [264] model a cluster within a computational grid as an open queueing network, and evaluate the impact of resource al-

location strategies on performance and energy consumption. The authors model the heterogeneous nature of clusters, though they model jobs as being nonpreemptable (i.e. once they commence execution, they cannot be suspended or abandoned until completion) which is unlikely given the potential for resource failures, and particularly in our context of multi-use clusters where jobs may be preempted by interactive users. The scheduling approach considered by Zikos *et al* also differs from ours; in their model jobs may be queued on compute resource prior to execution, where under our model, jobs are only allocated to idle resources. The proposed resource allocation strategies consider queue length at each node and the performance of the nodes; energy consumption is considered but only as a secondary optimisation criteria in the event of multiple servers existing with empty queues and identical performance. Policies are evaluated by simulation for various levels of system load, and the authors acknowledge the trade-off between energy consumption and performance, and the significance of system load on the effectiveness of each resource allocation policy.

Faria *et al* [83] explore network and energy-aware resource allocation strategies for opportunistic grids. The authors extend the Workqueue (WQ) [62] scheduling strategy to consider network traffic, distance between input files and the execution node, as well as the current state of the execution node. Their proposed scheduling strategy is similar to our resource allocation policy targeting the most energy-efficient computers (referred to in Chapter 5 as **S2**), though rather than using the energy consumption of the execute node in the selection process, the full energy cost of transferring files to and from the execute node are considered. However, resources are considered to be heterogeneous in performance (and resultant execution time required to execute a given task), and in our scenario where bandwidth between nodes and the time to wake resources is considered to be uniform, these policies will be equivalent. The authors construct a testing environment comprising 30 workstations across three sub-networks, as the basis for their experimental evaluation. They further simulate this environment using GridSim [45] and GreenCloud [129] for three sample workloads with input file sizes of 10MB, 100MB and 1GB respectively. In the 10MB and 100MB cases their proposed strategy was shown to make little improvement compared to the HTCondor default

policy (referred to in Chapter 5 as **S1**), though in the third case with large 1GB input sizes, an improvement of 10.5% is observed as data transfer begins to dominate the cost of resource allocation.

Finally, a number of resource allocations applied in practice have been documented by the administrators of high-throughput computing systems. A detailed discussion of these approaches may be found in Chapter 5.2.

2.3.4 Reducing the number of miscreant tasks executions in a multi-use cluster

Here we discuss work directly related to Chapter 6 of this thesis. The issue of task failures is a general one, not specific to our own cluster. [200, 208, 260]. Lingrand *et al* [145] analyse logs of over 33 million jobs submitted to the EGEE European production grid environment between September 2005 and June 2007 and find 19% of jobs failed, with 35% not completing normal execution. The most common and default policy for handling task failures in an unreliable environment is resubmission.

Berten and Jeannot [31] performed a numerical analysis of resubmissions in a fault prone Grid environment. Their approach studies the effect of bounded and unbounded reallocation polices (equivalent to **X0** and **N1(*n*)** as outlined in Chapter 5). The authors consider *global* and *local* resubmission schemes; under the *global* scheme, failing jobs are reallocated to the main scheduler, while under the *local* scheme, failing jobs are resubmitted to the compute resource to which it was originally allocated. Throughout our work we consider task reallocations to the main scheduler (*global* rescheduling). However, energy consumption is not considered and tasks are assumed not to be faulty, where evidence exists across multiple classes of system to suggest software failures dominate hardware failures [148].

Hwang and Keselman [110] present an architecture in which extra tasks are run alongside the main task in order to more closely identify the state of the main task. This we see as complementary to our work and could be used to help aid 'good' and 'bad' task detection.

Haider *et al* [95] provide a literature review for the different fault tolerance mecha-

nisms provided by different distributed systems along with an argument for the need for such techniques.

Estimates of task execution times can be used as a criteria for selecting when to abandon a task. However, the use of estimates, provided by users at submission, have been widely criticised by the scheduling community for their inaccuracy [15, 16, 218]. Niu *et al* [181] analyse the traces of four large-scale systems from the Parallel Workloads Archive [1] and find only 17% of jobs completed within 90-110% of their estimated time.

Furthermore, many papers reporting the majority of task taking less than 30% of their requested allocation [55, 58, 251]. This may be due to tasks misconfiguration causing immediate termination [173] but is often due to wide variation in execution times [120] – especially if the cluster is heterogeneous – or since tasks are often terminated at the end of their estimated time interval users 'pad' their estimate to increase the chance of completing.

2.3.5 Energy efficient checkpointing

Here we discuss work directly related to Chapter 7 of this thesis. For a comprehensive survey of fault tolerance mechanisms and their applications in Grid and HPC environments, refer to the survey by Egwutuoha *et al* [74].

Checkpointing in real-time systems

Previous works in energy-aware checkpointing have primarily focused on real-time systems [166, 240, 261] subject to strict energy and deadline constraints.

Zhang *et al* [261] propose an adaptive checkpointing scheme to maximise the probability of satisfying a task's deadline in the presence of k faults, specified by a predefined fault tolerance requirement. Energy consumption is then introduced as a secondary optimisation criteria, with Dynamic Voltage Scaling (DVS) employed to maintain a processor in low power state, transitioning to higher frequency operating modes when required to satisfy a task's deadline.

Melhem *et al* [166] propose a similar approach, employing DVS in the absence of

failures to leverage *'slack'* time between a task's deadline and expected completion time, transitioning a processor into a less performant but more energy efficient operating state.

Unsal *et al* [240] evaluate the energy characteristics of an Application-Level Fault Tolerance (ALFT) scheme, where redundancy and recovery logic is incorporated at the application level, rather than being provided at the system or hardware level and propose a task scheduling heuristic reducing energy consumption by up to 40%.

In contrast, our scenario of a high-throughput computing environment is not subject to the same budgetary constraints as real-time systems. HTC systems tend to place an emphasis on overall system throughput rather than the completion time for individual tasks, instead adopting a best effort policy to execution completion, and often do not consider deadline constraints in during resource allocation. However, these approaches may be considered complementary to our own.

Checkpointing in HPC

More recently, research has sought to understand the overheads and energy implications of fault tolerance mechanisms, including checkpointing, in anticipation of exascale High-Performance Computing (HPC). Elnozahy *et al* [76] present a comprehensive survey of checkpointing and fault tolerance approaches in HPC systems. Bouguerra *et al* [39] investigate the impact of combined proactive and preventative checkpointing schemes in HPC systems, achieving up to a 30% increase in computational efficiency with negligible increase in overheads, but without consideration for its impact on energy consumption.

At exascale, increased frequency of faults are anticipated and energy consumption is a key issue [47]. To this end, Diouri *et al* explore the energy consumption impact of uncoordinated and coordinated checkpointing protocols on an MPI HPC workload [75], while Mills *et al* demonstrate energy savings by applying Dynamic Voltage and Frequency Scaling (DVFS) during checkpointing [168].

The potential performance impact of checkpointing is particularly great in large distributed-memory HPC systems. In these systems, all compute nodes are required

to quiesce while a snapshot of application state across all nodes is taken. Here the time taken to checkpoint, and thus the duration nodes must quiesce, is significant, with a number of works seeking to minimise this. Ferreira *et al* [87] propose one such approach, employing hash-based incremental checkpointing to reduce the overheads incurred by traditional coordinated checkpointing approaches.

Further works focus on energy and scalability issues relating to persisting checkpoint images to stable storage. Saito *et al* [201] consider energy saving when persisting checkpoint images, employing profile-based I/O optimisation to reduce the energy consumption of checkpointing to NAND flash memory by ~40-67%.

We consider the application of DVS [240, 261] and DVFS [168] to reduce the energy consumption of checkpoint operations to be complementary to our approaches.

Checkpointing in HTC systems

The application of checkpointing in High-Throughput Computing environments and Fine-Grained Cycle Sharing (FGCS) systems is explored extensively in [38, 182, 197], though without consideration for its implications for energy consumption.

Aupy *et al* [13] investigate energy-aware checkpointing strategies in the context of arbitrarily divisible tasks. While divisible tasks encompasses a number of common applications including BLAST sequencing and parallel video processing [255], such tasks represents only a proportion of our workload, and HTC systems do not typically have control over the division of batched tasks.

Chapter 3

Energy efficient content distribution with BitTorrent

Summary

In this chapter, energy efficiency considerations are investigated in a decentralised context, using BitTorrent. We provide mechanisms to facilitate energy efficiency and energy proportionality, and propose an energy-efficient content distribution system employing these mechanisms to minimise energy consumption and reduce cost. Our preliminary investigation highlights the challenges and issues in enacting energy saving operating policies in an environment where decision-making is decentralised.

3.1 Introduction

BitTorrent [61] is a peer-to-peer (P2P) file sharing protocol, accounting for approximately 17.9% [190] of overall Internet bandwidth use. Compared to traditional client-server approaches, BitTorrent relies less on the distributor's centralised infrastructure and bandwidth, offering a scalable content distribution solution with reduced provider-side power consumption and cost. This scalability makes BitTorrent particularly resilient to *flash crowds* [113], vast numbers of users accessing content simultaneously, a behaviour often observed for new and popular content. BitTorrent is employed not only in residential settings [106], but also within datacentres for the distribution of software updates [135] and in Infrastructure as a Service cloud computing environments [136].

In this chapter we investigate provider-side mechanisms to promote energy-efficient and energy-proportional operation of a BitTorrent based content distribution system. Our approach is complementary to the proxy scheme proposed in [5], and alleviates the need for centralised peer control imposed in [8] and [35]. We consider situations where such centralised control cannot be guaranteed, and present mecha-

nisms which do not require alterations to client logic. These relaxed conditions make our approach more broadly applicable as well as simplifying deployment.

3.2 BitTorrent

When a downloader (*peer*) initiates a download via BitTorrent, they first obtain a *torrent file*, a file containing metadata for the requested content. This metadata includes an endpoint to a BitTorrent tracker node. The *tracker* is essential to the operation of any BitTorrent system. The tracker maintains records of all peers uploading or downloading particular content (known collectively as the *swarm*), and coordinates content distribution and enables peer discovery. The tracker component must remain online at all times in order for newly arriving peers to be able to connect.

Once the peer has established a connection with the tracker, the tracker responds with a *peer list* containing the details of a random subset of the other peers transferring the requested content. The peer may then connect to, and obtain content from, these peers. Additionally, the peer may elect to obtain up-to-date peer lists from the tracker periodically according to an *announce interval* specified by the tracker.

Files in BitTorrent are split into multiple *pieces*, allowing peers to share pieces of the file they hold while obtaining the pieces they require. BitTorrent peers' ability to download and upload simultaneously benefits performance and makes BitTorrent significantly more scalable than client-server file distribution approaches.

BitTorrent peers may belong to one of two states; *leeching* or *seeding*. Peers actively downloading in the system but who do not currently hold a full copy of the file are referred to as *leechers*. Once a peer has obtained all the pieces of their download, they may either depart from the system or remain active as a *seed*. Seeds remain active participants in the system, altruistically sharing upload bandwidth to distribute content to other peers.

3.3 System Models and Objectives

In our model we represent peer power consumption as *nameplate* power consumption figures, values specified by the manufacturer of the computer hardware. Selecting readily available power consumption values provides sufficient accuracy for our system to make valuable energy savings while minimising the overhead associated with collecting the information. We also maintain details of the download and upload capacity of individual peers. These may be bandwidth figures obtained out of band or taken from real-time observations of the running system.

We model a *seed pool* as a group of servers under centralised control, heterogeneous in terms of power consumption and upload capacity. The upload capacity of these servers is assumed to be considerably greater than that of typical peers. Membership is assumed to be dynamic, with servers arriving to and departing from the pool periodically. Where members of the seed pool may be considered internal architecture across one or more data centre facilities, we may assume physical access for detailed in-situ power profiling. Multiple linear regression models calibrated for each resource will provide accurate estimates based on real-time resource utilisation measurements, including CPU, memory and disk activity. Software agents instrumenting each machine communicate this utilisation data to the tracker.

Our model considers tracker and seed instances to belong to one of two distinct states; *sleep* or *active*. An active resource is fully powered up and is able to execute operations and serve requests from the system. A resource may be placed in a sleep state, where the machine is no longer able to serve requests and consumes significantly less power. While asleep, system state is stored in memory allowing the machine to transition into an active state quickly. We model the time taken to transition between these two states, during which the resources consume power but are unable to contribute to the system.

Content distribution networks are typically large shared infrastructures, distributed across multiple data centre facilities nationally or globally. Hence, it is imperative that our system model adequately represents the differences between data centre

facilities and global variation in the cost and cleanliness of their power sources. Facility modeling includes the Power Usage Effectiveness (PUE) [21] rating, a metric representing the proportion of facility overheads (for example, power, cooling and lighting infrastructure) in terms of the power consumption of the IT equipment. We account for variations in the price and ecological impact of energy supply in our model, representing these in pence and kg CO_2 per kWh respectively.

We consider modeling of network devices outside the data centre facility as beyond of the scope for this research. Peer-to-peer approaches have greater total bandwidth requirements than client-server approaches due to peers communicating with one another. The impact of this communication overhead on power consumption is difficult to assess. Despite significant recent improvements in energy-efficiency of hardware [199], typical network hardware is found to be energy-disproportional [151]. This power characteristic results in a narrow dynamic power range, limiting the potential impact of variable traffic workload on power consumption. Furthermore, these network devices must remain online at all times and are outside of the administrative control of content providers. Existing research has compared client-server and peer-to-peer approaches, finding peer-to-peer to demonstrate greater network-related power consumption but lower overall power consumption in a communication-intensive scenario such as file distribution [177].

It is unrealistic for an organisation to minimise its power consumption without first considering the trade-offs between energy efficiency, cost and reliability. In an inter-organisational scenario such as software patch distribution in an office environment or large-scale deployment across a cluster [135], stakeholders of the system will most likely be concerned with minimising the aggregate energy consumption and cost of a system. Conversely, in situations where peers are external to the organisation (e.g. video on demand or public content distribution), stakeholders are likely to prioritise provider-side energy efficiency and cost over those of the peers. Our approach must remain flexible in order to satisfy the various optimisation goals of the stakeholder.

3.4 Approach

In this section we outline three key approaches with potential to reduce the energy consumption of BitTorrent systems. The impact of these approaches should be both equitable and proportional, such that energy-efficient peers are not penalised excessively in terms of download performance, and be beneficial to the swarm as a whole. We do not currently envisage the download performance offered to the peers would serve as a sufficiently significant driver to motivate energy-inefficient machine procurement. Decisions made by the system are informed by comprehensive measures of system performance collected by the tracker, and are subject to the optimisation goals of the policy currently being enforced by the service provider, and the current state of the system.

3.4.1 Energy Proportional Tracker Migration

Energy Proportional Tracker Migration leverages heterogeneous hardware to promote energy proportionality of the tracker component. During periods of low utilisation the tracker will reside on a computationally constrained but energy-efficient machine, autonomically migrating to a more performant (but more costly in terms of power) server during periods of increased load. This aims to minimise the load-independent component of our system's overall power consumption and achieve near energy proportional operation. Our approach differs from those in the literature by explicitly considering the characteristics of the BitTorrent workload.

Existing research has demonstrated the ability to compose a number of non energy-proportional servers, combining power saving mechanisms to deliver an energy-proportional aggregate system [234] [132]. We acknowledge the heterogeneous nature of typical real-world data centres (often caused by machine failures, and upgrades, etc) [100] and contribute mechanisms which specifically leverage hardware heterogeneity to achieve aggregate energy proportionality.

3.4.2 Elastic Capacity Provisioning

In *Elastic Capacity Provisioning*, we propose a variation of typical BitTorrent use, whereby a content distributor operates a pool of specialised seeds. It is the role of these seeds to share content to other peers, ensuring satisfactory levels of performance, energy consumption and cost. Instances are provisioned dynamically in response to real-time service demand. We consider the heterogeneous nature of this pool of specialised seeds when periodically recalculating and provisioning the minimum active set of seed resources to achieve desired performance, cost and energy optimisations.

Traditionally, BitTorrent seeds operate according to a strategy where seeder upload capacity is allocated proportionally to those peers with higher download rates, optimistic that those peers may themselves become seeds more quickly and serve other peers. We propose a scheme whereby upload bandwidth is allocated on a combination of observed download rates and peer energy efficiency. Peers who are particularly energy-inefficient relative to the rest of the swarm will be provided with a larger proportion of the seeder's upload capacity, enabling these peers to complete their download and depart from the system more quickly, reducing their power consumption. In situations where upload capacity is limited among members of the swarm, and such actions threaten the overall health of the swarm, the traditional strategy is observed to prevent starvation.

3.4.3 Peer Connectivity Shaping

Peer Connectivity Shaping augments the peer lists returned by the tracker, giving some peers preferential treatment by providing them with the details of a larger peer set, or of peers with greater available upload bandwidth. This aims to promote greater connectivity between the peer and the swarm, lowering the peer's download time and consequently reduces its energy consumption.

Once a peer list has been received, a client typically selects a random subset of peers with which to connect to in the first instance. Peers are unaware of the upload capacity of the peers when they select which peers to connect to, so it is important when a peer requests its initial peer list that the list comprises a smaller proportion

of peers with slow upload rates. Subsequent peer lists may include a wider range of peer upload capabilities, as BitTorrent's "tit-for-tat" mechanism will favour peers with higher upload rates and ensure the peer receives fair download rates. In the case of a particularly energy-inefficient peer, it may be more beneficial to provide small peer lists to increase download performance at the expense of increasing the peer's connectivity with the swarm.

The interval between a peer's requests to the tracker may also be optimised to improve performance and lower energy consumption and cost. In highly dynamic systems where peers and seeds are arriving and departing frequently, it may be preferable to lower the interval between peer requests in order for them to remain responsive to the changing state of the system. Increased requests to the tracker will place the tracker under greater load so there exists a trade-off between increasing performance for peers without incurring greater power consumption on the tracker.

3.5 Experimentation

To evaluate the efficacy of our approach we have developed a simulation environment based on TorrentSim [17]. TorrentSim was chosen as the basis for our work due to ease of extension and operating system independence. In order to evaluate our proposed approaches, we extend the underlying simulation framework in a number of key areas.

Dynamic provisioning of nodes throughout execution The TorrentSim simulation environment, and the underlying Simmcast [18] framework, support only static provisioning of nodes. The number and capability of nodes must be specified in configuration files prior to the start of a simulation run, and they are fixed throughout the simulation's execution. We obtain the source code for Simmcast and extend its API to expose the operations required to add and remove nodes while the simulation is running, and make the requistite changes to the TorrentSim to allow peers to be provisioned and de-provisioned dynamically throughout the execution of the simulation.

We employ the Moving Window Average (MWA) approach to dynamic provisioning presented in [132], with a window size n of 5 minutes.

Pseudocode for the algorithm responsible for enacting provisioning decisions [132] is presented below.

```
for server_type in server_types
    start_time = server_type.startup_time
    # Predict and start servers
    pred = predict_load (now + start_time)
    clust = make_cluster(pred)
            - current_cluster
            - nodes waking up in time
    start server_type servers in clust
```

The pseudocode for the algorithm responsible for de-provisioning servers [132] is presented below.

```
for server_type in rev(server_types):
    start_time = server_type.startup_time
    max_clust = empty_cluster
    for t in range(1, start_time)
        pred = predict_load(now + t)
        temp_clust = make_cluster(pred)
        max_clust = max (max_clust, temp_clust)
    temp_clust = current_cluster - max_clust
    turn off servers in temp_clust
```

Our extensions to the TorrentSim framework have been designed such that they are as modular as possible as possible, allowing alternative scaling algorithms to be plugged easily.

Power consumption modelling We represent peer power consumption as manufacturer specified *nameplate* power consumption figures, as explained in Section 3.3. As shown in the state transition diagram in Figure 4.2, resources are modelled as being in one of three states based on the Advanced Configuration and Power Interface Specification (ACPI) specification [103], an open standard describing device configuration power management mechanisms for the operating systems. The states we consider are as follows:

- **Active**: actively participating in the BitTorrent system, either as a leech or seed. This equates to ACPI state S0.
- **Sleep**: computer state stored in RAM which remains powered. All other components are powered down. This allows for quick system resume without the need to restart the operating system. ACPI state S3.

These values are used in conjunction with the ammount of time nodes spend in each operating state, to calculate total energy consumption. Total energy con-

sumption is calculated as follows:

$$\sum_{c=0}^{n} \sum_{p=0}^{m} t_{c,p} E_{c,p} \tag{3.1}$$

where n is the number of peers, m is the number of power states, $t_{c,p}$ is the time spent by peer c in state p (as in Figure 3.1) and $E_{c,p}$ is the energy consumed by peer c in state p.

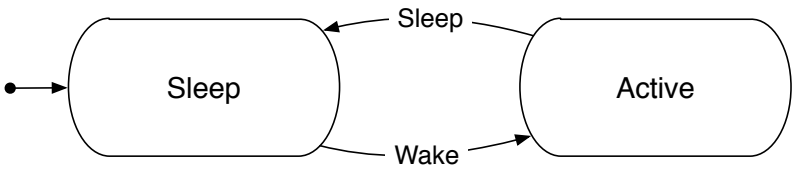

Fig. 3.1: State transition diagram for a compute resource

The representations of the tracker and peers - both seeds or leeches - were updated to include power consumptino values for each of these operating states.

Correct calculation of performance metrics for long-running simulations In many areas of the TorrentSim [17] simulation environment, the `double` Java type was frequently used to hold statistical information related to performance characteristics of the simulated environment. We observe for long-running simulations that values being stored would exceed the maximum value for this datatype of $(2-2^{-52}) \cdot 2^{1023}$, hence were susceptible to overflow. We address this issue by making use of `BigDecimal` to store statistical information.

Augmenting the BitTorrent tracker component We augment the BitTorrent tracker component by providing a finer granularity of performance metrics. As standard, the BitTorrent tracker does not record or expose statistics on the number of (and type of) requests it has received from peers in a given time period. We extend the BitTorrent tracker such that this information is readily available programmatically, and also written to a log file to allow for additional analysis once the simulation run has completed.

Performance and power measurement collection We extend the simulation to add a *Monitor* component which runs periodically (by default, once per second) and records a variety of performance and power measurement figures for offline analysis. Information collected by the monitor includes the number and type of peers active within the system, and their average power consumption and bandwidth utilisation for each peer.

3.6 Results

In the evaluation of the Energy Proportional Tracker Migration approach we consider two normalised tracker workload traces shown in Figures 3.2 and 3.3. Workload traces were obtained through the execution of our simulation environment, in the presence of our extended performance measurement collection described in Section 3.5. These workloads were selected as we found them to be representative examples of typical BitTorrent operation.

Workload traces WL_1 and WL_2 represent tracker requests during the arrival and service of 100 and 200 peers respectively. While our simulation envrionment is capable of simulating larger numbers of peers, we selected 100 and 200 based on indicative figures from the literature of the scale of typical usage of BitTorrent in a datacentre environment [92]. In each case three seeds are active in the system, and all peers depart from the system upon completing their download.

Workload WL_1 is characterised by larger peer inter-arrival times and greater availability, resulting in smaller mean peer service time. Conversely, in WL_2 peer inter-arrival times are much smaller and peer download rates are constrained by limited availability and greater competition for available upload capacity. The request rate at a given period is largely dependant on the number of peers and seeds active in the system. Observed increases in request rate over time indicate the arrival of new peers, while decreases signify peers' completion and subsequent departure from the system.

The efficacy of our provisioning approach is evaluated for two groups of servers. The first group is homogeneous in terms of both performance and power consumption, while the second comprises servers from two heterogeneous classes of server. Table 3.1 outlines the performance and power characteristics of the classes of server we consider in this work.

Type	Performance (ops/sec)	Wake time (seconds)	Power Consumption	
			Active	*Sleep*
Low Performance	200	30	60W	2W
High Performance	500	120	180W	5W

Table 3.1: Computer Types

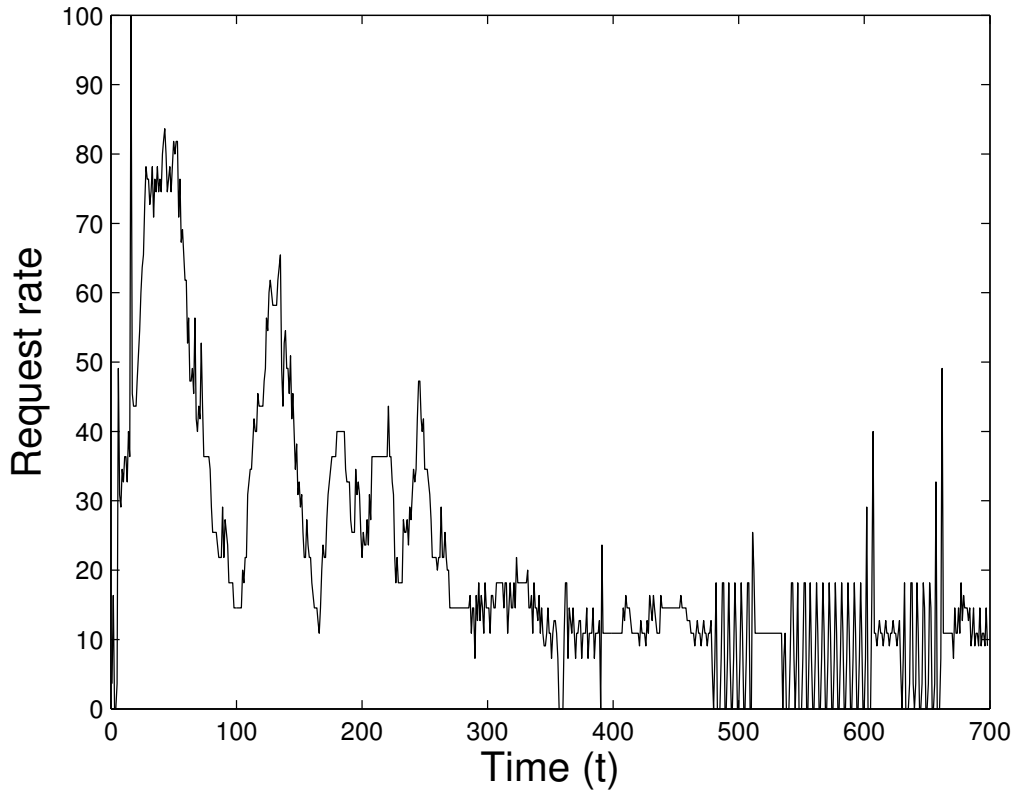

Fig. 3.2: BitTorrent tracker workload trace WL_1.

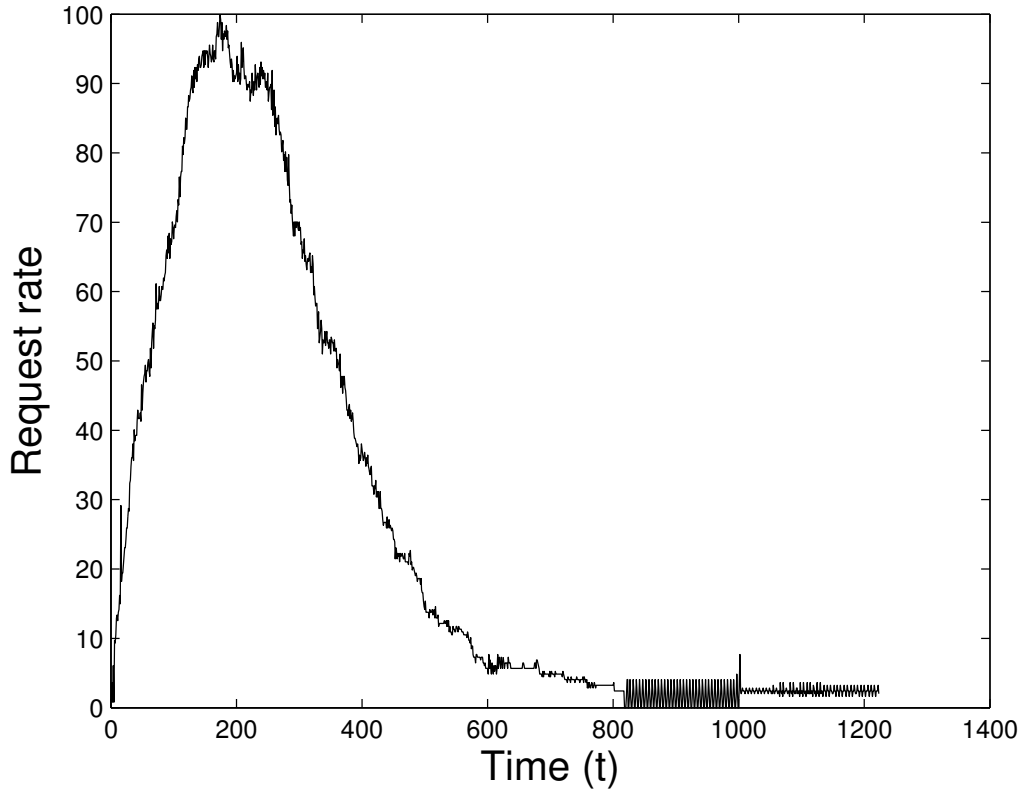

Fig. 3.3: BitTorrent tracker workload trace WL_2.

In Figure 3.4 we present relative energy savings for our approach when compared to a group of servers right-sized to satisfy the peak request rate observed over the duration of the traces. In each case we find increasing the number of servers is beneficial in reducing energy consumption, allowing for finer grained provisioning of resources to satisfy the offered workload.

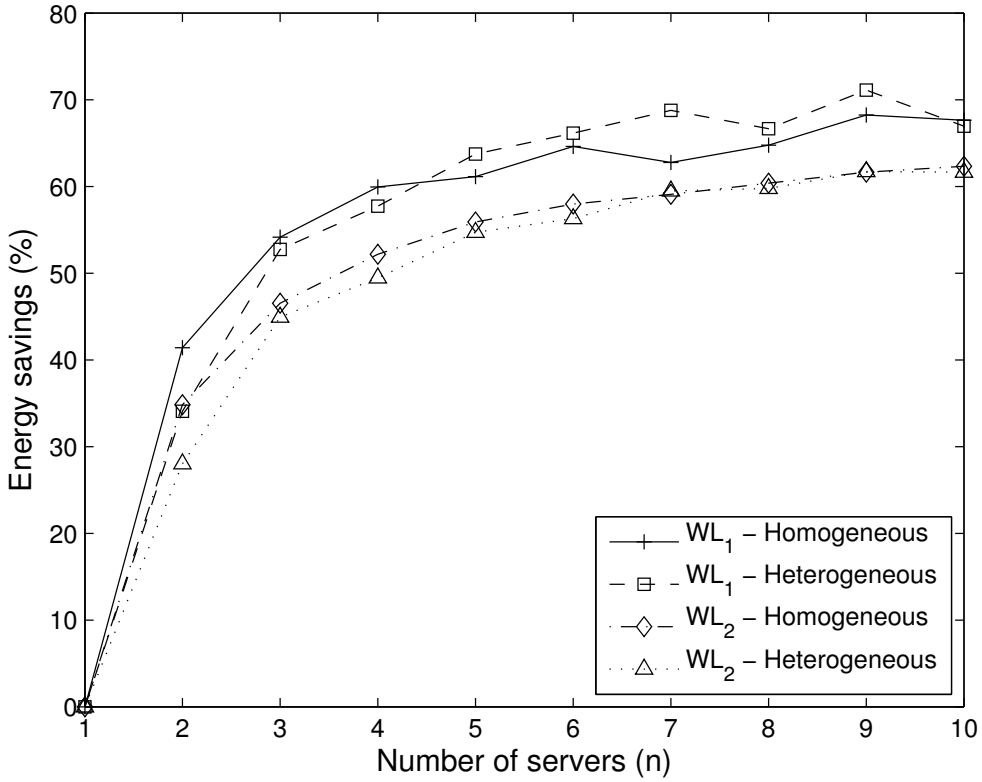

Fig. 3.4: Comparison of energy savings for two workload traces with homogeneous and heterogeneous groups of servers of size n.

Our results for Energy Proportional Tracker Migration demonstrate the potential for considerable energy savings and reduction in the load-independent portion of aggregate power consumption, compared to a system provisioned to meet peak demand. However, due to variability introduced by distributed decision making in BitTorrent, the Peer Connectivity Shaping approach was found to have little impact on the energy consumption of the system.

3.7 Conclusions and Further Work

In this chapter we have presented a preliminary investigation into energy saving policies within a BitTorrent system. From this preliminary investigation we learn a number of key lessons in terms of the context of this thesis. The lack of control observed in environments where decision-making is decentralised severely limits our ability to enact energy-efficient policies effectively. This is exacerbated by issues of trust; a fundamental issue with decentralised systems is the need to trust the energy consumption reported by the peers, but where there is an inherent incentive to be dishonest in order to receive better levels of service, and no simple mechanism to verify such claims, there exists the opportunity for peers to provide a pessimistic view of their own energy consumption in exchange for preferential download performance.

Though we find the application of energy saving schemes within public BitTorrent networks to be limited, we envisage scope for further work within organisational setting. Gadea *et al* [135] consider the use of BitTorrent for distributing codebases among datacentres of thousands of servers at Twitter. In this scenario, where explicit control over the provisioning of servers may be assumed, more convincing results may be sought.

Within the remainder of this thesis we will consider more centralised systems, where we are able to control each of the individual units contributing to the overall energy profile of the system.

3.7.1 Further Work

In the area of energy-efficiency of peer-to-peer networks there are a number of interesting areas of further work to be explored, though these are not explored in this thesis.

Federation This paper considers the use of BitTorrent as a content distribution mechanism in a single management domain. An interesting area of future research is to extend our approach to facilitate energy-efficient use of BitTorrent in a federated network of interconnected content distribution networks. Such a federated approach would allow organisations to share resources, further reducing the

need to over-provision to meet peak demand. A common challenge in peer-to-peer systems is accountability [238]. Service Level Agreements (SLAs) between these organisations may be enforced on a combination of utility, cost and energy efficiency. Audit and accountability information may be used to facilitate billing for service between organisations. Of particular interest is the ability to reconcile the conflicting optimisation goals of multiple service providers on shared infrastructure, and energy-aware incentive mechanisms in a federated context.

Internet-based media streaming BitTorrent and other peer-to-peer (P2P) solutions are increasingly being considered to handle the enormous bandwidth requirements in the context of on-demand media streaming [147]. This context differs from ours, with more stringent requirements on end-user Quality of Experience (QoE). A number of works have explored the feasibility of a BitTorrent-like system for this purpose [63, 248], but without consideration for the impact on energy consumption.

Chapter 4

Trace-driven simulation for energy consumption in High Throughput Computing systems

Summary

This chapter presents the approach we adopt to trace-driven simulation for energy consumption in high-throughput computing systems employed in subsequent chapters. We present a generalised model of resources and jobs within an HTC environment, detail our power modelling approach, and apply this to our case study of the Newcastle University HTCondor pool.

We will present details of HTC-Sim for simulating High Throughput Computing systems comprising both dedicated resources and resources shared with interactive users. We shall present the core model of the simulation along with discussion of the trace logs required and the methods needed to produce such logs. Though we focus on the modelling of our HTCondor system, our simulation base and system model is easily generalisable to other HTC systems.

We evaluate the impact of running the simulation software both in terms of memory footprint and execution time and show, through the use of synthetic trace logs that the simulation software scales linearly in both memory and execution time as the number of jobs to simulate increases.

4.1 Introduction

Modern computational power allows researchers to perform work hitherto unimaginable. This is often achieved through the processing of vast quantities of data (Big Data), performing large scale simulations or ensembles of smaller simulations. However, our desire to solve such problems has now far-outstripped the computational power of a single computer. Parallel computing, where multiple processing units are employed in solving a single piece of work, is a common solution to such problems. Where this

work may be sub-divided into separate jobs that can be run independently of each other we refer to this as an *'embarrassingly parallel'* or *'pleasingly parallel'* problem and solve it using High Throughput Computing (HTC). Many HTC systems exist, such as HTCondor [146] and BOINC [6], with these systems being used to help solve research problems from small scale up to grand research challenges.

Traditionally HTC systems were provisioned as either dedicated resources or as a shared facility (often referred to as a Desktop Grid) with resources powered up all the time either servicing jobs or sitting idle. The performability and reliability of such systems is generally well understood [117]. With IT operations facing increased scrutiny for their energy consumption and a strong desire to reduce the impact of these systems, HTC systems would appear to be a prime candidate for such savings.

Aggressive power management policies, over the resources which constitute an HTC system, are often proposed, though these policies could have significant impact not only on the energy consumption but also performance, reliability and availability of resources for HTC users. Placing idle resources into a sleep state too rapidly could lead to HTC resource starvation, while weak policies may offer little energy savings.

It is therefore highly desirable to determine the 'best' set of energy conservation policies which can be applied to both the HTC system and the underlying hardware. Controlling such factors as when idle resources are sent to sleep, when to wake up resources, the selection of the resource to use in order to minimise energy consumption or how to deal with jobs which fail to complete. This is particularly important for Desktop Grids, with priority for interactive users, as job eviction does not imply that the job cannot complete with more time on a different resource.

One solution to determining an optimal policy set is to test policy changes on the live system. This has three significant drawbacks: running the system under the new policy for a significant amount of time to ensure statistical relevance; detailed logging to determine energy consumption and monitoring of the high-throughput architecture is required; and a danger that changes could have unpredicted (negative) consequences. This leads to making minor modifications to the policy set where we are confident that the impact on users will be low; significant changes being considered

too dangerous.

Two alternatives exist: a test environment or a simulation. Test environments remove the need for site-wide monitoring and do not affect the production system, however time is required to evaluate changes and we need to justify how results would map to the whole system. Instead we present here HTC-Sim, a Java based trace-driven simulation we have been developing as part of our work in energy saving for HTC systems. The simulation system allows for the quantifiable, and quick, evaluation of different policies against the same workload and interactive user patterns.

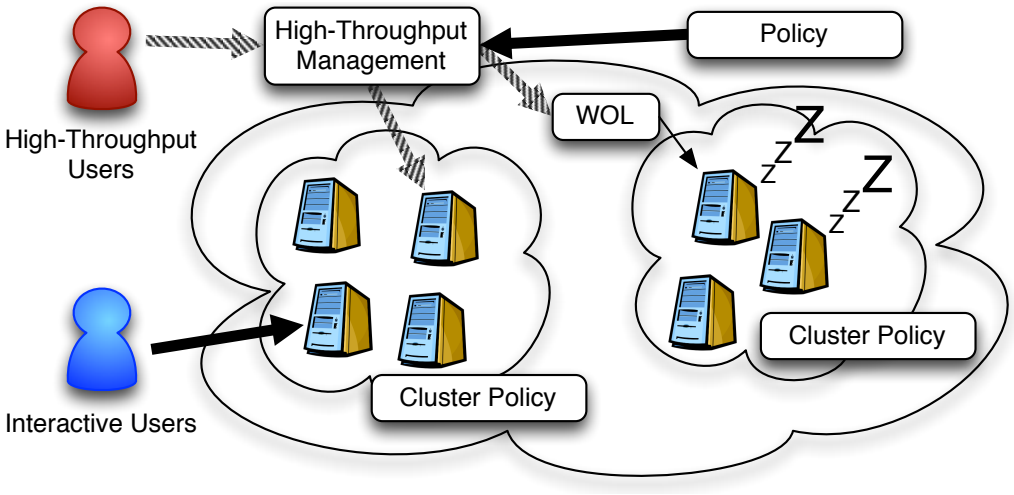

Fig. 4.1: Model of an HTC system and multi-use environment

The simulation system allows the modelling of energy consumption and performance characteristics of the HTC system. Thus it can be seen as a powerful tool for administrators to evaluate new policies as well as the impact of changes to the infrastructure itself.

The overall model for HTC-Sim is shown in Figure 4.1, where two types of user can interact with the system – HTC users and interactive users. These are handled through historical trace logs for both user types, allowing us to replay system behaviour for the period over which the traces were collected. Interactive user trace logs contain the login and logout time along with the resource used – it is assumed that this is a fixed interaction. However, for the HTC workload only the job submission time and the execution time are considered – the execution start time and resource used may change due to the active policy set. Resources within the system are grouped into *clusters*, each representing a set of homogeneous resources under the same policy set. In this way we

can model both sets of resources purchased together or resources co-located and acting under identical policies. The HTC system has its own policy set. We consider the resources within our system to feature a Wake-on-LAN (WOL) capability such that the system may power up these resources on demand.

The remainder of this chapter is structured as follows. In Section 4.2 we provide details of the simulation model and the policy decisions of high-throughput systems we plan to model in our system. We present a case study of using our simulation with the trace logs obtained from our HTCondor cluster in Section 4.3. Performance evaluation of the HTC-Sim is presented in Section 4.4, evaluating the performance of our simulation framework in terms of execution time and memory consumption. Conclusions and future plans for extending HTC-Sim are presented in Section 4.5.

4.2 System Model

We introduce our generic model of the entities and resources within a HTC system along with our metrics for user impact, energy, cost and environmental implication.

4.2.1 Compute resources

We model compute resources as being either dedicated - whether local or cloud infrastructure - or multi-use cluster machines shared with interactive users. We model a number of characteristics for machines, namely architecture type, operating system, performance measures (e.g. CPU speed, number of cores, memory) and energy profile. We further allow users of the simulation to specify custom attributes for machines which are specific to the environment which they are modelling.

We adopt the SPECpower [219] model for energy consumption within a system, as discussed in Chapter 2.1.1. Here discrete values for CPU load are equated with specific energy consumption levels. This allows the energy consumption of a resource to be derived from the current CPU load, if known. As shown in the state transition diagram in Figure 4.2, resources are modelled as being in one of three states based on the Advanced Configuration and Power Interface Specification (ACPI) specification [103], an open standard describing device configuration power management mechanisms for

the operating systems. The states we consider are as follows:

- **Active**: in use either by an interactive user or a high-throughput job. This equates to ACPI state S0. We consider resources to belong to the active state if they are in *'User'*, *'HTC'* or *'HTC+User'* states in Figure 4.2. If CPU load is known then energy consumption can be derived from this figure.
- **Idle**: powered up but not actively processing work for interactive user or high-throughput job, with lower energy consumption than in active state. Also S0. Equates to approximately 5-10% CPU load.
- **Sleep**: computer state stored in RAM which remains powered. All other components are powered down. This allows for quick system resume without the need to restart the operating system. ACPI state S3. The CPU is inactive consuming only a base level of energy.

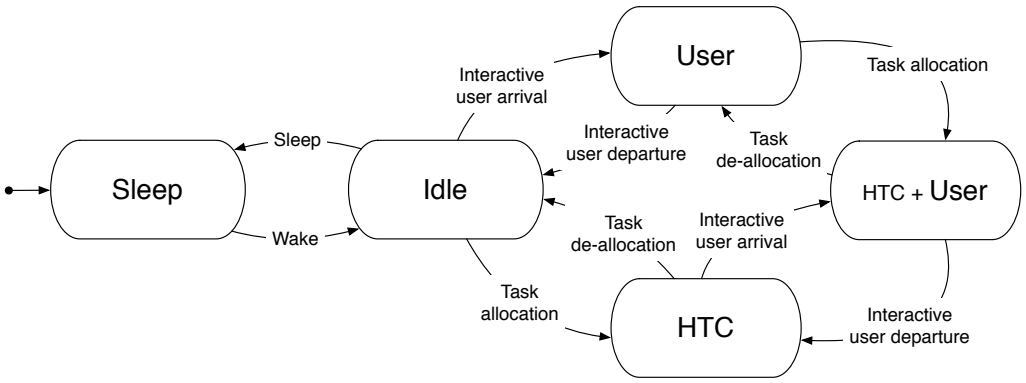

Fig. 4.2: State transition diagram for a compute resource

4.2.2 Interactive user sessions

We model interactive user sessions as a tuple $\langle s_i, c, u, e_i \rangle$ where s_i and e_i are the login and logout timestamps respectively, c is the name of the computer, and u is a hash of the interactive users identity. Hashing of the user identifier provides anonymity to the user, while allowing us to correlate multiple sessions from a particular user.

4.2.3 Cluster

We group resources into 'clusters' defined as a homogeneous group of machines[1], whose specifications are identical, provisioned at the same time, co-located in the same physical space, and governed by the same operating policies. The Power Usage Effectiveness (PUE) [21] of the cluster can also be taken into account here. We model the changing behaviour of cluster machines over time. Factors include: times of scheduled reboots, whether HTC jobs are currently permitted, whether machines are currently available for use by interactive users, whether HTC jobs are allowed to run on a machine currently occupied by an interactive user, how long must a resource remain idle before transitioning into a low power state, and how long after a resource enters the idle state does it become available to run HTC jobs.

We are further able to model *'special'* events through the course of the simulation where the policies enacted on the cluster may vary. Examples of this include clusters being closed for upgrades, different policies for different days of the week, or bank holidays.

4.2.4 HTC Job

The HTC workload comprises of jobs which may be part of a batch. We define a job by the tuple $\langle j, b, q, d, h, e, f, u, d \rangle$, where:

- j is the identifier of a job (or batch of jobs)
- b is the identifier of a job within a batch (if present)
- q is the time the job was submitted into the system
- d was the job duration observed in the original system
- h is the hash of the user who submitted the job

[1] Though we acknowledge the heterogeneous nature of real-world clusters [100] (often a consequence of manufacturing variation, machine failures and upgrades, etc), the variation in energy profile of machines with a group are smaller than the variation between classes of machine (see Table 4.1) and a homogeneous cluster model has been shown to be sufficient for our work. A cluster exhibiting system significant variation between machines may easily be modeled as two distinct clusters within our framework.

- e is the HTC result state of running the job (either 'success' or 'terminated')
- if a job was terminated (result state e equals 'terminated') then f represents the time that the job termination was submitted.
- u, d represent the data transfer to and from the resource which ran the job.

Although most HTC systems can provide much more information on the jobs which were run these are the core elements currently used within the simulation.

Each job will transition through a number of states as depicted in Figure 4.3. Jobs arriving into the system will be initially queued, though if possible they will be allocated immediately to a resource and enter the running state. In the ideal case the running job will finish without any further state transitions. However, if an interactive user takes possession of a resource then the job will enter a suspend state where execution is temporarily suspended – in the hope that the user will leave soon afterward – after which the job can resume running. If the suspension time becomes too great then the job will be evicted back to the queue. If checkpointing is being used then jobs will be checkpointed at intervals defined by policy (for a full discussion of checkpointing, see Chapter 7). Jobs may be terminated at any time in which case they end up in the final 'Job Removed' state.

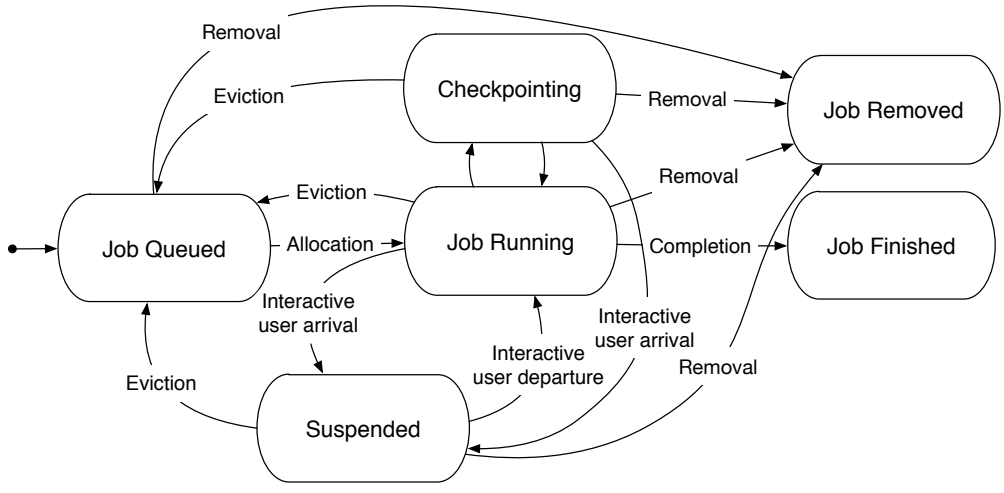

Fig. 4.3: State transition diagram for a job within an HTC system

4.2.5 Policy decisions - HTC

A number of common policy decisions exist within HTC systems. We discuss those which have already been built into our simulation model here:

Resource allocation: Given a set of available resources, the HTC system must select the most appropriate resources to optimise the required metrics. These may include random allocation, lowest energy consumption, least chance of eviction, or fastest resource. Further discussion of these policies can be found in [159, 161] and Chapter 5 of this thesis.

Job resubmission: In an system where jobs can be evicted through activities outside of its own control (reboots and interactive users) the decision of whether to try and re-submit a job which has previously been evicted needs to be made. This is non-trivial as a job which has been evicted many times may indicate that it is 'broken' and will never complete or might just indicate that the job has been unfortunate in its previous allocation to resources [163]. Job resubmission strategies to promote the successful completion of good tasks within HTC systems are discussed in detail in Chapter 6 of this thesis.

Suspension: Suspending jobs offers great potential for 'saving' the effort already exerted on a job. However, if the suspension timeout is too short then this benefit can be lost, whilst if the timeout is too long then a significant penalty is imposed on the time a job takes to complete [161]. Suspension policies in HTC systems is discussed in Chapter 5 of this thesis.

Reboots (deferral): Many Desktop Grid installations have nightly reboot policies. Given that the best time for running HTC workloads tends to be at night the ability for HTC jobs to defer these reboots can significantly improve the chance of nightly jobs to completing [161]. The impact of machine reboots and their potential deferral are discussed in Chapter 5 of this thesis.

Checkpointing: Checkpointing can save both time and energy by allowing jobs which are evicted to resume from the last checkpoint. However, as the process of check-

pointing consumes both time and energy a careful balance is required to minimise energy consumption [89, 91]. The impact of checkpointing in HTC systems is discussed in Chapter 7 of this thesis.

4.2.6 Policy decisions - Infrastructure

A number of policy decisions can be made for the underlying infrastructure [161], these include:

Time before HTC usage: Once a computer becomes idle it is a potential target for HTC work. However, in busy clusters a logout could be quickly followed by a login causing a job eviction. Therefore allowing some time between user logouts and HTC use is desirable [161] (see Section 5).

Time to sleep: Energy is saved by sending resources to sleep as soon as they become idle. However, this increases the time for HTC jobs as the resources need to return from the sleeping state first. This is exacerbated if resources are required to be idle for a set amount of time before they can be used for HTC work.

Can HTC wake up computers: If the HTC system can not wake up resources then this can lead to resource starvation once the resources have gone to sleep. Likewise if they can wake up computers then this leads to potentially more energy usage.

Allow HTC usage: At busy times of day it may be desirable to disable HTC workload on specific clusters.

4.2.7 Metrics

When evaluating proposed policies, a number of metrics are of particular interest, providing insight into the performance, energy consumption and cost of operation. Below we outline the range of metrics currently supported by HTC-Sim.

Performance overhead: is measured as mean average job overhead - defined as the time difference between the job entering and departing the system, and the actual job execution time (d in our job tuple in Section 4.2.4). Overheads may in-

clude suspension, checkpointing or delays incurred while awaiting resource allocation.

Energy consumption: reporting fine grained energy consumption results, at per-computer, cluster and system levels. Providing a breakdown of energy consumption for each state, e.g. sleep, idle, HTC and/or interactive user. Total energy consumption is calculated as follows:

$$\sum_{c=0}^{n} \sum_{p=0}^{m} t_{c,p} E_{c,p} \tag{4.1}$$

where n is the number of computers, m is the number of power states, $t_{c,p}$ is the time spent by computer c in state p (as in Figure 4.2) and $E_{c,p}$ is the energy consumed by computer c in state p.

In the case of resources based in data centres / machine rooms, we utilise the Power Usage Effectiveness (PUE) [21] value for the environment, describing the ratio of power consumed by compute resources to the power consumed by the cooling and lighting infrastructure to support the resources. It is important to note that PUE values may not legitimately be applied to desktop machines based on users' clusters due to the multi-use nature of the environment in which the machines reside, and variations introduced by user occupancy.

Good jobs terminated: Policies governing the resubmission of evicted jobs may lead to good jobs being terminated.

Data transfer: Data transfer is often a significant overhead. This is particularly evident for jobs with large datasets, or when using checkpointing. The simulation models the bandwidth available between nodes, imposing time delays on data ingress/egress. Estimated data transfer delays may then be used to inform resource allocation and other decisions. The *iperf* [216] bandwidth measurement tool was used to ascertain the peak bandwidth available between cluster machines and an average value of 94.75 MBits/s used as an approximate in our simulation.

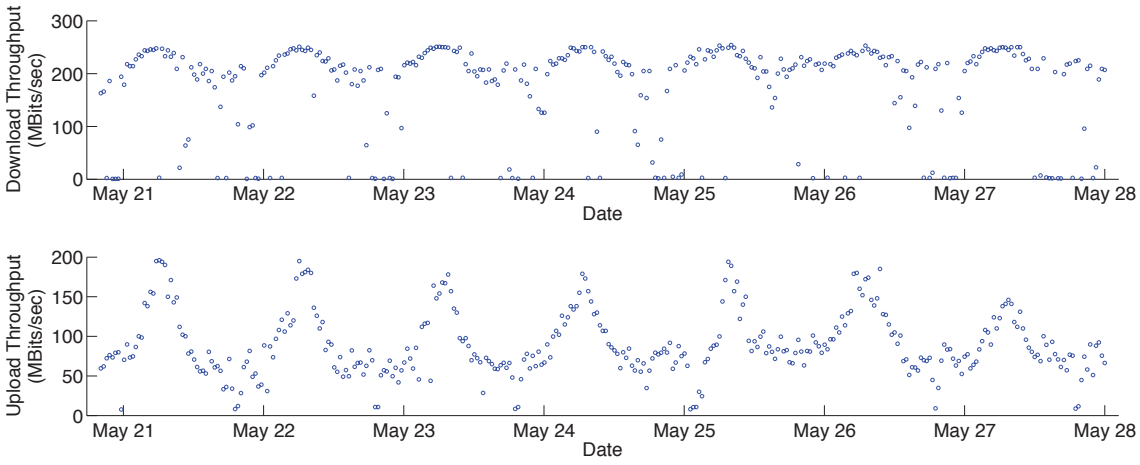

Fig. 4.4: Bandwidth measurements for EC2 upload and download throughput

In [162] we further evaluate the data transfer overheads of file transfers to cloud instances based in the Amazon EC2 (US East Northern Virginia Region) datacentre. The results of this experimentation is shown in Figure 4.4, capturing the network bandwidth every thirty minutes between Monday May 20th 2013 and Tuesday May 28th 2013 based on the GMT time zone. There is a clear day and night pattern to this data, although there are a number of outlying points. Bandwidth potentials appear to be greatest during the early hours of each day (GMT) with the upload speeds showing the greatest variation. A full analysis and modelling of this variation in bandwidth forms the basis for further investigation; in our works we take average bandwidth values from our test period for our simulations, those being an upload speed of 90.08 MBits/s (11,811 bytes/ms) and download speed of 174.88 MBits/s (22,925 bytes/ms). It should be noted that the largest data transfer observed in the data set was 903 MB with our transfer experiments running for five minutes reaching up to 9.4 GB of data transfer. It should also be noted that these are maximum bandwidth potentials for the connections; real use is likely to be less, thus these give a lower estimate on data transfer times.

Cluster utilisation and throughput: Policies such as fault tolerance and replication have the potential for significant impact on throughput and overall cluster utilisation. We report utilisation both in terms of the HTC workload in isolation, and also including interactive user load, and report measures of average and peak

throughput.

Cost and environmental impact: It is insufficient to evaluate energy consumption and performance of policies without also considering their implications for cost. We model electricity cost per kWh, and a carbon emissions charge for each kilogram of CO_2 produced from energy [71] (currently £16 per metric tonne in the UK). These figures may be specified at a system- or cluster-specific level to reflect the costs associated with the users' infrastructure, and any cost differences in federated and cloud contexts. Thus, the total operating cost C for set of resources r is calculated as:

$$C(r) = \sum_{r=0} u_r * p_r + \frac{u_r}{1000 e_r} * t_r \tag{4.2}$$

where u_r is the energy consumed by resource r (measured in kWh), p_r is the energy price per kWh for resource r, e_r is the emissions factor for resource r, and t is the current tax rate per metric tonne of CO_2 for resource r.

We have in previous works extended this energy model to account for additional costs including the hardware and network infrastructure [162].

Logging: The simulation employs a multi-level logging approach. Within the configuration file the logging level is specified. Detailed logging is available during development and debugging, while lower levels of logging may be selected to minimise output size for large sets of simulation runs.

4.3 Case Study of HTCondor

In this section we validate our simulation environment by modelling the HTCondor deployment at Newcastle University and use the simulation environment to explore a set of simple resource selection policies. We also discuss the process of obtaining interactive user and HTCondor workload trace data across a twelve month period to drive the simulation.

4.3.1 Newcastle University HTCondor pool

In 2010 the Newcastle University's HTCondor pool comprised ~1400 desktop computers spread through 35 clusters on campus. The opening hours of these clusters varied, with some respecting office hours, and others available for use 24 hours a day. Clusters may belong to a particular department within the University and serve a particular subset of users, or may be part of a common area such as the University Library or Students' Union building. Computers within the clusters are replaced on a four-year rolling programme with computers falling into one of three broad categories as outlined in Table 4.1. In this work we lack resource utilisation information for the HTC worker nodes, so adopt a power model employing easily obtained manufacturer *'nameplate'* power consumption values for each of the operating states as outlined in Section 4.2.1. These nameplate values are typically estimated by manufacturers as the sum of the worst case power draw of all components in the system [171], so may be considered a worst-case estimate of energy consumption.

The University has a policy to minimise energy consumption on all computational infrastructure which has been in place for a number of years. Hence the 'Normal' computers have been chosen to be energy efficient. 'High End' computers are provisioned for courses requiring large computational and/or rendering requirements such as CAD or video editing, and as such they have higher energy requirements. 'Legacy' computers pre-date the policy of purchasing energy efficient computers and are also the oldest equipment within the infrastructure. All computers within a cluster are provisioned at the same time and will contain equivalent computing resources.

These computer clusters are provisioned for the needs of the primary (interactive) users of the system. Students generally demand Windows-based machines so the pro-

Type	Cores	Speed	Power Consumption		
			Active	*Idle*	*Sleep*
Normal	2	~3Ghz	57W	40W	2W
High End	4	~3Ghz	114W	67W	3W
Legacy	2	~2Ghz	100-180W	50-80W	4W

Table 4.1: Computer Types

portion of resources capable of checkpointing (i.e. Linux) is limited. At Newcastle University, Linux-based machines constitute only ~5% of resources available to HTCondor.

All cluster machines within the pool reboot between 3am and 5am each day to install new software, perform updates and install patches. The reboot also helps to clear any temporary faults which may be present on the machine.

4.3.2 HTCondor-specifics

We extend our generalised simulation environment to model the operation of an HTCondor environment. The operation of HTCondor is modelled around the description of core components presented in [229]. HTCondor uses ClassAds [215] to define jobs. A ClassAd is a attribute-value pair document containing all information about a given job. A ClassAd can contain any number of element pairs, our system producing over 50, however, there are only currently nine elements we require for our simulations. Table 4.2 maps these to characteristics of a job which we identify in Section 4.2.4. Note that `JobStatus` can have values here of '4' for completed jobs and '3' for terminated jobs. Note also that the computation for d neglects the fact that jobs can accumulate time through suspensions which would be included here. This can easily be removed by subtracting `CumulativeSuspensionTime`.

Job characteristic	Tuple term	HTCondor parameter or expression
Job identifier	j	`ClusterId`
Batch identifier	b	`ProcId`
Submission time	q	`QDate`
Job duration	d	`EnteredCurrentStatus` `-JobCurrentStartDate`
Owner	h	`Owner`
Result state	e	`JobStatus`
Data transfer in	u	`BytesSent`
Data transfer out	d	`BytesRecvd`

Table 4.2: Job Characteristics to HTCondor mappings

HTCondor provides powerful resource matching through the 'Matchmaker' which takes in two ClassAd pairs namely `Requirements` and `Rank`. Requirements is used to

indicate characteristics which must be present on a resource for successful matching, such as type of operating system and minimum memory, whilst rank indicates how to order all those resources which match the requirements – with the top-raking resource being used. As our main intention here is comparison of energy consumption and overheads, and `Requirements` and `Rank` were almost completely unused in our log [158], we have ignored this information here. However, it would not be difficult to extend the resource allocation code to take this into account.

4.3.3 Preparing User logs

Interactive logins on resources at Newcastle University are handled through a central Managed Desktop Service (MDS). Extracting the user logins and user logouts from 2010, we are able to construct an amalgamated user trace log. Unfortunately the MDS provides login and logout data separately and each file can contain duplicate records – both identical in time and separated by a few milliseconds. This is a consequence of the login to the resource and the mounting of remote user file-space. Further to this the records are not generated in chronological order. We have developed a tool which is able to remove the duplicates, match logins to logouts and order the trace log. A further complication arises in the case where a computer crashes or is powered off manually during a logged in session. In this case there will be no corresponding logout. As this accounted for less than 0.1% of the trace log these were ignored.

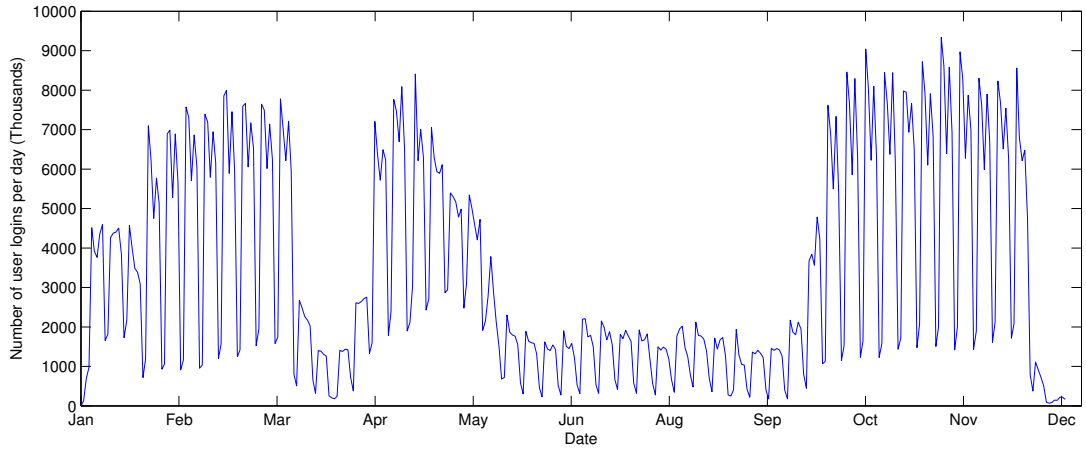

Fig. 4.5: Newcastle University Interactive user activity trace for 2010

Figure 4.5 illustrates the user profile from 2010, representing 1,229,820 user logins. It is easy to see the weekly cycle of computer usage, with lower usage at the weekends, along with term patterns indicating when the easter and summer breaks occurred. We currently do not possess resource utilisation information from user sessions therefore assume 100% resource utilisation of the computers whilst users are active.

4.3.4 Preparing HTCondor logs

HTCondor collects historical logs of jobs which have either completed or been terminated. This can be extracted with the `condor_history -long` command. However, this history may only contain the previous n jobs (where n is configurable in HTCondor) and the jobs are ordered by completion rather than submission time. In order to overcome the former a regular capture of the history can be performed, however, this may lead to duplicates. To solve this and the ordering of records we have produced a tool which orders jobs by submission time and removes duplicates. The simulation itself is then able to read the processed HTCondor log directly through an HTCondor translator.

Figure 4.6 illustrates the number of jobs submitted each day during 2010. In total 561,851 jobs were submitted, with a mean job submission rate of 1,454 jobs per day. There is no clearly visible pattern to this trace log.

Furthermore, since December 2012 we have extend our data collection to include

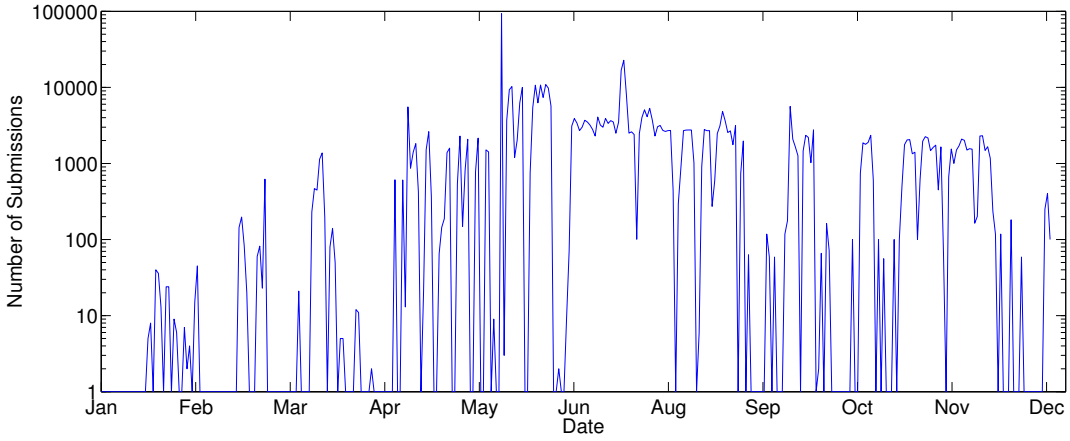

Fig. 4.6: Newcastle University HTCondor workload trace for 2010

event logs which include additional information including periodic memory and disk utilisation information throughout a jobs execution, and complete logs for resource re-allocation, suspension and checkpointing. This fine-grained event logging is typically only provided to the submitting user of a job, but centralised collection of this data may be enabled by including the following options in an HTCondor configuration.

```
EVENT_LOG = /some/file/path
EVENT_LOG_USE_XML = True
EVENT_LOG_MAX_SIZE = 52428800
EVENT_LOG_MAX_ROTATIONS = 3
```

The HTCondor log files comprising our dataset were collected using Condor v6.6, but our simulation remains compatible with current versions of HTCondor (currently v8.3.0).

To facilitate the sharing of HTCondor traces across organisational boundaries, we provide tooling support to automatically sanitise logs obtained from running systems, removing sensitive or personally identifiable information. Fields such as job owner and executable name are replaced with hashes to facilitate more detailed analysis of workload traces.

Figure 4.7 shows the proportion of cluster time used by interactive users, HTC workload and time spent in an idle state for our HTC pool in 2010. We may observe that the offered workload to our system in 2010 results in very low system utilisation (12%).

Figure 4.8 shows the probability that a job of length x hours will complete given that it is submitted during hour y of the day. Probabilities are obtained through simulation based on our Newcastle University trace logs for interactive users, and knowledge of computer reboots. Note that this is assuming that no other jobs are running at the time and should th erefore be considered as an overestimate of the probability. As all computers are rebooted at 3am this leads to the diagonal cut-off within the heat map going from a 50% chance of completion to 0% in the lower right hand side of the figure. There is only one hour slot under which a 24 hour job can complete - when started immediately after a computer reboot at 3am. The highest chance of short jobs

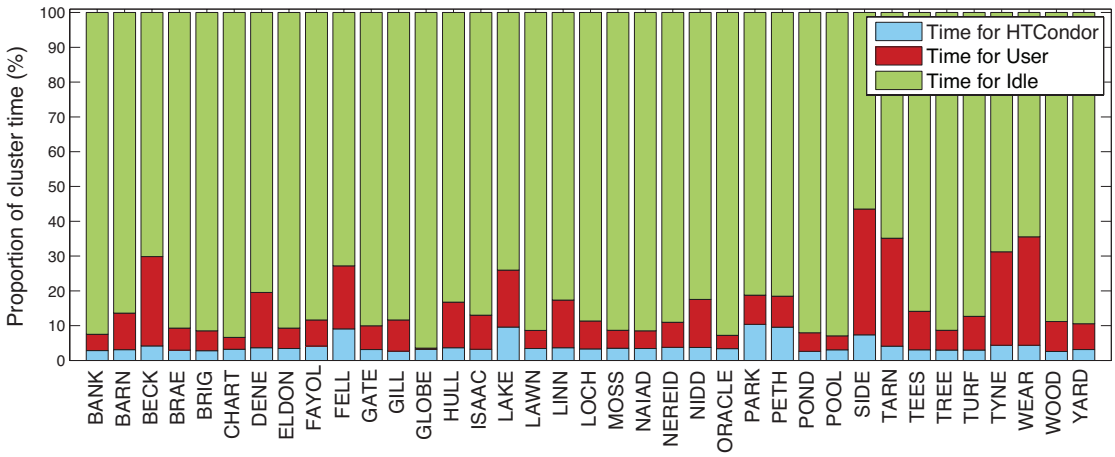

Fig. 4.7: Proportion of cluster time used by interactive users and HTCondor

completing successfully falls between 3am and 8am. By using Figure 4.8 along with largest prior execution time for an evicted job we can determine with some degree of confidence the chances that the job will complete at the time of submission. The prediction of task completion time for our institutional workload is explored in greater detail by Bradley *et al* [41].

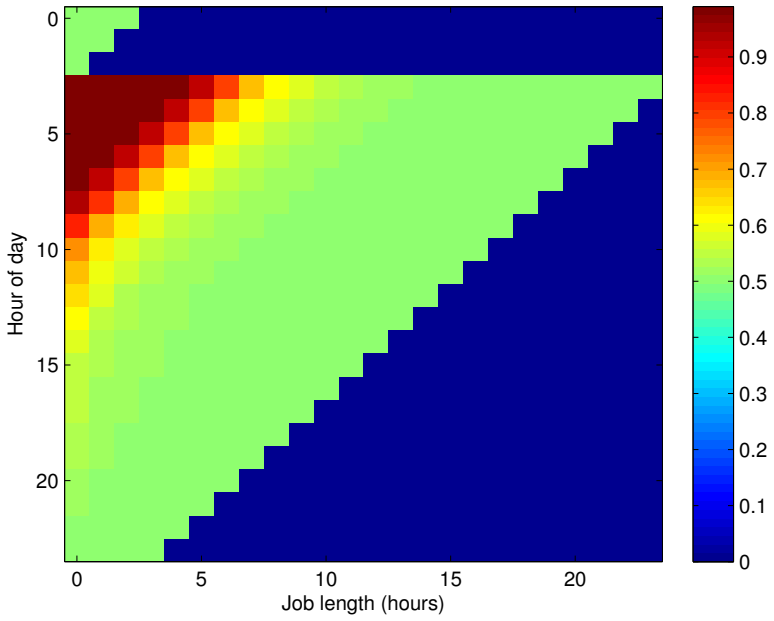

Fig. 4.8: Heat map showing the probability of successful job completion given job duration and submission time

Figure 4.9 shows the percentage of jobs each day which required y hours of execution time – ignoring any time wasted through evictions. Note that this does not include

jobs which failed to terminate as these jobs do not have a meaningful execution time. Most *'good'* jobs have an execution time less than three hours. However, there are a number of anomalies. Thus it is not safe to assume any job which has received over y hours of service will automatically be a *'bad'* job.

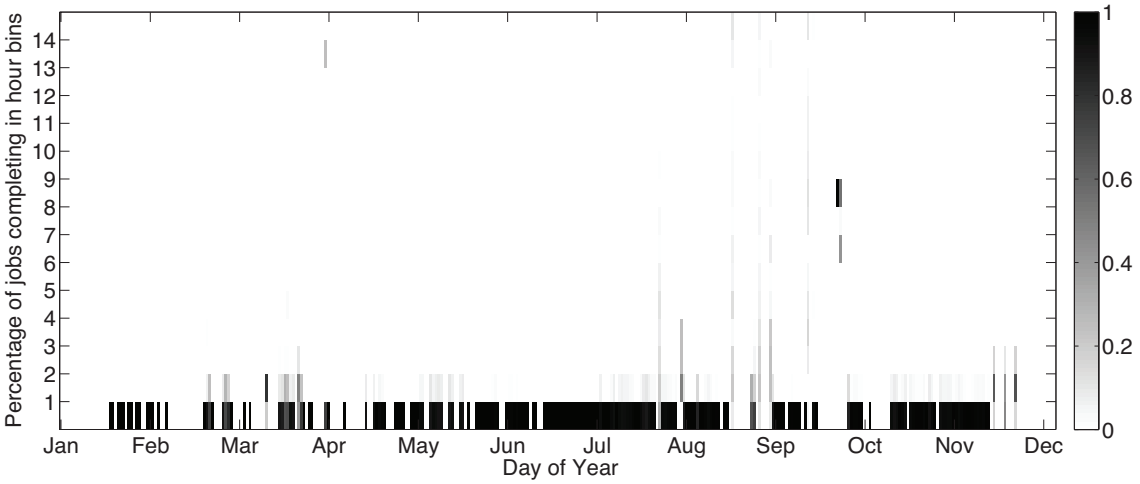

Fig. 4.9: Breakdown of Job durations per day

4.4 Performance Evaluation

Here we evaluate the performance of our simulation software and justify its applicability to arbitrary sized HTC data sets. We do so using the resource allocation policies outlined in Chapter 5. We evaluate this in terms of the wall-clock time to run the simulation and the maximum memory footprint. The timing for a simulation and the memory footprint will be a direct consequence of the policy set being evaluated. For example a simulation such as **S6** which holds a sliding window of prior user logins will require more memory to maintain this set along with more time to process the set than a simulation based solely on random resource selection. We therefore present figures here for simulations based on policy **S1**.

Each simulation was run on a machine with an Intel Core i7 860 2.80GHz processor with 4GB RAM and 500GB 7,200RPM Western Digital Blue hard drive, running the Fedora 19 operating system. Results are based on ten simulation runs using different machines to reduce random effects.

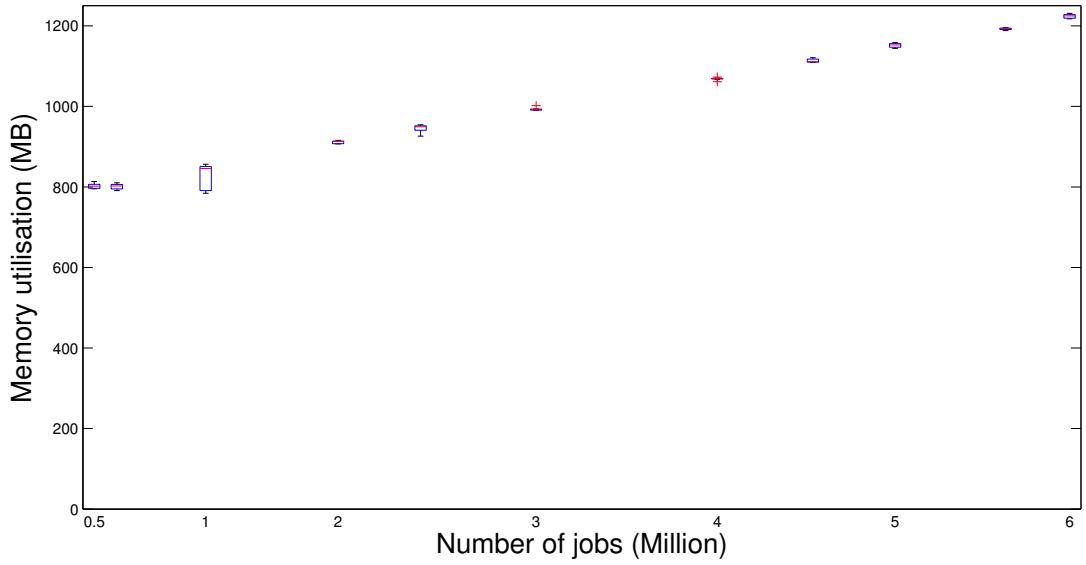

Fig. 4.10: HTC-Sim performance analysis: Maximum memory footprint

Running our real historical trace log of HTCondor workload requires an average of 3:03 minutes. Running the simulation without the HTCondor requires 2:06 minutes, whist running without interactive users (representing a dedicated cluster) requires 1:13 minutes. Note that you cannot just sum these two times to give the overall simulation time due to simulation book-keeping and the processing of cluster events such as computer reboots and clusters opening and closing. The memory footprint for these simulations are 802MB, 750MB and 795MB respectively. The higher memory footprint from the HTCondor only simulation most likely a consequence of the larger ClassAds log file.

In order to evaluate the scalability of our simulation software we investigate the execution time and memory footprint when running larger (synthetic) workloads [161] – over ten times our real workload (~six million jobs). Figures 4.10 and 4.11 show the memory footprint and execution times respectively for both our original simulation and synthetic trace logs. In both cases the memory consumption and execution time increases linearly with workload indicating the simulation scales well with workload. The only exception to this is the execution time for the largest synthetic workload. However, as this requires a memory footprint close to the normal Java memory allocation this is likely to be a consequence of aggressive garbage collection.

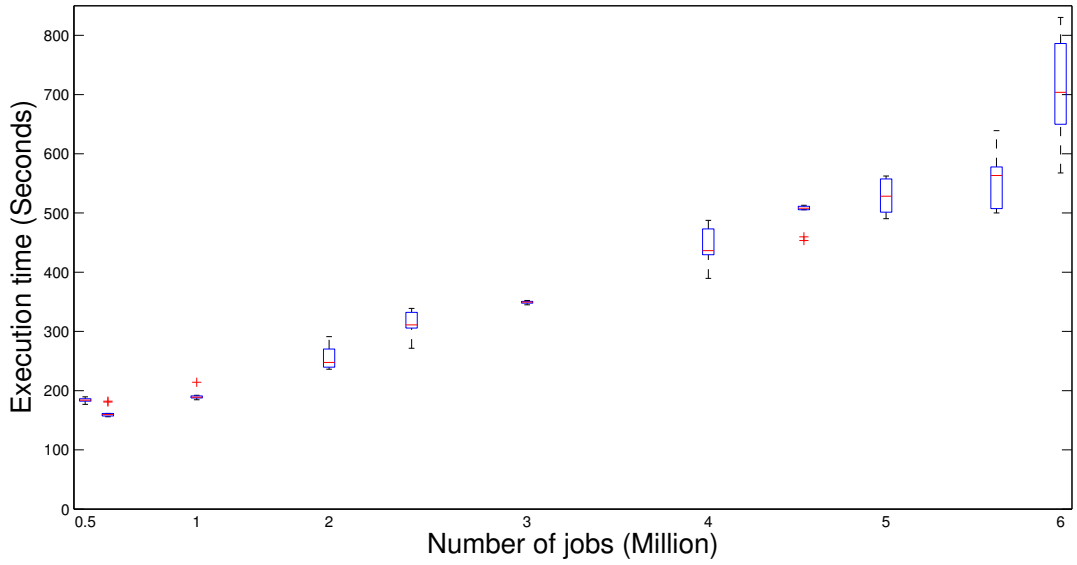

Fig. 4.11: HTC-Sim performance analysis: Execution time

4.5 Conclusions and Further Work

In this chapter we have introduced HTC-Sim, a trace-driven simulation framework for the simulation of High Throughput Computing systems comprising both dedicated resources and resources shared with interactive users. We outline our generalisable model of HTC systems and apply these to HTCondor [146], forming the basis of the remainder of this thesis. We evaluate the impact of running HTC-Sim both in terms of memory footprint and execution time, both for our own institutional trace data and larger synthetic trace logs derived in [161]. We show the simulation scales linearly in both memory consumption and execution time as the size of the workload trace increases.

4.5.1 Further Work

As part of ongoing research in the field of energy efficient high throughput computing, we are working to extend the capabilities of the HTC-Sim framework in a number of key areas:

Workflow and MPI workload support Computational grids may be used to execute jobs belonging to one of a number of categories [192]; Bag-of-task, Message

Passing Interface (MPI), and Workflow. In this work we consider a Bag-of-task workload [59], comprising multiple independent tasks with no communication among each other. In contrast, MPI workloads are needed to communicate with one another throughout execution, so there is a need for a number of resources to be made available at the same time. In further work we shall extend our simulation to; a) support MPI workloads spanning multiple compute nodes, b) incorporate support for Workflow workloads, e.g. by modeling the functioning of the Directed Acyclic Graph Manager (DAGMan) [227], the meta-scheduler used by HTCondor to handle the dependencies between Workflow jobs. This would allow the extension of existing DAG-based application mapping techniques [97–99] to be evaluated for energy consumption in the context of multi-use clusters. An overview of the workflow support of grid systems including HTCondor is avilable in [257].

Extended HTC system support Of particular interest is the ability to evaluate policies for a wider range of HTC systems, and for workload traces available in the literature [1, 10, 84, 112]. To this end we are currently adding a workload translator to support the use of workload traces in Grid Workload Format (GWF) [111], and those in formats of other HTC systems e.g. BOINC.

Network modelling As discussed in Section 4.2.7, we model the bandwidth between nodes in our system as the basis for data transfer delay calculation and decision making within our simulation. However, we do not currently support modeling of bandwidth sharing or the ability to report network contention between nodes. This is increasingly desirable when considering data-intensive workloads and will form the basis of further work.

Multi-tenancy In our current simulation model we assume a given resource is capable of executing a single task at a time. Modern grid systems commonly comprise multi-core systems, and with not all workloads supporting multi-core operation, there is significant scope for the conslidation of workloads.

The inclusion of multi-tenancy also introduces a further policy decision, governing the batching of jobs during quiet periods. Kamitsos *et al* [121] explore determining the optimal time to delay applications under low load to reduce energy consumption, with considerations for temperature and reliability. While they deal with web workloads, we envisage a similar appropach to work in the context of high-throughput computing environments.

Resource failure modeling Multi-use clusters such as our own are often underrepresented in the public grid [1, 10, 84, 112] and failure [81, 118, 206, 207] datasets. We indent do address this by instating monitoring and data collection among our institutional clusters, such that we may posess failure traces which may be incorporated into the simulation environment. Similarly, we consider the integration of statistical models of resource failures into our simulation model.

Dynamic Voltage and Frequency Scaling (DVFS) Dynamic Voltage and Frequency Scaling (DVFS) is commonly used within the literature to reduce energy consumption in HTC and HPC systems [168, 240, 261]. To date we consider many of these works to be complementary to our own but have been unable to evaluate them. We shall extend our model of our compute resources to facilitate DVFS.

Chapter 5

Resource Allocation

Summary

In this chapter we demonstrate the energy and performance impact of resource allocation strategies in high-throughput computing systems in the context of multi-use clusters. Such strategies govern the power management of the underlying resources, as well as resource selection and the use of suspension strategies to promote successful execution. We extend our simulation model previously introduced in Chapter 4, and demonstrate that these policies could save 55% of the currently used energy for our high-throughput jobs over our current cluster policies without affecting the high-throughput users' experience.

5.1 Introduction

In this chapter we investigate the impact of resource allocation strategies on energy consumption policies, identifying policies capable of directing high-throughput jobs to more energy efficient computers, reduce the amount of time that is wasted before a job starts execution and reduce the chances of a job being evicted. This will reduce the time between job submission and completion with the further advantage of decreasing energy consumption. We also investigate policies to reduce the amount of time a computer is idle awaiting either an interactive user or a high-throughput job.

The rest of this chapter is structured as follows. In Section 5.2 we discuss existing examples of policy that are currently, or have previously been, enacted in various production environments within academic institutions. Section 5.3 discusses a number of policies that have been identified, this includes ones that we have implemented in our production environment, and also those we wish to evaluate before implementation. In Section 5.4 we describe the results of our experimentation, before concluding and

motivating future work in Section 5.5.

5.2 Existing Examples of Policy

Historically many institutions, like Newcastle University, allowed their computer labs to remain powered up at all times. With the increasing requirements for institutions to save energy this waste of power was identified as an early target for savings. The initial policy was to place the computer into a shutdown (ACPI S5) state. This has the disadvantage that these computers are no longer available to HTCondor (see Section 4.3.2) leading to high-throughput starvation.

The base Condor system allows for the definition of policies which describe how jobs will run and under what circumstances jobs will be evicted from computers. These policies can affect the amount of time a computer needs to remain idle before it can be used by Condor and which computers should be used, having a direct effect on the responsiveness of the Condor cluster and also on the power consumed by the cluster.

These policies have not been used in the past to provide energy efficiency, however, newer versions of Condor now take such things into account. Condor now includes the ability to send computers to sleep and then wake them up, via *Rooster* [49], as and when needed [51]. This policy system does require that Condor take control of the power-management of the cluster. If Condor has exclusive use of the computers this is not a problem, though it may cause contention if pre-existing power management tools are in use [157, 213].

There are a number of different operating policies employed at UK HE institutions, for example:

Cardiff University has taken an approach in which computers will send themselves to sleep after a set time (normally cluster closure time) if there is no Condor job running on the resource. Powering themselves down as soon as there are no jobs left to service. Any Condor job arriving after this time will be unable to use the computer until it is re-started. This can lead to backlogs of Condor jobs if submitted after the computers have entered an offline state.

Liverpool University have implemented a more advanced policy. Computers still go to sleep when they have no work to service without informing Condor. However, a script run at regular intervals looks for computers which have 'disappeared', due to going to sleep (without informing Condor), and inserts a 'fake' resource description so that Condor thinks of it as being asleep allowing Rooster to wake it up as necessary [213]. This relies on having a pre-defined list of known computers and leads to intervals when Condor 'looses' computers, as it is unaware that a computer is sleeping but available for execution.

University College London uses a thin-client system for its open access computers though runs the client on a full-spec computer. This allows them to exploit the unused computing power of the computers as part of their high-throughput computing system. Though in itself this doesn't reduce power consumption the ability to use computers for multiple purposes simultaneously helps them cut down on capital expense and maintenance.

In previous work McGough *et al* [157] proposed a model in which computers send the 'fake' sleep notification themselves just before entering the sleep state, along with a script, run at regular intervals, to catch 'lost' computers (ones marked as being awake, but with no update for a pre-defined amount of time), which have failed to send the 'fake' sleep notification. This has the advantage of not requiring a list of known computers and reduces the time that Condor is unaware that a computer is sleeping.

5.3 Policy

In this section we discuss a number of polices which can be applied to a multi-use cluster similar to the one at Newcastle University. These may be broadly categorised into cluster management, selection of computers to use and job management.

5.3.1 Cluster management

Power management of computers covering when the computer can be awake (active or idle) and when the computer can sleep. The four power management policies are:

P1 Computers are permanently awake. This was the default policy used by most high-throughput computing installations before power saving. This policy can lead to large amounts of wasted energy when a computer is idle, though as computers are always available it minimises overheads.

P2 Computers are on during cluster opening times or powered off otherwise with no ability to wake up. If the computer is servicing jobs at cluster close time it remains active until this and any subsequent jobs are completed.

P3(n) Computers sleep after n minutes of inactivity with no wakeup for high-throughput jobs. Initially we used a value of one hour as the resume time from shutdown was significant. However, the reliable sleep feature of Microsoft Windows 7 has made this process almost instantaneous so smaller values may now be feasible without causing significant inconvenient to interactive users. However, at present we still adopt the one hour time interval.

P4(n) Computers sleep after n minutes of inactivity with HTCondor being made aware of their availability. This policy is an extension of policy **P3**, which additionally allows Rooster to wake up computers when needed to prevent resource starvation for high-throughput jobs.

P5(m,n) Computers sleep after m minutes of inactivity with sleeping computers being advertised every n minutes. When a computer goes to sleep no information is sent to Condor. A service runs every n minutes checking for sleeping computers, posting a 'fake' advertisement for them. This policy is originally proposed by Smith [213] and is included here for comparison.

5.3.2 Selecting computers to use

These policies allow us to determine which computer to select for job execution.

S1 HTCondor default: note that this devolves into a random selection policy favouring powered up computers.

S2 Target the most energy efficient computers. Energy consumption for each computer is defined along with a Power Usage Effectiveness (PUE) – the ratio of total amount of power used by a computer facility to the power delivered to computing equipment. This allows us to order the computers by PUE × energy, targeting jobs as appropriate. It is important to note that in this instance we use PUE as a relative measure of energy efficiency against which resource selection decisions may be made, rather than an absolute measure.

This process is an approximation to the efficiency of a job, as different computers will handle different computational tasks with different degrees of efficiency. One computer may be most efficient on memory intensive jobs whilst another may be more efficient on floating point dominant jobs. However, this policy aims only to steer jobs towards the more energy efficient computers based on our benchmarking.

5.3.3 Job management

These policies allow us to alter the behaviour of Condor in terms of when to allow jobs to start running and when to cease attempting to process a job:

M1(*n*) A computer may not be used until it has been idle for n minutes. This Condor default is intended to prevent computers that are frequently used from being matched.

C1(*n*) Detection of 'miscreant' jobs. Condor attempts to run jobs to completion, this includes re-submitting evicted job due to computer crash, reboot or user precedence. If this happens n times we mark the job as 'miscreant'. Selection of the value n needs to be made carefully: too small a value will create false positives whilst large values will waste time and energy.

Although a miscreant job may not be broken it may not complete, continuing to consume resources. We may then choose to terminate these jobs. Care needs to be taken as such jobs may be performing good computational work through some other out of bounds mechanism.

Policies to detect and mitigate the impact of 'miscreant' jobs are explored in further detail in Chapter 6.

C2(m,n) Provision of dedicated computing resources. Extending the repeatedly evicted policy **C1**(n). Once a job has been evicted n times it is allowed to continue execution on a dedicated set of m computers. This would throttle the problem of long running jobs never completing due to repeated eviction though we would still need to monitor these jobs for non-completion.

C3(m,n,t) Timeout for dedicated computers. If there are more miscreant jobs than dedicated computers then policy **C2**(m,n) then all dedicated computers are blocked with jobs and the policy degrades to **C1**(n). However, if we select a time interval t over which we assume the job will not complete and can therefore be safely killed we regain the ability for the dedicated resources to allow long running 'good' tasks to complete.

5.3.4 New Proposed Policy

We wish to evaluate a number of proposed policies in terms of how much power they might save and the potential impact on the high-throughput users. Some of the policies may also have an impact on the interactive users of the cluster. It is not possible to determine those effects here, though by using the simulation we can evaluate the impact on high-throughput users and power consumption enabling us to evaluate whether such a policy would lead to a large enough advantage that it was worth considering the policy and potential impact on the interactive users. These can be grouped into cluster management and computer selection.

M2 High-throughput jobs defer nightly reboots. Allow high-throughput jobs to run through the night and thus for longer than 24 hours. This policy addresses the same issue as policy **C2**.

M3 High-throughput jobs use computers at the same time as interactive users. Desktop computers are now more powerful. All computers at Newcastle University are at least dual core with many quad core. From observation the interactive

load is often far less than the available computing power. Although some applications are capable of exploiting multi-core (e.g. CAD and MATLAB [154]) many are only capable of exploiting a single core leading to under-utilisation, which can be exploited through HTCondor.

In this case rather than jobs being evicted when an interactive user logs in an eviction can be triggered when the load placed on the computer exceeds the requirements of both the interactive user and the high-throughput job. Our trace of interactive uses of the computers does not include information on the load the user placed on the computer and we therefore assume for these simulations that the load of both high-throughput jobs and the interactive user does not exceed the capability of the computer.

An evaluation of the potential energy savings and reduction in overhead time needs to be performed in order to determine if this policy could provide enough of an improvement to warrant live evaluation over the real cluster.

S3(i) Targeting less used computers. Our interactive user workload traces demonstrate that computers placed in locations frequented by students tend to have short durations between interactive users and are many users each day. In contrast, computers in less popular locations typically observe much greater inter-user durations, and are used by far fewer users. Computer usage can also be affected by the 'opening hours'. It would be beneficial to select less used computers, thus reducing the chance of job eviction and hence less wasted power on incomplete execution.

It is not possible to know *a-priori* which computers will be unused in the near future, also this information would be seasonally affected. However, we can look for general trends in the usage patterns of computers from historical evidence and use this to help select the computers least likely to have a log in. We can favour computers based on their current state – an idle computer with greater chance of a login is used above a computer which is asleep but has a lower chance of login – or selecting the computer with the lest chance of login irrespective of current state. We define the following 14 options for computer ordering:

- [1, 5, 8, 12] : Largest individual average interval: logout – login
- [2, 6, 9, 13] : Largest individual minimum interval: logout – login
- [2, 6, 9, 13] : Largest individual maximum interval: logout – login
- [4, 11] : Smallest number of interactive users

Where options (1, 2, 3, 8, 9, 10) assume the computer will not be rebooted each night, while (5, 6, 7, 12, 13, 14) assume the computers will be rebooted. Options (1, 2, 3, 4, 5, 6, 7) assume that the current state will be taken into account (idle computers before sleeping computers) whilst options (8, 9, 10, 11, 12, 13, 14) will use computers irrespective of whether the computer is idle or sleeping.

S4 Targeting clusters closed for public use. Each computer within the university is part of a cluster with each cluster having pre-defined opening and closing hours. Here we propose selecting computers in clusters which have the greatest amount of time remaining before the cluster is re-opened, thus minimising the chances of the jobs being evicted through by interactive users.

S5(*i*) Target less used clusters. Similar to policy **S3**(i) this policy targets the least used computers. Though differs by the fact that it is simpler to implement. Clusters can be selected based on the following criteria:

1. Largest individual average interval: logout - login
2. Largest individual maximum interval: logout - login
3. Smallest number of interactive users.
4. Smallest total interactive user duration
5. Smallest mean interactive user duration
6. Number of interactive users.

HTCondor contains the ability to suspend jobs when an interactive user logs into the computer. This allows the job to resume if the user logs out of the computer quickly after. If this interval is short enough then this will prevent the eviction of the job and allow it to continue, thus saving energy and overhead. If the interval

is long then this will increase the overhead though save on energy. We extend the notion of suspensions here to allow more fine-grained control over when a job should be suspended and when a job should be evicted.

S6(Δ) A policy observing the number of interactive user arrivals to each cluster across a sliding window of Δ minutes, with arriving jobs allocated to resources ordered by availability. This policy may be expressed as:

$$\min_{c \in C} \left\{ |E_{c,t,\Delta}| \right\} \tag{5.1}$$

where $E_{c,t,\Delta}$ is the set of interactive user sessions starting in cluster c during the time frame $[t-\Delta, t)$, C is the set of all clusters and t is the current time. Contrary to the resource selection proposed thus far, S6 does not rely on prior knowledge of interactive user arrivals.

Policy **S6** was first introduced in [90] as an extension to the resource selection policies first explored in [161].

H(*initial*, *subsequent*) Allow a job to be suspended given the *initial* policy is satisfied for the first suspension and the *subsequent* policy is satisfied for all future suspensions, otherwise the job is evicted.

The *initial* polices can be defined as:

- **None** : Jobs will be immediately evicted.
- t : Allow the job to be suspended for up to t minutes. After this time the job should be evicted. This is the default Condor policy. A small value of t allows jobs to remain active if there is a brief use by an interactive user.
- p : Allow the job to be suspended for up to p % of its current execution time. This allows jobs which have received little execution to be evicted quickly as this gives the best chance of keeping the overheads low. Whilst tasks which have received significant amounts of service are suspended for longer as their is greater impact if these are evicted.

The *subsequent* suspension policy determines if the job can be re-suspended:

- **None** : Jobs will be terminated when the second user attempts to log in.
- n : If the job has not been suspended n times already it will be suspended, Otherwise it will be evicted. This helps prevent jobs which are allocated to high turnover computers from receiving short burst of execution thus leading to high overheads.
- T : If the total time the job has been suspended is less than T then the job is suspended, otherwise it is evicted. This helps prevent jobs from spending significant amounts of time suspended and not completing.
- P : If the proportion of time that the job has been suspended is less than a given threshold P then the job can be suspended, otherwise it is evicted. This helps prevent tasks which are only achieving a small amount of progress through suspensions.

In all cases if the job can be suspended then the maximum time interval for suspension in *initial* is used.

5.3.5 Policy Combinations

The polices described above are not mutually exclusive. Most can be used in combination with each other. Table 5.1 indicates the groupings of polices which cannot be used in combination with each other. Policies in different policy groups can always be combined with each other.

Policy Group name	Combinable policies	Non-combinable polices
Power		P1, P2, P3, P4, P5
Selection		S1, S2, S3, S4, S5
Management	M1, M2, M3, H	
Job Termination		C1, C2, C3

Table 5.1: Resource Allocation : Policy Combinations

5.4 Simulations and Results

In this section we evaluate the previously described policies in order to assess an optimal set of policies for our cluster. These tests could easily be performed on other clusters and we believe that the general conclusions from this work will be applicable to other similar clusters. These tests are grouped into baseline tests, power management tests, computer selection tests and cluster change tests. As the simulations presented here, apart from the default policy for selection of computer to run on, are entirely trace-driven only a single run of the simulation is considered. For simulations based around the default selection policy **S1** (HTCondor default, random selection favouring powered up computers.) the average of 10 simulation results are reported.

5.4.1 Baseline Evaluation

We first perform a simulation aimed at providing baseline energy consumption and performance figures against which each other policy will be evaluated. In this simulation power policy **P4**(n) (Computers sleep after n minutes of inactivity with HTCondor being made aware of their availability.) was used and only interactive users were simulated. This simulation generated 120.7MWh of active power consumption, 33.8MWh idle time consumption and 28.5MWh of energy consumption for sleep time. The energy consumption for Condor is then calculated separately from this. If we add in the execution of HTCondor jobs this adds ~120.9MWh of energy consumed for these jobs along with an average overhead of ~13.33 minutes.

In the remainder of this section we test each of the proposed polices in isolation against our default policy, to determine the effectiveness of each. We then combine the 'best' policies and evaluating these for both real and synthetic workloads.

5.4.2 Power management policies

Here we evaluate power management policies **P1** (Computers are permanently awake.), **P2** (Computers are on during cluster opening times or powered off otherwise with no ability to wake up.), **P3**(n) (Computers sleep after n minutes of inactivity

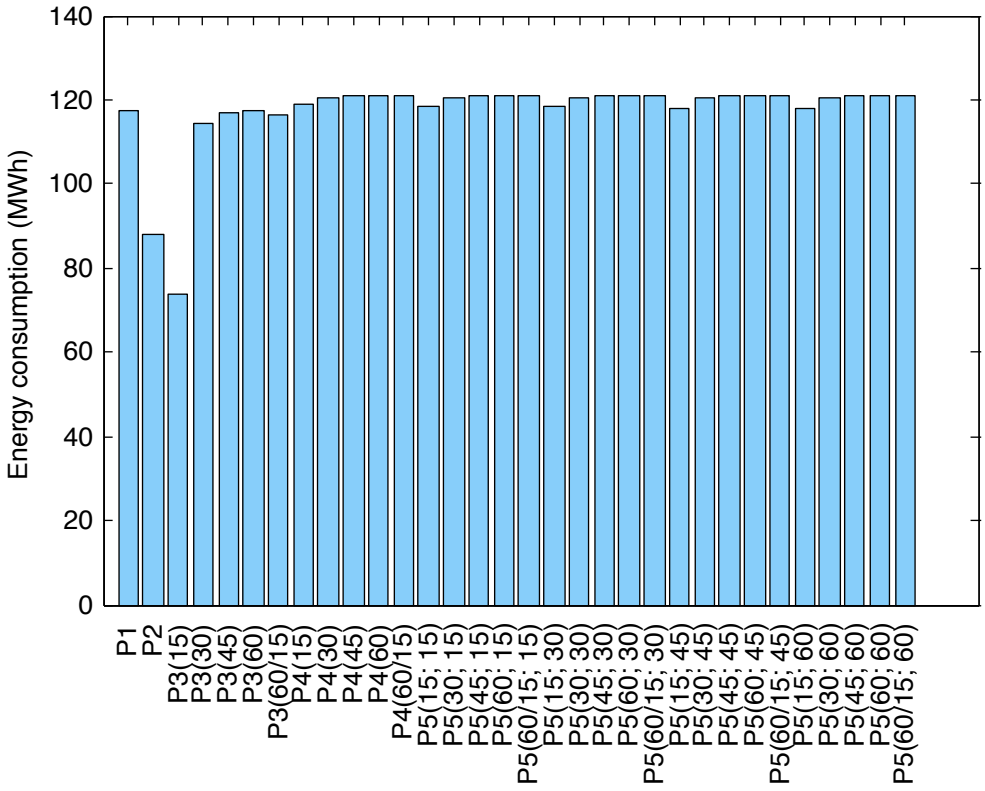

Fig. 5.1: The impact of Power Management policies on energy consumed

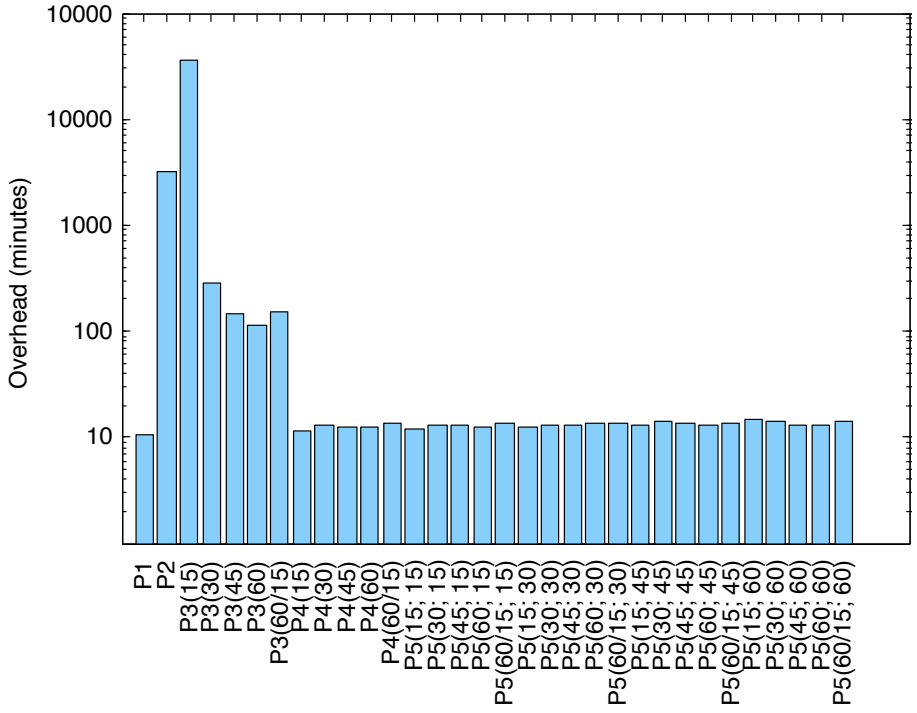

Fig. 5.2: The impact of Power Management policies vs. overheads

with no wakeup for high-throughput jobs.), **P4**(n) (Computers sleep after n minutes of inactivity with HTCondor being made aware of their availability.) and **P5**(m, n) (Computers sleep after m minutes of inactivity with sleeping computers being advertised

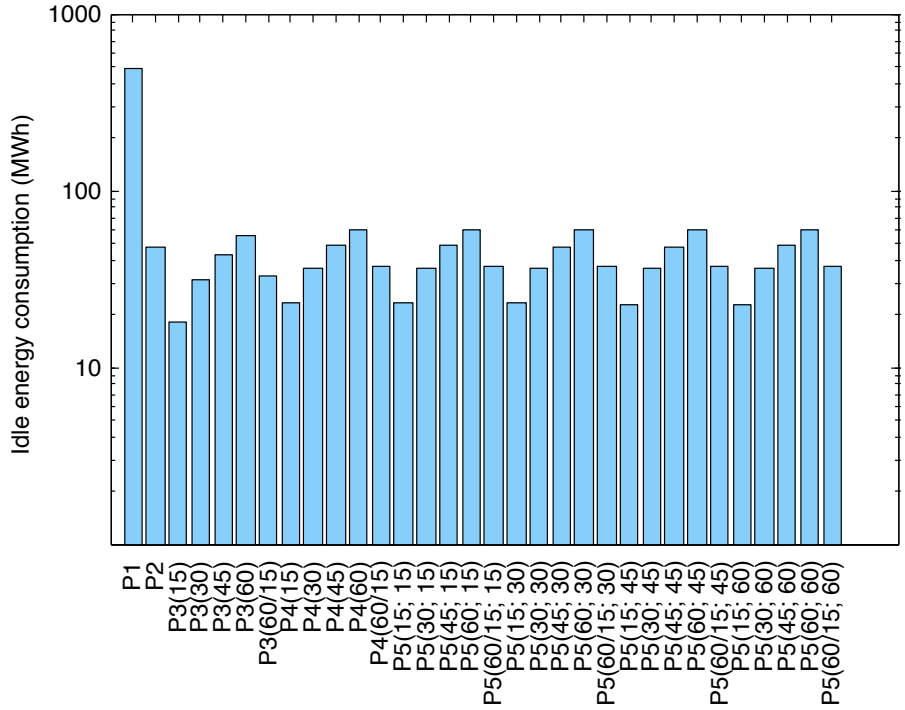

Fig. 5.3: The impact of Power Management policies on energy consumed

every *n* minutes.). All tests were performed with selection policy **S1** (HTCondor default, random selection favouring powered up computers.). The amount of time before computers were allowed to go was also varied for policies **P3**, **P4** and **P5**, taking values of 15, 30, 45 and 60 minutes. The current policy of 60 minutes during open hours and 15 minutes outside was also evaluated *(60;15)*. Figures 5.1 and 5.2 illustrate the results from these simulations. Policies **P2** and **P5**(*15,15*) would appear to have the 'best' energy consumption result (Figure 5.1) however when the average overhead (Figure 5.2) is taken into account these policies clearly starve high-throughput users of their resources. Policy **P5**(*15,15*) would seem to be a consequence of computers going offline at the same time as the sweep happening thus leading to computers being absent for longer. The remaining polices show little significant statistical difference even from **P1**. Thus indicating that these policies have little impact on the high-throughput users. Although the policy of changing the time clusters are powered down has no impact on the high-throughput use of the cluster, Figure 5.3 illustrates that this has a marked impact on the energy consumed by idle computers. As this policy can be combined with the other policies this would make sense to adopt and have a low (~ 15 minute) value

such that energy consumption due to idle resources may be reduced.

5.4.3 Computer Selection policies

Here we evaluate the selection policies **S1** (HTCondor default, random selection favouring powered up computers.), **S2** (Target the most energy efficient computers.), **S3**(i) (Targeting less used computers.), **S4** (Targeting clusters closed for public use.) and **S5**(i) (Target less used clusters.) under power policy **P4**(*60;15*). Figures 5.4 and 5.5 shows the result of these simulations. All polices apart from **S4** and **S5** reduce the overall energy consumed, with **S2** and **S5** showing the best improvement. All polices apart from **S3**(1-7) produce no significant change to the overheads for jobs. Thus selection policies **S2** and **S5** would appear the best choice. We observe that policy **S6** is capable of achieving savings comparable with **S5** which assumes perfect knowledge, with sliding window size having little impact on results. Although policies **S3**(1-7) select computers with the greatest chance of being unused for the duration of the job execution the resources are selected by initial state first – idle over sleeping. Hence an idle computer with little chance of remaining idle during the job's duration will be selected over a sleeping computer which would most likely be idle for the job's duration.

5.4.4 Management Policies

Policies **M1**(n) (A computer may not be used until it has been idle for n minutes.), **M2** (High-throughput jobs defer nightly reboots.), **M3** (High-throughput jobs use computers at the same time as interactive users.) and **H**(*initial*, *subsequent*) (Hierarchical policies.) are evaluated here with default selection policy **S1** (HTCondor default, random selection favouring powered up computers.) and power policy **P4**(*60;15*). Figures 5.4 and 5.5 illustrates these results for policies **M1**, **M2** and **M3**. Policy **M1** has little perceivable impact on the power or overhead of jobs. However, this policy does have an impact on the overall energy consumed by the whole system by increasing, by a factor of 10, the amount of energy consumed by idle computers by reducing the energy for sleeping computers when the value of n is low. This is a consequence of HTCondor waking up computers for short running jobs which then leaves the computer

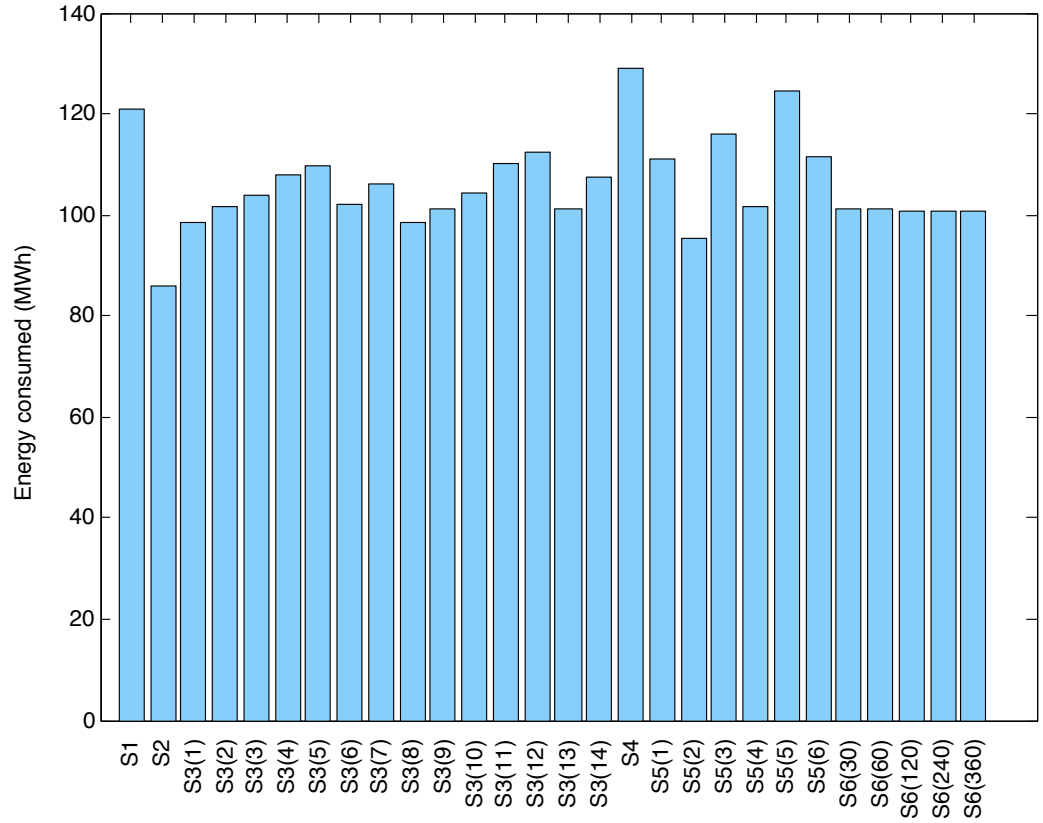

Fig. 5.4: The impact of Computer Selection policies on energy consumed

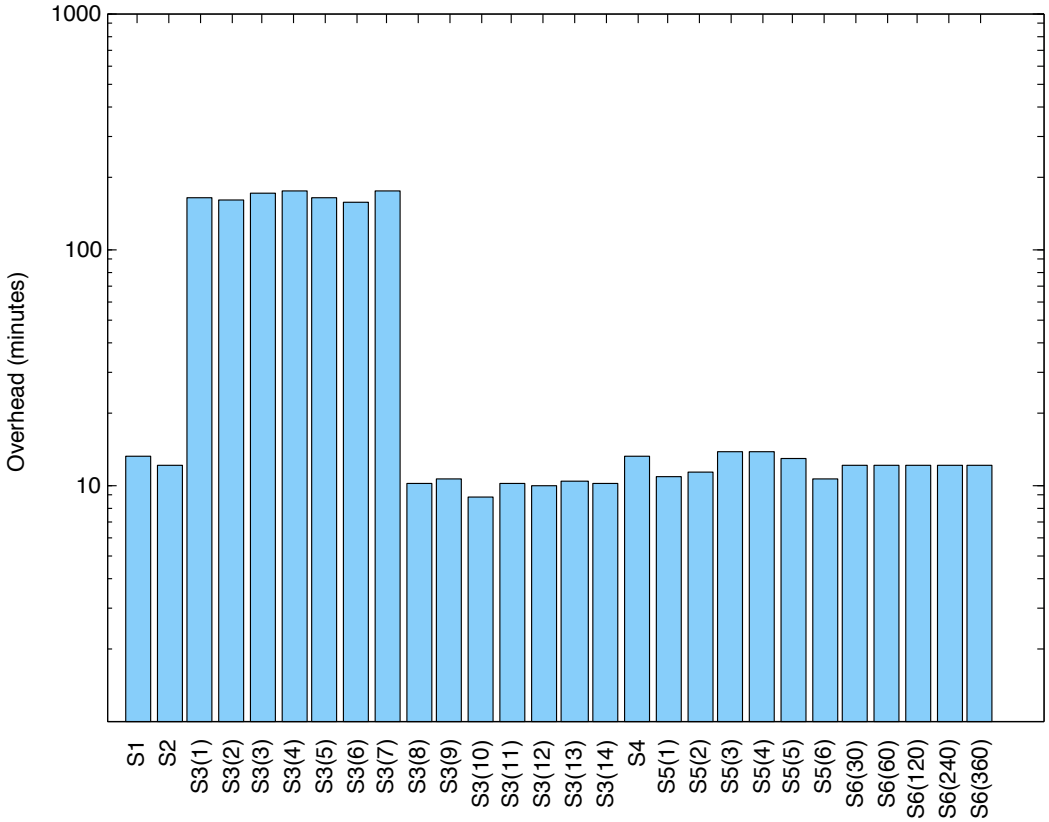

Fig. 5.5: The impact of Computer Selection policies on overheads

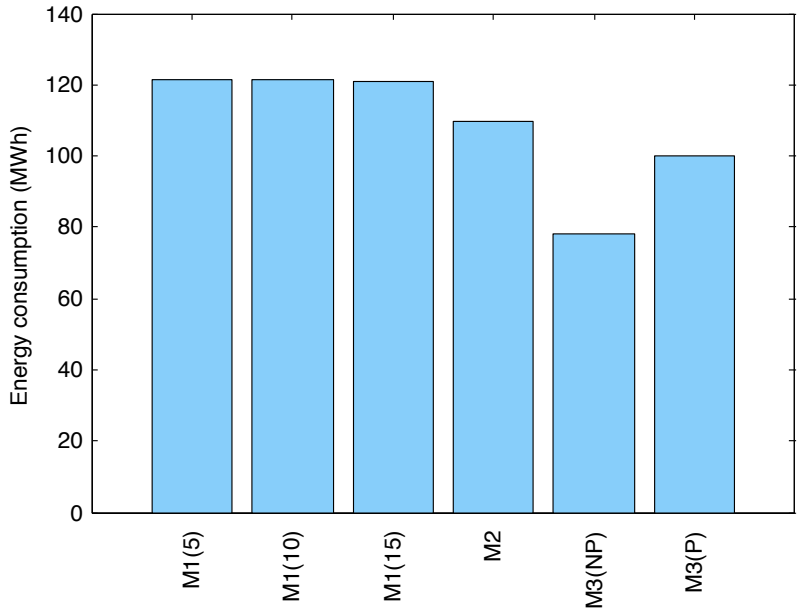

Fig. 5.6: The impact of Management policies on energy consumed

idle. Whilst for larger values of n these short jobs accumulate up and run continuously. There is a slight energy advantage in using $n = 10$ and should be selected. Policy **M2** (jobs prevent reboots) provides an advantage for both energy consumed and overheads and should be used.

Policy **M3** (High-throughput jobs use computers at the same time as interactive users.) is depicted for both the case where we assume that no energy charge is allocated to the HTCondor job (**M3**(*NP*)) and the case where we assume that there is an energy charge for using the computer (**M3**(*P*)). As we do not know the power consumption of the HTCondor job we assume the worst case – the HTCondor job is consuming all the processing power. Using the SPECpower [219] power evaluation software we have benchmarked one of the high-end computers at 117W active and 65W idle. Thus in the worst case scenario HTCondor would consume 52W. Although this policy decreases the overall energy consumed it has a negative impact on the overhead. This is a consequence of 'bad' jobs not being evicted when users log in allowing 'good' jobs a chance of execution.

Figures 5.8, 5.9, 5.10 and 5.11 exemplify policy **H**(*initial, subsequent*) (Hierarchical policies.). For the case of maximum suspension time (Figures 5.8 and 5.9) the 'best' policy appears to be **H**(t, *None*). With all other policies increasing both energy con-

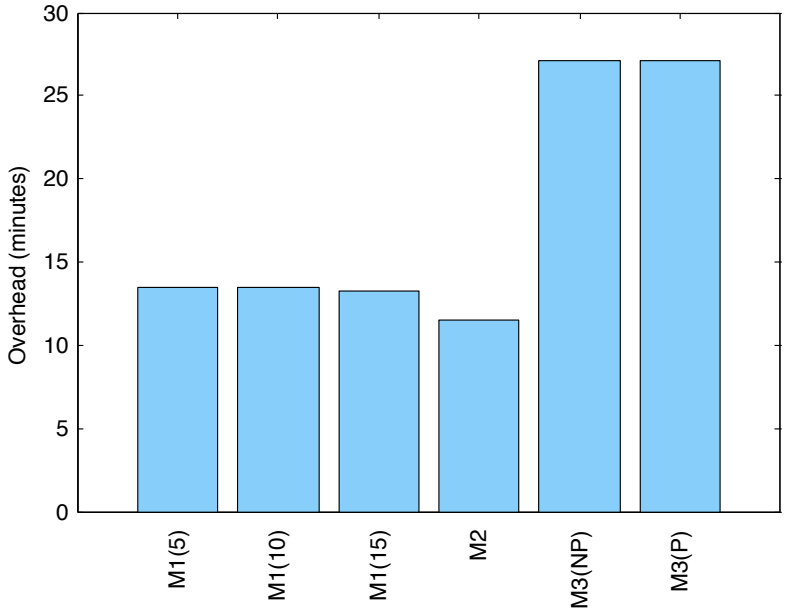

Fig. 5.7: The impact of Management polices on overheads

sumed and overhead. Likewise for percentage of execution time (Figures 5.10 and 5.11) there appears to be no benefit in using any policy over **H**(*p, None*). In fact there appears to be little benefit in using the suspension policy as it increases both energy consumed and overhead over the baseline.

It should be noted that all of the polices in this set can be combined with each other. However, the policies which show the 'best' chance of improvement are **M1**(*10*) and **M2**.

5.4.5 Cluster termination policies

Here we evaluate polices **C1**(n) (Detection of 'miscreant' jobs.), **C2**(n) (Provision of dedicated computing resources.) and **C3**(m, n, t) (Timeout for dedicated computers.). Figures 5.12 and 5.13 illustrate the energy consumption and overheads for these policies. Note that Figure 5.13 only shows the lower retry values to help distinguish the different policies. Policy **C1** leads to significant numbers of good jobs being killed (defined as a job which originally completed successfully now being terminated) – Figure 5.14. Addition of dedicated resources (**C2**) leads to fewer good jobs being killed but can lead to bottlenecks for job overheads if the number of retries are low and excessive energy consumption if retries are high. By the inclusion of a time limit on dedicated

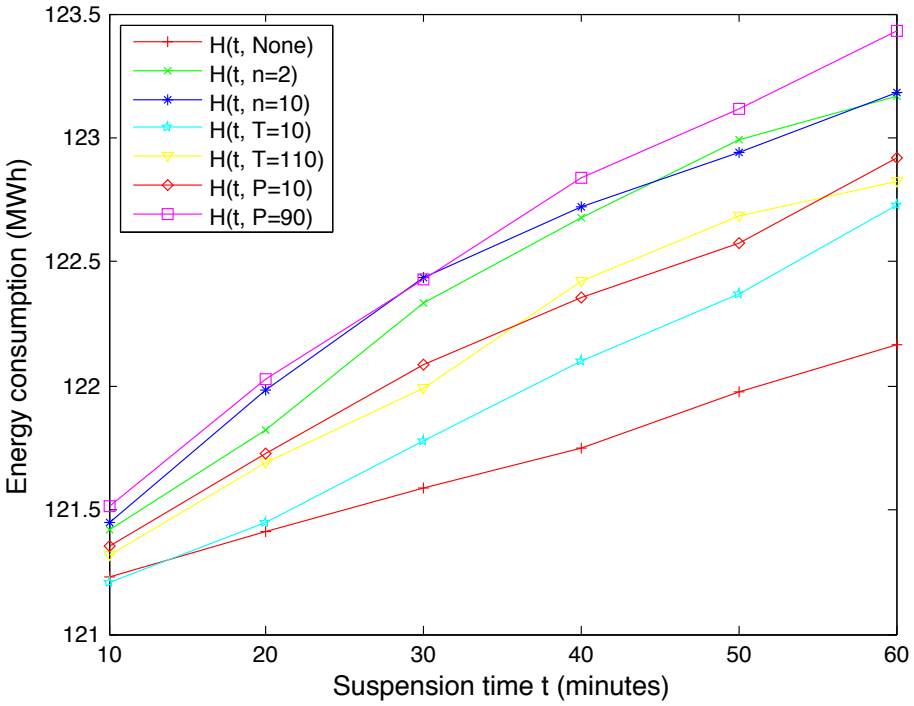

Fig. 5.8: The impact of Suspension time on energy consumed

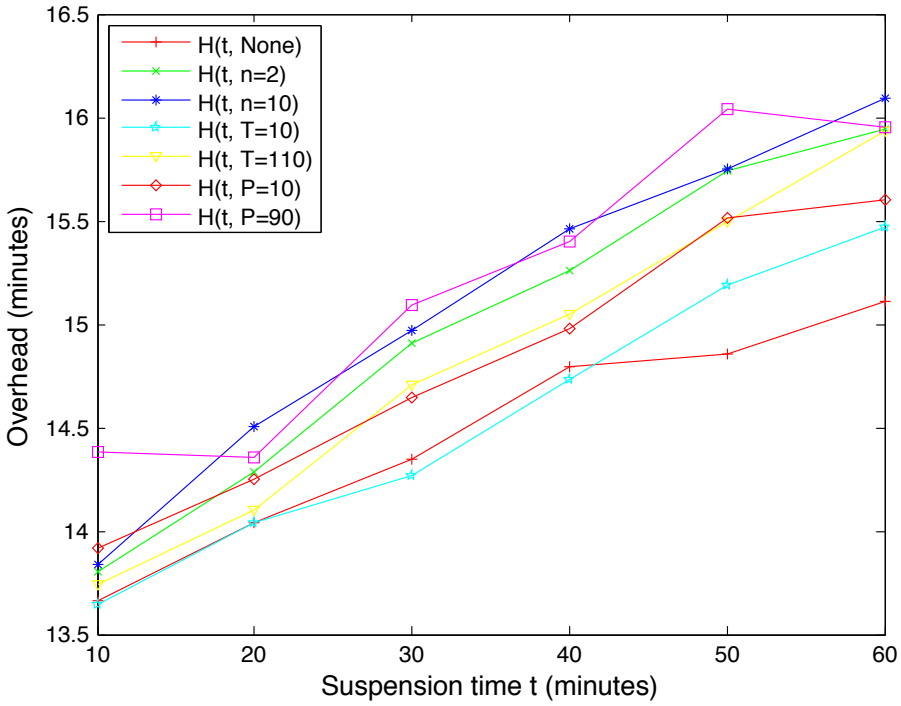

Fig. 5.9: The impact of Suspension time on overheads

resource usage we can bring the energy usage down, by keeping the retires low and preventing 'bad' jobs from running indefinitely on the dedicated resources, allowing us to still maintain good overheads and low numbers of good jobs killed. The policy

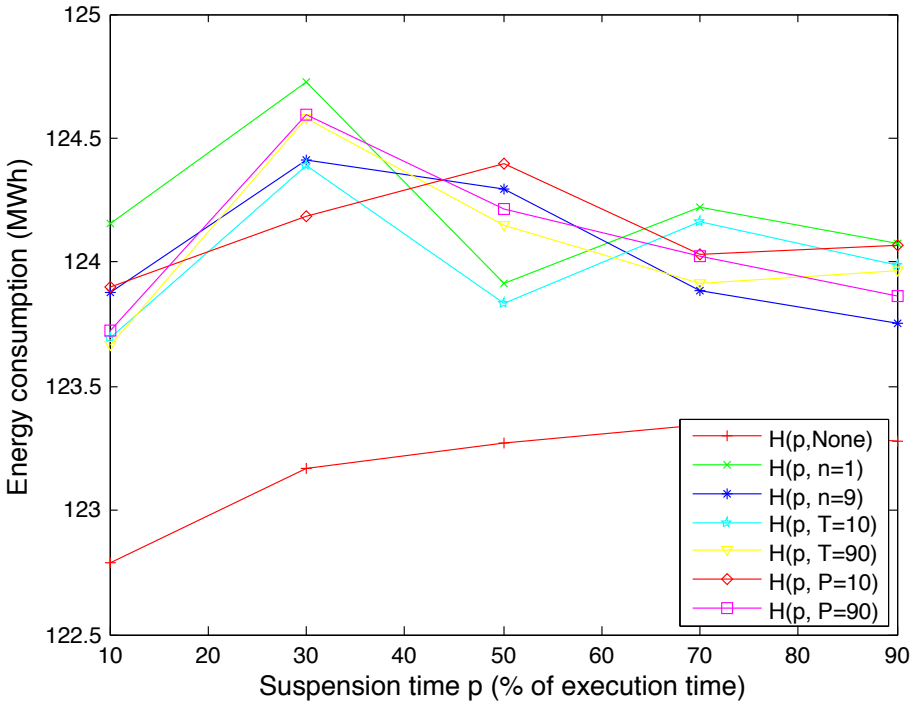

Fig. 5.10: The impact of Suspension percentage on energy consumed

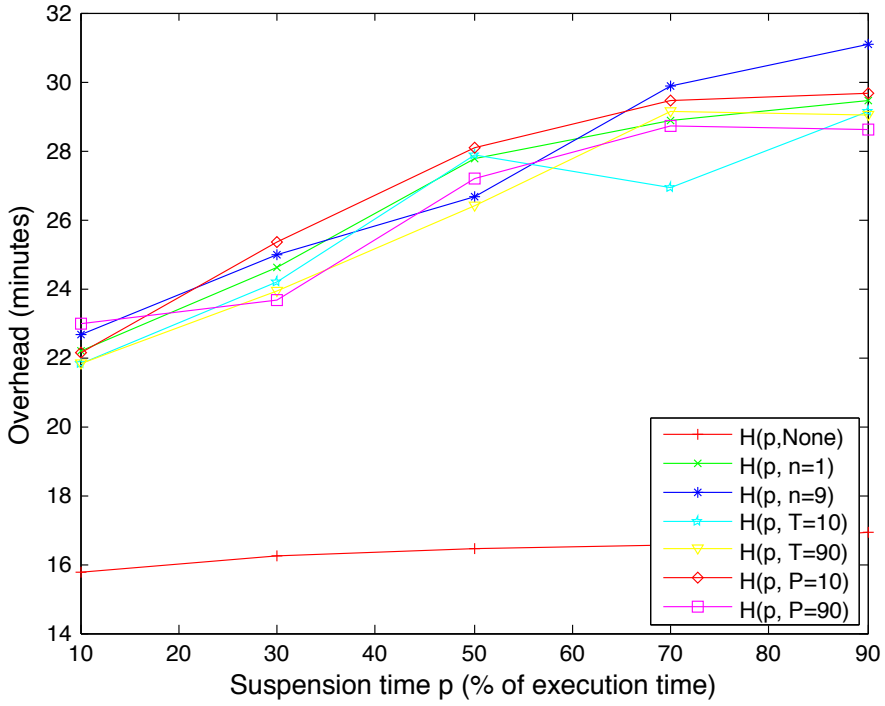

Fig. 5.11: The impact of Suspension percentage on overhead

C3(*40;6;60*) gives a good combination as it gives no good job kills.

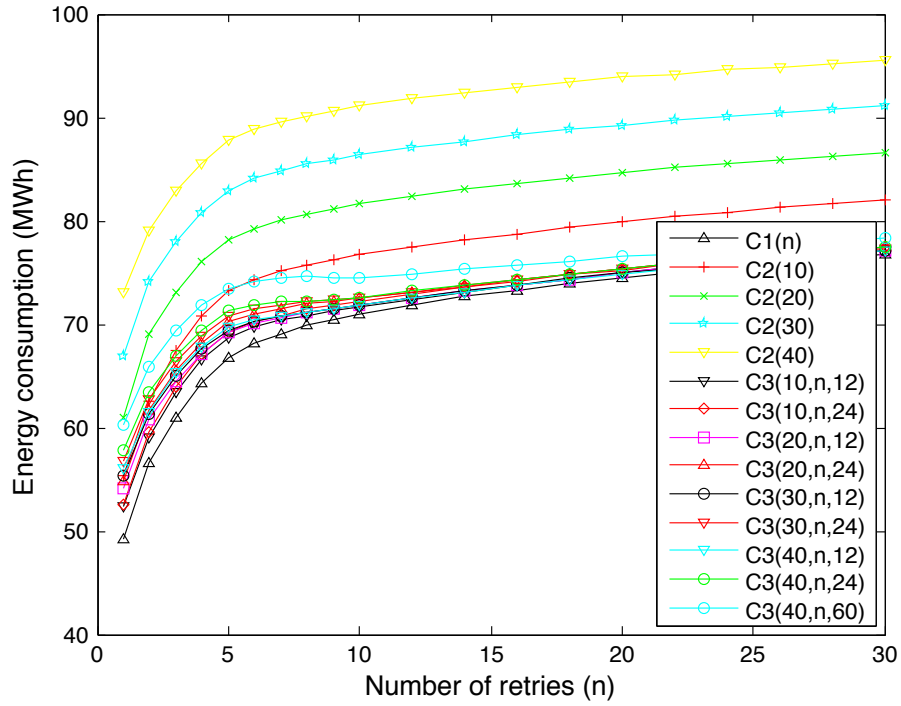

Fig. 5.12: The impact of Job Termination policies on energy consumed

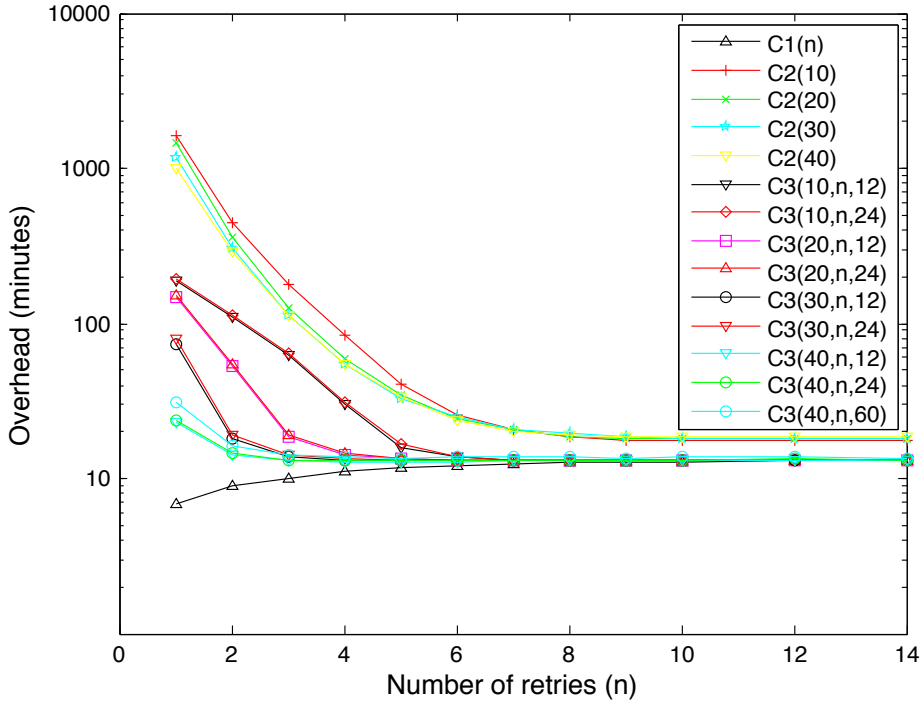

Fig. 5.13: The impact of Job Termination policies on overheads

5.4.6 Combined polices with synthetic jobs

Here we evaluate the 'best' policies identified above against larger (synthetic) workloads [161] derived from our workload trace from 2010 – over ten times our real work-

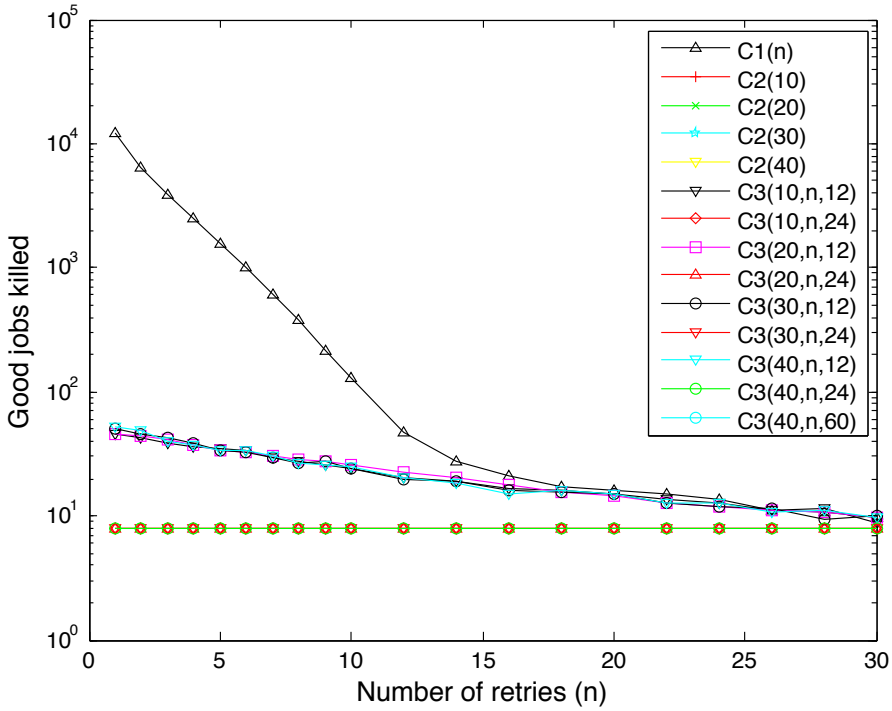

Fig. 5.14: The impact of Job Termination policies on 'good' jobs killed

load (~six million jobs). We have identified three power polices (**P3**(*15*), **P4**(*15*) and **P5**(*15,15*) along with the selection polices (**S2**, **S5**(*2*)), the management polices (**M1**(*10*) and **M2**) and the job termination policy (**C3**(*40,6,60*)). As the management polices are not mutually exclusive we use both simultaneously here. Thus the four policy combinations are:

com-1 {**M1**; **M2**; **P4**; **S2**; **C3**(*40;6;60*) }

com-2 {**M1** ; **M2**; **P5**; **S2**; **C3**(*40;6;60*) }

com-3 {**M1** ; **M2**; **P4**; **S5**(*2*); **C3**(*40;6;60*) }

com-4 {**M1** ; **M2**; **P5**; **S5**(*2*); **C3**(*40;6;60*) }

Figures 5.15 and 5.16 illustrate the effectiveness of these policies over different workloads. Note that termination policy **C3**(*40;6;60*) prevented any good jobs from being terminated. Although this is not guaranteed, the simulated workloads here exhibited this property.

All four policy sets scale consistently with increased workload with policy set **com-2** showing slightly worse performance in almost all cases. The power increase in all

cases is sub-linear – i.e. doubling the number of jobs does not double the energy consumed. However, overheads do increase in a greater than linear manner. Suggesting that a more stringent policy set for removing bad jobs would be beneficial for higher workloads.

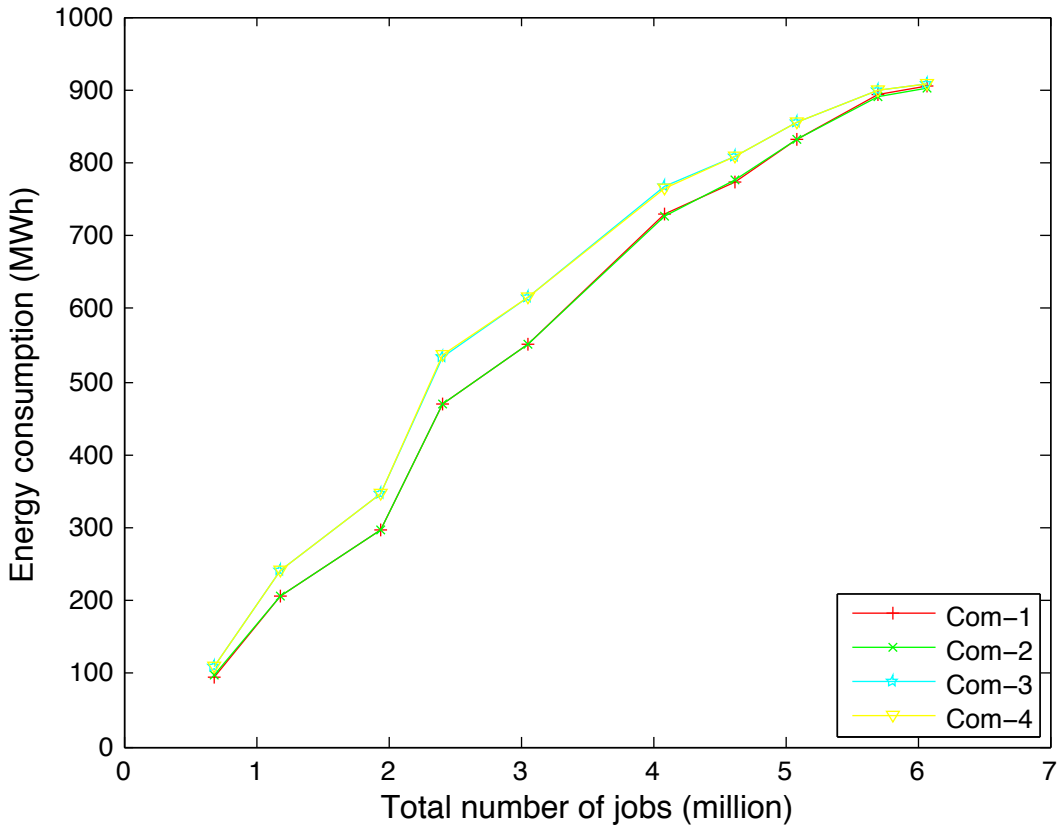

Fig. 5.15: The impact of Combined policies on energy consumed

5.5 Conclusion

The selection of an optimal set of policies for energy consumption across a multi-use cluster is complicated. Many policies have a significant impact on the power consumed, though also have a (detrimental) impact on the usability of the cluster for high-throughput users.

Power management polices **P4** (Computers sleep after n minutes of inactivity with HTCondor being made aware of their availability.) and **P5** (Computers sleep after m minutes of inactivity with sleeping computers being advertised every n minutes.) appear to be the most optimal polices to select. Whilst selection policy **S2** (Target the

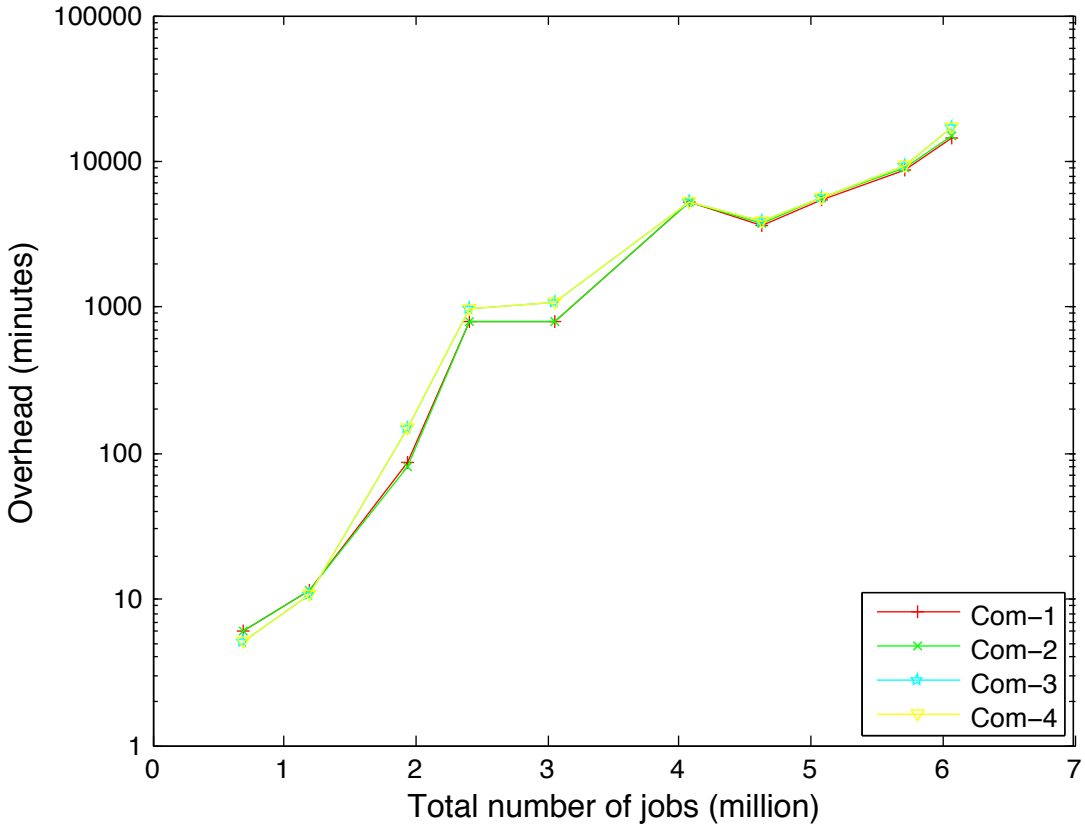

Fig. 5.16: The impact of Combined policies on overhead

most energy efficient computers.) has the greater impact on overhead and power consumption, with **S5**(*2*) being a close second. We anticipate for the interactive user usage and HTC workload observed by our system, that merging these policies could offer both advantages, though computing this ordering may be difficult. Though we obtain our greatest energy savings with policies **P3** (Computers sleep after n minutes of inactivity with no wakeup for high-throughput jobs.) and **P2** (Computers are on during cluster opening times or powered off otherwise with no ability to wake up.), we do not consider this policy as a good choice due to the significant resource starvation for high throughput jobs as evidenced in Figure 5.2.

Management policy **M1**(n) (A computer may not be used until it has been idle for n minutes.) appears to have little effect, though this could be masked since a significant proportion of time the cluster is closed to interactive users where **M1**(*0*) applies. Policy **M2** (High-throughput jobs defer nightly reboots.) has a significant impact in both power saving and job overhead, though **M3** (High-throughput jobs use computers at the same time as interactive users.) fails to provide a good reduction in energy and

increases overheads markedly. This is due to bad jobs continuing to run on resources blocking other jobs and wasting energy. The combination of this policy with a timeout interval similar to that of **C3**(m, n, t) (Timeout for dedicated computers.) could make this policy more attractive. Unfortunately suspension policy **H** (Hierarchical policies.) fails to deliver any significant benefit.

Changing polices for the cluster (dedicated resources for evictees, postponing reboots and allowing jobs to run on the same computers as interactive users) each show the potential to save power and reduce overheads for users. The best effect is likely to come from a combination of these policies. These policies have been combined and evaluated over larger (synthetic) workloads showing that they remain similar in benefit.

The main advantage with these polices comes as they are not mutually exclusive. Combinations of these polices can be produced increasing the overall energy savings without a significant impact on users of the high-throughput resources. The most significant energy saving that can be made is simply allowing computers to go to sleep when not needed. We have shown that changes to the cluster policy can further reduce energy consumption without significantly affecting the high-throughput users. For the Newcastle HTCondor pool and the offered workload we consider throughout this work, this can lead to an energy saving of ~65MWh, ~55% of the energy currently consumed by the high-throughput system. We anticipate workload size to be a significant factor affecting the generalisability of the energy savings we observe for the Newcastle HTCondor pool to other systems.

5.5.1 Further Work

Scheduling of parallel workloads Our workload trace does not currently contain parallel tasks but resource allocation in the context of parallel and mixed workloads is of significant interest in future work, evaluating the impact of previously proposed strategies such as dynamic partitioning [156] and gang scheduling [152] on the energy consumption of an HTC system.

Scheduling strategies Throughout the work presented in this chapter we model the

resource allocation process around the HTCondor [146] matchmaking [193] process. This process is an adaptation of a First In, First Out (FIFO) queueing discipline employed by numerous grid scheduling systems in FCFS (First Come, First Served) based algorithms. In future work we may explore a number of alternatives, e.g. Shortest Job First (SJF), data locality-aware scheduling [26], fair scheduling [72] and proportional-share resource allocation [85].

Under a Shortest Job First scheduling strategy [96], the workload is prioritised based on the size of the job and also the selected printer's anticipated service rate for the job. Though an accurate estimate of task execution may not be known *a priori* [16], in the event of batch submissions and jobs who have previously been allocated to a resource, a lower bound on execution time may be inferred.

Harchol-Balter *et al* suggest a rule of thumb for the partitioning of jobs into priority groupings [96], namely a lower cut-off such that $\frac{1}{2}$ of the workload is smaller, and an upper cutoff, above which 0.5-1% fall. Correct parameterisation of these cutoffs is required to correctly prioritise short-running tasks while preventing starvation of larger tasks.

Advanced Reservations and Backfilling strategies Some grid applications have particularly large resource requirements and require simultaneous access to these resources. The notion of *'advanced reservations'* have been shown to be beneficial for such applications [214]. This in turn necessitates Backfilling strategies to make use of the idle time on notes surrounding these reservations.

Deadline- and priority-aware strategies Jobs modeled within our system do not presently have user-specified deadlines or priority values, and these are not currently considered by our resource allocation strategies. Though resource allocation strategies would still strive to reduce overall average task makespan for the offered workload, particular emphasis would be placed on jobs of high priority or those which may exceed specified deadline. Furthermore, we intend to investigate policies considering contention between HTC users, promoting fair distribution of compute resources among HTC users, subject ot the aftorementioned

deadline and priority constraints.

Chapter 6

Reducing the number of miscreant tasks executions in a multi-use cluster

Summary

Exploiting computational resources within an organisation for more than their primary task offers great benefits – making better use of capital expenditure and provides a pool of computational power. This can be achieved through the deployment of a cycle stealing distributed system, where tasks execute during the idle time on computers. However, if a task has not completed when a computer returns to its primary function the task will be preempted, wasting time (and energy), and is often reallocated to a new resource in an attempt to complete. This becomes exacerbated when tasks are incapable of completing due to excessive execution time or faulty hardware / software, leading to a situation where tasks are perpetually reallocated between computers – wasting time and energy. In this chapter we investigate techniques to increase the chance of 'good' tasks completing whilst curtailing the execution of 'bad' tasks. We demonstrate, by extending the simulation presented in Chapter 4, that we could have reduce the energy consumption of the Newcastle University cycle stealing system in 2010 by approximately 50%.

6.1 Introduction

A key issue when using cycle stealing systems such as HTCondor [146], particularly within a multi-use cluster setting such as our own, that of ensuring that all 'good' tasks complete. We define a 'good' task as one which given enough time on a dedicated resource would run to a natural completion. Computers within the cluster can appear and disappear arbitrarily, the computers may be heterogeneous (or broken) making it difficult for tasks to execute correctly, or computers may have to preempt tasks in order to return to their primary role. Thus, if a task fails to complete on a given resource we

cannot assume that it is a 'bad' task.

To alleviate the effects of the system on task execution, an approach is adopted in which tasks that do not reach a natural completion are reallocated to a new resource. This leads to potential wasted energy from tasks repeatedly allocated to resources either because the task will never complete or the resource is incapable of satisfying task requirements (e.g. appropriate environment or a sufficiently long period for execution). This can be alleviated by limiting the number of resubmissions, though if the value is too low 'good' tasks, unfortunate in their allocation, will fail to complete whilst if the value is too high 'bad' tasks will waste time and energy. We define a *'miscreant'* task as one which exhibits multiple reallocation attempts and seek to minimise energy consumption by reducing the number of reallocations of 'bad' tasks whilst increasing the chance that 'good' tasks are reallocated to resources capable of servicing their needs. It should be noted that miscreant does not imply 'good' or 'bad', just that a task has required multiple reallocations.

Traditionally this has led to a trade-off between the number of failed 'good' tasks and overheads on task execution, with each organisation selecting a local optimal – an open question which received significant discussion at Condor Week 2012 [230]. However, due to energy conservation – now a more important criteria – this has become a three-way problem. Reducing reallocation attempts reduces energy consumption through removal of 'bad' executions, though increases 'good' tasks failures.

Tasks can be allocated to resources whilst they are idle or sleeping (through Wake on LAN) executing until the resource is required for its primary purpose (interactive user, system maintenance or reboot). This would suggest that the 'best' option is to have tasks shorter than the intervals between primary use and/or only run task during expected long periods of primary inactivity (e.g. overnight). However, such a policy would require unrealistically short execution times and would incur significant delays in task execution.

In this chapter we investigate a number of policies for curtailing 'bad' executions whilst still minimising the number of 'good' task terminations and the average task overhead – allowing us to minimise energy consumption.

The rest of the chapter is set out as follows. In Section 6.2 we discuss the job characteristics and circumstances under which a task may be deallocated in an HTC system. Section 6.3 extends our analysis of our Newcastle University 2010 trace datasets presented in Chapter 4, focusing on resource reliability and the impact of task deallocation on energy consumption and makespan. Section 6.4 describes policies aimed at identifying which miscreant tasks should be re-run and which should be terminated. Section 6.5 presents the simulated results for these different polices, before concluding in Section 6.6.

6.2 Task Deallocation

Tasks may become deallocated from the resource they were previously allocated to for several reasons:

Task preemption: Condor has decided to deallocate the task. Condor [51] identifies four preemption cases: *i*) Higher priority task is identified which will start once this task has been preempted; *ii*) Policy of the resource – this can include an interactive user logging in or a pre-defined time during when tasks can't run; *iii*) Resource ranking – the resource determines a more appropriate task to execute (e.g. a maths department owned computer preempts non-maths tasks for maths tasks); *iv*) Condor is shutting down – during shut-down Condor will preempt running tasks. Many managed clusters have a regular shutdown policy allowing updates and resetting. These preemptions will mark a task as miscreant though none indicate the tasks is 'bad'.

Hardware / Software failures If a resource becomes unreachable by the system for an appropriate interval it will be deemed no longer part of the pool. This can be for a myriad of reasons including hardware failure, Operating System failure, catastrophic software failure (including the running task) or network failure. Note that these may be transient in nature. Again none of these issues implies that the running task was 'bad'.

Although the above indicate under what circumstances a task is deallocated from a resource they don't distinguish whether the task could complete on a subsequent execution. In all cases the task is deallocated before it has reached its own natural exit point. The reason for this can be:

Execution time longer than time available: The time between allocation and deallocation, t_r, is less than the task execution time. If t_r is small the task is likely to complete on a new allocation, whilst if t_r is close to the maximum time available then it is most likely to be deallocated again. Note that the task may be 'good' but require more time than the system can provide.

Code has malfunctioned: The code crashes but does not terminate (infinite loop, awaiting user interaction) remaining active until deallocated. Re-running the task is unlikely to change this scenario. Reducing the chance of a re-run here is highly desirable.

Hardware / Software malfunction: A fault in the environment causes the task to fail to terminate (e.g. broken library, CPU failure). Reallocating to a different resource is likely to allow the task to complete.

Task requirements not satisfied: Although many failures in task requirements would prevent the task from starting or fail upon starting, there are circumstances where an apparent code malfunction would occur. However, in this case allocation to a new computational resource could resolve these requirements.

This problem becomes exacerbated by the fact that it is not possible to distinguish easily these cases from each other. A piece of code which malfunctions and is deallocated after only a few minutes exhibits the same properties as a 'good' task which is also evicted after only a few minutes. Hence the use of the term 'miscreant' indicating that, although not definitively 'bad' tasks, the task is behaving in a manner which is not desirable. An assumption could be taken in which any task failing to complete on the first attempt is abandoned by the system; however, this will lead to a significant number of 'good' tasks being terminated, though this will reduce energy consumption

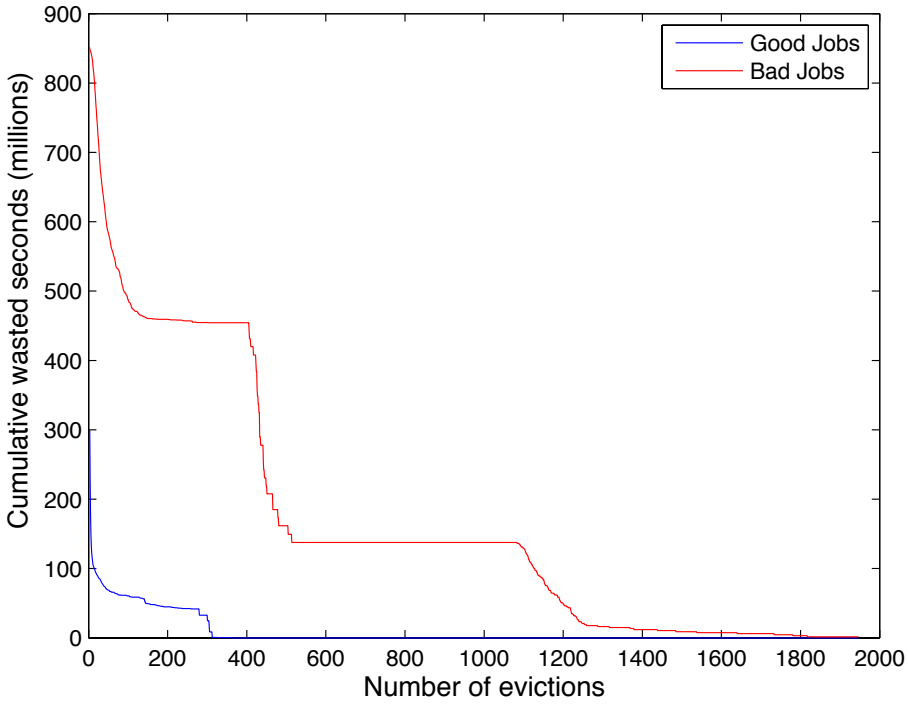

Fig. 6.1: Graph of total wasted time against evictions

and the overheads on those tasks which complete, as 'bad' tasks will not be consuming resources. Alternatively allowing miscreant tasks to be re-run an arbitrary number of times allows 'bad' tasks to consume significant amounts of energy and increase the overheads for all tasks due to bad tasks consuming resources. Historically, this problem has only been considered in terms of the metrics of overhead and number of 'good' tasks being terminated with the administrators of HTC system deployments selecting a value for the number of retries which keeps the number of 'good' tasks terminated to an acceptable level and keeps the overheads to a reasonable level. Although it is desirable to obtain the right balance for these metrics there is little penalty for not getting the balance right. Including energy as the third metric thus imposes a significant penalty for wasting computational resources.

6.2.1 Definitions

There exists a clear need for precise definitions of what constitutes *'good'*, *'bad'* and *'miscreant'* tasks. These are provided below.

'Good' tasks We define a 'good' task as one which given enough time on a dedicated resource would run to a natural completion.

'Bad' tasks We define a 'bad' task as one which is subsequently removed by the submitter of the job, or the system administrator.

'Miscreant' tasks We define a *'miscreant'* task as one which exhibits multiple reallocation attempts. This task may be a *'good'* task which has been unfortunate in its resource allocation, or may be a *'bad'* task which will never reach a natural completion.

6.3 Analysis of the Newcastle Condor System

Here we further our analysis of the Newcastle University HTCondor pool presented in Section 4.3.1, with an emphasis on wasted execution. Here we investigate the two implicit policy assumptions made by many high-throughput cluster managers. In general it is assumed that a (fairly low) value for reallocations will allow the vast majority of 'good' tasks to be completed and that choosing a small enough task duration will allow the majority of 'good' tasks to complete without reallocation.

Newcastle University has been running a largely unmanaged HTCondor pool since October 2005 [157]. We analyse the tasks from 2010 in order to exemplify the effects of miscreant tasks on the cluster and to address the two assumptions. In total 561,851 tasks were submitted through HTCondor consuming 1,684,940,087 seconds (~53 years), of which 1,218,729,685 (~39 years) was wasted. This wasted time comprised 851,989,414 seconds (~27 years) for the 4,729 tasks which were subsequently killed by the user – 'bad' tasks – and 366,740,271 seconds (~12 years) wasted on the 557,121 tasks which did complete – 'good' tasks. Although it is not possible to determine, from our trace data, the time consumed by each unsuccessful allocation of a task terminated by the user the total time for tasks with at least one deallocation is relatively close to the total wasted time for killed tasks (849,725,325 seconds, ~27 years). Thus indicating most 'bad' tasks accrued at least one reallocation. For 'good' tasks this is only the wasted time, thus all of these tasks have accrued at least one reallocation.

Although a maximum number of reallocations can be specified in HTCondor this property was not activated in the Newcastle cluster in 2010. Figure 6.1 shows the max-

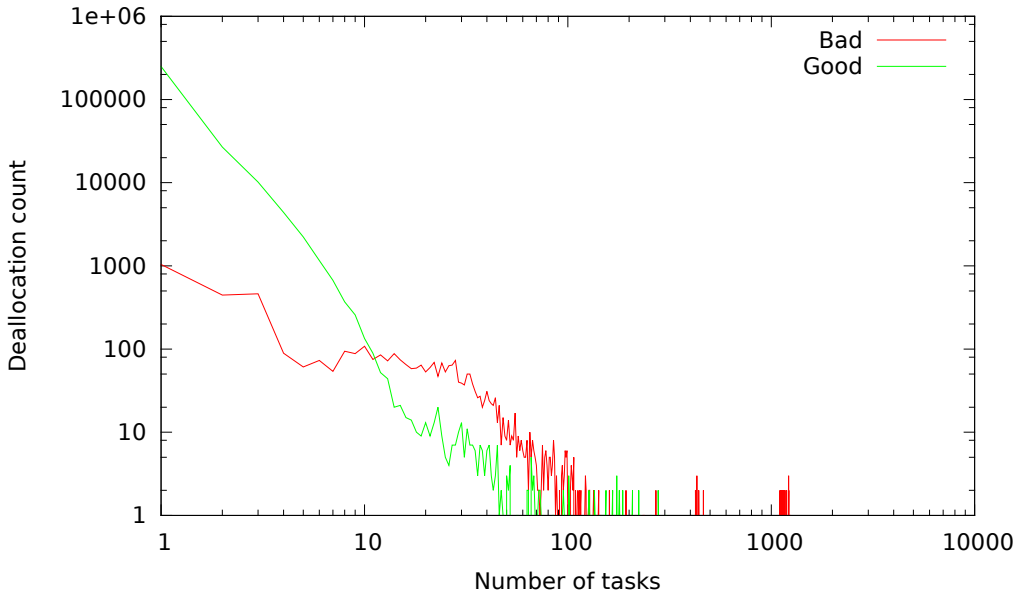

Fig. 6.2: Histogram of good task evictions

imum number of retries for 'bad' tasks was 1,946, whilst the maximum for 'good' tasks was 360. The average number of retries for 'good' and 'bad' tasks was 1.20 and 44.89 respectively. Figure 6.1 also illustrates that the majority of wasted time is associated with low eviction counts. It should be noted that HTCondor history does not explicitly record the number of times a task ran on a resource but the number of times that the task was allocated, thus resource state changes could cause a task to be deallocated before execution starts. However, as we are interested here in the number of times a task is allocated these rapid deallocations can simply be ignored as fortuitous in terms of energy consumption. Thus to ensure all 'good' tasks are successful we need a maximum reallocation count of 360. Reducing wasted 'bad' task time to 395,373,483 seconds (~13 years). It should be noted that this does not take into account the effect that changing the policy would have on the operation of the cluster or the way users would interact with the cluster.

Figure 6.2 illustrates the number of 'good' and 'bad' deallocations. In both cases the average number of deallocations is relatively low (1.38 and 44.89 respectively). In order to ensure 95% 'good' task completion we need a reallocation maximum of three, whist for 99% we need a threshold of 6 – this matches nicely with the intuitive value quoted by many cluster managers. However, a maximum of six reallocations would mean 2,022 'good' tasks failures, though reducing wasted time on 'bad' tasks to 7,534,050 seconds

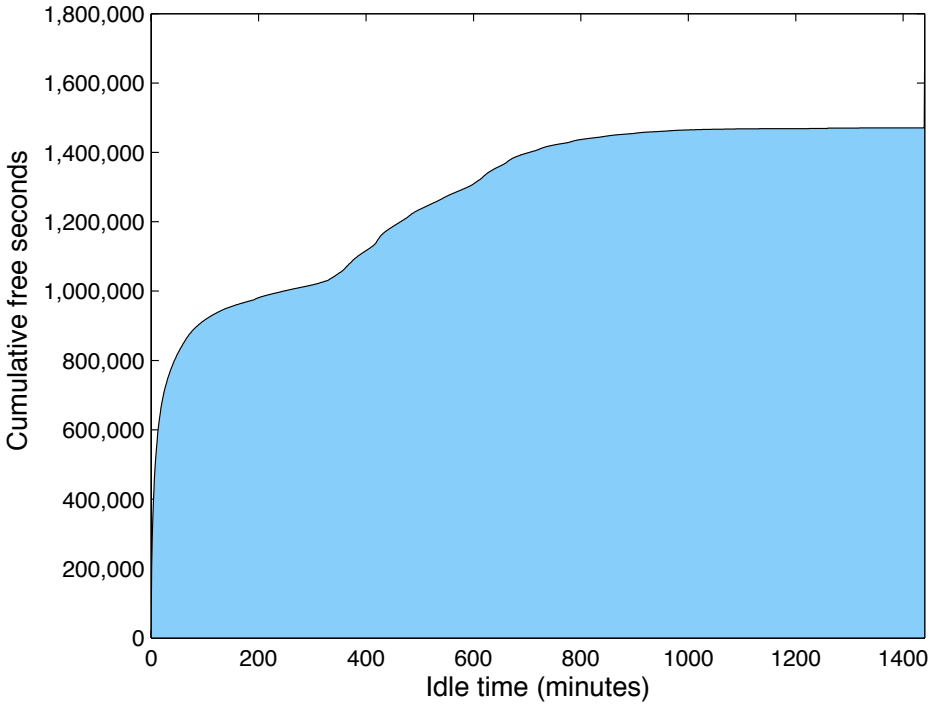

Fig. 6.3: Cumulative idle time

(~87 days).

Figure 6.3 illustrates the idle intervals for computers, defined as the time between a user logging off and the next user logging in. The average idle length is 371 minutes. A task would need to be no longer than two minutes to ensure that 95% of idle intervals are long enough, whilst it would have to be no longer than one minute to ensure that 99% of intervals are long enough. This is clearly unobtainable.

We can clearly reduce the wasted time in a cluster by reducing the number of reallocations at the expense of failing to complete 'good' tasks. Hence we need a better approach in which 'good' tasks which have been unfortunate in their allocations are reallocated whilst 'bad' tasks are curtailed.

6.4 Policy for handling miscreant tasks

We evaluate a number of policies to identify and handle miscreant tasks. These policies govern the circumstances under which deallocated tasks are abandoned or reallocated. Policies are divided into three groups; those governing resource selection, those determining the use of dedicated resources, and those used to identify miscreant tasks.

6.4.1 Baseline policy

X0: There is no limit on the number of times a task can be reallocated, with termination only occurring if the task is removed by the submitter or an administrator.

6.4.2 Computer selection policy

C1: Tasks are allocated to resources at random, favouring awake resources. This represents the HTCondor default policy.

C2: Targeting less used computers [160]. By selecting resources with longer idle times between users reduces the chance that a task will be deallocated due to preemption.

C3: tasks are allocated to computers in clusters with the least amount of time used by interactive users. This reduces the chances of task preemption and exploits the less popular clusters around campus. Computers can be ranked using:

$$Rank(c) = \frac{\sum_{s}^{s \in c} s_{idle} / s_{total}}{|c|}$$

where c is the set of computers in a cluster, s is a computer in c, s_{idle} is the total idle time on computer s, and s_{total} is the total time for computer s.

A detailed evaluation of the impact of computer selection policies on energy consumption and average task makespan is presented in Chapter 5.

6.4.3 Dedicated resources

D1(m, d): Tasks identified as miscreant are permitted to continue executing on a dedicated set of m computers. Tasks running on these dedicated resources are not susceptible to interruption through interactive users arrival or reboots. A maximum execution duration d prevents the task from running indefinitely.

6.4.4 Miscreant task identification

Conventional *n* reallocation policies do not distinguish the causes of deallocation, thus are poorly suited to the multi-use cluster context. Evictions due to the arrival of interactive users and planned machine reboots do not in any way imply a task to be miscreant. We propose two variations on **N1** which discount these evictions from a task's reallocation count:

N1(n): Termination after n reallocations. If a user still believes that the task is good they may resubmit it. This represents the HTCondor default policy for reallocation.

N2(n): A task will be abandoned if it deallocated n times, ignoring deallocations due to interactive users.

N3(n): A task will be abandoned if it is deallocated n times, ignoring deallocation due to computer reboots.

We present a number of random policies to allow for comparison:

R1(p): a task is abandoned with probability p ($0 \leq p \leq 1$).

A deallocated task j is retried according to exponential function $P(f) = (1 - e^{-kf}), 0 \leq k \leq 1$, where k is a scaling factor:

E1($f = n$): Exponential decay on deallocation count n.

E2($f = t$): Exponential decay on the total *accrued time* from all executions.

Tasks are subject to an upper bound t on their cumulative execution time, and are abandoned if deallocated and over this bound. Furthermore, we investigate the impact of discounting deallocations due to interactive users and reboots from a task's accrued execution time:

A1(t) Abandon if accrued time $> t$ and task deallocated.

A2(t): Abandon if accrued time $> t$ and task is deallocated, discounting deallocations due to interactive users.

Policy	Overheads	Power	Good tasks killed
X0 C1	20.03 minutes	137.54 MWh	0
X0 C2	15.05 minutes	123.58 MWh	0
X0 C3	15.77 minutes	117.43 MWh	0

Table 6.1: Miscreant tasks policies: Baseline Results

A3(t): Abandon if accrued time > t and tasks is deallocated, discounting deallocations due to reboots.

I1(t): Abandon if individual time > t. Nightly reboots bound individual execution times to 24 hours. We investigate the impact of lowering this threshold.

By leveraging historical information it is possible to more closely identify 'bad' tasks by looking at the percentile values for different properties:

P1(n, p): Abandon if $f(n) > p$.

P2(t, p): Abandon if $f(accruedtime) > p$.

where $p \in [0, 100]$ and function $f(p)$ estimates the value y of the p-th percentile using the linear interpolation, as presented in [232]:

$$y = f(p) = y_1 + \frac{p - x_1}{x_2 - x_1}(y_2 - y_1) \tag{6.1}$$

where $(x_1,\ y_1)$ and $(x_2,\ y_2)$ are data points such that $y_1 = f(x_1)$ and $y_2 = f(x_2)$ and $x_1 < p < x_2$.

6.5 Simulation results

Table 6.1 depicts the results for running the baseline case of no abandonment policy (**X0**) against the three resource selection policies (**C1**, **C2**, **C3**) with the two usage-based selection policies **C2** and **C3** providing a better overhead for tasks though providing alternative optimalities for energy or overhead between themselves. It should be noted that as all of these policies have no limit on the number of reallocations of tasks this leads to zero 'good' tasks being killed.

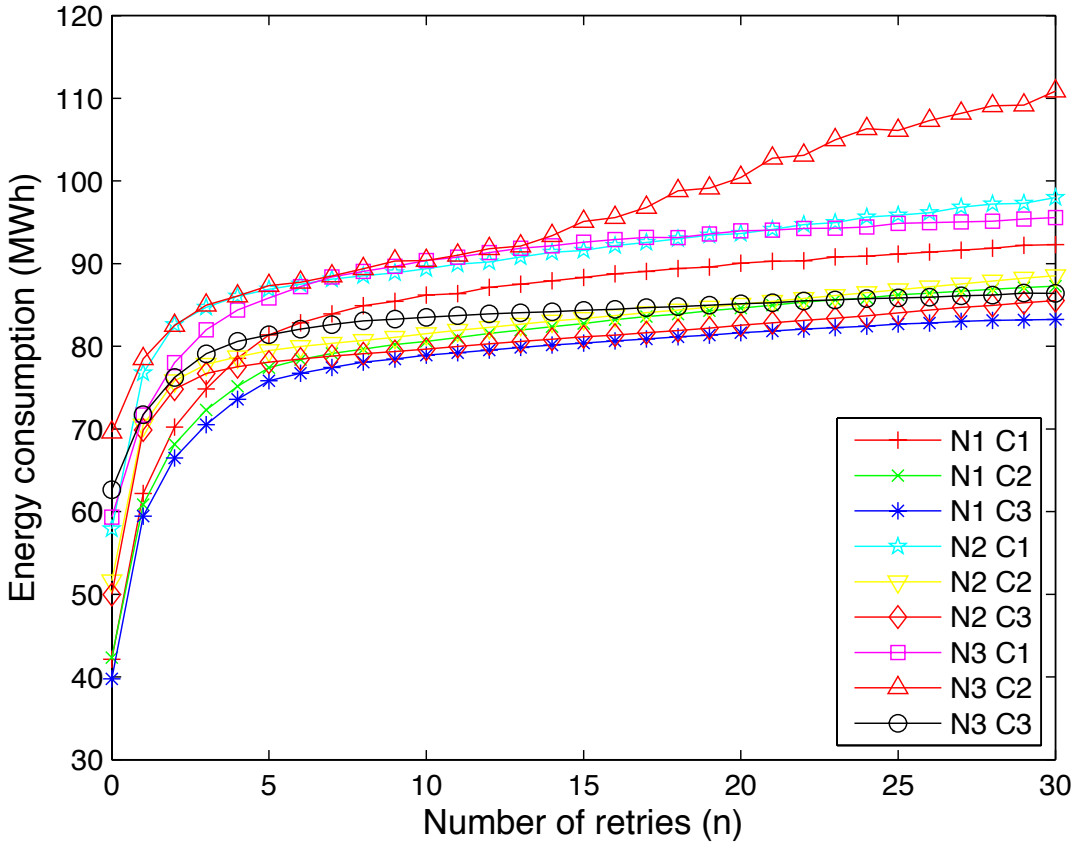

Fig. 6.4: The impact of Terminate after N allocations policy on Energy consumption

In the rest of this section we compare the energy, good tasks killed and overheads for all policies. Although not illustrated here the cost for these polices can easily be derived by multiplying energy by the cost per unit. In some cases the key has been omitted from a graph for clarity; for these the key on the other graphs in the set can be used.

Figures 6.4, 6.5 and 6.6 illustrate the differences between abandonment polices **N1**(n) (Termination after n reallocations.), **N2**(n) (A task will be abandoned if it deallocated n times, ignoring deallocations due to interactive users.) and **N3**(n) (A task will be abandoned if it is deallocated n times, ignoring deallocation due to computer reboots.) along with selection polices **C1**, **C2** and **C3**. Policy **C3** (Tasks are allocated to computers in clusters with the least amount of time used by interactive users.) gives a significant improvement for 'good' tasks terminated when the value of n is small. Selection policy **C2** (Targeting less used computers [160].) and **C3** (Tasks are allocated to computers in clusters with the least amount of time used by interactive users.) work

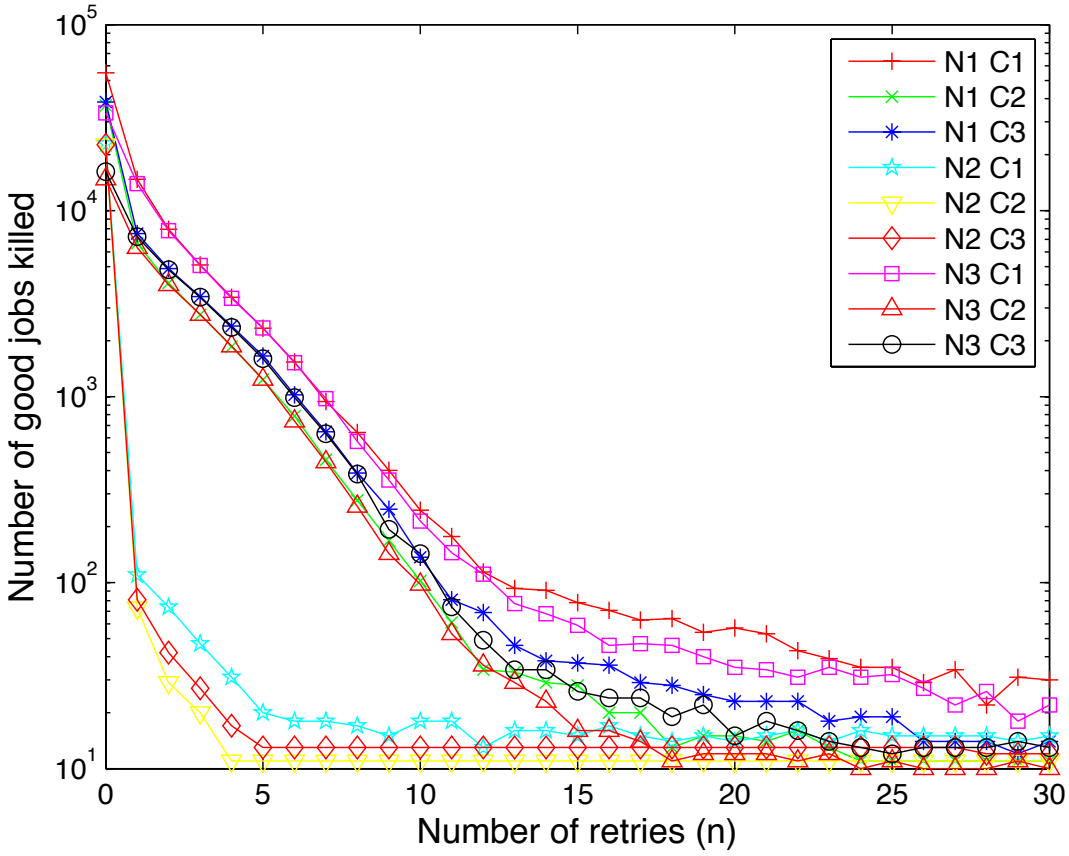

Fig. 6.5: The impact of Terminate after N allocations policy on Good tasks killed

well with this by keeping the energy levels and overheads low.

The individual time accrued policy **I1**(t) (Abandon if individual time > t) is explored in Figures 6.7, 6.8 and 6.9. This policy gives better energy performance and overheads in comparison with **N1**(n) (Termination after n reallocations.). However, this is at significant impact on the number of 'good' tasks which are terminated. It should be noted that the steep step in energy in Figure 6.7 corresponds with an individual execution time of 24 hours. This effectively allows the task to run indefinitely.

Exponential abandonment polices **E1**($f = n$) (Exponential decay on deallocation count n.) and **E1**($f = n$) (Exponential decay on deallocation count n.) are shown in Figures 6.10, 6.11 and 6.12. Note the scaling factors for these have been adjusted to allow both data sets to be drawn on the same graph **E1** needs to be scaled by 10^{-3} and **E2** by 10^{-10}. Although for this policy energy and overheads fall as the growth factor increases the number of good tasks terminated increases and from a high initial value (~4,500). Likewise for the random selection policy **R1** (a task is abandoned with prob-

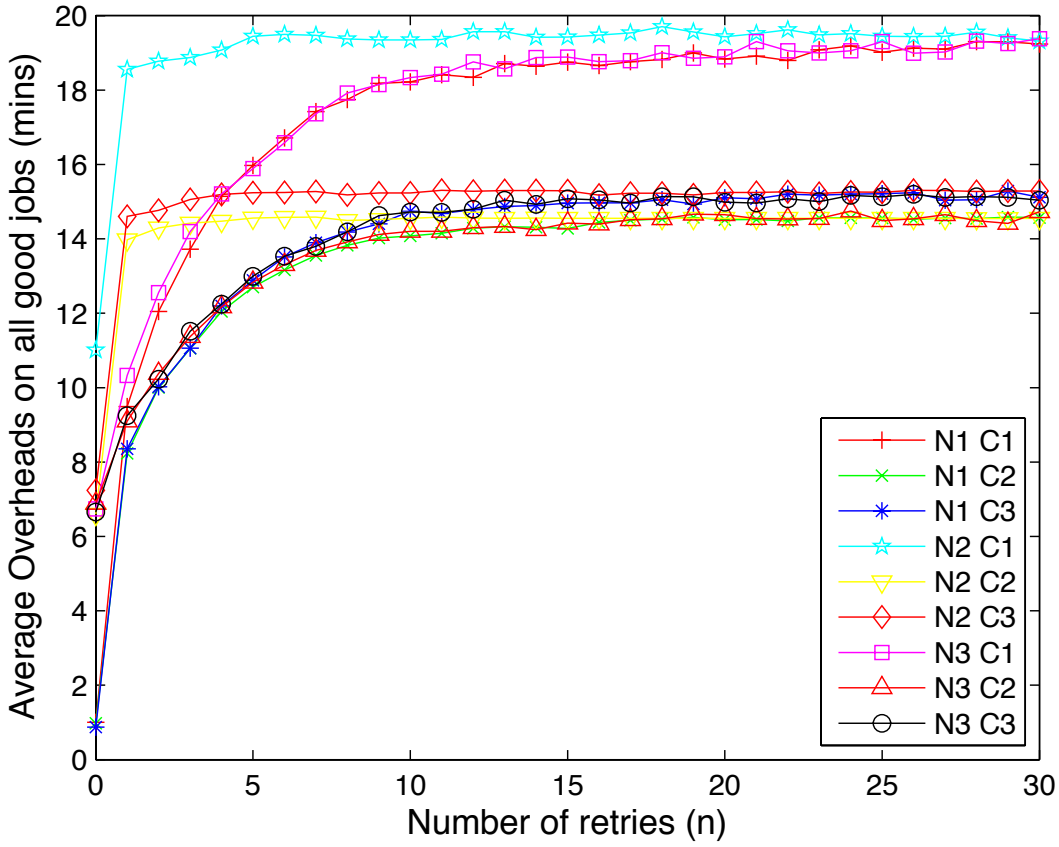

Fig. 6.6: The impact of Terminate after N allocations policy on Overheads

ability p ($0 \leq p \leq 1$)) – Figures 6.13, 6.14 and 6.15 the number of 'good' tasks killed is high and increases despite offering good overheads and energy results.

The policy of dedicated resources **D1**(m,d) (Tasks identified as miscreant are permitted to continue executing on a dedicated set of m computers) is explored in Figures 6.16, 6.17 and 6.18. There is a significant advantage here for energy in keeping the number of retries (n) low, but the other factors (dedicated resources and maximum dedicated time) have relatively small impact on the energy consumed. This is due to the maximum run time increasing on the dedicated resources as a consequence of increasing these factors. Only the maximum dedicated time has an impact on the number of good tasks killed. A dedicated maximum execution time of ~90 hours then allows for zero 'good' task terminations with little effect on the overall overheads, though the overheads are in general poor. Note that dedicated resources are assumed to use the same energy as our top-end computers, with further savings possible by consolidating multiple tasks onto a single dedicated resource.

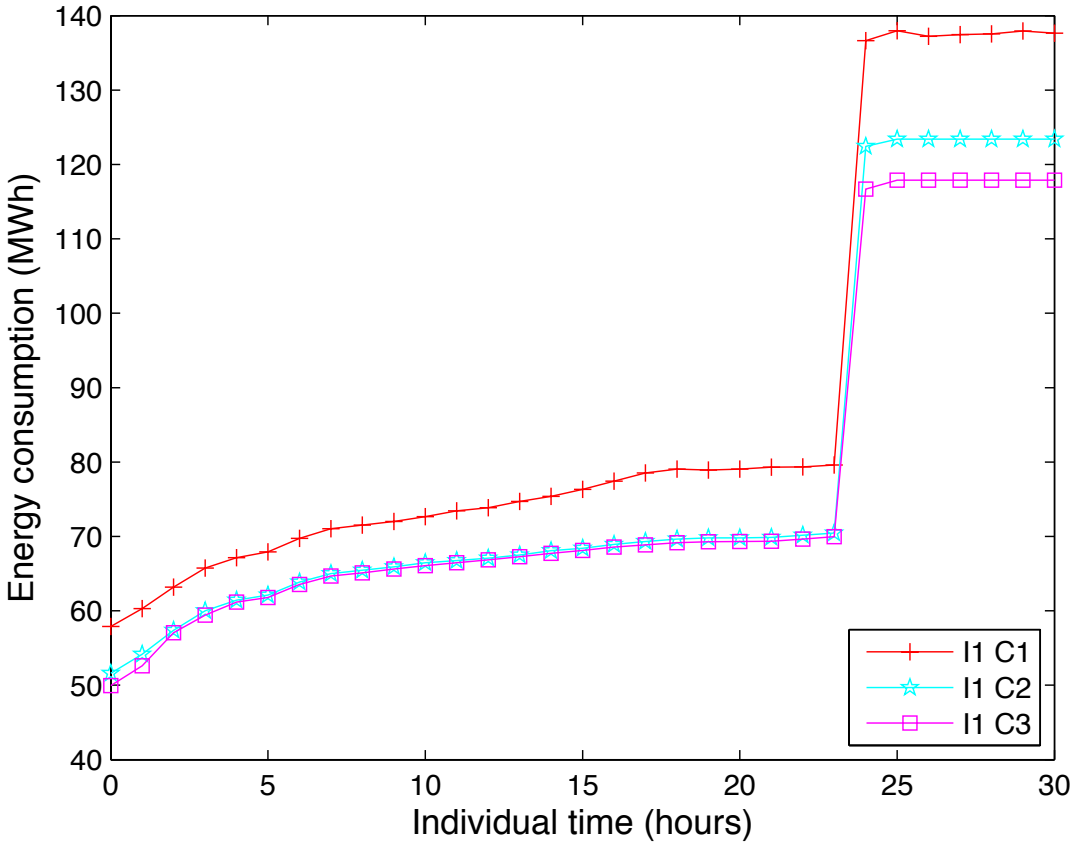

Fig. 6.7: The impact of Individual policy on Energy

Accrued policy **A1**(t) (Abandon if accrued time > t and task deallocated), **A2**(t) (Abandon if accrued time > t and task is deallocated, discounting deallocations due to interactive users) and **A3**(t) (Abandon if accrued time > t and tasks is deallocated, discounting deallocations due to reboots) are explored in Figures 6.19, 6.20 and 6.21. Low accrued times offer lower energy consumption at the expense of 'good' tasks killed. Apart from the combination '**A3 C2**' there is no significant advantage in selecting an accrued total over ~40 hours.

The percentile policies depicted in Figures 6.22, 6.23 and 6.24 show that the consumed energy comes down to an equivalent level as the other polices, however, only as the percentile tends to 100% do the number of 'good' tasks terminated reduce significantly. In order to benefit from using policy P2 the percentile needs to be almost exactly 100% giving little advantage over policy N1. Whilst policy P1 benefit as low as 90%.

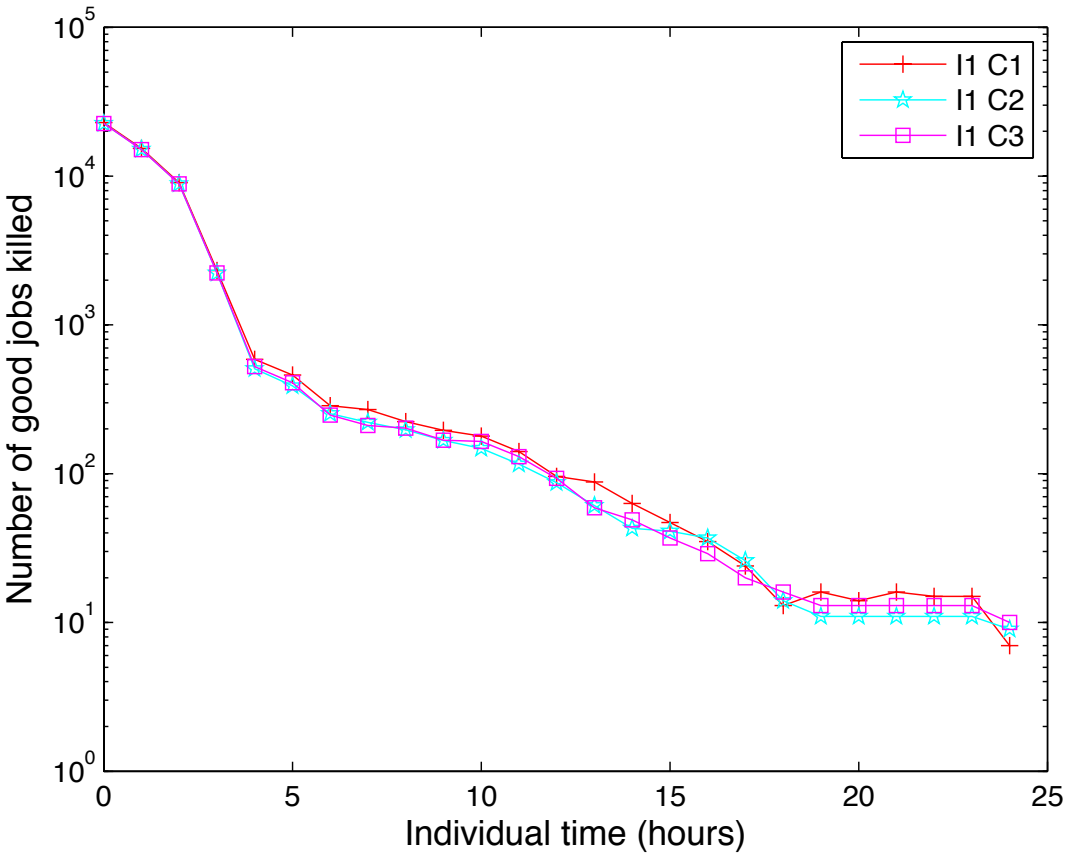

Fig. 6.8: The impact of Individual policy on Good Jobs Killed

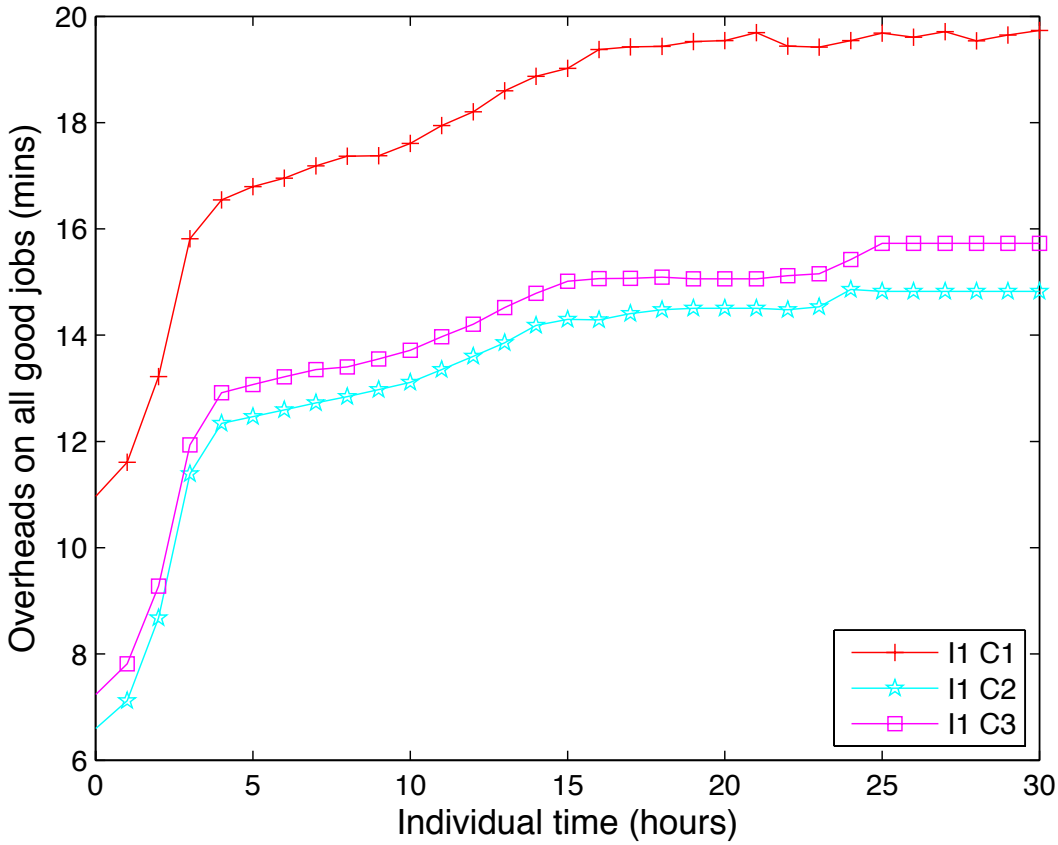

Fig. 6.9: The impact of Individual policy on Overheads

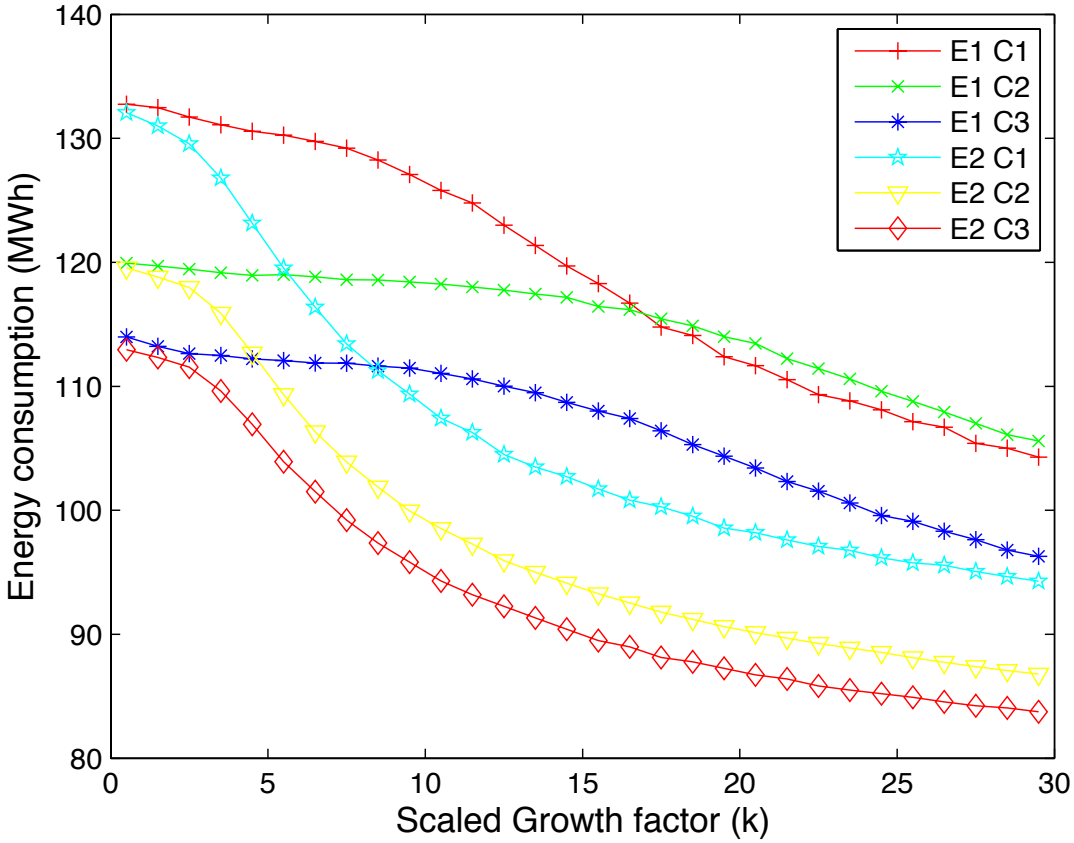

Fig. 6.10: The impact of Exponential policy on Energy

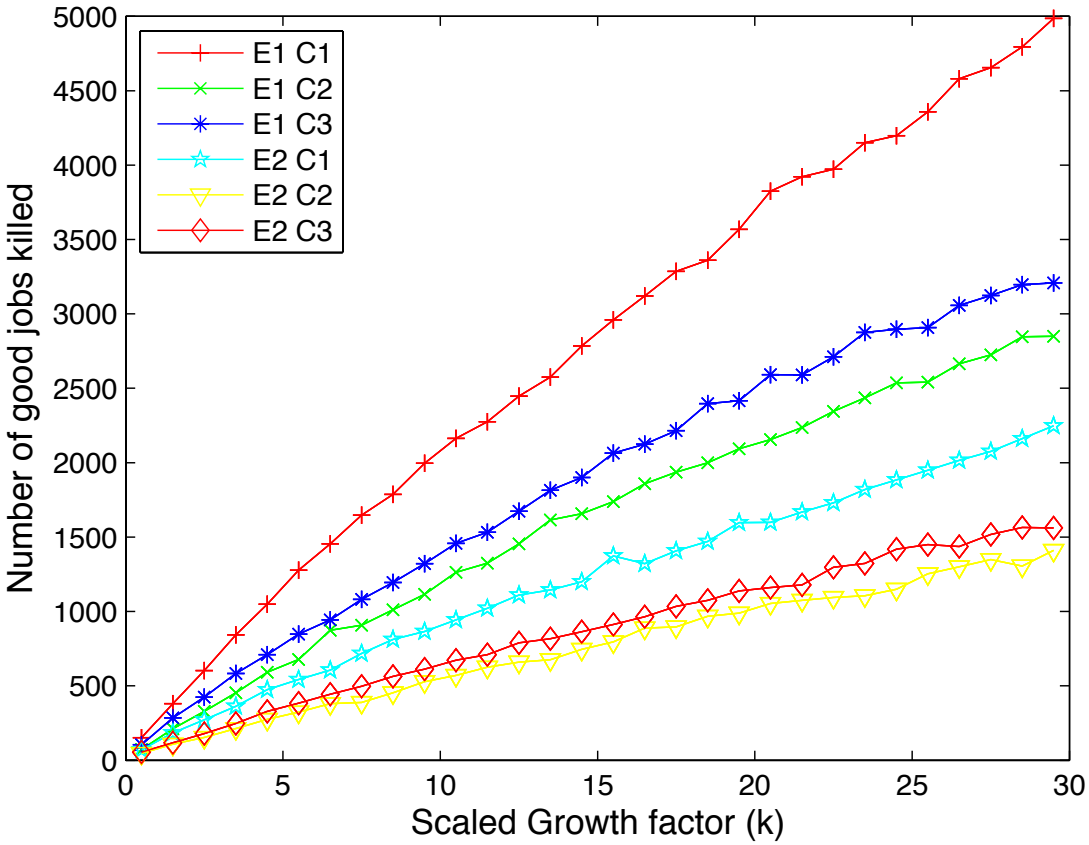

Fig. 6.11: The impact of Exponential policy on Good Jobs Killed

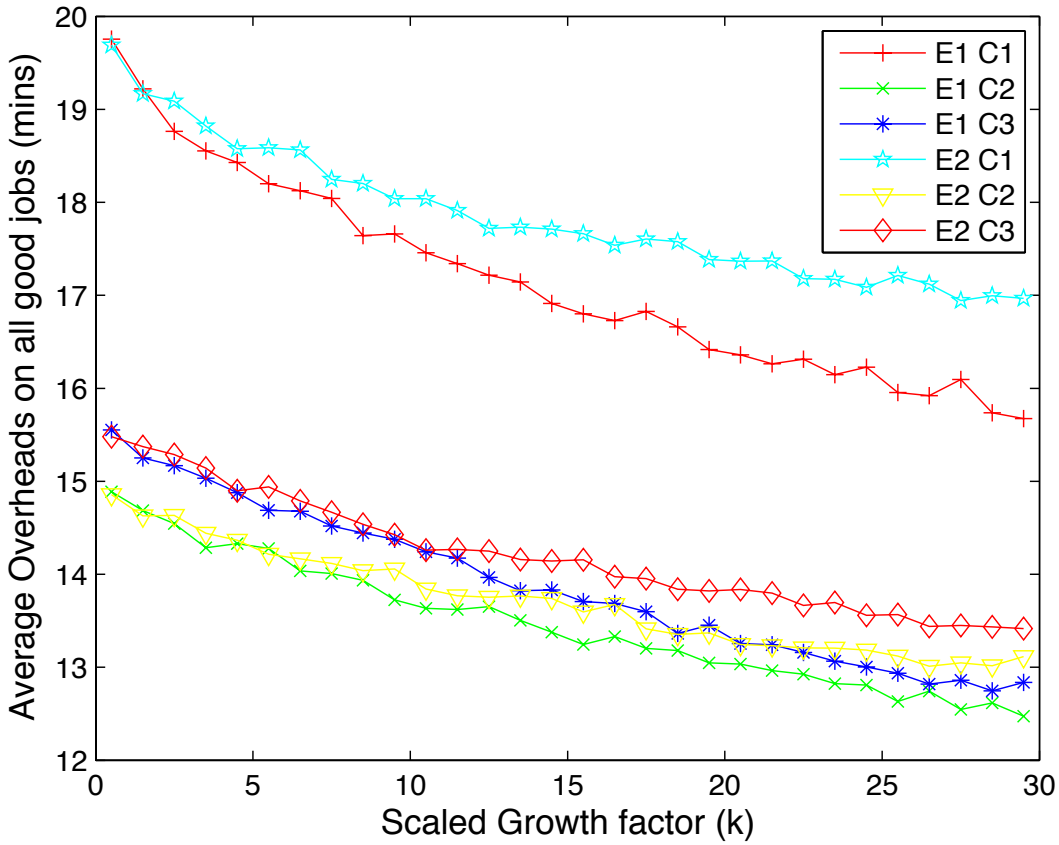

Fig. 6.12: The impact of Exponential policy on Overheads

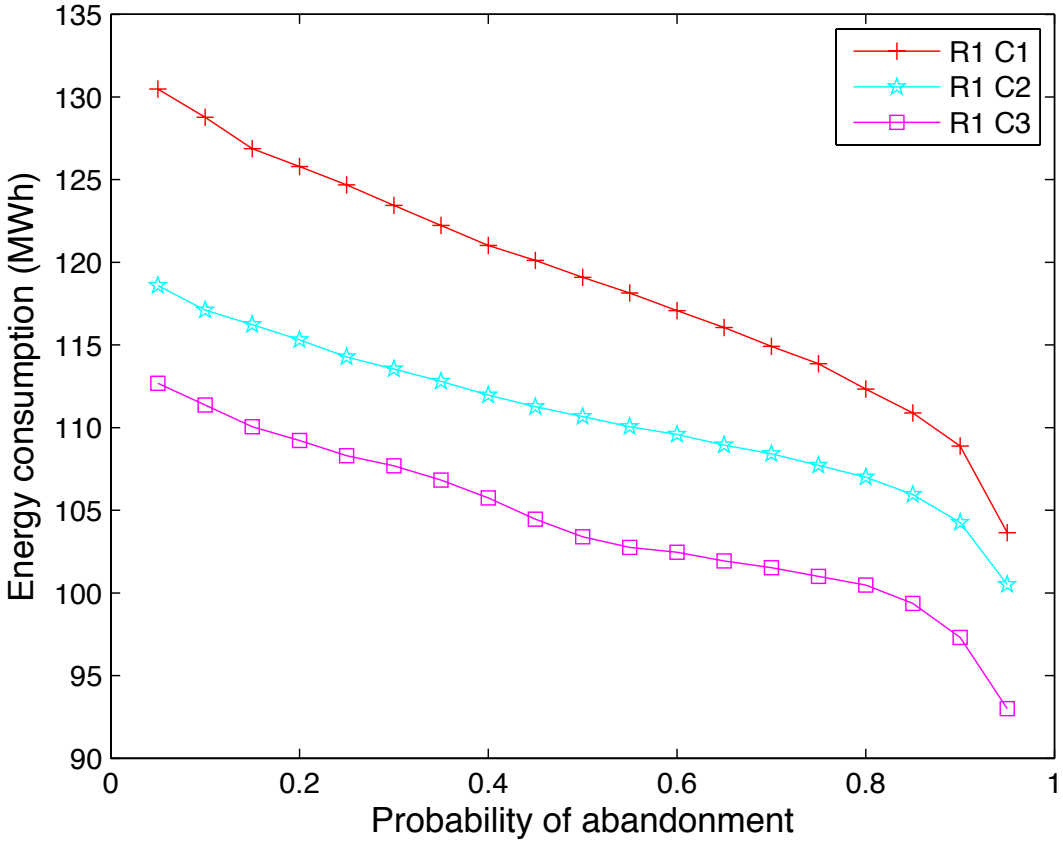

Fig. 6.13: The impact of Random policy on Energy

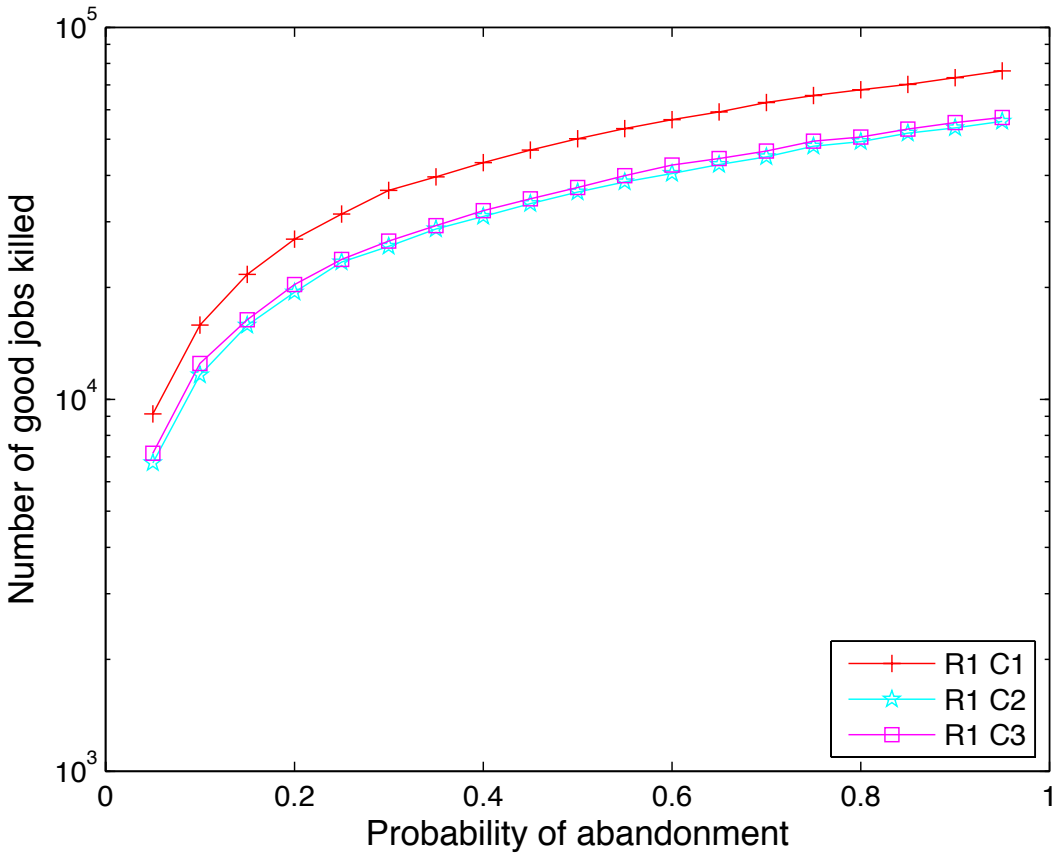

Fig. 6.14: The impact of Random policy on Good Jobs Killed

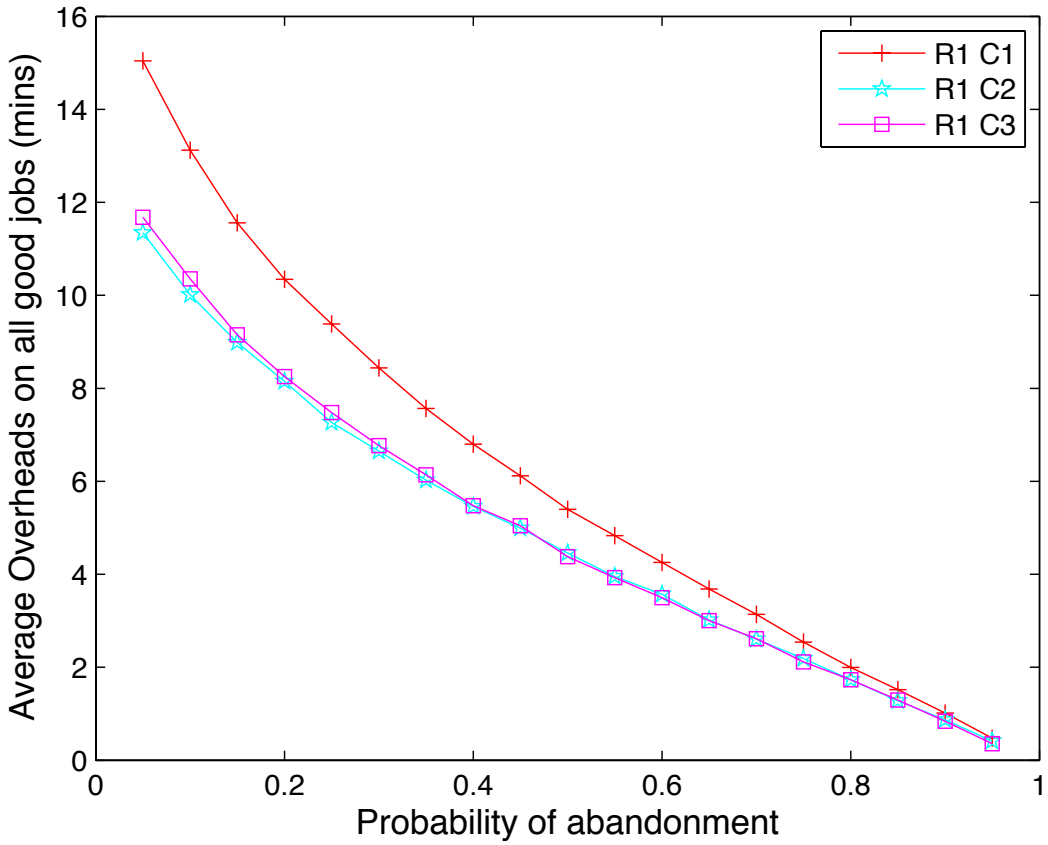

Fig. 6.15: The impact of Random policy on Overheads

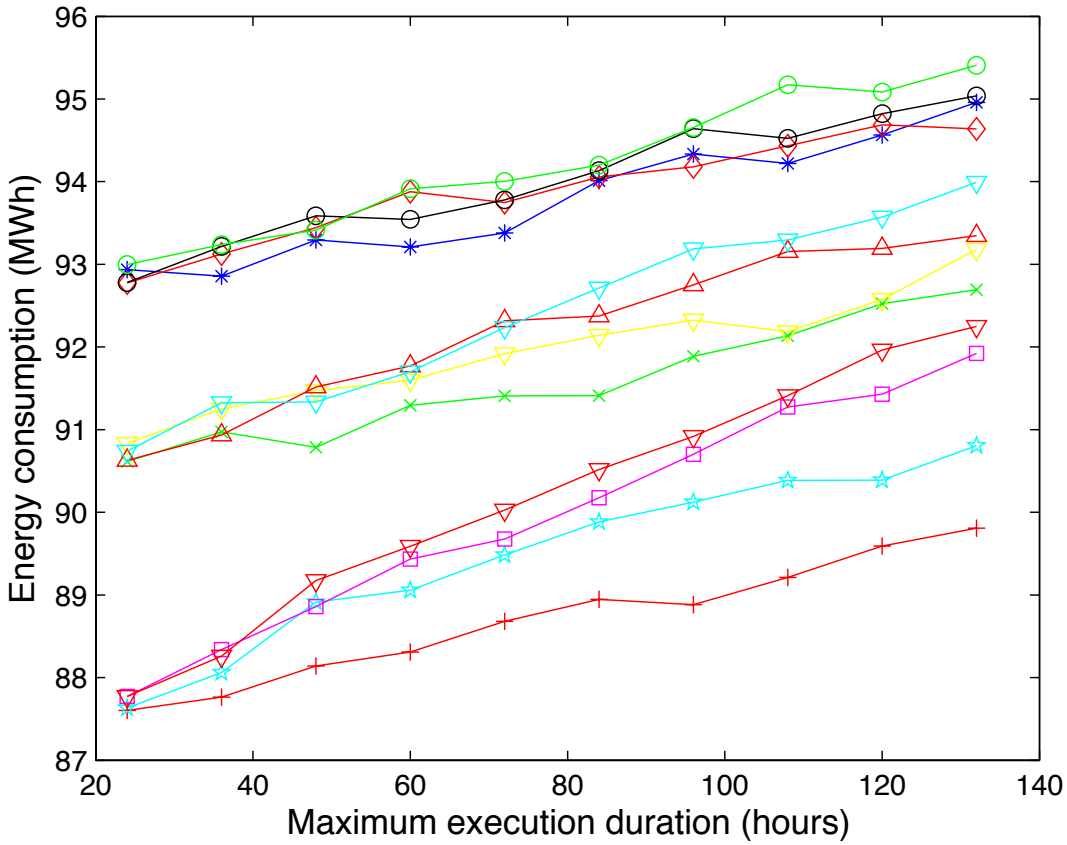

Fig. 6.16: The impact of Dedicated policy on Energy

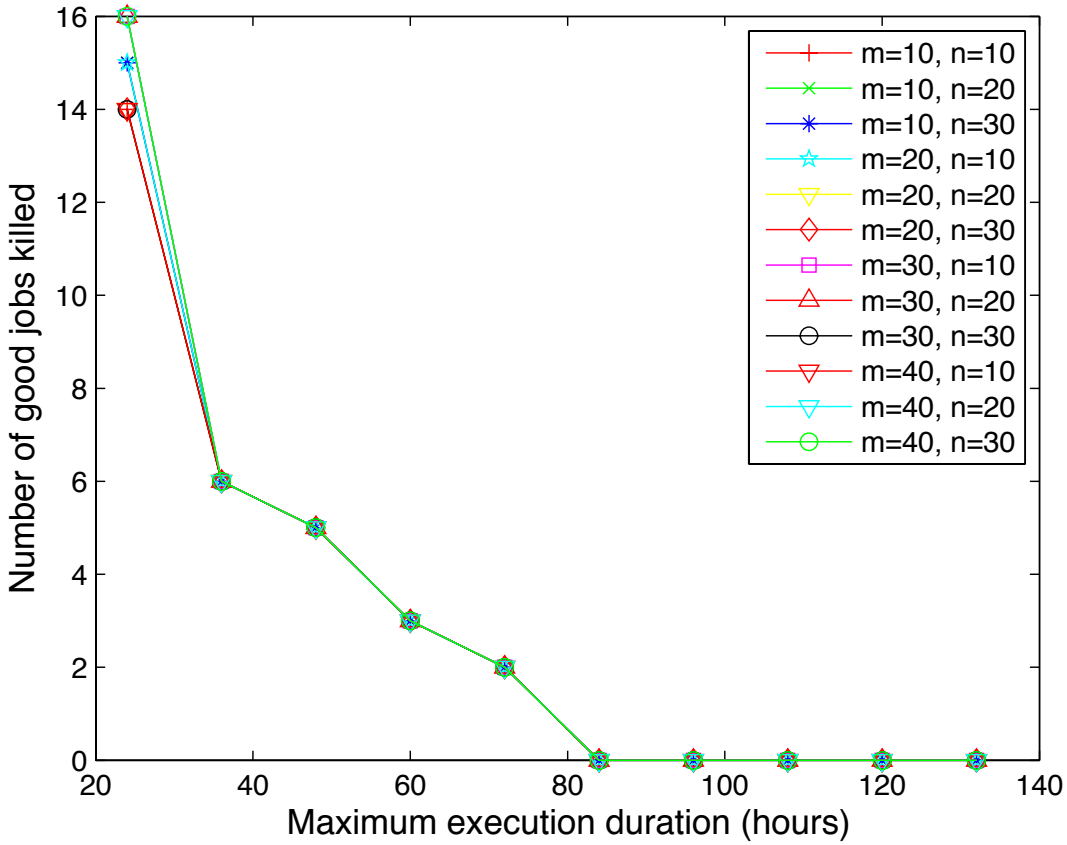

Fig. 6.17: The impact of Dedicated policy on Good Jobs Killed

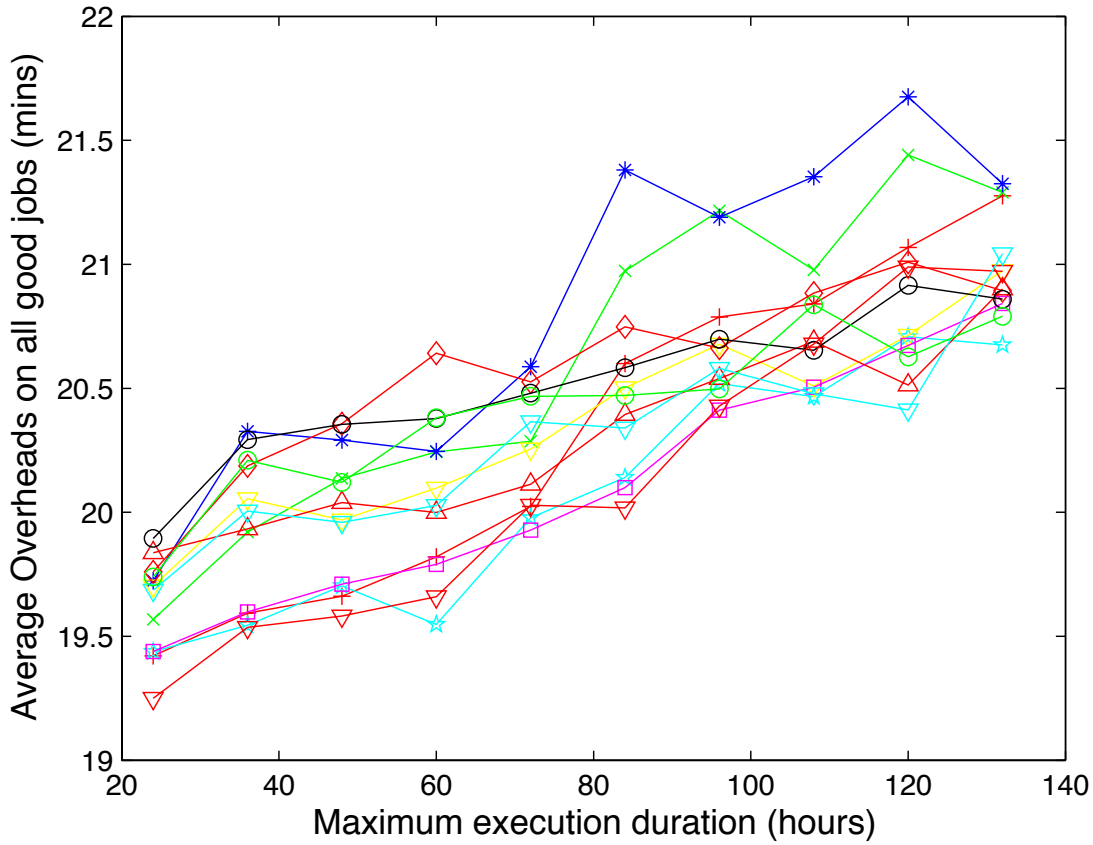

Fig. 6.18: The impact of Dedicated policy on Overheads

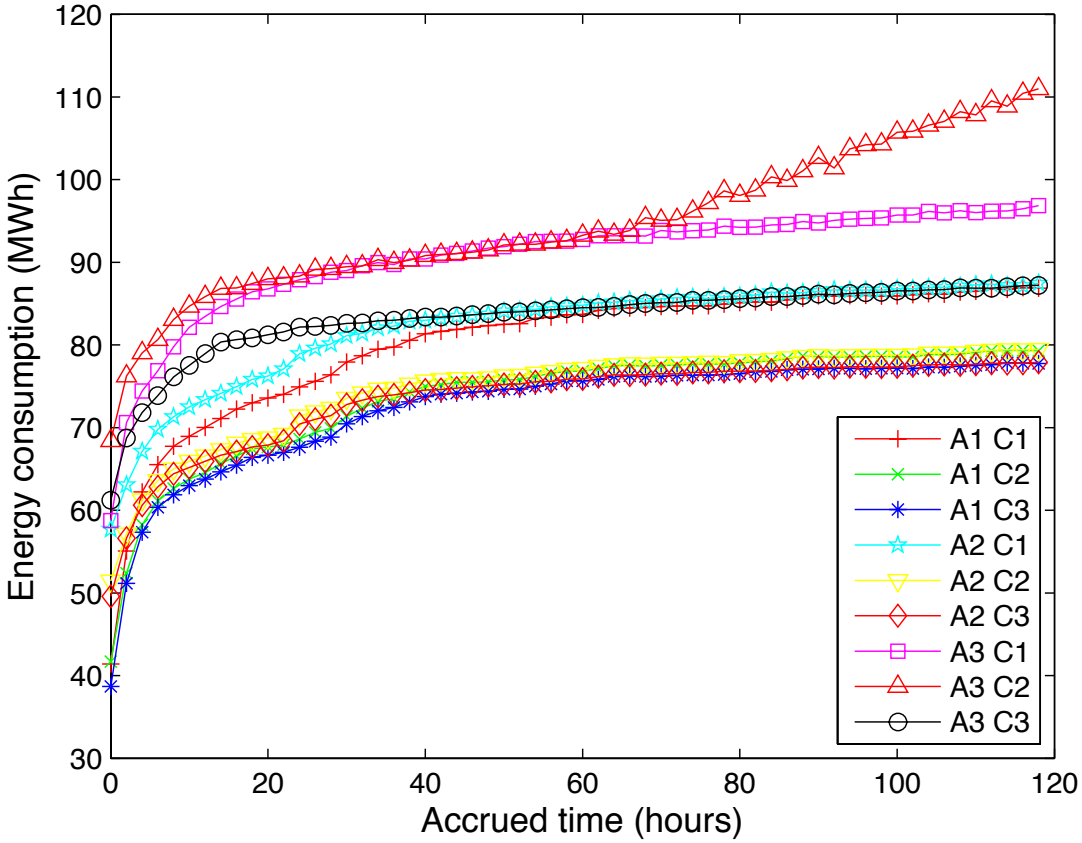

Fig. 6.19: The impact of Accrued policy on Energy

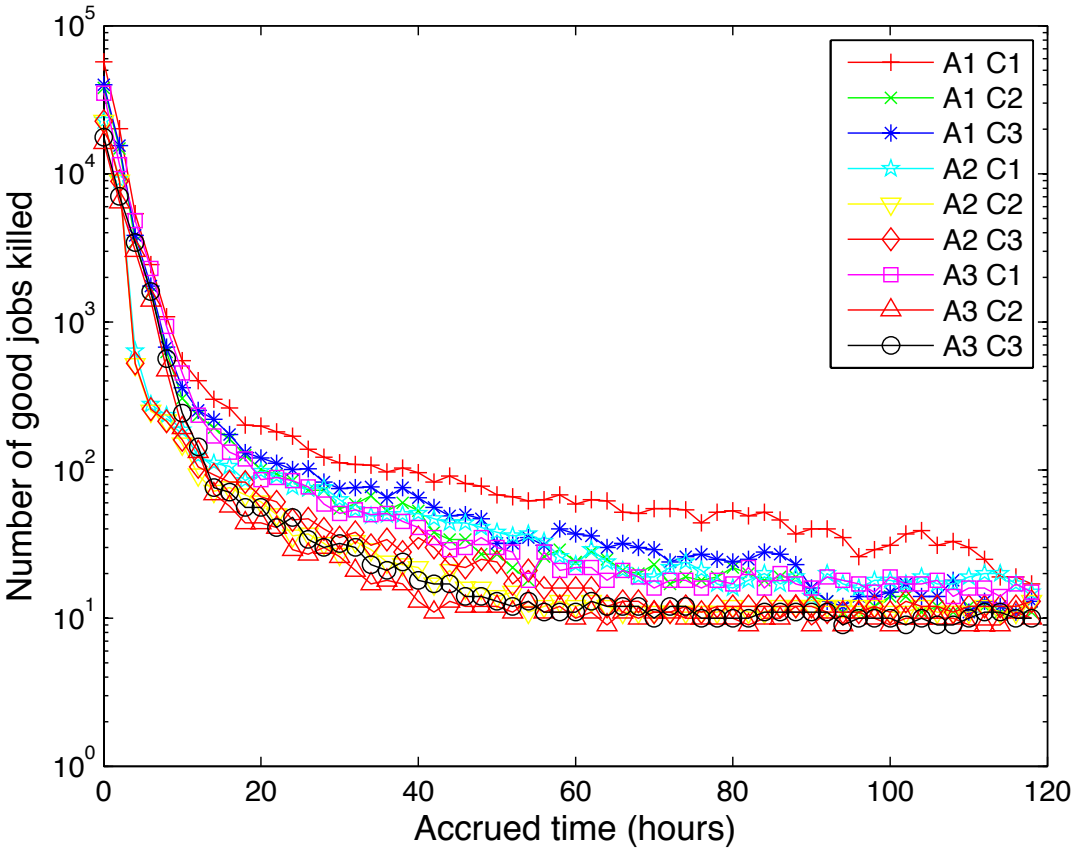

Fig. 6.20: The impact of Accrued policy on Good Jobs Killed

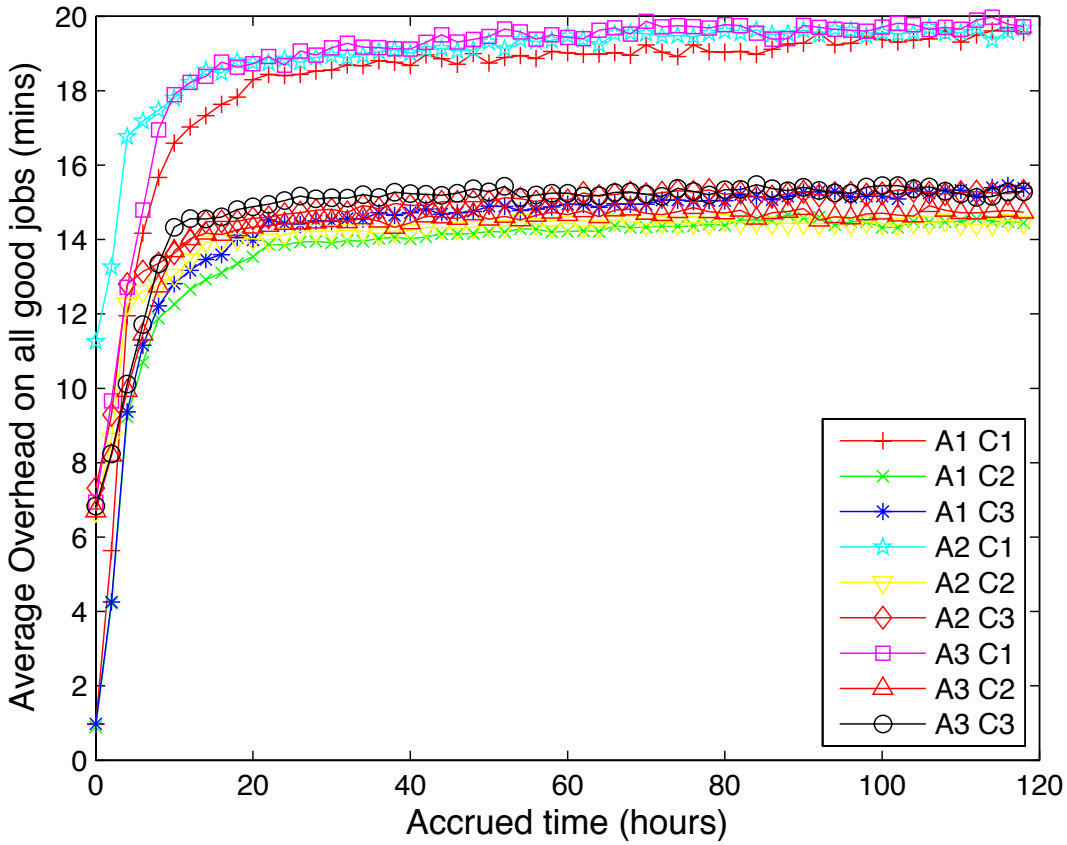

Fig. 6.21: The impact of Accrued policy on Overheads

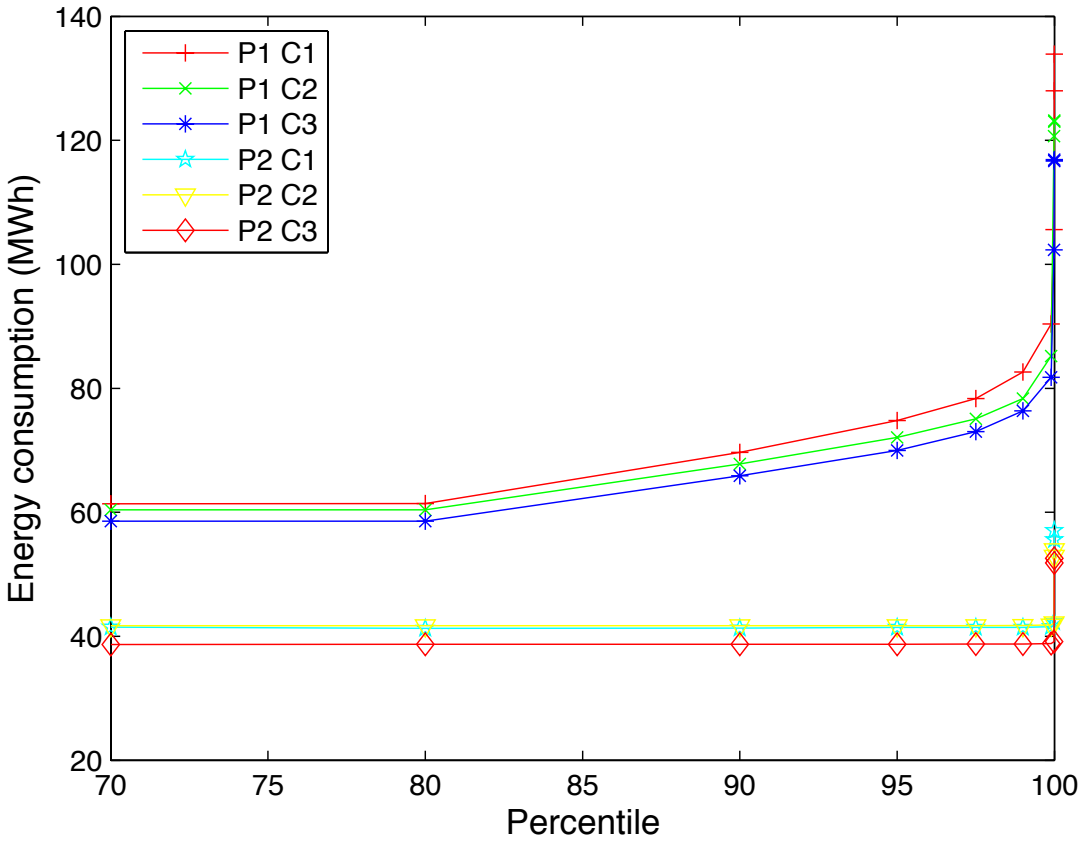

Fig. 6.22: The impact of Percentile policy on Energy

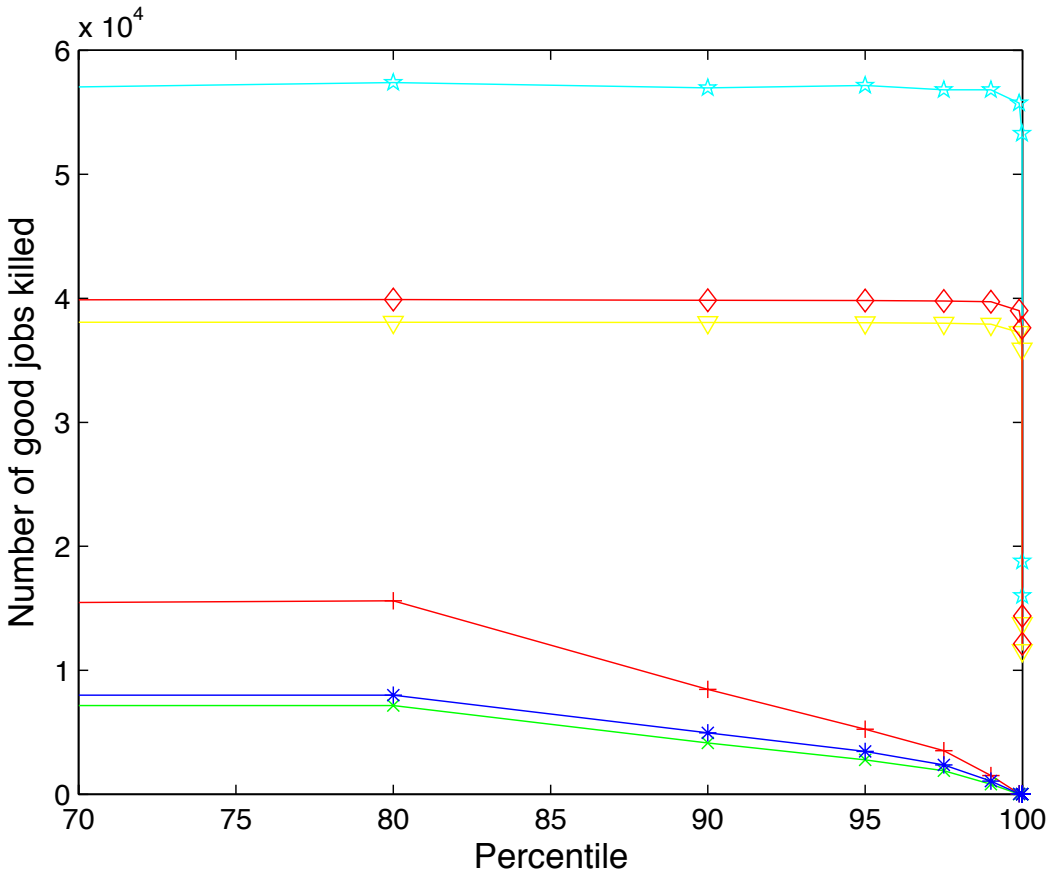

Fig. 6.23: The impact of Percentile policy on Good Jobs Killed

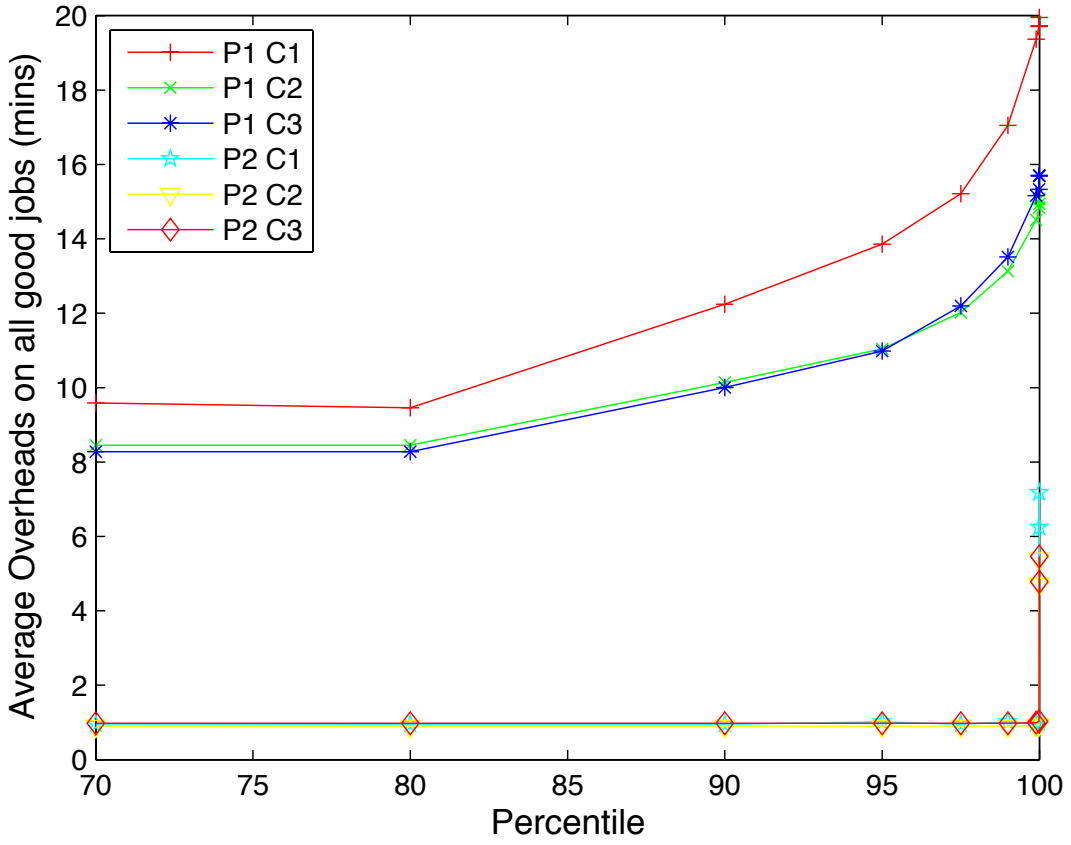

Fig. 6.24: The impact of Percentile policy on Overheads

6.6 Conclusion

In this work we demonstrate, through simulation, a number of policies which can be used to reduce the effect of miscreant tasks in a multi-use cycle stealing cluster. Each policy is capable of dramatically reducing the energy consumption for tasks in the system – to around $\frac{1}{2}$ of the original. This is largely attributed to reducing the amount of effort wasted on tasks that will never complete, but also by ensuring that tasks are placed onto computers which are less likely to be required for their primary task.

Although we are able to reduce the energy consumption significantly in all cases this can often be to the detriment of the users of the high-throughput system. Choosing a policy such as N2 (total deallocation count ignoring interactive user preemption) allows a significant decrease in energy consumption without loosing a significant number of good tasks and a minor increase in average task overhead, albeit still significantly

less than the values for the baseline results. By using dedicated computers we are able to reduce the number of good tasks lost to zero for a relatively small increase in energy consumption.

The policy D(m, d) with a low value for reallocations and a value of ~90 for maximum dedicated resource time would appear to give a 'good' solution. However, this policy could easily be adapted to incorporate the advantages of policy N2 or even the individual or accrued polices. In fact most of the policies presented in this work could be combined with each other to maximise the potential energy savings.

6.6.1 Future Work

One area of future work in the area of miscreant task detection is to extend the policies presented in this chapter to support the detection of miscreant *batches* of tasks. Tasks in historical workload traces may be grouped retrospectively based on data we currently hold about the job (e.g. executable name, submitting user). Our current dataset does not contain a sufficient number of batches to allow for the evaluation of such policies, but more recent datasets obtained in ongoing work (as discussed in Chapter 8) should offer a good basis for this work. We will further evaluate these policies for other public grid workoads [1, 10, 112].

Chapter 7

Energy efficient checkpointing in HTC systems

Summary

Checkpointing is a fault-tolerance mechanism commonly used in High Throughput Computing (HTC) environments to allow the execution of long-running computational tasks on compute resources subject to hardware or software failures as well as interruptions from resource owners and more important tasks. Until recently many researchers have focused on the performance gains achieved through checkpointing, but now with growing scrutiny of the energy consumption of IT infrastructures it is increasingly important to understand the energy impact of checkpointing within an HTC environment.

In this chapter we extend our trace-driven simulation introduced in Chapter 4, and use real-world datasets to demonstrate that existing checkpointing strategies are inadequate at maintaining an acceptable level of energy consumption whilst retaining the performance gains expected with checkpointing. Furthermore, we identify factors important in deciding whether to exploit checkpointing within an HTC environment, and propose novel strategies to curtail the energy consumption of checkpointing approaches whist maintaining the performance benefits.

7.1 Introduction

The issues of performance and reliability in cluster computing have been studied extensively over many years [117], resulting in techniques to improve these properties. The issue of cluster *'performability'* is relatively well understood, though until recently little consideration has been given to the energy impact of cluster performability.

High-throughput cycle stealing distributed systems such as HTCondor [146] and

BOINC [6] allow organisations to leverage spare capacity on existing infrastructure to undertake valuable computation. These High Throughput Computing (HTC) systems are frequently used to execute large numbers of long-running computational tasks, so are susceptible to interruption due to hardware and software failures. Furthermore, like many organisations we leverage institutional *'multi-use'* clusters comprised of student and staff machines, where jobs may also be interrupted when an interactive user starts to use a machine. Such interruptions lead to the tasks being evicted from the resource, increasing task makespan and wasted energy.

The execution time of these long-running tasks often exceeds the mean time to failure (MTTF) of the resources on which they execute. Furthermore, running thousands of jobs increases dramatically the chances of one of the computers failing during the run. Consequently, failures of resources lead to significant wasted computation and energy consumption. These overheads in turn lead to increased makespan (also referred in the literature as *sojourn time*) of tasks in the system.

Checkpointing is a fault-tolerance mechanism commonly used to increase reliability and predictability by periodically storing snapshots of application state to stable storage. These snapshots may then be used to resume execution in the event of a failure, reducing wasted execution time to that performed since the last checkpoint. Checkpointing has previously been employed on HTC clusters with little consideration for the energy consumption incurred by checkpointing overheads.

In this chapter we provide insights into the energy impact of checkpointing within high-throughput computing environments, making the following key contributions:

- Evaluate the energy impact of the two checkpoint schemes previously proposed in the literature [50, 183] for a real workload.
- Propose novel checkpoint policies for high-throughput computing environments and evaluate their performance for a real workload in terms of average task makespan, energy consumption and checkpoint utilisation.
- Develop a trace-driven simulation environment as a basis for research into energy-efficient fault tolerance approaches for HTC systems.

The rest of this chapter is organised as follows. In Section 7.2 we introduce our model for jobs executing within our system in the presence of failures, and state our assumptions surrounding the checkpointing progress and our checkpointing energy model. Section 7.3 describes a number of existing checkpointing strategies from the literature, and we propose a number of novel energy- and failure-aware checkpoint strategies. In Section 7.4 we evaluate the performance of the proposed checkpointing policies in terms of their impact on energy consumption, average task makespan and checkpoint utilisation. In Section 7.5 we discuss key considerations when adopting checkpointing in HTC clusters. Finally, we conclude and motivate further work in the area in Section 7.6.

7.2 Checkpointing and Failure Model

Choi *et al* [56] present a classification of two types of failures encountered on desktop grid environments: *volatility failures* including machine crashes and unavailability due to network issues, and *interference failures* arising from the volunteer nature of the resources. It is these *interference failures* which we consider throughout this work. Furthermore, we consider resource volatility in the form of scheduled nightly reboots for maintenance.

Figure 7.1 shows the state transition diagram for the execution of a single job in our system in the presence of these failures. Jobs are submitted by users and join a queue prior to being allocated on a resource. Once running, jobs are susceptible to interruption due to interactive users arriving on the resource. Jobs may be evicted immediately, or suspended for a period of time, after which jobs are evicted if the interactive user has not departed. Furthermore, jobs may be manually removed by their owner or a system administrator while in any non-final state.

Jobs may also periodically checkpoint, during which time their execution is paused while a snapshot of application state is taken. While High-Performance Computing (HPC) workloads such as MPI-based parallel applications rely on low-latency interconnects and significant bandwdith between nodes, HTC jobs typically have minimal network requirements so we expect the impact of checkpointing on the resident job to

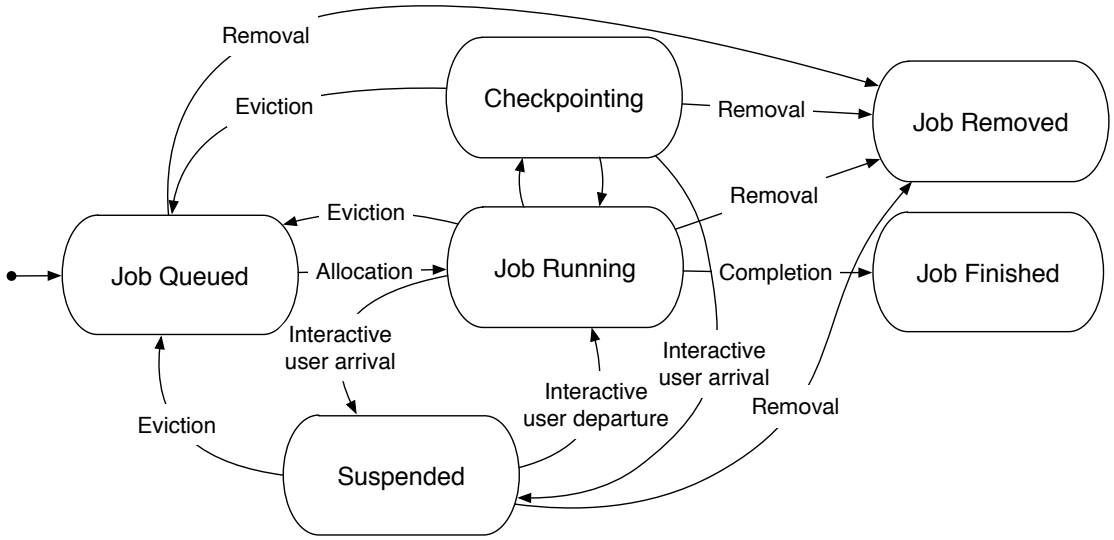

Fig. 7.1: Job state transition diagram

be negligible. Therefore, we assume the transfer of a checkpoint image may occur once the execution of a checkpointed job resumes.

Our checkpoint model differs from those presented in the literature as we assume interruptions may occur during checkpointing operations and subsequent recoveries.

7.2.1 Power model

In this work we assume checkpoints are stored on the stable storage of the existing servers provisioned to act as the central manager and submit nodes for HTCondor, so are able to discount their energy consumption. Consequently we model the energy cost of a checkpoint operation as the energy consumption of the compute resource during the checkpoint operation.

7.3 Policies

In this section we introduce the checkpointing policies investigated throughout this work. We divide these into policies to determine the interval between checkpoint evaluation events and policies determining whether a checkpoint operation should take place for a given evaluation event. Furthermore, we propose a class of migration policies which proactively checkpoint in anticipation of failure events, and migrate tasks

to resources less susceptible to failure.

When devising checkpointing strategies we ensure they rely only upon readily available system information and avoid expensive computation, such that they may be easily implemented in a real HTC system. The policies outlined below make use of system information exposed through the HTCondor ClassAd mechanism [193] and other HTC systems, so we consider each of these policies to be realistic.

7.3.1 Baseline policies

The following checkpointing policies are proposed to form a baseline against which the competitiveness of our proposed policies may be assessed.

None: This represents the policy enacted during 2010 in the Newcastle University HTCondor pool, where no jobs were checkpointed.

Opt: An optimal checkpointing strategy for best case comparison, whereby jobs are checkpointed immediately prior to eviction within our simulation. The results of this policy represent the greatest possible reduction in energy consumption and makespan achievable using checkpointing mechanisms, assuming perfect knowledge of future events. In order to provide a more realistic optimal policy against which we base our comparisons, under the Opt scheme checkpoints are only performed where current execution time of the job is greater than or equal to the duration required for the checkpoint operation to complete. Otherwise, a checkpoint is not taken, resulting in some loss of computation.

7.3.2 Checkpoint Interval

Here we present a number of policies determining the interval between checkpoint operations for a job.

C(n): Each job is checkpointed every n minutes. Hourly checkpointing (**C**(*60*)) is frequently considered in the literature and the HTCondor default strategy equates to **C**(*180*) [50].

Multi(n_{open}, n_{closed}, t): This policy leverages easily obtained system knowledge, considering computer cluster open/closed state to be analogous to high and low rates of user arrivals respectively. We define the time to the next checkpoint interval for a job in cluster j at time τ as:

$$I_{j,\tau} = \begin{cases} n_{open} & \text{if } \exists s_{i,j}, f_{i,j} : s_{i,j} - c_j \leq \tau \leq f_{i,j} - c_j \\ n_{closed} & \text{otherwise} \end{cases} \tag{7.1}$$

where $s_{i,j}$ is the ordered set of all start of open periods in cluster j, $f_{i,j}$ is the corresponding ordered set of all closed periods in cluster j and c_j is a time interval to mitigate the effect of checkpoints intervals selected close to a boundary being allocated a bad checkpoint interval with respect to the next interval.

MinuteInHour(m, t, R): In our analysis of our institutional workload, we observe a large proportion of interruptions from interactive users occur close to hour boundaries during office hours. This occurs due to the interactive users of the system mostly comprising of taught students, with students arriving to and departing from computers ahead of scheduled practical sessions and lectures. In this policy we leverage this observation, setting checkpoint intervals such that checkpoint operations are enacted prior to this period of increased interruptions

The next checkpointing interval i is derived using the following equation:

$$i = \begin{cases} m - j_{min} + R & \text{if } j_{min} < (m - t) \\ 60 + (m - j_{min} + R) & \text{otherwise} \end{cases} \tag{7.2}$$

where $j_{min} (0 \leq j_{min} \leq 59)$ is the number of minutes past the hour at which we are computing the next checkpoint interval, threshold value t represents a minimum job runtime before a job may be checkpointed and m is the number of minutes past the hour at which we wish to perform a checkpoint.

In situations where large batches of jobs are submitted to the system at the same time, this policy may result in a large number of checkpoints being taken simultaneously. In a real system this could impose significant load on the network

and storage nodes. In order to mitigate these potential effects, we introduce a random component in the checkpoint interval R, where R is a random variable uniformly distributed on $[-r, r]$, measured in minutes. As the value of r increases the system will become less susceptible to large numbers of simultaneous checkpoints caused by batch arrivals, but limit the ability of the policy to leverage the minute-in-hour period behaviour in checkpoint scheduling.

Ratio(p): In this policy we place an upper bound on the proportion of execution time consumed through checkpointing operations. The checkpoint interval i for a given job j is calculated as $i_j = \frac{d_j}{p}$ where d_j is the estimated checkpoint duration for job j, and p the maximum proportion of execution time to be occupied by checkpointing.

StartDelay(n, d): Through preliminary investigation we observe a significant proportion of wasted checkpoints occurred as a result of checkpointing of short-running jobs. While execution time of tasks is not known *a priori* and user estimates of task execution in grid have been shown to be inaccurate [15, 16, 218], this policy aims to curtail this waste, applying a start delay d before which a newly allocated task may not be checkpointed, after which tasks are checkpointed every n minutes.

GeometricProgression(a, r): Here we propose a generalised backoff policy based on a geometric progression, where the duration of the n^{th} checkpoint interval for job j is given by:

$$i_j^n = \begin{cases} a & \text{if } n = 0 \\ ar^{n-1} & \text{if } n \geq 1 \end{cases} \tag{7.3}$$

where a represents the initial checkpoint interval, r ($r \geq 0$) represents the *'common ratio'* for the sequence. The *'Exponential backoff'* policy proposed by Oliner *et al* [183] is equivalent to the geometric progression policy where $r = 2$.

7.3.3 Defer checkpoint policies

At each checkpoint interval, a decision must be made whether to proceed with carrying out a checkpoint operation, or defer to the next checkpoint interval. These decisions may be static, or may be dynamic and informed by the state of the system or job.

ClosedCluster: A simple policy incorporating easily obtained information about the institutional computer clusters, checkpoint operations are deferred when the cluster running the job is closed for use by interactive users.

Interarrival(w, m, l, d)**:** A policy requiring a greater insight into the global state of the HTC system, in this policy we observe the number of interactive user arrivals in a sliding window of w minutes. The feasibility of a checkpoint operation is evaluated every m minutes, with a checkpoint operation enacted if the number of arrivals in the period e_i from event set E is greater than threshold l and the job has not previously been checkpointed in the last d minutes. This policy may be expressed as follows:

$$\begin{cases} (t - c_j) \leq d & \text{if } \left| \left\{ e_i \middle| e_i \in E \wedge t - w \leq T(e_i) \leq t \right\} \right| \geq l \\ defer & \text{otherwise} \end{cases} \tag{7.4}$$

where current time is t, $T(e)$ is the arrival time for interactive user event e, c_j represents the time job j was last checkpointed (or 0 for jobs who have not previously been checkpointed).

We consider two variations of this policy, one considering the number of arrivals in the cluster of machines local to the job, and another considering the number of interactive user arrivals to the whole system.

7.3.4 Proactive migration

In addition to enabling recovery from failures, checkpointing mechanisms may also be used to support proactive migration of computational tasks to reduce makespan and energy consumption.

Scheduled: Tasks are migrated to avoid scheduled interruptions, e.g. all campus computers at Newcastle University reboot daily between 3am and 5am to perform routine maintenance and apply updates.

ClusterOpening: An event-driven checkpointing policy, where checkpoint operations are scheduled immediately prior to a cluster transitioning from being closed to open for use by interactive users.

7.4 Results

The impact on average task overhead and energy consumption for **None** and **Opt** policies on average task makespan and energy consumption is shown in Figures 7.2 and 7.3 respectively. All results presented are mean values obtained from fifty simulation runs, with error bars signifying 95% confidence interval values.

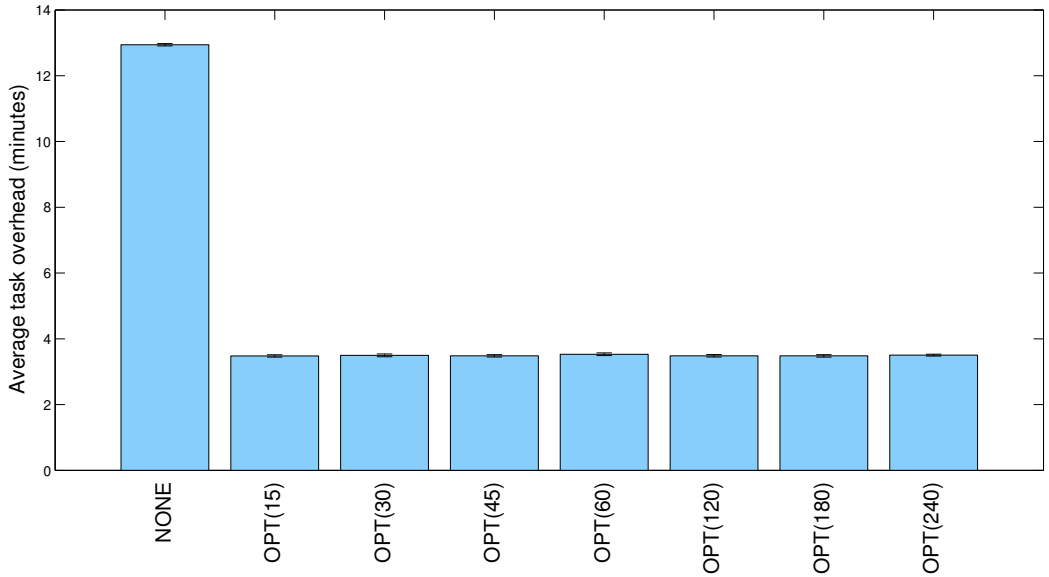

Fig. 7.2: Average Task Overheads

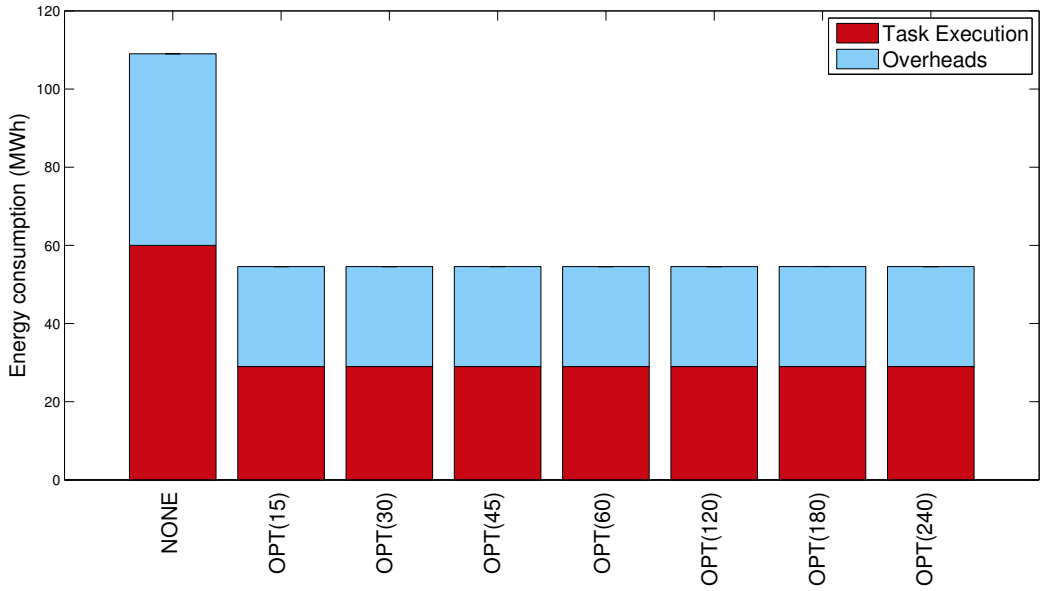

Fig. 7.3: Energy Consumption

The HTCondor workload from 2010 with no checkpointing mechanism applied results in an average task overhead of 12.94 minutes and energy consumption of 112 MWh. In this scenario, task overheads result from time spent by newly arrived or evicted jobs awaiting resources to become available. Under our optimal policy, which assumes perfect knowledge of failures, overheads are reduced to 3.48 minutes, with resulting energy consumption of 54.6 MWh. Here the time taken to generate checkpoints is shown to have little impact on the efficacy of checkpointing in the presence of optimal checkpoint interval selection.

7.4.1 Policy Results

We assess the impact of the proposed policies as the proportion of maximal benefit from checkpoint approaches. We define our benefit function as follows:

$$\text{Benefit} = 1 - \left(\frac{v_x - v_{opt}}{v_{none} - v_{opt}} \right) \tag{7.5}$$

where v_x may refer to either average task makespan, energy consumption or checkpoint utilisation for a given policy x, and v_{none} and v_{opt} refer to the these values for the **None** and **Opt** baseline policies respectively. We define checkpoint utilisation as the proportion of completed checkpoint operations which are subsequently used for recovery, indicating a given policy's ability to identify situations where a checkpoint

will be required.

Figure 7.4 show the impact of the policy on makespan, energy consumption and checkpoint utilisation for our Fixed (periodic) checkpointing policy $\mathbf{C}(n)$. Results are shown for checkpoint durations of one, two, three and four minutes respectively. Checkpoint duration is taken to include both the generation of the checkpoint and persisting this image to stable storage. We acknowledge that checkpoint duration is heavily dependent on data transfer costs, and incorporate estimates of these costs in our simulation, based on our investigation presented in Section 4.2.7. We observe this policy has the potential to achieve energy and makespan savings which are as great as 60% of optimal when the policy is correctly parameterised. The optimal checkpoint interval is shown to be dependent on the checkpoint duration for the workload, with the optimal interval for one- and four-minute jobs centred around 30 and 55 minutes respectively. In all cases, where a checkpoint interval shorter than 30 minutes are selected performance degrades significantly, with the cost of checkpoint operations exceeding the possible savings, leading to worsening overall performance and energy consumption. As the length of checkpoint intervals increase, the benefits of checkpointing tends towards zero, representing no checkpointing of jobs. We observe only a small proportion of successfully generated checkpoints are utilised under the Fixed policy, with the time taken to generate checkpoints having negligible impact. Though utilisation rises to approximately 15% for a checkpoint interval of 180 minutes, the benefit of a job resuming from a checkpoint generated that far in the past would be limited. When considering the checkpoint strategies previously considered in the literature, hourly checkpointing ($\mathbf{C}(60)$) delivers good performance dependent on the time required to generate checkpoints for jobs, but we show the HTCondor default of $\mathbf{C}(180)$ [50] to have little benefit for our workload. The observable decline in checkpoint utilisation for checkpoint intervals of approximately 130 minutes are an artefact of the relatively short execution time of the jobs comprising our workload.

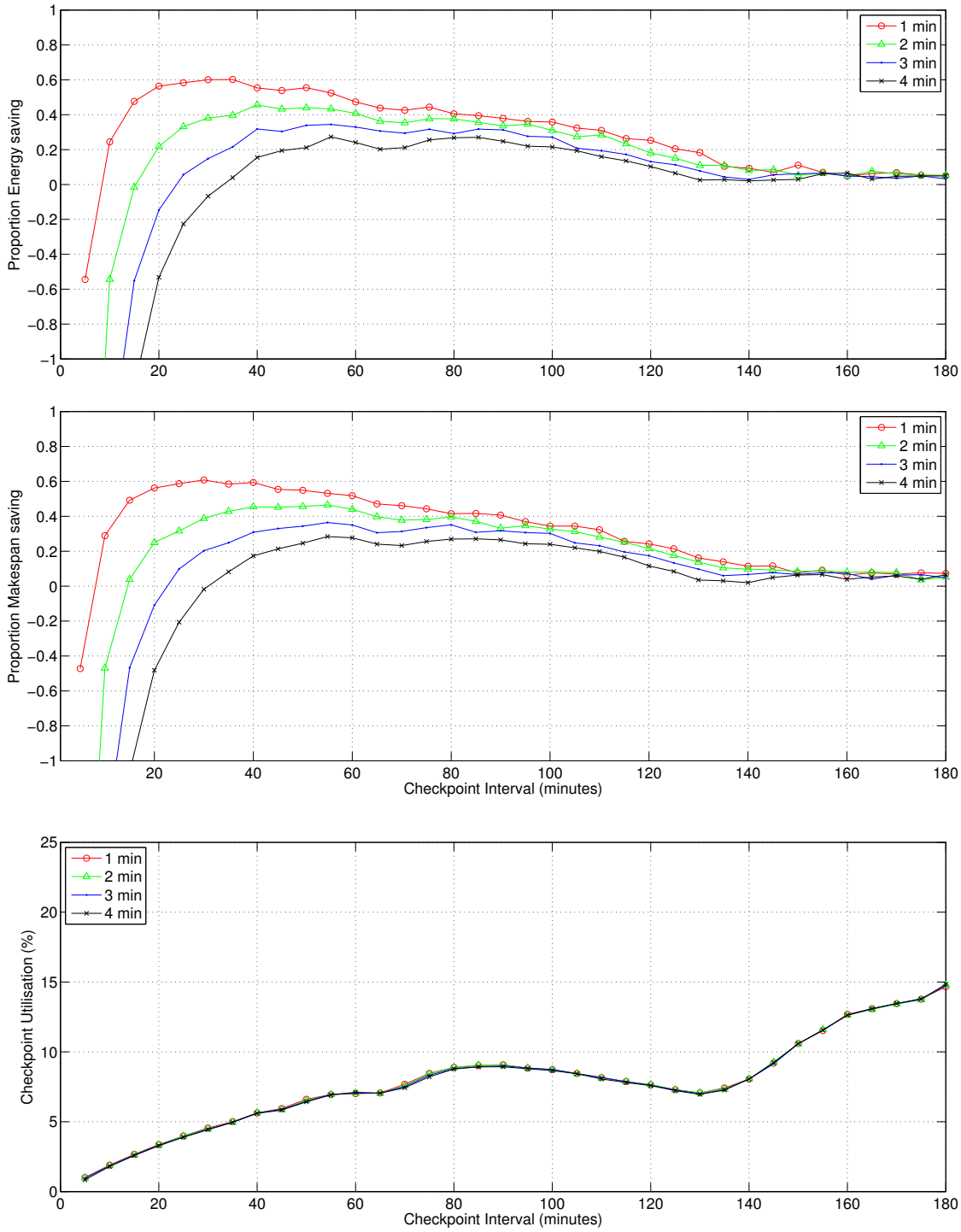

Fig. 7.4: The impact of Fixed checkpoint policy on energy consumption, overhead and checkpoint utilisation

In Figure 7.5 we compare our Fixed periodic scheme ($\mathbf{C}(n)$) with our Scheduled proactive migration policy, both in isolation (SR) and in combination with our Closed-Cluster defferal policy (CCSR). To aid readability we provide results for each policy for

checkpoint durations of one and four minutes. When considering the ClosedCluster policy with Scheduled reboot proactive migration (CCSR), we observe significant improvements in average task overhead and energy consumption, with the policy outperforming the Fixed periodic checkpointing scheme ($\mathbf{C}(n)$) for all lengths of checkpoint interval. Though the greatest proportional makespan and energy saving is only found to rise from 0.6 for the Fixed periodic scheme ($\mathbf{C}(n)$) to 0.7 for the CCSR scheme, this improvement is observed across a much wider range of checkpoint intervals, making these policies much less susceptible to poor performance due to sub-optimal checkpoint interval selection. Furthermore, we observe a significant increase in the utilisation of checkpoints generated in all cases.

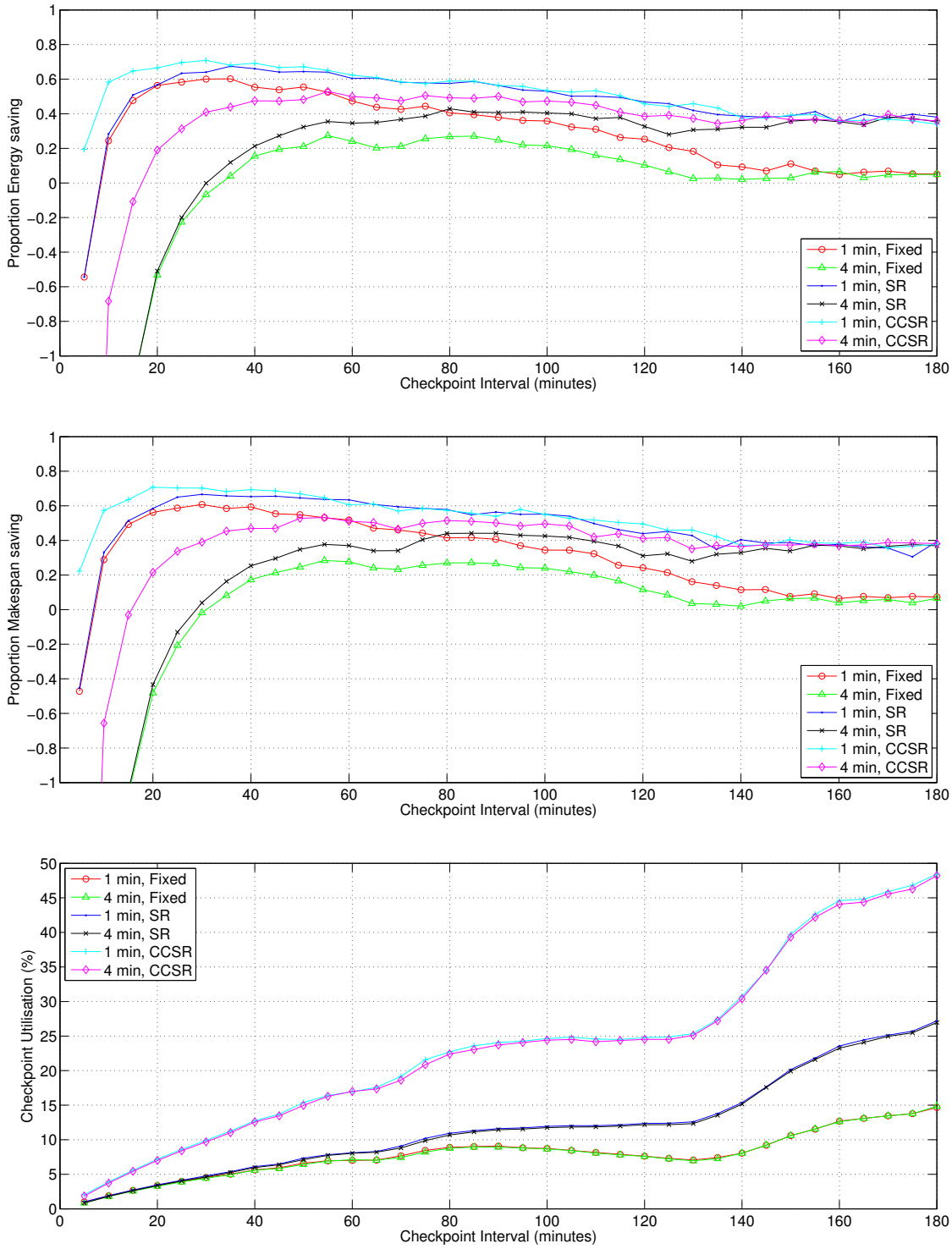

Fig. 7.5: The impact of ClosedCluster policy and Scheduled proactive migration on energy consumption, overhead and checkpoint utilisation

In Figure 7.6 we present the results of our Geometric policy. Results are shown for a 30 minute checkpoint interval, and varying common ratio parameter r. We find this policy to provide benefits to energy and makespan for all values of r. The best selection of parameter r is dependent on checkpoint duration, as $r \approx 1$ for 1 minute checkpoints,

and $r \approx 2$ for 4 minute checkpoints. Furthermore, the selection of this common ratio is dependent on the composition of the HTC workload, with a greater proportion of shorter or longer jobs impacting on the best value to select. An interesting extension of this policy would be to explore the selection of r based on the expected execution time of the workload.

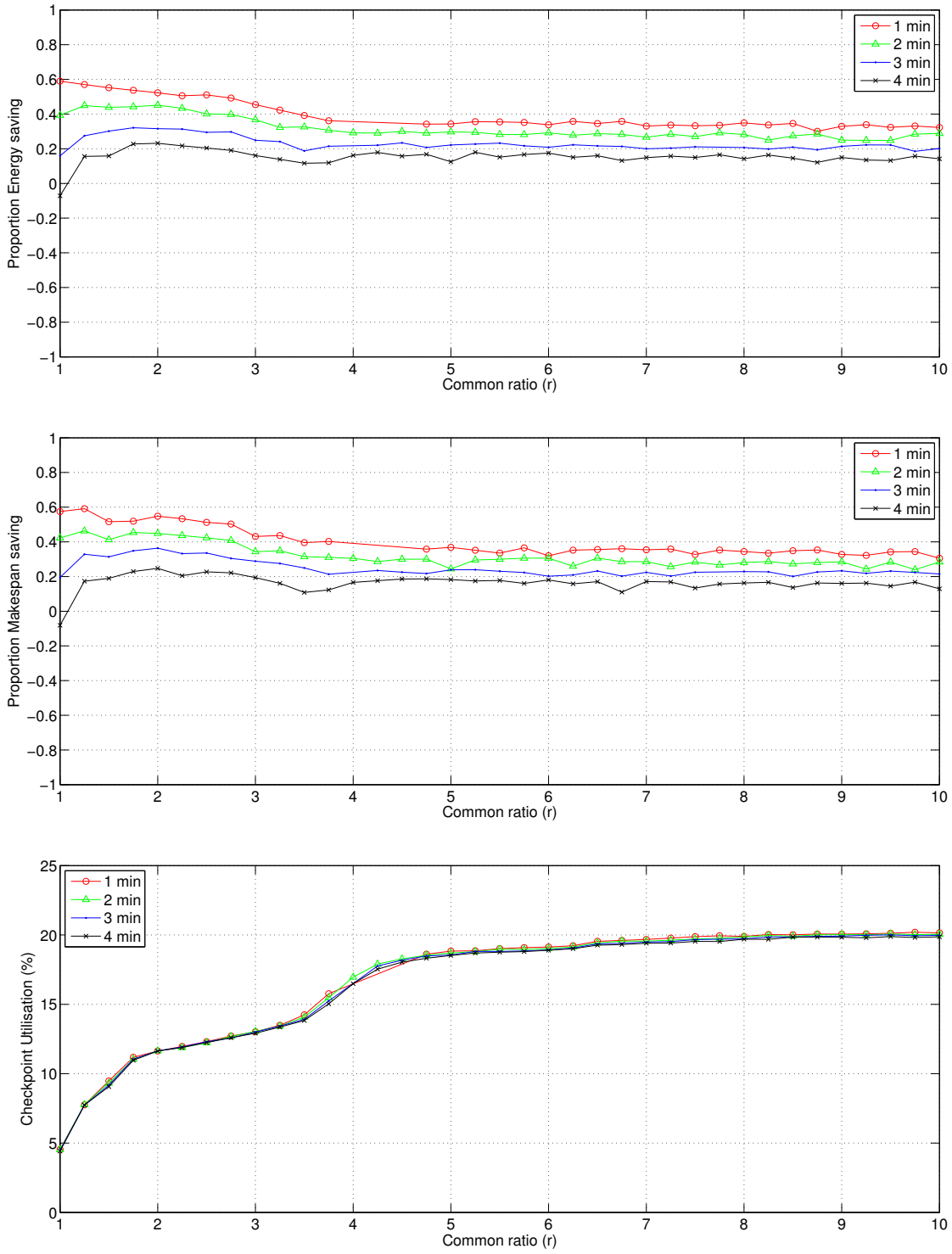

Fig. 7.6: The impact of Geometric policy on energy consumption, overhead and checkpoint utilisation

Results of our MinuteInHour policy are shown in Figure 7.7. Using knowledge of interactive user activity to inform the placement of checkpoint operations is found to result in an $\approx 20\%$ improvement in energy and makespan saving where $m = 55$ compared to the checkpoints carried out on the hour boundary. We introduce the random

component r to prevent large numbers of checkpoints scheduled at the same time, leading to network congestion and increased transfer delays. To exemplify the potential impact of such an adjustment, we show the results for a deliberately conservative value of $r = 5$ minutes. Under this policy, energy and makespan savings are lessened, particularly for the case of four minute checkpoints due to checkpoint operations being deferred towards the hour boundary, increasing their likelihood of interruption. Utilisation remains largely unaffected by the choice of parameter m. In a real system we anticipate a much smaller value of r to be adequate.

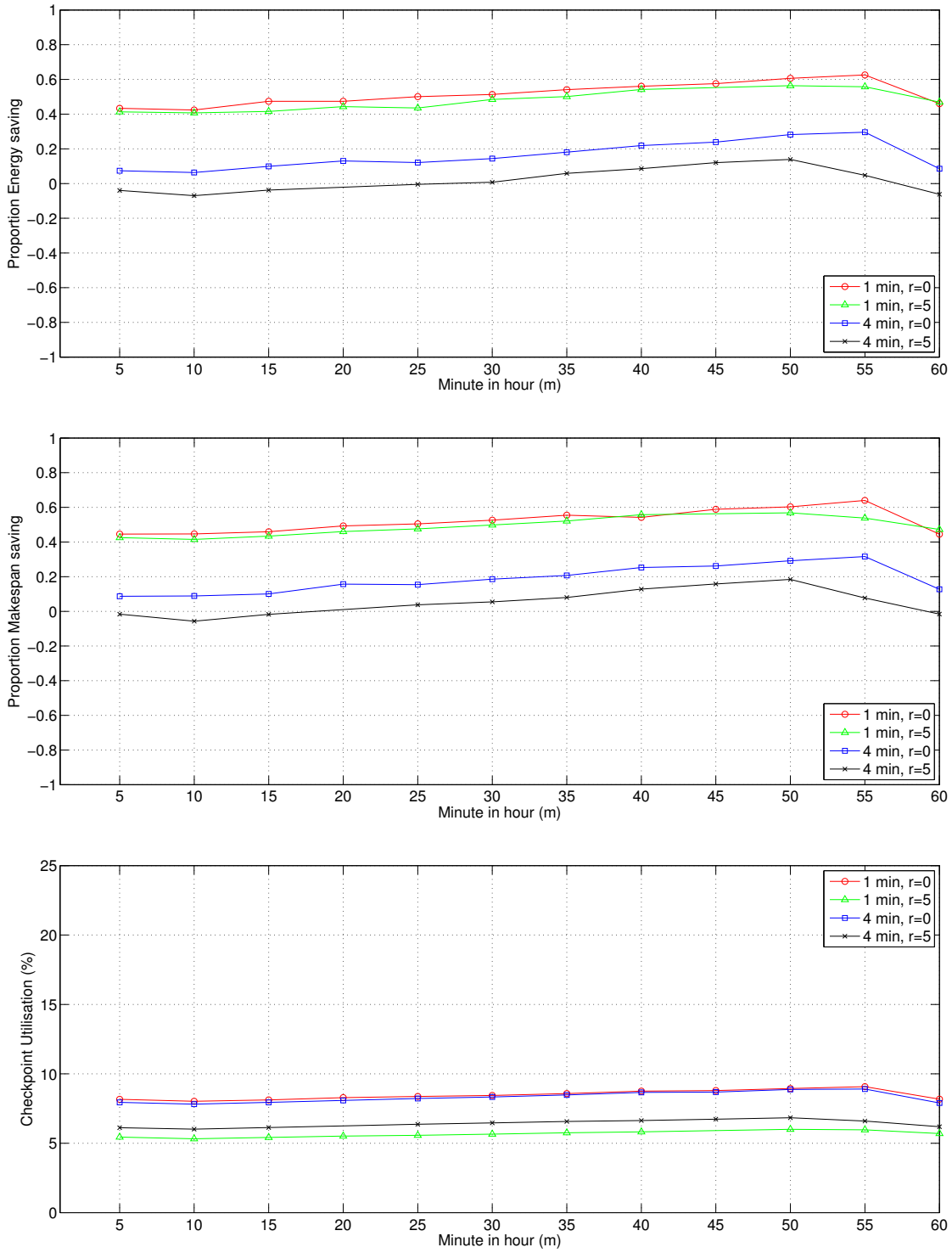

Fig. 7.7: The impact of MinuteInHour policy on energy consumption, overhead and checkpoint utilisation

Figure 7.8 show the results for the Ratio policy. This policy makes use of estimates of the time required to generate a checkpoint for a given job, and here we demonstrate the policy's ability to deliver equivalent benefits to jobs, irrespective of checkpoint duration. We observe the greatest benefit for our workload where checkpointing is con-

figured to take ~ 4% of execution time. Beyond this point, benefits begin to curtail and at ~ 15%, the cost of checkpoint operations exceeds that of lost execution due to interruptions. When considering checkpoint utilisation under the Ratio policy, utilisation falls as the proportion of execution time spent checkpointing (and thus the number of checkpoint operations) increases.

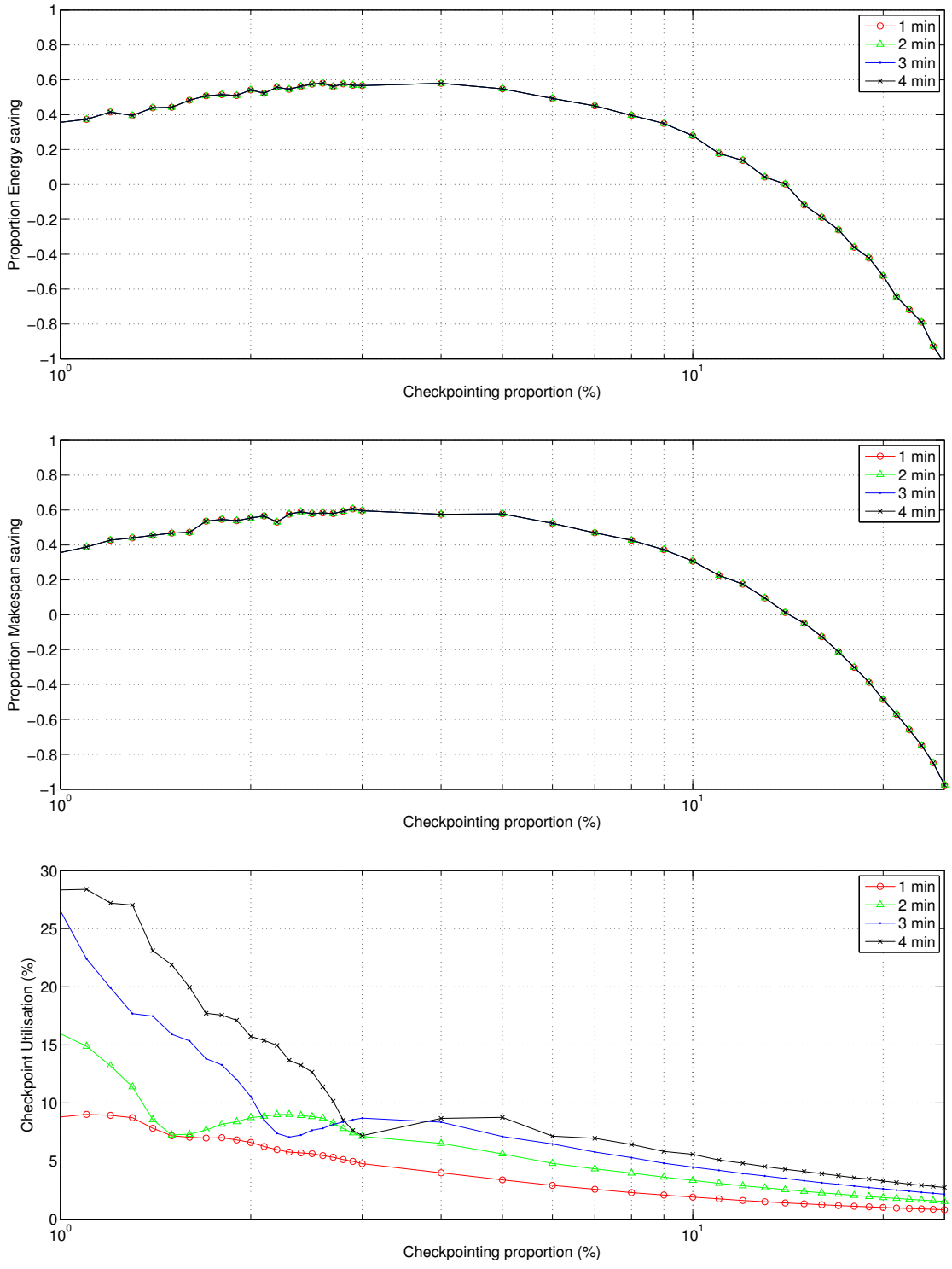

Fig. 7.8: The impact of Ratio policy on energy consumption, overhead and checkpoint utilisation

Figure 7.9 show the results for our policy placing a delay on the start of checkpointing for a job. With the exception of $\mathbf{C}(60)$ for one minute checkpoints, we observe a modest benefit to delaying the start of checkpointing during the first hour of a task's execution. Due to the relatively short execution time of the jobs comprising our work-

load, results begin to decrease beyond a start delay of ~ 90 minutes, due to the start delay being longer than the execution time of the task. The observable drop in the checkpoint utilisation graph centred at approximately 120 mins is also an artefact of this interaction between task execution time and start delay.

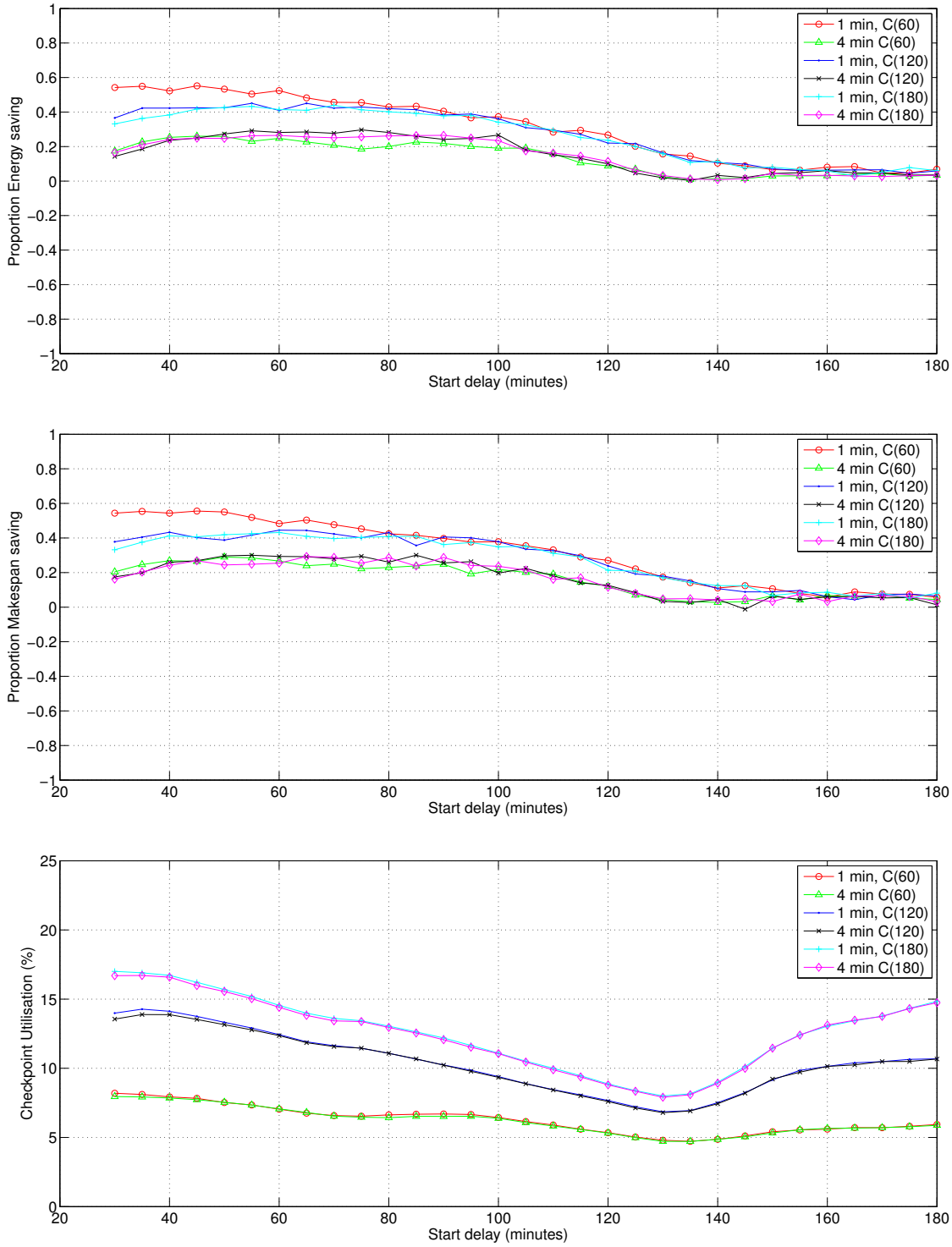

Fig. 7.9: The impact of Start Delay policy on energy consumption, overhead and checkpoint utilisation

In Figure 7.10 we show results for our Interarrival policy determining the conditions under which a scheduled checkpointing operation should proceed. Each of these results are shown for one minute checkpoint duration, and for sliding windows of length one, ten and twenty minutes. We present results for two variations of the policy, one which enacts checkpoints for a job based on the interactive user arrivals at the cluster where the job is executing, and the other based on interactive user arrivals throughout the entire system. The System-level checkpointing strategy is shown to provide greater improvements to energy consumption and overhead when compared to the Cluster-based approach, despite significantly lower checkpoint utilisation. The results for policies using a one minute sliding window are shown to be more sensitive to selection of interactive user arrival threshold (l) than those with longer window lengths. In both cases the benefits are greatest for small values of l, but we do not find user arrivals in such low quantities to be a sufficiently good predictor of task preemption for our workload.

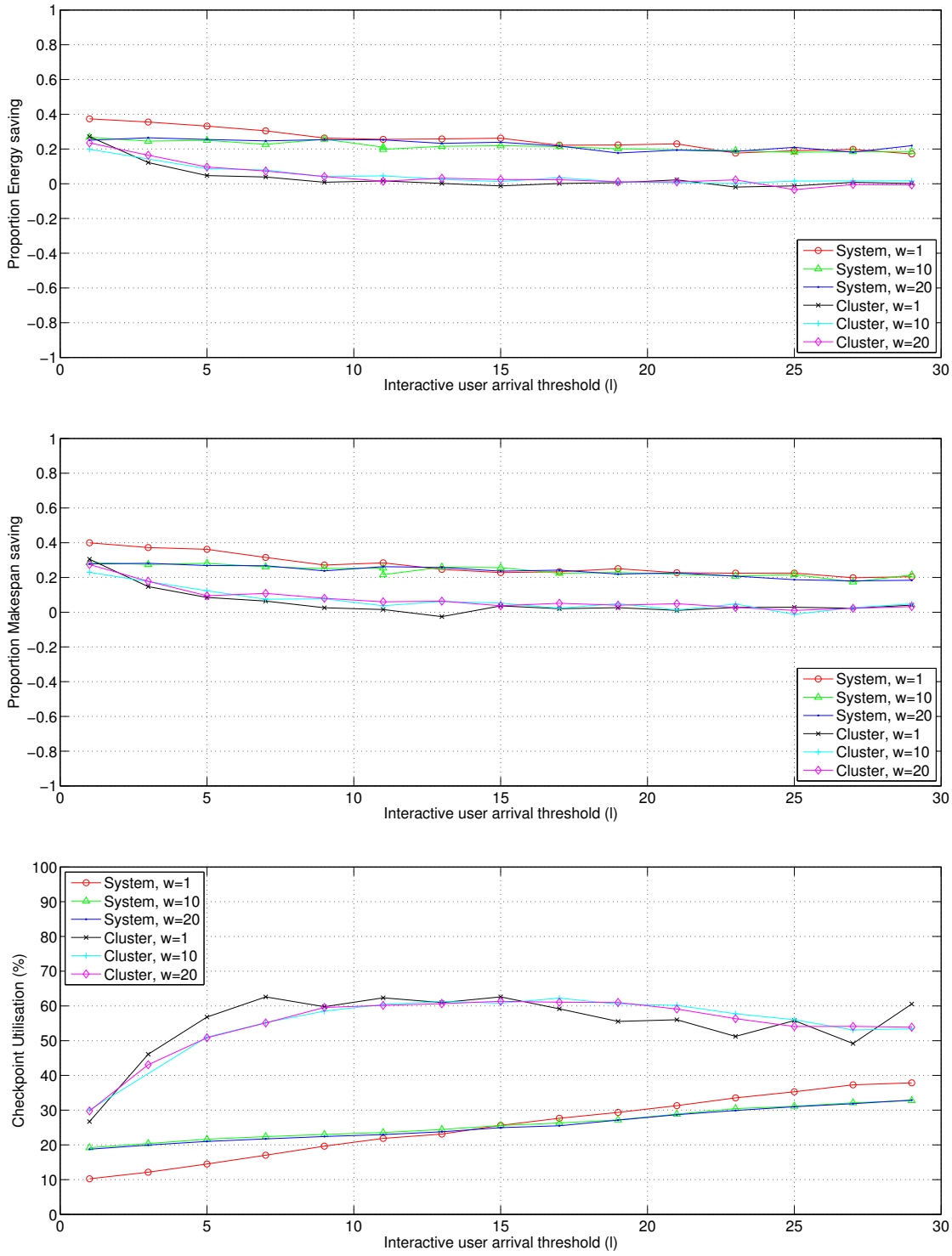

Fig. 7.10: The impact of Interarrival policy on energy consumption, overhead and checkpoint utilisation

7.4.2 Summary

From the results of our preliminary investigation, we note that for periodic checkpointing schemes, checkpoint duration is often as important as the checkpointing interval

chosen. This highlights the importance of a combined approach between checkpoint scheduling policies and the efficiency of the checkpointing mechanisms themselves. With checkpoint duration dominated by data transfer costs, significant gains may also be sought by reducing the size of the resulting checkpoint snapshots.

Though we find checkpointing results in significant improvements to task overheads, for many policies including periodic checkpointing, the benefits rely on correct parameterisation of policies. The exploration of approaches to adaptive checkpointing policies with the ability to adapt parameters to the observed interactive user and HTC workloads shall form the basis of ongoing work in this area.

Furthermore, a significant contributing factor in the significant potential for checkpointing to reduce average makespan is the relatively low load observed in the Newcastle University HTCondor cluster (approximately 12% for 2010 [1]). Consequently, evicted jobs are reallocated quickly, incurring only a short delay while waiting for resource to become available. We anticipate these makespan savings to be more modest for more heavily utilised pools of resources.

A key finding of this work relates to the effectiveness of load-based measures to govern the operation of a checkpointing scheme. While we found policies leveraging knowledge of scheduled interruptions and periods where clusters will be closed to interactive users, our threshold-based user interarrival policy was not found to offer significant benefits. In a real world system where the collection of such detailed knowledge is non-trivial, simple measures such as cluster opening times and the knowledge of scheduled interruptions seem sufficient in achieving favourable results.

Finally, we acknowledge that the efficacy of each of the checkpointing strategies presented here is dependent on the operating conditions. As many of the policies outlined in this chapter are not mutually exclusive, there is scope to yield further improvements by combining these approaches and targeting the scenarios in which they operate most effectively.

[1] We present a full analysis of our institutional workload from 2010 in Chapter 4.

7.5 Discussion

In this section, we outline the considerations the administrator of an HTC cluster should make when deciding whether to employ a checkpointing mechanism within their environment. In doing so, we highlight a number of areas of research interest, both with respect to energy-efficient checkpointing generally, and also issues specific to the application of these approaches in the context of multi-use clusters.

7.5.1 Operating policies

High Throughput Computing and Fine-Grained Cycle Sharing systems are typically configured to operate conservatively, with the interactive user of a machine given priority over the HTC workload running on the machine. Historically there was significant potential of interference from an HTC job, degrading performance and responsiveness for interactive users of a system. However, now in multi-core systems, and with the additional separation afforded by virtualisation technologies, the impact of HTC workloads on interactive users has been shown to be negligible [142]. Relaxing operational constrains preventing HTC jobs from running on resources with interactive users not only increases the capacity and throughput of the system, but also offers significant reduction in energy consumption. We demonstrate the energy and performance benefits made possible when leveraging knowledge of scheduled interruptions and user activity, highlighting the benefit of communication between cluster and HTC system administrators. Furthermore, we demonstrate the potential for checkpointing informing the management decisions made at the cluster level. For example, nightly reboots may be staggered to reduce the interference caused by many jobs checkpointing simultaneously, or reboots may be scheduled for shortly after clusters close to interactive users, increasing resource availability.

7.5.2 Workload

The efficacy of checkpointing is largely dependent on cluster workload. Checkpointing is most useful when the execution time of a large proportion of the workload exceeds

typical resource mean time to failure or user inter-arrival durations, increasing the likelihood of interruption. Checkpointing in other situations is likely to have a detrimental effect on energy consumption and makespan. Furthermore, some jobs do not support checkpointing, or are unsuitable for checkpointing e.g. those with particularly large application states.

7.5.3 User base

The Newcastle University HTC cluster supports a diverse user base, from experienced system administrators and Computer Scientists interacting directly with the system, to scientists leveraging its capabilities through user interfaces or submission mechanisms provided to them. Consequently there is a need for checkpointing mechanisms to be transparent and not require in-depth understanding of HTC or programming ability for users to benefit. Furthermore it is essential that such checkpointing mechanisms are capable of achieving energy savings in the absence of user knowledge.

7.5.4 Resource composition

Modern HTC clusters commonly comprise both volunteer and dedicated resources, and increasingly leverage Cloud resources to handle peak loads and offer runtime environments not supported locally. The composition of a cluster is an important factor in determining whether checkpoint mechanisms should be employed. In clusters solely relying on volunteer resources, checkpointing offers an attractive means to deliver favourable makespan and reduced energy consumption in the presence of interruptions. As the proportion of dedicated resources increase, similar benefits may be sought by steering longer-running jobs to these more reliable resources. The implications of checkpointing on workloads running on Cloud resources has not previously been investigated in the literature, but data transfer/storage and instance costs will exacerbate the impact of any checkpoint overheads.

7.6 Conclusion

In this chapter we have shown existing checkpointing mechanisms to be inadequate in reducing makespan while maintaining acceptable levels of energy consumption in multi-use clusters with interactive user interruptions. Our experimentation demonstrates that the naive application of checkpointing approaches has the potential to negatively impact energy consumption. We go on to propose and evaluate novel energy- and load-aware checkpointing strategies to curtail the energy consumption of checkpointing approaches whist maintaining the performance benefits. We highlight key considerations when adopting checkpointing in an HTC cluster and motivate a number of areas of further research interest in energy-efficient checkpointing.

7.6.1 Further work

There are a number of areas where we intend to extend our work into energy-efficient checkpointing in high-throughput computing systems.

Checkpoint replication

In this work we assume checkpoints are stored on the stable storage of the existing servers provisioned to act as the central manager and submit nodes for HTCondor, an assumption we wish to relax in further work.

Critical to the efficacy of checkpoint projects is the availability of application snapshots in the event a recovery is required. By using unreliable worker nodes to store checkpoint images, replication is required to reduce the likelihood of a checkpoint image being unavailable when required. Prior works have explored the decision of selecting a location for checkpoint replicas both on dedicated resources and worker nodes [150, 194].

Provisioning dedicated resources to act as checkpoint repositories is an approach commonly found in the literature, but would incur a penalty on energy consumption which may exceed the savings sought through the fault tolerance mechanism itself. This energy penalty may be mitigated in a number of ways. Firstly, we consider dy-

namic consolidation of dedicated checkpointing repositories to meet offered workload. More promising is the potential to use existing computers within the HTC pool - either idle or currently executing other work - as checkpoint repositories. The use of non-dedicated checkpoint repositories has been proposed the literature [9, 67, 68], though without consideration for potential energy savings.

Multi-version checkpointing schemes are often employed in scenarios where *latent errors* (i.e. errors which remain undetected for a period of time) are present [149], but we employ such a scheme to allow jobs to resume execution from previous checkpoints, where the latest snapshot of application state was placed on a non-dedicated checkpoint repository which is no longer available.

A challenge we foresee in the use of non-dedicated checkpoint repositories is that of network transfer, particularly for large checkpoint images. Here we consider the application of a BitTorrent-based system, similar to that proposed by Gadea *et al* [135], to replicate checkpoint images among workers as an area of further interest.

Job redundancy

Job redundancy, in which multiple copies of each task are deployed increasing the chance that at least one will complete in the first attempt, has been employed within grids to reduce the impact of failures on task makespan [110] but few consider the impact of job redundancy on energy consumption, with job duplication considered to be too costly in terms of energy.

Jensen *et al* [119] propose a task duplication scheme to mitigate the impact of failures early in a task's execution, whereby a job is submitted to multiple compute nodes at the beginning of execution, and replicas are cancelled after a number of minutes. However, the assumption that failures occur at the beginning of execution is questionable, and certainly not sufficient in out multi-use cluster context with interruptions from interactive users at any point during task execution.

Mills *et al* [169] propose *'shadow computing'*, a variant of typical job redundancy whereby DVFS techniques are applied to execute replicas at lower processor speeds (referred to as *shadows*). Consequently, replicas do not progress through execution as

quickly as the primary instance of the job, but nor do they consume as much energy in doing so. In the event of a failure, a shadow continues execution of the job, perhaps at an elevated processor speed. The authors develop an analytical model to evaluate the approach and find shadow replication to provide energy savings of 15-30% compared to traditional replication strategies.

Enokido *et al* [77] explore the application of redundant execution of workloads with consideration for energy-efficiency. The paper investigates these strategies for a cluster of nine servers, either homogeneous or heterogeneous in composition, in terms of computational power and energy consumption. However, such policies have not yet been explored in the context of multi-use clusters in the presence of user interruptions, nor at the scale of typical computational grids.

Despite these initial efforts, task duplication has not been considered in the presence of checkpointing mechanisms. As a basis of ongoing work, we consider the exploration of job duplication schemes leveraging the following additional knowledge to reduce the energy impact of their operation; *a*) knowledge of current HTC system load *b*) likelihood of interruption from interactive users *c*) estimated job execution time.

Spot pricing

Amazon EC2 provides Spot instances [3] which offer compute resources at significantly lower cost than typical EC2 instances. There exists a trade-off between cost, latency incurred when current instance costs exceed the bid rate, and the ungraceful interruption of tasks running on the instance. Traditionally this has precluded Spot instances from being used for long-running computation. In ongoing work we plan to leverage our previous research into Checkpointing strategies for HTCondor, adapting these for the spot instance environment to promote interruption tolerance and reduce cost/wasted computation.

Checkpointing strategies for spot instances have been explored in the literature. Voorsluys et al [249] explore the impact of price history and on-demand price based bidding strategies in the presence of hourly checkpointing, but do not explore varying the checkpointing interval. Experimentation is carried out using the CloudSim [46]

simulation framework with spot price data obtained between July and October 2011, and for a trace workload of 100,000 *'embarrasingly parallel tasks'* from the LHC Grid at CERN [1]. Results find higher bids to perform better both in terms of monetary cost and reduced deadline violations. Significant benefits sought through the use of fault-tolerance mechanisms (checkpoint and migration) but job duplication is shown to perform poorly in all cases. This study considers only the use of spot instances. It would be interesting to investigate the use of spot pricing in addition to normal on-demand cloud resources as well as local resources. Furthermore, the study does not consider the combination of fault-tolerance approaches to achieve further improvements.

Yi [256] investigates checkpointing strategies for EC2 Spot Instances, exploring the impact of checkpointing at hour boundaries and when the current spot price increases. The authors demonstrate the ability to tolerate the failures incurred by out-of-bid situations while reducing costs in comparison with standard Cloud instances.

Khatua *et al* [127] extend the work of [256] proposing a checkpointing scheme based on current spot history price. Their approach adopts two threshold bid values, one for the spot instance (kept sufficiently high such that out-bit events are unlikely to occur) and an application budget bid price used to determine when to checkpoint, terminate and provision spot instances. The authors claim a 5.56% performance improvement over the Optimal policy presented in [256].

In the pursuit of this area of research, we are currently obtaining trace data of Spot pricing for the last 12 months and have extended the HTC-Sim simulation framework to consume pricing traces.

Chapter 8

Conclusions

Summary

In this chapter, we summarise the research work presented in this thesis, investigating the trade-off of energy consumption and performance in large-scale distributed systems. We outline the contributions and limitations of these works, and discuss open research problems in the field, motivating a number of ongoing research efforts.

8.1 Thesis Summary

In this thesis we have explored the impact of operating policies on the energy efficiency and performance of large scale distributed systems.

In this work we adopt a number of approaches to exert energy efficient operation in large-scale computing environments. In Chapter 3 we explore energy-saving mechanisms in a decentralised BitTorrent environment which do not directly control resources comprising the system. We find these approaches to have limited applicability for energy reduction in decentralised systems, so in Chapter 4 we turn our attention to systems with more centralised control, namely high-throughput computing (HTC) environments. Here we define a generalised model of energy consumption in HTC systems and detail works undertaken to develop a trace-driven simulation capturing the behaviour of these systems. In Chapter 5 we explore resource allocation decisions in HTC systems, assuming full control over the power management of the resources, while in Chapters 6 and 7 we inform system behaviour to reduce energy consumption, without relying on control over the power management of resources.

We adopt a trace-driven simulation approach to explore policies governing the operation of large-scale distributed systems. Our simulation approach offers a number of benefits over a measurement approach, allowing us to rapidly evaluate new pol-

icy ideas and scheduling decisions in a controlled and repeatable manner, without the need for a costly testing environment, and with isolation from variability introduced by evaluations based on a live printing environment. As the workload observed in our environment is highly seasonal (as is commonplace in HTC systems [145]), trace-driven simulation allows us to compare policies across various workloads. The simulation environment, HTC-Sim, outlined in Chapter 4 is designed in such a way that policies evaluated in simulation may then be easily deployed into a real production environment.

8.2 Limitations

The verification and validating of simulations is a well documented issue [128]. In our original 2010 dataset we possess only input to the system (as trace logs), and limited summary statistics of the operation of the system during the period, as obtained from the running production environment. As discussed in Section 4.3.4, since December 2012 we have extended our data collection to include event logs which include additional information including periodic memory and disk utilisation information throughout job execution, and comprehensive logs for resource re-allocation, suspension and checkpointing. This information is useful in a number of key areas. Firstly, by providing a greater insight into the performance and behaviours of the applications running in our HTC system, we may exploit this information in our scheduling decisions and operating policies. Secondly, it shall enable us to evaluate the intermediate output of our simulation to establish consistency between the simulated results and real world outcomes. While this strategy will offer greater confidence in the results for the policies enacted on the real cluster, this approach offers little ability to guarantee the validity of simulations for novel policies not enacted on the production environment. To offer greater confidence in the simulated results, we intend to extend the works of McGough *et al* [157] to enact our policies within a live environment.

A further limitation of a trace-driven simulation approach is the generalisability of results beyond the trace logs available. To this end, we are currently in discussion with the Computer Sciences Department at University of Wisconsin-Madison, and are ac-

tively seeking contributions from other HTC system operators, to obtain further trace logs for more heavily loaded systems to further validate our results.

Finally, an open challenge common to many of the policies proposed throughout this thesis is determining the correct choice of parameters for a given configuration and offered load. In some cases, policies achieve favourable performance and energy consumption results for a broad range of parameter choices, while others are more reliant on correct parameterisation, with sub-optimal parameter selection sometimes leading to degradation of performance. In [159] we explore this challenge using Reinforcement Learning [226] to tune parameters of a resource allocation policy, and found it capable of yielding energy reductions of 30% with no impact on task completion, or up to 53% in situations where a modest overhead increase may be incurred. We see the application of Reinforcement Learning and similar machine learning approaches to other policy decisions within HTC environments as a key area of ongoing exploration.

8.3 Future Research Directions

Here we motivate a number of areas of future research, arising from lessons learnt throughout the PhD.

8.3.1 Generalise operating policies to other environments

An area of key interest is the generalisability of our developed approaches to other operating environments. As organisations strive to reduce the energy cost of their desktop IT estate, many now look towards centralising and consolidation of computational power through virtualisation and thin client architectures. Though the adoption of thin clients reduce energy consumption, they lack the computational power required for serving HTC workloads, thus reducing the available capacity of HTC systems. In order for organisations to continue offering HTC services, the purchase and provisioning of dedicated resources is required. Cloud Computing [12] offers an alternative in which organisations can offload the HTC work they are no longer capable of processing locally for the operational expense of pay-as-you-go Cloud charging. Using the Cloud for excessive local demand has been proposed in the literature [66, 70, 153, 155, 241], how-

ever, this has largely been through bursting to the Cloud [4] when all local resources are exhausted, with no consideration of the 'bursty' nature of HTC workload or other uses of the local resources. Other approaches have looked at deployment of the entire workload to the Cloud [70]. Although this may lead to favourable makespan, this is unlikely to offer the most cost effective solution due to the cost overheads and built-in profit margins of Cloud providers. Likewise it may be more economical to utilise the Cloud prior to full loading of the organisations resource pool due to the risk of tasks being evicted from a resource, requiring re-computation on an alternative resource.

In previous work we have evaluated the viability of running institutional high-throughput computing workloads on the Cloud, exploring a number of policy decisions governing resource allocation decisions, cloud instance keep-alive, and the delayed deployment of jobs [162]. In future work we intend to investigate the economics of running HTC jobs over both the local cluster and Cloud computers, investigating the effect this will have both on the overall cost to the organisation (local costs and Cloud costs) along with the effect these policies will have on the HTC users.

8.3.2 Combined with analytical approach

Throughout this thesis we have adopted a trace-driven simulation approach to investigating the trade-offs between performance and energy consumption of large-scale systems. While trace-driven simulation approaches allow the practitioner to closely model the behaviour of the target system, large-scale parameter sweep experimentation is computationally expensive and generalising results to other trace workloads is non-trivial. Conversely, analytical models rely on making certain assumptions to arrive at a tractable solution, but computation is inexpensive, and results more readily generalisable.

In [41] we motivate the need for combined approach, with analytical approach to the performance/energy trade-offs of large distributed systems. A Population Continuous Time Markov Chain (PCTMC) model of our HTC system is presented, and once fitted to the Newcastle Uni HTCondor trace data from 2010, and the PCTMC model is solved using the Grouped PEPA Analyser (GPA) tool [221]. This approach is shown to

be capable of capturing the performance and energy consumption characteristics of computational grid systems at scale.

One area of particular interest is the application of PCTMC models running alongside a production HTC system environment, with model predictions used to inform policy decisions on the running system.

8.3.3 Energy efficient printing

Printing is estimated to account for 10-16% of ICT related energy electricity consumption within higher education [115], but to date printing has received less attention than power saving techniques for desktop and server infrastructure. Contrary to server and commodity hardware which has seen significant improvements in energy consumption in recent years [199], this does not hold for printer hardware. Programmes such as EPA EnergyStar [78] provide guidance for the energy efficient operation of print devices, but to date the impact of these standards has not been evaluated for real-world print workloads. With energy consumption of active printers considerably greater than that of desktop machines, and printers reportedly only powered down 15-30% of the time [116], there is clearly a demand for intelligent approaches to handling the energy consumption of printers.

Prior efforts to promote sustainability in printing have primarily focused on informing and altering user behaviour to reduce usage of consumables [93, 253]. Ciriza *et al* [57] develop a statistical model for the optimisation of printer power consumption by determining the optimal printer timeout period, though in doing so significantly increase the number of shutdown and wake-up transitions, posing significant implications on long-term printer reliability. Stefanek *et al* [222] present energy consumption data of a single centrally-managed shared printer within a University, though without consideration for potential changes to their operating policies. Andreoli *et al* [7] employ a clustering approach to discover communities of users within a shared print environment. A survey and critical evaluation of life-cycle analysis of the environmental impact of print resources is presented in [40]. Despite clear opportunities for significant reductions in cost and environmental impact, few in the literature have consid-

ered energy-efficient management policies for shared printing.

As part of this ongoing work we have collected two extensive datasets of usage on shared printer infrastructure, the largest corpus of print trace data currently available for shared printing infrastructures, comprising *7,562,680* print jobs processed between *01/08/2004* and *13/12/2012* by the *189* centrally managed printers at Newcastle University. Datasets were obtained from the mining of historical account data and from interrogating logs of a running departmental print server.

Predictive models of energy consumption [73] and energy-aware performance benchmarks such as SPECpower [134] are well established in the context of server and commodity hardware but similar efforts have not extended to printers. Here we propose a predictive model of full-system energy consumption of print devices. Figure 8.1 demonstrates one power trace obtained during preliminary testing of a Konica BizHub C280 laser printer for *warm-up*, *active-idle* and *printing* modes. [1]

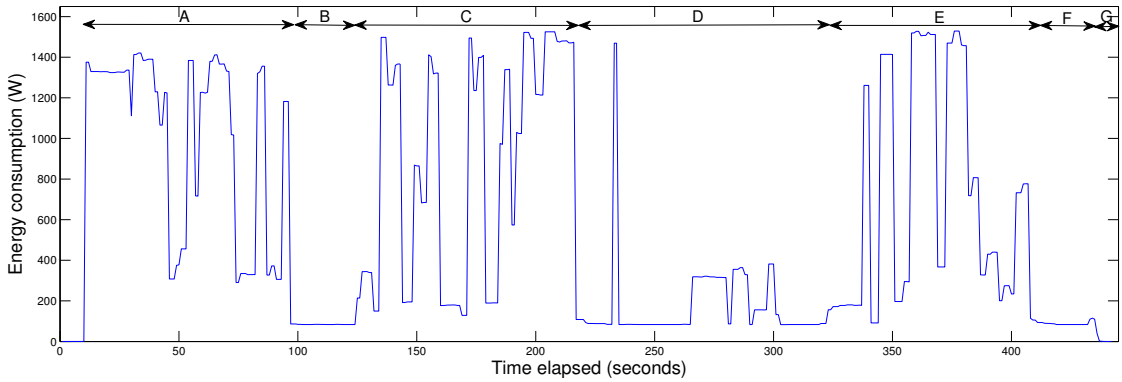

Fig. 8.1: Energy consumption trace for Konica BizHub C280

In this area of future work we shall motivate the need for energy-aware printer management techniques governing device timeouts, batching of jobs, and printer selec-

[1] We can see in segment *'A'* the printer transition from its *off* state where energy consumption is negligible, to an *active-idle* state in segment *'B'* where the printer consumes approximately 83.5W. The transition in *'A'* takes approximately 67 seconds to complete, during which time there are significant increases in energy consumption, with peak consumption reaching 1,413W. Due to this bursty power profile we take a mean average value (1,002.5W) to represent the wakeup period in our model. Segment *'C'* shows the energy profile for printing a 45-page simplex text document in colour mode, taking 89 seconds to complete Meanwhile, segment *'D'* shows the printer again in an *active-idle* state consuming approximately 82.9W, but highlights the impact of periodic mechanical operations on the energy consumption of the printer under this mode. Through further experimentation we find these operations to occur shortly after jobs but not during longer periods of idle time. Segments *'E'* and *'F'* refer to further periods of printing and idle time respectively. Finally, segment *'G'* shows the energy consumption trace of the printer transitioning once again into a low power state, incurring only a brief period of slightly increased energy consumption.

tion. We will demonstrate through trace-driven simulation the impact of these policy decisions at single-printer and ensemble levels for a large institutional print workload.

References

[1] http://www.cs.huji.ac.il/labs/parallel/workload/.

[2] Muhammad Abdullah Adnan, Ryo Sugihara, Yan Ma, and Rajesh K Gupta. Energy-optimized dynamic deferral of workload for capacity provisioning in data centers. In *Green Computing Conference (IGCC), 2013 International*, pages 1–10. IEEE, 2013.

[3] Amazon Web Services, Inc. Amazon Elastic Compute Cloud (Amazon EC2), 2014. URL http://aws.amazon.com/ec2/.

[4] Brian Amedro, Françoise Baude, Denis Caromel, Christian Delbé, Imen Filali, Fabrice Huet, Elton Mathias, and Oleg Smirnov. An efficient framework for running applications on clusters, grids, and clouds. *Cloud Computing*, pages 163–178, 2010.

[5] Giuseppe Anastasi, Ilaria Giannetti, and Andrea Passarella. A bittorrent proxy for green internet file sharing: Design and experimental evaluation. *Comput. Commun.*, 33:794–802, 2010. ISSN 0140-3664.

[6] David P Anderson. Boinc: A system for public-resource computing and storage. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 4–10. IEEE, 2004.

[7] Jean-Marc Andreoli and Guillaume Bouchard. Probabilistic latent clustering of device usage. In *Advances in Intelligent Data Analysis VI*, pages 1–11. Springer, 2005.

[8] Lachlan L. H. Andrew, Andrew Sucevic, and Thuy T. T. Nguyen. Balancing peer and server energy consumption in large peer-to-peer file distribution systems. In *Online Conference on Green Communications (GreenCom), 2011 IEEE*, 2011.

[9] Filipe Araujo, Patricio Domingues, Derrick Kondo, and Luis Moura Silva. Using cliques of nodes to store desktop grid checkpoints. In *Grid Computing*, pages 25–36. Springer, 2008.

[10] The Grid Workloads Archive. http://gwa.ewi.tudelft.nl/, 2014.

[11] Eduardo Argollo, Ayose Falcón, Paolo Faraboschi, Matteo Monchiero, and Daniel Ortega. Cotson: infrastructure for full system simulation. *ACM SIGOPS Operating Systems Review*, 43(1):52–61, 2009.

[12] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010. ISSN 0001-0782. doi: 10.1145/1721654.1721672.

[13] Guillaume Aupy, Anne Benoit, Rami G. Melhem, Paul Renaud-Goud, and Yves Robert. Energy-aware checkpointing of divisible tasks with soft or hard deadlines. *CoRR*, abs/1302.3720, 2013.

[14] Axel Auweter, Arndt Bode, Matthias Brehm, Luigi Brochard, Nicolay Hammer, Herbert Huber, Raj Panda, Francois Thomas, and Torsten Wilde. A Case Study of Energy Aware Scheduling on SuperMUC. In *Supercomputing*, pages 394–409. Springer, 2014.

[15] Cynthia Bailey Lee, Yael Schwartzman, Jennifer Hardy, and Allan Snavely. Are user runtime estimates inherently inaccurate? In *Job Scheduling Strategies for Parallel Processing*, pages 253–263. Springer, 2005.

[16] Cynthia Bailey Lee, Yael Schwartzman, Jennifer Hardy, and Allan Snavely. Are user runtime estimates inherently inaccurate? In DrorG. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 3277 of *LNCS*, pages 253–263. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25330-3. URL http://dx.doi.org/10.1007/11407522_14.

[17] Marinho P. Barcellos, Rodolfo B. Mansilha, and Francisco V. Brasileiro. Torrentlab: investigating bittorrent through simulation and live experiments. In *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, pages 507 –512, July 2008.

[18] Marinho P Barcellos, Rodolfo S Antunes, Hisham H Muhammad, and Ruthiano S Munaretti. Beyond network simulators: Fostering novel distributed applications and protocols through extendible design. *Journal of Network and Computer Applications*, 35(1):328–339, 2012.

[19] Luiz A. Barroso and Urs Holzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007. ISSN 0018-9162. doi: 10.1109/MC.2007.443.

[20] Luiz André Barroso. The price of performance. *Queue*, 3(7):48–53, September 2005. ISSN 1542-7730. doi: 10.1145/1095408.1095420. URL http://doi.acm.org/10.1145/1095408.1095420.

[21] Christian Belady, Andy Rawson, John Pfleuger, and Tahir Cader. Green grid data center power efficiency metrics: Pue and dcie. Technical report, Technical report, Green Grid, 2008.

[22] Christian Belady, Dan Azevedo, Michael Patterson, Jack Pouchet, and Roger Tipley. Carbon usage effectiveness (CUE): a green grid data center sustainability metric. *White Paper*, 32, 2010.

[23] Christian L. Belady. In the data center, power and cooling costs more than the it equipment it supports. *Electronics cooling*, 13(1):24, 2007.

[24] William H. Bell, David G. Cameron, Luigi Capozza, A. Paul Millar, Kurt Stockinger, and Floriano Zini. Optorsim - a grid simulator for studying dynamic data replication strategies. *International Journal of High Performance Computing Applications*, 2003.

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

[26] Frank Bellosa and Martin Steckermeier. The performance implications of locality information usage in shared-memory multiprocessors. *Journal of Parallel and Distributed Computing*, 37(1):113–121, 1996.

[27] Anton Beloglazov and Rajkumar Buyya. OpenStack neat: A framework for dynamic consolidation of virtual machines in OpenStack clouds–A blueprint. *Cloud Computing and Distributed Systems (CLOUDS) Laboratory*, 2012.

[28] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, and Albert Zomaya. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers*, 82(2):47–111, 2011.

[29] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755 – 768, 2012. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2011.04.017. URL http://www.sciencedirect.com/science/article/pii/S0167739X11000689. Special Section: Energy efficiency in large-scale distributed systems.

[30] Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Kerry Hill, and Jon Hiller. Exascale computing study: Technology challenges in achieving exascale systems. *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep*, 15, 2008.

[31] Vandy Berten and Emmanuel Jeannot. Modeling Resubmission in Unreliable Grids: the Bottom-Up Approach. In *Seventh International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks - heteroPar'09*, Delft, Netherlands, 2009.

[32] Paulo Bertoldi and Bogdan Anatasiu. Electricity Consumption and Efficiency Trends in European Union – Status Report 2009, 2009.

[33] CB Bhattacharya, Sankar Sen, and Daniel Korschun. Using corporate social responsibility to win the war for talent. *MIT Sloan management review*, 49, 2012.

[34] Christian Bienia and Kai Li. Parsec 2.0: A new benchmark suite for chip-multiprocessors. In *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, 2009.

[35] Jeremy Blackburn and Ken Christensen. A Simulation Study of a New Green BitTorrent. In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pages 1–6, 2009.

[36] Raffaele Bolla, Roberto Bruschi, Franco Davoli, and Flavio Cucchietti. Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures. *Communications Surveys & Tutorials, IEEE*, 13(2):223–244, 2011.

[37] BonFIRE Consortium. Bonfire (homepage), 2014. URL http://www.bonfire-project.eu/.

[38] Mohamed-Slim Bouguerra, Derrick Kondo, and Denis Trystram. On the Scheduling of Checkpoints in Desktop Grids. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, CCGrid '13, pages 305–313, 2011. doi: 10.1109/CCGrid.2011.63.

[39] Mohamed-Slim Bouguerra, Ana Gainaru, Leonardo Bautista Gomez, Franck Cappello, Satoshi Matsuoka, and Naoya Maruyama. Improving the computing efficiency of hpc systems using a combination of proactive and preventive checkpointing. In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 501–512, May 2013. doi: 10.1109/IPDPS.2013.74.

[40] Justin Bousquin, Marcos Esterman, and Sandra Rothenberg. Life cycle analysis in the printing industry: A review. In *NIP*, pages 709–715, 2011.

[41] Jeremy T. Bradley, Matthew Forshaw, Anton Stefanek, and Nigel Thomas. Time-inhomogeneous population models of a cycle-stealing distributed system. In *29th Annual UK Performance Engineering Workshop (UKPEW) 2013*, pages 8–13. Loughborough University, 2013.

[42] David Brooks, Vivek Tiwari, and Margaret Martonosi. *Wattch: a framework for architectural-level power analysis and optimizations*, volume 28. ACM, 2000.

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

[44] John S Bucy, Jiri Schindler, Steven W Schlosser, and Gregory R Ganger. The disksim simulation environment version 4.0 reference manual (cmu-pdl-08-101). *Parallel Data Laboratory*, page 26, 2008.

[45] Rajkumar Buyya and Manzur Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience*, 14(13):1175–1220, 2002.

[46] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, CŽsar A. F. De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011. ISSN 1097-024X. doi: 10.1002/spe.995. URL http://dx.doi.org/10.1002/spe.995.

[47] Franck Cappello, Al Geist, Bill Gropp, Laxmikant Kale, Bill Kramer, and Marc Snir. Toward exascale resilience. *Int. J. High Perform. Comput. Appl.*, 23(4):374–388, November 2009. ISSN 1094-3420. doi: 10.1177/1094342009347767. URL http://dx.doi.org/10.1177/1094342009347767.

[48] Enrique V. Carrera, Eduardo Pinheiro, and Ricardo Bianchini. Conserving disk energy in network servers. In *Proceedings of the 17th annual international conference on Supercomputing*, pages 86–97. ACM, 2003.

[49] Center for High Throughput Computing, Computer Sciences Department, University of Wisconsin-Madison, WI. HTCondor manual - Power Management. http://research.cs.wisc.edu/htcondor/manual/v8.2/3_15Power_Management.html.

[50] Center for High Throughput Computing, Computer Sciences Department, University of Wisconsin-Madison, WI. UW-Madison CS Dept. HTCondor Pool Policies, 2013. URL http://research.cs.wisc.edu/htcondor/uwcs/policy.html.

[51] Center for High Throughput Computing, Computer Sciences Department, University of Wisconsin-Madison, WI. HTCondor manual. http://www.cs.wisc.edu/condor/manual/, 2014. Oct 2014, University of Wisconsin.

[52] Joseph Chabarek, Joel Sommers, Paul Barford, Cristian Estan, David Tsiang, and Steve Wright. Power awareness in network design and routing. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008.

[53] Xinuo Chen and Stephen A. Jarvis. Analysing BitTorrent's seeding strategies. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 2, pages 140–149. IEEE, 2009.

[54] Yanpei Chen, Archana Ganapathi, and Randy H. Katz. To compress or not to compress-compute vs. IO tradeoffs for mapreduce energy efficiency. In *Proceedings of the first ACM SIGCOMM workshop on Green networking*, pages 23–28. ACM, 2010.

[55] Su-Hui Chiang, Andrea C. Arpaci-Dusseau, and Mary K. Vernon. The impact of more accurate requested runtimes on production job scheduling performance. In *Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing*, JSSPP '02, pages 103–127, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-00172-7.

[56] SungJin Choi, MaengSoon Baik, ChongSun Hwang, JoonMin Gil, and Heon-Chang Yu. Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment. In *Network Computing and Applications, 2004. (NCA 2004). Proceedings. Third IEEE International Symposium on*, NCA '04, pages 366–371, 2004. doi: 10.1109/NCA.2004.1347802.

[57] Victor Ciriza, Laurent Donini, Jean-Baptiste Durand, and Stéphane Girard. A statistical model for optimizing power consumption of printers. In *Presentation during a joint meeting of the Statistical Society of Canada & the Société Française de Statistique, in Otawa Congress Centre*, 2008.

[58] Walfredo Cirne and Francine Berman. A comprehensive model of the supercomputer workload. In *Proceedings of the Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop*, WWC '01, pages 140–148, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7803-7315-4.

[59] Walfredo Cirne, Francisco Brasileiro, Jacques Sauvé, Nazareno Andrade, Daniel Paranhos, Elizeu Santos-Neto, and Raissa Medeiros. Grid computing for bag of tasks applications. In *Proc. of the 3rd IFIP Conference on E-Commerce, E-Business and EGovernment*. Citeseer, 2003.

[60] Ryan Cochran, Can Hankendi, Ayse K. Coskun, and Sherief Reda. Pack & Cap: adaptive DVFS and thread packing under power caps. In *Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture*, pages 175–185. ACM, 2011.

[61] Bram Cohen. Incentives build robustness in bittorrent, 2003.

[62] Daniel Paranhos Da Silva, Walfredo Cirne, and Francisco Vilar Brasileiro. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In *Euro-Par 2003 Parallel Processing*, pages 169–180. Springer, 2003.

[63] Chris Dana, Danjue Li, David Harrison, and Chen-Nee Chuah. Bass: Bittorrent assisted streaming system for video-on-demand. In *Multimedia Signal Processing, 2005 IEEE 7th Workshop on*, pages 1–4. IEEE, 2005.

[64] John D. Davis, Suzanne Rivoire, Moises Goldszmidt, and Ehsan K Ardestani. Accounting for variability in large-scale cluster power models. Exascale Evaluation and Research Techniques Workshop (EXERT), 2011.

[65] John D. Davis, Suzanne Rivoire, Moises Goldszmidt, and Ehsan K Ardestani. No hardware required: building and validating composable highly accurate os-based power models. Technical report, Technical Report, Microsoft Research Technical Report No. MSR-TR-2011-89, 2011.

[66] Marcos Dias de Assuncao, Alexandre di Costanzo, and Rajkumar Buyya. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In *Proceedings of the 18th ACM international symposium on High performance distributed computing*, HPDC '09, pages 141–150, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-587-1.

[67] Raphael Y. De Camargo, Renato Cerqueira, and Fabio Kon. Strategies for storage of checkpointing data using non-dedicated repositories on grid systems. In *Proceedings of the 3rd international workshop on Middleware for grid computing*, pages 1–6. ACM, 2005.

[68] Raphael Y de Camargo, Fabio Kon, and Renato Cerqueira. Strategies for checkpoint storage on opportunistic grids. *Distributed Systems Online, IEEE*, 7(9):1–1, 2006.

[69] Răzvan Deaconescu, George Milescu, Bogdan Aurelian, Răzvan Rughiniş, and Nicolae Ţăpuş. A Virtualized Infrastructure for Automated BitTorrent Performance Testing and Evaluation. *Internation Journal on Advances in Systems and Measurements*, 2(2&3):236–247, 2009.

[70] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman, and John Good. The cost of doing science on the cloud: the montage example. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 50:1–50:12, Piscataway, NJ, USA, 2008. IEEE Press. ISBN 978-1-4244-2835-9.

[71] Department of Energy and Climate Change, UK Government. CRC Energy Efficiency Scheme Order: Table of Conversion Factors 2013/14. 2014.

[72] Nikolaos D. Doulamis, Anastasios D. Doulamis, Emmanouel A. Varvarigos, and Theodora A. Varvarigou. Fair scheduling algorithms in grids. *Parallel and Distributed Systems, IEEE Transactions on*, 18(11):1630–1648, 2007.

[73] Dimitris Economou, Suzanne Rivoire, Christos Kozyrakis, and Partha Ranganathan. Full-system power analysis and modeling for server environments. International Symposium on Computer Architecture-IEEE, 2006.

[74] Ifeanyi P. Egwutuoha, David Levy, Bran Selic, and Shiping Chen. A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems. *The Journal of Supercomputing*, 65(3):1302–1326, 2013.

[75] Mehdi El Mehdi Diouri, Oliver Gluck, Laurent Lefevre, and Frank Cappello. Energy considerations in checkpointing and fault tolerance protocols. In *Dependable Systems and Networks Workshops (DSN-W), 2012 IEEE/IFIP 42nd International Conference on*, DSN-W '12, pages 1–6, 2012.

[76] Elmootazbellah Nabil Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys (CSUR)*, 34(3):375–408, 2002.

[77] Tomoya Enokido, Ailixier Aikebaier, and Makoto Takizawa. Energy-efficient redundant execution of processes in a fault-tolerant cluster of servers. *International Journal of Parallel Programming*, 42(5):798–819, 2014.

[78] EPA Energy Star. Energy Star®Program Requirements for Imaging Equipment, June 2013.

[79] Pedro Evangelista, Marcelo Amaral, Charles Miers, Walter Goya, Marcos Simplicio, Tereza Carvalho, and Victor Souza. Ebitsim: An enhanced bittorrent simulation using omnet++ 4. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*, pages 437–440. IEEE, 2011.

[80] Nathan S. Evans and Christian Grothoff. Beyond simulation: Large-scale distributed emulation of p2p protocols. In *Proceedings of the 4th Conference on Cyber Security Experimentation and Test*, CSET'11, pages 4–4, Berkeley, CA, USA, 2011. USENIX Association. URL http://dl.acm.org/citation.cfm?id=2027999.2028003.

[81] Failure Trace Archive (FTA). Failure Trace Archive (FTA) (Homepage). http://fta.scem.uws.edu.au/, 2014.

[82] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *ACM SIGARCH Computer Architecture News*, volume 35, pages 13–23. ACM, 2007.

[83] Izaias Faria, Mario Dantas, Miriam A.M. Capretz, and Wilson Higashino. Network and Energy-Aware Resource Selection Model for Opportunistic Grids. In *Convergence of Distributed Clouds, Grids and their Management (CDCGM) track of Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2014 IEEE 23rd International Workshop on*, CDCGM '14, 2014.

[84] Dror G. Feitelson, Dan Tsafrir, and David Krakov. Experience with using the Parallel Workloads Archive. *J. Parallel & Distributed Comput.*, 74(10):2967–2982, Oct 2014. doi: 10.1016/j.jpdc.2014.06.013.

[85] Michal Feldman, Kevin Lai, and Li Zhang. The proportional-share allocation market for computational resources. *Parallel and Distributed Systems, IEEE Transactions on*, 20(8):1075–1088, 2009.

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

[87] Kurt B. Ferreira, Rolf Riesen, Patrick Bridges, Dorian Arnold, and Ron Brightwell. Accelerating incremental checkpointing for extreme-scale computing. *Future Generation Computer Systems*, 30:66–77, 2014.

[88] Matthew Forshaw and Nigel Thomas. A novel approach to energy efficient content distribution with BitTorrent. In *Computer Performance Engineering, Lecture Notes in Computer Science (LNCS) 7587*, pages 188–196. Springer-Verlag Berlin Heidelberg, 2013.

[89] Matthew Forshaw, A. Stephen McGough, and Nigel Thomas. On energy-efficient checkpointing in high-throughput cycle-stealing distributed systems. In *3rd International Conference on Smart Grids and Green IT Systems (SMARTGREENS)*, 2014.

[90] Matthew Forshaw, Nigel Thomas, and A. Stephen McGough. Trace-driven simulation for energy consumption in High Throughput Computing systems. In *Distributed Simulation and Real Time Applications (DS-RT), 2014 IEEE/ACM 18th International Symposium on*, 2014.

[91] Matthew Forshaw, A. Stephen McGough, and Nigel Thomas. Energy-efficient checkpointing in high-throughput cycle-stealing distributed systems. *Electronic Notes in Theoretical Computer Science*, 310:65–90, 2015.

[92] Larry Gadea. Using bittorrent for fast website deploys. In *CUSEC 2010*, January 2010.

[93] Antonietta Grasso, Jutta Willamowski, Victor Ciriza, and Yves Hoppenot. The personal assessment tool: a system providing environmental feedback to users of shared printers for providing environmental feedback. In *ICMLA*, pages 704–709. IEEE, 2010.

[94] Sudhanva Gurumurthi, Anand Sivasubramaniam, Mary Jane Irwin, Narayanan Vijaykrishnan, and Mahmut Kandemir. Using complete machine simulation for software power estimation: The softwatt approach. In *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, pages 141–150. IEEE, 2002.

[95] Sajjad Haider, Naveed Riaz Ansari, Muhammad Akbar, Mohammad Raza Perwez, and Khawaja MoyeezUllah Ghori. Fault Tolerance in Distributed Paradigms. 2011.

[96] Mor Harchol-Balter, Bianca Schroeder, Nikhil Bansal, and Mukesh Agrawal. Size-based scheduling to improve web performance. *ACM Trans. Comput. Syst.*, 21 (2):207–233, 2003. ISSN 0734-2071. doi: 10.1145/762483.762486. URL http://doi.acm.org/10.1145/762483.762486.

[97] Ligang He, Stephen A. Jarvis, Daniel P. Spooner, and Graham R. Nudd. Dynamic, capability-driven scheduling of dag-based real-time jobs in heterogeneous clusters. *International Journal of High Performance Computing and Networking (IJHPCN)*, 2(2/3/4):165–177, 2004.

[98] Ligang He, Stephen A. Jarvis, Daniel P. Spooner, David Bacigalupo, Guang Tan, and Graham R. Nudd. Mapping dag-based applications to multiclusters with background workload. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 2, pages 855–862 Vol. 2, May 2005. doi: 10.1109/CCGRID.2005.1558651.

[99] Ligang He, Stephen A. Jarvis, Daniel P. Spooner, and Graham R. Nudd. Performance evaluation of scheduling applications with dag topologies on multiclusters with independent local schedulers. In *International Parallel & Distributed Processing Symposium*, 2006.

[100] Taliver Heath, Bruno Diniz, Enrique V. Carrera, Wagner Meira, Jr., and Ricardo Bianchini. Energy conservation in heterogeneous server clusters. In *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, PPoPP '05, pages 186–195, New York, NY, USA, 2005. ACM. ISBN 1-59593-080-9.

[101] Thomas R. Henderson, Mathieu Lacage, and George F. Riley. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 2008.

[102] Magnus K. Herrlin. Rack cooling effectiveness in data centers and telecom central offices: The rack cooling index (rci). *Transactions-American Society of Heating Refrigerating and Air conditioning Engineers*, 111(2):725, 2005.

[103] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd and Toshiba Corporation. ACPI Specification. http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf, 2010.

[104] Mike Hibler, Robert Ricci, Leigh Stoller, Jonathon Duerig, Shashi Guruprasad, Tim Stack, Kirk Webb, and Jay Lepreau. Large-scale virtualization in the emulab network testbed. In *USENIX Annual Technical Conference*, pages 113–128, 2008.

[105] Helmut Hlavacs, Georges Da Costa, and Jean-Marc Pierson. Energy consumption of residential and professional switches. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 1, pages 240–246, Aug 2009. doi: 10.1109/CSE.2009.244.

[106] Helmut Hlavacs, Roman Weidlich, and Thomas Treutner. Energy efficient peer-to-peer file sharing. *The Journal of Supercomputing*, 62(3):1167–1188, 2012. ISSN 0920-8542. doi: 10.1007/s11227-011-0602-8. URL http://dx.doi.org/10.1007/s11227-011-0602-8.

[107] Fred Howell and Ross McNab. SimJava: A discrete event simulation library for java. *Simulation Series*, 30:51–56, 1998.

[108] Chung-Hsing Hsu and Stephen W. Poole. Power signature analysis of the SPECpower_ssj2008 benchmark. In *Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on*, pages 227–236. IEEE, 2011.

[109] Chung-Hsing Hsu and Stephen W Poole. Revisiting server energy proportionality. In *Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 834–840, Oct 2013. doi: 10.1109/ICPP.2013.99.

[110] Soonwook Hwang and Carl Kesselman. A flexible framework for fault tolerance in the grid. *Journal of Grid Computing*, 1:251–272, 2003. ISSN 1570-7873.

[111] Alexandru Iosup, Hui Li, Catalin Dumitrescu, Lex Wolters, and Dick Epema. The Grid Workload Format. http://gwa.ewi.tudelft.nl/fileadmin/pds/trace-archives/grid-workloads-archive/docs/TheGridWorkloadFormat_v001.pdf, 2006.

[112] Alexandru Iosup, Hui Li, Mathieu Jan, Shanny Anoep, Catalin Dumitrescu, Lex Wolters, and Dick HJ Epema. The grid workloads archive. *Future Generation Computer Systems*, 24(7):672–686, 2008.

[113] Mikel Izal, Guillaume Urvoy-Keller, Ernst W Biersack, Pascal A Felber, Anwar Al Hamra, and Luis Garces-Erice. Dissecting bittorrent: Five months in torrent's lifetime. In *Passive and Active Network Measurement*, pages 1–11. Springer, 2004.

[114] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[115] Peter James and Lisa Hopkinson. Results of the 2008 susteit survey. *Environmental management*, 50(27):25, 2008.

[116] Peter James and Lisa Hopkinson. Energy efficient printing and imaging in further and higher education. *A Best Practice Review prepared for the Joint Information Services Committee (JISC)*, 2008.

[117] Stephen A. Jarvis, Nigel Thomas, and Aad van Moorsel. Open issues in grid performability. *International Journal of Simulation and Process Modelling (IJSPM)*, 5(5):3–12, 2004.

[118] Bahman Javadi, Derrick Kondo, Alexandru Iosup, and Dick Epema. The Failure Trace Archive: Enabling the comparison of failure measurements and models of distributed systems. *Journal of Parallel and Distributed Computing*, 73(8):1208–1223, 2013.

[119] Henrik Thostrup Jensen and Jesper Ryge Leth. Automatic job resubmission in the nordugrid middleware. Technical report, Citeseer, 2004.

[120] James Patton Jones and Bill Nitzberg. Scheduling for parallel supercomputing: A historical perspective of achievable utilization. In *Proceedings of the Job Scheduling Strategies for Parallel Processing*, IPPS/SPDP '99/JSSPP '99, pages 1–16, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-66676-1.

[121] 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*, pages 31–40. ACM, 2010.

[122] Aman Kansal and Feng Zhao. Fine-grained energy profiling for power-aware application design. *ACM SIGMETRICS Performance Evaluation Review*, 36(2):26–31, 2008.

[123] Konstantinos Katsaros, Vasileios P Kemerlis, Charilaos Stais, and George Xylomenos. A BitTorrent module for the OMNeT++ simulator. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2009. MASCOTS'09. IEEE International Symposium on*, pages 1–10. IEEE, 2009.

[124] Konstantinos Kavoussanakis, Alastair Hume, Josep Martrat, Carmelo Ragusa, Michael Gienger, Konrad Campowsky, Gregory Van Seghbroeck, Constantino Vázquez, Celia Velayos, and Frédéric Gittler. Bonfire: the clouds and services testbed. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 2, pages 321–326. IEEE, 2013.

[125] Maria Kazandjieva, Brandon Heller, Philip Levis, and Christos Kozyrakis. Energy dumpster diving. In *Proc. 2nd Workshop on Power Aware Computing (HotPower'09)*, pages 1–5, 2009.

[126] Imre Kelényi, Ákos Ludányi, and Jukka K. Nurminen. Energy-efficient bittorrent downloads to mobile phones through memory-limited proxies. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 715–719, 2011.

[127] Sunirmal Khatua and Nandini Mukherjee. A Novel Checkpointing Scheme for Amazon EC2 Spot Instances. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 180–181. IEEE, 2013.

[128] Jack P. C. Kleijnen. Verification and validation of simulation models. *European Journal of Operational Research*, 82(1):145 – 162, 1995. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/0377-2217(94)00016-6. URL http://www.sciencedirect.com/science/article/pii/0377221794000166.

[129] Dzmitry Kliazovich, Pascal Bouvry, Yury Audzevich, and Samee Ullah Khan. Greencloud: A packet-level simulator of energy-aware cloud computing data centers. In *GLOBECOM*, pages 1–5, 2010.

[130] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283, 2012.

[131] Ricardo Koller, Akshat Verma, and Anindya Neogi. Wattapp: an application aware power meter for shared data centers. In *Proceedings of the 7th international conference on Autonomic computing*, pages 31–40. ACM, 2010.

[132] Andrew Krioukov, Prashanth Mohan, Sara Alspaugh, Laura Keys, David Culler, and Randy H. Katz. Napsac: design and implementation of a power-proportional web cluster. In *Proceedings of the first ACM SIGCOMM workshop on Green networking*, Green Networking '10, pages 15–22, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0196-1.

[133] Andrew Krioukov, Prashanth Mohan, Sara Alspaugh, Laura Keys, David Culler, and Randy Katz. Napsac: Design and implementation of a power-proportional web cluster. *ACM SIGCOMM computer communication review*, 41(1):102–108, 2011.

[134] Klaus-Dieter Lange. Identifying Shades of Green: The SPECpower Benchmarks. *IEEE Computer*, 42(3):95–97, 2009.

[135] Larry Gadea and Matt Freels. Murder: Fast datacenter code deploys using BitTorrent. https://blog.twitter.com/2010/murder-fast-datacenter-code-deploys-using-bittorrent, 2010.

[136] Preston V Lee and Valentin Dinu. Bittorious: global controlled genomics data publication, research and archiving via bittorrent extensions. *BMC bioinformatics*, 15(1):6601, 2014.

[137] Uichin Lee, Ivica Rimac, Daniel Kilper, and Volker Hilt. Toward energy-efficient content dissemination. *Network, IEEE*, 25(2):14–19, 2011.

[138] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. Power capping: a prelude to power shifting. *Cluster Computing*, 11(2):183–195, 2008. ISSN 1386-7857. doi: 10.1007/s10586-007-0045-4. URL http://dx.doi.org/10.1007/s10586-007-0045-4.

[139] Arnaud Legrand and Loris Marchal. Scheduling distributed applications: The simgrid simulation framework. In *In Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid*, pages 138–145, 2003.

[140] Daan Leijen and Erik Meijer. Parsec: Direct style monadic parser combinators for the real world. Technical report, Technical Report UU-CS-2001-27, Department of Computer Science, Universiteit Utrecht, 2001.

[141] Bo Li, Jianxin Li, Jinpeng Huai, Tianyu Wo, Qin Li, and Liang Zhong. Enacloud: An energy-saving application live placement approach for cloud computing environments. In *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*, pages 17–24. IEEE, 2009.

[142] Jiangtian Li, Amey Deshpande, Jagan Srinivasan, and Xiaosong Ma. Energy and performance impact of aggressive volunteer computing with multi-core computers. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*, MASCOTS '09, pages 1–10, 2009. doi: 10.1109/MASCOT.2009.5366968.

[143] Seung-Hwan Lim, Bikash Sharma, Gunwoo Nam, Eun Kyoung Kim, and Chita R Das. Mdcsim: A multi-tier data center simulation, platform. In *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, pages 1–9, 2009. doi: 10.1109/CLUSTR.2009.5289159.

[144] Minghong Lin, Adam Wierman, Lachlan LH Andrew, and Eno Thereska. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking (TON)*, 21(5):1378–1391, 2013.

[145] Diane Lingrand and Johan Montagnat. Efficient resubmission strategies to design robust grid production environments. In *e-Science (e-Science), 2010 IEEE Sixth International Conference on*, pages 198–205. IEEE, 2010.

[146] Michael J Litzkow, Miron Livny, and Matt W Mutka. Condor-a hunter of idle workstations. In *8th International Conference on Distributed Computing Systems*, ICDCS '88, pages 104–111, 1998.

[147] Yong Liu, Yang Guo, and Chao Liang. A survey on peer-to-peer video streaming systems. *Peer-to-peer Networking and Applications*, 1(1):18–28, 2008.

[148] Charng-Da Lu. *Scalable diskless checkpointing for large parallel systems*. PhD thesis, University of Illinois at Urbana-Champaign, 2005.

[149] Guoming Lu, Ziming Zheng, and Andrew A. Chien. When is multi-version checkpointing needed? In *Proceedings of the 3rd Workshop on Fault-tolerance for HPC at extreme scale*, pages 49–56. ACM, 2013.

[150] Andre Luckow and Bettina Schnor. Adaptive checkpoint replication for supporting the fault tolerance of applications in the grid. In *Network Computing and Applications, 2008. NCA'08. Seventh IEEE International Symposium on*, pages 299–306. IEEE, 2008.

[151] Priya Mahadevan, Puneet Sharma, Sujata Banerjee, and Parthasarathy Ranganathan. A power benchmarking framework for network devices. In *Proceedings of the 8th International IFIP-TC 6 Networking Conference*, NETWORKING '09, pages 795–808, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-01398-0.

[152] S. Majumdar, D. L. Eager, and R. B. Bunt. *Scheduling in Multiprogrammed Parallel Systems*. SIGMETRICS '88. ACM, New York, NY, USA, 1988. ISBN 0-89791-254-3. doi: 10.1145/55595.55608. URL http://doi.acm.org/10.1145/55595.55608.

[153] Paul Marshall, Kate Keahey, and Tim Freeman. Elastic site: Using clouds to elastically extend site resources. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 43 –52, may 2010. doi: 10.1109/CCGRID.2010.80.

[154] MathWorks. MATLAB. http://www.mathworks.co.uk/.

[155] Michael Mattess, Christian Vecchiola, and Rajkumar Buyya. Managing peak loads by leasing cloud infrastructure services from a spot market. In *Proceedings of the 2010 IEEE 12th International Conference on High Performance Computing and Communications*, HPCC '10, pages 180–188, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4214-0.

[156] Cathy McCann, Raj Vaswani, and John Zahorjan. A dynamic processor allocation policy for multiprogrammed shared-memory multiprocessors. *ACM Transactions on Computer Systems (TOCS)*, 11(2):146–178, 1993.

[157] A. S. McGough, Paul Robinson, Clive Gerrard, Paul Haldane, Sindre Hamlander, Dave Sharples, Dan Swan, and Stuart Wheater. Intelligent power management over large clusters. In *International Conference on Green Computing and Communications (GreenCom2010)*, 2010.

[158] A. Stephen McGough, Clive Gerrard, Paul Haldane, Dave Sharples, Dan Swan, Paul Robinson, Sindre Hamlander, and Stuart Wheater. Intelligent Power Management Over Large Clusters. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 88–95, 2010. doi: 10.1109/GreenCom-CPSCom.2010.131.

[159] A. Stephen McGough, Matthew Forshaw, Gerrard Clive, Wheater Stuart, Allen Ben, and Robinson Paul. Reduction of wasted energy in a volunteer computing system through reinforcement learning. *Sustainable Computing, Informatics and Systems*, 2014. doi: 10.1016/j.suscom.2014.08.014.

[160] Andrew Stephen McGough, Clive Gerrard, Jonathan Noble, Paul Robinson, and Stuart Wheater. Analysis of power-saving techniques over a large multi-use cluster. In *International Conference on Cloud and Green Computing (CGC2011)*, 2011.

[161] Andrew Stephen McGough, Matthew Forshaw, Clive Gerrard, Paul Robinson, and Stuart Wheater. Analysis of power-saving techniques over a large multi-use cluster with variable workload. *Concurrency and Computation: Practice and Experience*, 25(18):2501–2522, 2013. ISSN 1532-0634. URL http://dx.doi.org/10.1002/cpe.3082.

[162] Andrew Stephen McGough, Matthew Forshaw, Clive Gerrard, Stuart Wheater, Ben Allen, and Paul Robinson. Comparison of a cost-effective virtual cloud cluster with an existing campus cluster. *Future Generation Computer Systems*, 2014. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2014.07.002.

[163] A.S. McGough, M. Forshaw, C. Gerrard, and S. Wheater. Reducing the number of miscreant tasks executions in a multi-use cluster. In *Cloud and Green Computing (CGC), 2012 Second International Conference on*, pages 296–303, 2012. doi: 10.1109/CGC.2012.111.

[164] David Meisner and Thomas F Wenisch. Peak power modeling for data center servers with switched-mode power supplies. In *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, pages 319–324. IEEE, 2010.

[165] David Meisner, Brian T Gold, and Thomas F Wenisch. PowerNap: eliminating server idle power. *ACM SIGARCH Computer Architecture News*, 37(1):205–216, 2009.

[166] R. Melhem, D. Mosse, and E. Elnozahy. The interplay of power management and fault recovery in real-time systems. *Computers, IEEE Transactions on*, 53(2):217–231, 2004. ISSN 0018-9340. doi: 10.1109/TC.2004.1261830.

[167] Víctor Méndez and Felix García. Sicogrid: A complete grid simulator for scheduling and algorithmical research, with emergent artificial intelligence data algorithms.

[168] Bryan Mills, Ryan E. Grant, Kurt B. Ferreira, and Rolf Riesen. Evaluating energy savings for checkpoint/restart. In *Proceedings of the 1st International Workshop on Energy Efficient Supercomputing*, E2SC '13, pages 6:1–6:8, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2504-2. doi: 10.1145/2536430.2536432. URL http://doi.acm.org/10.1145/2536430.2536432.

[169] Bryan Mills, Taieb Znati, and Rami Melhem. Shadow computing: An energy-aware fault tolerant computing model. In *Computing, Networking and Communications (ICNC), 2014 International Conference on*, pages 73–77. IEEE, 2014.

[170] Timo Minartz, Julian Kunkel, and Thomas Ludwig. Simulation of power consumption of energy efficient cluster hardware. *Computer Science - Research and Development*, 25:165–175, 2010. ISSN 1865-2034.

[171] Jennifer Mitchell-Jackson, Jonathan G Koomey, Bruce Nordman, and Michele Blazek. Data center power requirements: measurements from silicon valley. *Energy*, 28(8):837–850, 2003.

[172] Justin D Moore, Jeffrey S Chase, Parthasarathy Ranganathan, and Ratnesh K Sharma. Making Scheduling "Cool": Temperature-Aware Workload Placement in Data Centers. In *USENIX annual technical conference, General Track*, pages 61–75, 2005.

[173] A.W. Mu'alem and D.G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. *Parallel and Distributed Systems, IEEE Transactions on*, 12(6):529 –543, jun 2001. ISSN 1045-9219.

[174] Matt W. Mutka and Miron Livny. The available capacity of a privately owned workstation environment. *Performance Evaluation*, 12(4):269 – 284, 1991. ISSN 0166-5316. doi: http://dx.doi.org/10.1016/0166-5316(91)90005-N. URL http://www.sciencedirect.com/science/article/pii/016653169190005N.

[175] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers. The state of peer-to-peer simulators and simulations. *SIGCOMM Comput. Commun. Rev.*, 37(2):95–98, March 2007. ISSN 0146-4833. doi: 10.1145/1232919.1232932. URL http://doi.acm.org/10.1145/1232919.1232932.

[176] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *NSDI*, volume 8, pages 323–336, 2008.

[177] Sergiu Nedevschi, Sylvia Ratnasamy, and Jitendra Padhye. Hot data centers vs. cool peers. In *Proceedings of the 2008 conference on Power aware computing and systems*, HotPower'08, pages 8–8, Berkeley, CA, USA, 2008. USENIX Association.

[178] Rolf Neugebauer and Derek McAuley. Energy is just another resource: Energy accounting and energy pricing in the Nemesis OS. In *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*, pages 67–72. IEEE, 2001.

[179] Thai Ha Nguyen, Matthew Forshaw, and Nigel Thomas. Operating policies for energy efficient dynamic server allocation. In *30th Annual UK Performance Engineering Workshop (UKPEW 2014)*, 2014.

[180] Tapio Niemi, Jukka Kommeri, Kalle Happonen, Jukka Klem, and Ari-Pekka Hameri. Improving energy-efficiency of grid computing clusters. In *Advances in Grid and Pervasive Computing*, volume 5529 of *LNCS*, pages 110–118. 2009.

[181] Shuangcheng Niu, Jidong Zhai, Xiaosong Ma, Mingliang Liu, Yan Zhai, Wenguang Chen, and Weimin Zheng. Employing checkpoint to improve job scheduling in large-scale systems. In *Job Scheduling Strategies for Parallel Processing*, pages 36–55. Springer, 2013.

[182] Daniel Nurmi, John Brevik, and Richard Wolski. Minimizing the network overhead of checkpointing in cycle-harvesting cluster environments. In *Cluster Computing, 2005. IEEE International*, pages 1–10. IEEE, 2005.

[183] Adam J. Oliner, Larry Rudolph, and Ramendra K. Sahoo. Cooperative checkpointing: A robust approach to large-scale systems reliability. In *Proceedings of the 20th Annual International Conference on Supercomputing*, ICS '06, pages 14–23, New York, NY, USA, 2006. ACM. ISBN 1-59593-282-8. URL http://doi.acm.org/10.1145/1183401.1183406.

[184] Steven Pelley, David Meisner, Thomas F Wenisch, and James W VanGilder. Understanding and abstracting total data center power. In *Workshop on Energy-Efficient Design*, 2009.

[185] Steven Pelley, David Meisner, Pooya Zandevakili, Thomas F Wenisch, and Jack Underwood. Power routing: dynamic power provisioning in the data center. In *ACM Sigplan Notices*, volume 45, pages 231–242. ACM, 2010.

[186] Cris Pettey. Gartner estimates ict industry accounts for 2 percent of global co2 emissions, 2007. URL http://www.gartner.com/newsroom/id/503867.

[187] Jean-Marc Pierson. Allocating resources greenly: reducing energy consumption or reducing ecological impact? In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pages 127–130. ACM, 2010.

[188] Meikel Poess, Raghunath Othayoth Nambiar, Kushagra Vaid, John M Stephens Jr, Karl Huppler, and Evan Haines. Energy benchmarks: a detailed analysis. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pages 131–140. ACM, 2010.

[189] Lesandro Ponciano and Francisco Brasileiro. On the impact of energy-saving strategies in opportunistic grids. In *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, pages 282–289. IEEE, 2010.

[190] David Price. An estimate of infringing use of the internet. Technical report, Envisional Ltd, 2011.

[191] Dongyu Qiu and Rayadurgam Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 367–378. ACM, 2004.

[192] Mustafizur Rahman, Rajiv Ranjan, Rajkumar Buyya, and Boualem Benatallah. A taxonomy and survey on autonomic management of applications in grid computing environments. *Concurrency and computation: practice and experience*, 23(16):1990–2019, 2011.

[193] Rajesh Raman, Miron Livny, and Marvin Solomon. Matchmaking: Distributed resource management for high throughput computing. In *High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on*, pages 140–146. IEEE, 1998.

[194] Kavitha Ranganathan and Ian Foster. Identifying dynamic replication strategies for a high-performance data grid. In *Grid Computing, GRID 2001*, pages 75–86. Springer, 2001.

[195] Parthasarathy Ranganathan, Phil Leech, David Irwin, and Jeffrey Chase. Ensemble-level power management for dense blade servers. In *ACM SIGARCH Computer Architecture News*, volume 34, pages 66–77. IEEE Computer Society, 2006.

[196] Sherief Reda and Abdullah N. Nowroz. Power modeling and characterization of computing devices: A survey. *Foundations and Trends in Electronic Design Automation*, 6(2):121–216, 2012.

[197] Xiaojuan Ren, Rudolf Eigenmann, and Saurabh Bagchi. Failure-aware Checkpointing in Fine-grained Cycle Sharing Systems. In *Proceedings of the 16th International Symposium on High Performance Distributed Computing*, HPDC '07, pages 33–42, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-673-8. doi: 10.1145/1272366.1272372. URL http://doi.acm.org/10.1145/1272366.1272372.

[198] Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. A comparison of high-level full-system power models. *HotPower*, 8:3–3, 2008.

[199] Frederick Ryckbosch, Stijn Polfliet, and Lieven Eeckhout. Trends in Server Energy Proportionality. *Computer*, 44(9):69 –72, 2011. ISSN 0018-9162.

[200] Ramendra K. Sahoo, Mark S. Squillante, Anand Sivasubramaniam, and Yanyong Zhang. Failure data analysis of a large-scale heterogeneous server environment. In *Dependable Systems and Networks, 2004 International Conference on*, pages 772–781. IEEE, 2004.

[201] Takafumi Saito, Kento Sato, Hitoshi Sato, and Satoshi Matsuoka. Energy-aware I/O Optimization for Checkpoint and Restart on a NAND Flash Memory System. In *Proceedings of the 3rd Workshop on Fault-tolerance for HPC at Extreme Scale*, FTXS '13, pages 41–48, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1983-6. doi: 10.1145/2465813.2465822.

[202] Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. Enerj: Approximate data types for safe and general low-power computation. In *ACM SIGPLAN Notices*, volume 46, pages 164–174. ACM, 2011.

[203] Kiavash Satvat, Matthew Forshaw, Feng Hao, and Ehsan. Toreini. On the privacy of private browsing - a forensic approach. In *8th DPM International Workshop on Data Privacy Management*. Royal Holloway, University of London, 2013.

[204] Kiavash Satvat, Matthew Forshaw, Feng Hao, and Ehsan Toreini. On the privacy of private browsing - a forensic approach. *Journal of Information Security and Applications*, 19(1):88–100, 2014.

[205] Richard Sawyer. Calculating total power requirements for data centers. *American Power Conversion, Tech. Rep*, 70:80–90, 2004.

[206] Bianca Schroeder and Garth A. Gibson. The computer failure data repository (CFDR). https://www.usenix.org/cfdr. Accessed: 2014-08-07.

[207] Bianca Schroeder and Garth A. Gibson. The computer failure data repository (CFDR). In *Workshop on Reliability Analysis of System Failure Data (RAF'07), MSR Cambridge, UK*, 2007.

[208] Bianca Schroeder and Garth A. Gibson. A large-scale study of failures in high-performance computing systems. *Dependable and Secure Computing, IEEE Transactions on*, 7(4):337–350, 2010.

[209] Herb Schwetman. CSIM: a C-based process-oriented simulation language. In *Proceedings of the 18th conference on Winter simulation*, pages 387–396. ACM, 1986.

[210] Li Shang, Li-Shiuan Peh, and Niraj K Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*, pages 91–102. IEEE, 2003.

[211] Ratnesh K Sharma, Cullen E Bash, Chandrakant D Patel, Richard J Friedrich, and Jeffrey S Chase. Balance of power: Dynamic thermal management for internet data centers. *Internet Computing, IEEE*, 9(1):42–49, 2005.

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

[213] Ian C. Smith. Experiences with running matlab applications on a power-saving condor pool. http://www.liv.ac.uk/csd/escience/condor/cardiff_condor.pdf. Oct. 2009.

[214] Warren Smith, Ian Foster, and Valerie Taylor. Scheduling with advanced reservations. In *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, pages 127–132. IEEE, 2000.

[215] Marvin Solomon. The ClassAd Language Reference Manual. *Computer Sciences Department, University of Wisconsin, Madison, WI, Oct*, 2003.

[216] Sourceforge project. The iperf project. http://iperf.sourceforge.net/.

[217] Leandro Souza, Ana Ripoll, X. Y. Yang, Porfidio Hernandez, and Fernando Cores. Designing a video-on-demand system for a brazilian high speed network. In *Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on*, pages 43–43. IEEE, 2006.

[218] Srividya Srinivasan, Rajkumar Kettimuthu, Vijay Subramani, and P Sadayappan. Characterization of backfilling strategies for parallel job scheduling. In *Parallel Processing Workshops, 2002. Proceedings. International Conference on*, pages 514–519. IEEE, 2002.

[219] Standard Performance Evaluation Corporation (SPEC). SPECpower_ssj2008. http://www.spec.org/power_ssj2008/.

[220] Standard Performance Evaluation Corporation (SPEC). Spec virt sc2013 (homepage), 2014. URL http://www.spec.org/virt_sc2013/.

[221] Anton Stefanek, Richard A Hayden, and Jeremy T. Bradley. Gpa-a tool for fluid scalability analysis of massively parallel systems. In *Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on*, pages 147–148. IEEE, 2011.

[222] Anton Stefanek, Uli Harder, and Jeremy T. Bradley. Energy consumption in the office. In *Computer Performance Engineering*, pages 224–236. Springer, 2013.

[223] Christopher Stewart and Kai Shen. Some joules are more precious than others: Managing renewable energy in the datacenter. In *Proceedings of the Workshop on Power Aware Computing and Systems*, 2009.

[224] Storage Performance Council (SPC). Storage performance council (spc) benchmark specifications, 2014. URL http://www.storageperformance.org/specs/.

[225] Robert F. Sullivan. Alternating cold and hot aisles provides more reliable cooling for server farms. *Uptime Institute*, 2000.

[226] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford book. Bradford Book, 1998. ISBN 9780262193986.

[227] Todd Tannenbaum, Derek Wright, Karen Miller, and Miron Livny. Condor: a distributed job scheduler. In *Beowulf cluster computing with Linux*, pages 307–350. MIT press, 2001.

[228] George Terzopoulos and Helen D. Karatza. Dynamic voltage scaling scheduling on power-aware clusters under power constraints. In *Distributed Simulation and Real Time Applications (DS-RT), 2013 IEEE/ACM 17th International Symposium on*, pages 72–78. IEEE, 2013.

[229] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: The condor experience. *Concurrency and Computation: Practice and Experience*, 17(2-4):323–356, 2005.

[230] The Condor Project. Condor week 2012. http://research.cs.wisc.edu/condor/CondorWeek2012/.

[231] The Green Grid. The Green Grid Opportunity: Decreasing datacenter and other IT usage patterns. Technical report, Technical report, The Green Grid, 2008.

[232] The MathWorks, Inc. http://www.mathworks.co.uk/help/stats/prctile.html, 2015.

[233] Ananta Tiwari, Michael Laurenzano, Joshua Peraza, Laura Carrington, and Allan Snavely. Green queue: Customized large-scale clock frequency scaling. In *Cloud and Green Computing (CGC), 2012 Second International Conference on*, pages 260–267. IEEE, 2012.

[234] Niraj Tolia, Zhikui Wang, Manish Marwah, Cullen Bash, Parthasarathy Ranganathan, and Xiaoyun Zhu. Delivering energy proportionality with non energy-proportional systems: optimizing the ensemble. In *Proceedings of the 2008 conference on Power aware computing and systems*, HotPower'08, pages 2–2, Berkeley, CA, USA, 2008. USENIX Association.

[235] Niraj Tolia, Zhikui Wang, Manish Marwah, Cullen Bash, Parthasarathy Ranganathan, and Xiaoyun Zhu. Delivering energy proportionality with non energy-proportional systems-optimizing the ensemble. *HotPower*, 8:2–2, 2008.

[236] Transaction Processing Performance Council (TPC). Transaction processing performance council (tpc) tpc-energy (homepage), 2014. URL http://www.tpc.org/tpc_energy/default.asp.

[237] Renlong Tu, Xin Wang, and Yue Yang. Energy-saving model for sdn data centers. *The Journal of Supercomputing*, pages 1–19, 2014.

[238] Eduard Turcan and Ross L. Graham. Getting the most from accountability in P2P. In *1st International Conference on Peer-to-Peer Computing (P2P'01)*, pages 95–96, 2001.

[239] UK Research Council End Use Energy Demand (EUED) Centres. EU Energy Efficiency target should be more ambitious and legally binding say leading energy demand researchers. http://www.eued.ac.uk/, 2014.

[240] Osman S. Unsal, Israel Koren, and C. Mani Krishna. Towards energy-aware software-based fault tolerance in real-time systems. In *Low Power Electronics and Design, 2002. ISLPED'02. Proceedings of the 2002 International Symposium on*, pages 124–129, 2002.

[241] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 228 – 235, July 2010. doi: 10.1109/CLOUD.2010.58.

[242] András Varga et al. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM 2001)*, volume 9, page 185. sn, 2001.

[243] Georgios Varsamopoulos and Sandeep KS Gupta. Energy proportionality and the future: Metrics and directions. In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pages 461–467. IEEE, 2010.

[244] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pMapper: power and migration cost aware application placement in virtualized systems. In *Middleware 2008*, pages 243–264. Springer, 2008.

[245] Akshat Verma, Puneet Ahuja, and Anindya Neogi. Power-aware dynamic placement of hpc applications. In *Proceedings of the 22nd annual international conference on Supercomputing*, pages 175–184. ACM, 2008.

[246] Akshat Verma, Gargi Dasgupta, Tapan Kumar Nayak, Pradipta De, and Ravi Kothari. Server workload analysis for power minimization using consolidation. In *Proceedings of the 2009 conference on USENIX Annual technical conference*, pages 28–28. USENIX Association, 2009.

[247] Gustavo M. D. Vieira and Luiz E. Buzato. Distributed checkpointing: Analysis and benchmarks. In *Proceedings of the 24th Brazilian Symposium on Computer Networks, SBRC*, volume 6, 2006.

[248] Angelos Vlavianos, Marios Iliofotou, and Michalis Faloutsos. Bitos: Enhancing bittorrent for supporting streaming applications. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–6. IEEE, 2006.

[249] William Voorsluys and Rajkumar Buyya. Reliable provisioning of spot instances for compute-intensive applications. In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pages 542–549. IEEE, 2012.

[250] Lizhe Wang and Samee Ullah Khan. Review of performance metrics for green data centers: a taxonomy study. *The Journal of Supercomputing*, 63(3):639–656, 2013. ISSN 0920-8542. doi: 10.1007/s11227-011-0704-3. URL http://dx.doi.org/10.1007/s11227-011-0704-3.

[251] William A. Ward, Jr., Carrie L. Mahood, and John E. West. Scheduling jobs on parallel systems using a relaxed backfill strategy. In *Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing*, JSSPP '02, pages 88–102, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-00172-7.

[252] Torsten Wilde, Axel Auweter, and Hayk Shoukourian. The 4 pillar framework for energy efficient hpc data centers. *Computer Science-Research and Development*, pages 1–11, 2013.

[253] Jutta K. Willamowski, Yves Hoppenot, and Antonietta Grasso. Promoting sustainable print behavior. In *CHI'13*, pages 1437–1442. ACM, 2013.

[254] Daniel Wong and Murali Annavaram. Knightshift: Scaling the energy proportionality wall through server-level heterogeneity. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 119–130. IEEE Computer Society, 2012.

[255] Yang Yang and Henri Casanova. Umr: A multi-round algorithm for scheduling divisible workloads. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pages 9–pp. IEEE, 2003.

[256] Sangho Yi, Artur Andrzejak, and Derrick Kondo. Monetary cost-aware checkpointing and migration on Amazon cloud spot instances. *Services Computing, IEEE Transactions on*, 5(4):512–524, 2012.

[257] Jia Yu and Rajkumar Buyya. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing*, 3(3-4):171–200, 2005.

[258] John Zedlewski, Sumeet Sobti, Nitin Garg, Fengzhou Zheng, Arvind Krishnamurthy, and Randolph Y. Wang. Modeling hard-disk power consumption. In *FAST*, volume 3, pages 217–230, 2003.

[259] Heng Zeng, Carla S. Ellis, Alvin R. Lebeck, and Amin Vahdat. ECOSystem: Managing energy as a first class operating system resource. In *ACM SIGPLAN Notices*, volume 37, pages 123–132. ACM, 2002.

[260] Yanyong Zhang, Mark S. Squillante, Anand Sivasubramaniam, and Ramendra K. Sahoo. Performance implications of failures in large-scale cluster scheduling. In *Job Scheduling Strategies for Parallel Processing*, pages 233–252. Springer, 2005.

[261] Ying Zhang and Krishnendu Chakrabarty. Energy-aware adaptive checkpointing in embedded real-time systems. In *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pages 918–923, 2003. doi: 10.1109/DATE.2003.1253723.

[262] Wei Zheng, Ana P. Centeno, Frederic Chong, and Ricardo Bianchini. Logstore: toward energy-proportional storage servers. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, pages 273–278. ACM, 2012.

[263] Ao Zhou, Shangguang Wang, Qibo Sun, Hua Zou, and Fangchun Yang. Ftcloudsim: A simulation tool for cloud service reliability enhancement mechanisms. In *Proceedings Demo & Poster Track of ACM/IFIP/USENIX International Middleware Conference*, MiddlewareDPT '13, pages 2:1–2:2, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2549-3. URL http://doi.acm.org/10.1145/2541614.2541616.

[264] Stylianos Zikos and Helen D Karatza. Performance and energy aware cluster-level scheduling of compute-intensive jobs with unknown service times. *Simulation Modelling Practice and Theory*, 19(1):239–250, 2011.