CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2015; **00**:1–32
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

HTC-Sim: A trace-driven simulation framework for energy consumption in High Throughput Computing systems

M. Forshaw[1]*, A.S. McGough[2], N. Thomas[1]

[1]*School of Computing Science, Newcastle University, Newcastle upon Tyne, NE1 7RU*
[2]*School of Engineering and Computing Sciences, Durham University, Durham, DH1 3LE*

SUMMARY

High Throughput Computing (HTC) is a powerful paradigm allowing vast quantities of independent work to be performed simultaneously across many loosely coupled computers. These systems often exploit the idle time available on computers provisioned for other purposes – volunteer computing. However, until recently, little evaluation has been performed on the energy impact of HTC. Many organisations now seek to minimise energy consumption across their IT infrastructure. However, it is unclear how this will affect the usability of HTC systems especially when exploiting volunteer computers. We present here HTC-Sim, a simulation system which allows the evaluation of different energy reduction policies across an HTC system. We model systems which are comprised of both collections of computational resources dedicated to HTC work and resources provided through volunteer computing – a Desktop Grid. We demonstrate that our simulation software scales linearly with increasing HTC workload. We go further to evaluate a number of resource selection policies in terms of the overheads / slowdown incurred, and the energy impact on the HTC system. Copyright © 2015 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Simulation, Energy, High-Throughput Computing

1. INTRODUCTION

In recent years a number of large scale distributed computing paradigms have been developed which facilitate parallel processing of computational work at a vast scale. These systems generally involve the analysis of large quantities of data (so-called 'Big data'), undertaking large scale simulations or large collections of smaller simulations. The underlying computer facilities supporting these applications typically consist of multiple processing units employed to solve a single problem. Situations where it is possible to subdivide this problem into separate jobs that can be executed independently is often termed as a pleasingly parallel problem which can be solved using High Throughput Computing (HTC). There exists a number of HTC systems, including HTCondor [1] and BOINC [2], which are typically used to help solve scientific problems at all scales.

Originally HTC systems were either created as dedicated resources or as shared facilities (Desktop Grids), which are constantly powered up, whether servicing jobs or sitting idle. Shared usage systems have the added complication that the execution of HTC jobs might be affected by other users. For example, in University research environments the HTC system is typically based on student accessible laboratories and, when these are required for teaching (or by individuals), the HTC jobs are expected to relinquish the resources – either by job eviction or suspension. Here we see job evictions as an unsuccessful execution.

*Correspondence to: School of Computing Science, Newcastle University, Newcastle upon Tyne, NE1 7RU

The performance and reliability challenges of large scale systems are generally well understood [3]. However, energy issues have received less attention but are of increasing concern due to high running costs, political pressures and potential environmental impact. The power consumption of western European data centres alone was estimated at 56 TWh/year in 2007 and is projected to double by 2020 [4] making it imperative to improve energy efficiency of IT operations. Hence, unsuccessful execution of HTC jobs can be considered not only as a performability problem, but also as an energy cost. In addition, measures to improve the reliability of HTC job execution, such as replication or checkpointing, carry an extra energy cost which needs to be considered against the performance benefits. HTC systems would therefore appear to offer significant potential for energy efficiency savings.

Many energy aware approaches have been suggested. However, resource management policies which favour reduced energy consumption over other attributes could have a significant impact on performance, reliability and availability of resources for HTC users. For example, placing idle resources into a sleep state too rapidly – without the ability to wake them up – could lead to increased failures or HTC resource starvation. Conversely, policies which emphasise availability may ultimately offer very little by way of energy savings. Therefore it is necessary to understand the factors which lead to intelligent selection of energy saving policies to apply to the underlying hardware and the management of HTC users and jobs. Suitable policies would control such factors as *a*) the energy state of a collection of resources; *b*) how to deal with jobs which fail to complete, and *c*) the choice of resources employed which potentially minimises energy consumption. Such policies become especially powerful on shared facilities where interactive users may lead to the eviction of an HTC job which could still complete; therefore selecting a less used (possibly higher energy) resource, or executing at a different time of day, may have a significant potential energy saving.

In [5] we described an architecture for managing policies for power management of resources in an HTC system. However, determining an optimal, or near-optimal, set of policies across a set of highly dependent shared resources is an extremely difficult and potentially costly process. This is due to the fact that the combined system behaviour is in general very difficult to determine in advance, as small policy changes can have significant influence on the overall system behaviour. One possible approach to determining a suitable policy set is to experiment with policy changes on a live system. This approach has three major drawbacks. Firstly, it is necessary to run the system under the new policy for a significant amount of time to ensure statistical relevance. Even with a long evaluation, some specific conditions may not occur during this time, which might lead to the wrong policy being adopted. Secondly, it is necessary to perform detailed logging and monitoring of the high throughput architecture in order to determine system wide energy consumption. Such in-situ monitoring often constitutes significant capital expense, potentially requires additional implementation, and may carry an overhead in performance and energy. Finally, there is a potential that policy changes could have unpredicted consequences which might have a negative impact, therefore leading to a significant degradation of the system during evaluation. Attempting to avoid this problem may lead to only minor policy modifications being made, so that there is a degree of certainty that the impact on users will be low; more significant changes being considered too risky.

Two potential alternatives to live experimentation are typically employed: these are either the establishment of a test bed environment or through simulation. Test beds offer the potential for detailed study but are generally costly and time consuming to develop and it can be hard to apply the derived results to a live system. Such test environments remove the need for site-wide monitoring and do not affect the production system. However, as test environments run in real time, or near real time, a significant duration is needed in order to evaluate changes. Simulation systems are also time-consuming to develop, but the same simulation environment can be tailored to different environments vastly reducing the individual development time. In addition, unlike live experimentation or test environments, many simulations can be run simultaneously to evaluate many possible policies and system parameters. Finally, the use of a simulation approach does not incur additional energy consumption on the real system; however, it should be noted that running simulations is not an energy-free process.

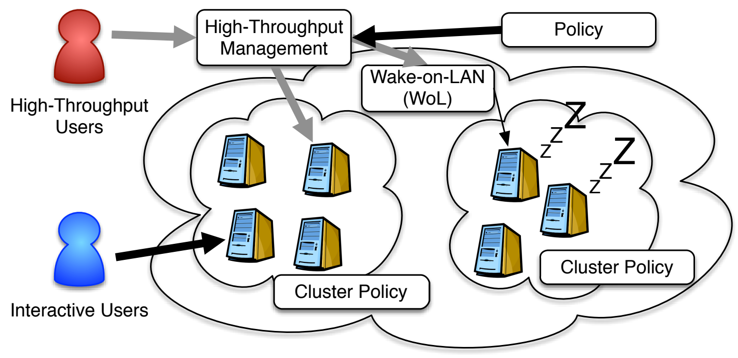

Figure 1. Model of an HTC system and multi-use environment

In this paper we describe our Java-based trace-driven simulation, HTC-Sim, which we have developed in order to study energy saving policies in HTC systems. HTC-Sim supports relatively fast and quantifiable evaluation of competing policies under identical workload and the presence of interactive users. The simulator supports a flexible approach to modelling energy consumption and performance of a specified HTC system. As such, HTC-Sim is a powerful tool which can be used by administrators to evaluate new policies and to assess the impact of changes to supporting infrastructure.

A high level model for HTC-Sim is illustrated in Figure 1. Resources are grouped into clusters, each representing a set of homogeneous resources subject to a given policy set. Thus we model sets of co-located resources purchased together acting under identical policies, though different sets of resources need not share these commonalities. The HTC system has its own policy set which specifies how resources are managed and how jobs are allocated to resources. There are two types of user who interact with the system - HTC users and interactive users. The actions of both types of user are specified through historical trace logs. Trace logs for interactive users contain login and logout times on the specific resource(s) used. These interactive sessions are assumed to be fixed. The trace logs for the HTC workload specify the job submission time and the execution duration. The job start time and the choice of resource used will be determined by the application of the active policy set. These traces can then be used, through the simulation along with an identified policy set to evaluate how the system would perform. Either user type can be removed from the system simply by providing an empty log file for that type, allowing a baseline performance to be established for each type of user.

The paper is organised as follows. Relevant related work is discussed in Section 2. We provide details of the simulation model in Section 3, followed by a discussion of the implementation of HTC-Sim in Section 4. A case study concerning the use of HTC-Sim to evaluate policies for an HTCondor cluster is presented in Section 5. In Section 6 we consider the performance of HTC-Sim, followed in Section 7 by a discussion of how we have used HTC-Sim in our work in recent years. In Section 8 we highlight some future work to extend HTC-Sim and we provide some conclusions in Section 9.

2. RELATED WORK

In this section we discuss a number of the related works pertaining to energy efficiency within computers and clusters.

2.1. *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.

Here we discuss alternative approaches to evaluating the energy consumption of large-scale distributed systems, classifying works based on their adopted approaches - namely experimental testbeds and emulation.

2.1.1. 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 dominated by private testbeds, with very few public infrastructures reporting energy metrics. One exception is BonFIRE [6, 7], 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 their users. A number of frameworks supporting private testbeds exist with emphasis on the evaluation of energy consumption, e.g. Enacloud [8] and Openstack Neat [9].

2.1.2. Emulation A further approach considered in a number of works is the emulation of large-scale systems. In an emulation approach, performance evaluation is conducted 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* [10] observed significant inconsistencies between results produced by multiple simulation frameworks modelling the same distributed environment. They attribute this variability to inconsistencies between abstract models and implementations, making the ability to closely align experimental and production code highly desirable. An emulation approach has been used in the context of peer-to-peer (P2P) [11] systems and networking [12], but few have accounted for the energy consumption of systems in emulation approaches, e.g. [13]. A significant constraint on emulation-based experiments is that of scale. Emulation-based approaches are commonly only capable of evaluating systems with small numbers of entities (orders of magnitude fewer entities than alternative approaches such as simulation). 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 large-scale, so we do not pursue an emulation approach further.

2.2. *Simulation frameworks*

A number of Grid and Cluster level simulators exist including SimGrid [14], GridSim [15], and OptorSim [16] 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 tradeoff between not only cost and Quality of Service, but also energy consumption. These include CloudSim [17], GreenCloud [18], and MDCSim [19].

In Table I we provide an overview of currently available simulation environments for the modelling of grid and cloud systems. We identify a number of commonalities between the capabilities of the simulation frameworks we consider. The '?' symbol is used where 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. We observe that all simulation frameworks support the modelling of performance and/or SLAs, with the exception of OptorSim. Similarly, all simulation frameworks except MDCSim support the modelling of heterogeneous compute resources. We find few simulation frameworks which support the modelling of virtualised 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 upon different platforms and due to its applicability to rapid development.

Table I illustrates that our simulation framework HTC-Sim exhibits novelty in its ability to model multi-use cluster and the presence of interactive users. Furthermore, HTC-Sim is one of only very few frameworks which support the simulation of real workloads and fault tolerance mechanisms.

2.3. *Energy efficient high throughput computing*

Techniques to reduce energy consumption within high throughput computing environments is receiving increasing attention.

Minartz *et al* [26] 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* [27] 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* [28] investigate the use of Dynamic Voltage Scaling techniques to reduce energy consumption in a heterogeneous cluster to conform to power budgets imposed by the infrastructure.

Niemi *et al* [29] 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 this.

Ponciano *et al* [30] 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* [31] model a cluster within a computational grid as an open queueing network and evaluate the impact of resource allocation 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 a compute resource prior to execution whereas, 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. 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* [32] explore network and energy-aware resource allocation strategies for opportunistic grids. The authors extend the Workqueue (WQ) [33] scheduling strategy to consider network traffic, distance between input files and the execution node, as well as the current state of the execution node.

Aupy *et al* [34] investigate energy-aware checkpointing strategies in the context of arbitrarily divisible jobs. While divisible jobs encompasses a number of common applications including BLAST sequencing and parallel video processing, such jobs represent only a proportion of our workload, and HTC systems do not typically have control over the division of batched jobs.

2.4. *Power modeling*

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 [35, 36] when developing predictive models of energy consumption, while others aimed to simulate individual [37, 38] or groups of system components [39, 40]. These models tend to require significant architecture knowledge and typically were not generalisable to other hardware, nor scalable to entire computer systems.

	CloudSim [17]	GreenCloud [20]	HTC-Sim	MDCSim [19]	OptorSim [16]	SiCoGrid [21]	SimGrid [14]	GridSim [15]
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	✓	✓	✓	?	✓	✓	✓	✓
Virtualisation	✓	✓	–	–	–	–	–	–
Heterogeneous resource models	✓	✓	✓	?	✓	✓	✓	✓
Underlying framework	SimJava [22] <=2.0, now custom core	NS-2	–	CSIM [23]	–	Parsec [24] and DiskSim [25]	–	–
Software license	LGPL	GPL	–	–	EUDatagrid	–	LGPL	GPL
Publicly available	✓	✓	–	–	✓	–	✓	✓
Latest release (dd/mm/yyyy) (as of 25/03/2015)	02/05/2013	19/12/2013	–	–	19/03/2008	–	02/06/2014	25/11/2010

Table I. Comparison of Simulation Frameworks

Fan *et al* [41] observed that total power consumption of server hardware was strongly correlated with CPU utilisation, and power consumption may be modelled 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* [42] introduce the Mantis model, which extends [41] 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 configuration).

More recently, Davis *et al* [43] 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 [44], 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 [43].

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* [45] 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.

2.5. Benchmarking

SPECpower_ssj2008 [46], released in November 2007, was the first industry-standard benchmark designed to evaluate and provide 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 [47], 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.

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* [48] present a survey of energy benchmarks for server systems.

3. SYSTEM MODEL

In this section we present an overall generic model for a HTC system examining the entities and resources found in all such systems. The atomic entities within our model are those of *compute resources* – either desktop computers or servers – and *jobs* (Figure 1) – representing the individual pieces of work which will be carried out. Third and fourth atomic entities exists within our model – those of interactive users, who may log into a desktop computer, and HTC users who submit HTC jobs into the system. However, as these entities are not under our direct control we model these entities as a consequence of their pre-observed actions. A number of compute resources are collectively called a *cluster* where a cluster is assumed to be under a common policy – such as opening hours and power down timings – and to comprise of equivalent hardware.

3.1. Compute resources

Compute resources are modelled as either multi-use computers – located within open-access clusters or found on individual workers desks – which are shared with interactive users or dedicated computers – normally located within a server machine room or as part of a cloud infrastructure.

The model characterises each compute resource through a set of parameters describing the resource in terms of operating system, architecture type, memory size, performance metrics (such as number of cores, CPU speed, MIPS) along with an energy profile. The model is extensible, allowing simulation developers to define their own custom parameters for resources which are bespoke to the environment they wish to model.

The energy profile is derived from the SPECpower [49] model of energy consumption within a compute resource. Within this model a set of CPU load values (such as 0%, 10%, ..., 100%) have known energy consumption values with interpolation being used to obtain intervening values. Thus we can model the energy consumption of a resource based on its current CPU load level. However, as a SPECpower model is not always available for a resource we can use the following two-point interpolation approach to approximate values [41]:

$$P(u) = P_{idle} + (P_{peak} - P_{idle})c \tag{1}$$

where P_{peak} is the peak power consumption of the resource (as defined by the manufacturer), P_{idle} the (manufacturer provided) idle energy consumption and c the CPU load level of the resource. If the value of c is unavailable it is assumed that the resource consumes P_{peak} while active with HTC workload or an interactive user and P_{idle} whilst idle.

Figure 2 depicts the five states of a compute resource along with the valid state transitions. These states are based on he ACPI specification [50] and are:

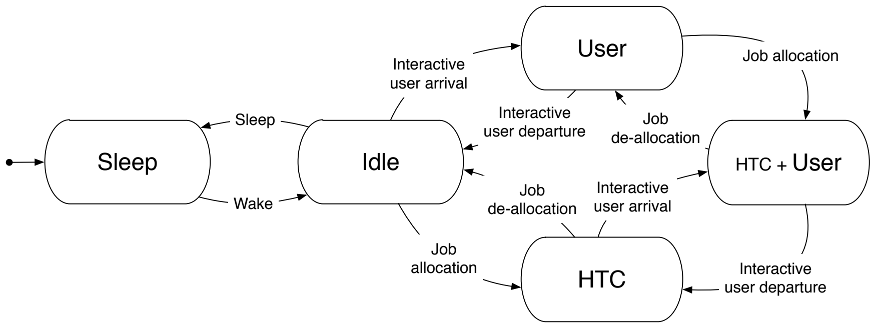

Figure 2. State transition diagram for an HTC resource

Active: the compute resource is actively performing work either executing a high throughput job or an interactive user is logged in. This maps to ACPI state *'G0 (S0)'*. This state is decomposed into three states User, HTC and HTC+User in Figure 2.

Idle: the compute resource is in a powered up state though no high throughput job is active and no user is logged in. The energy consumption in this state is much lower than the active state and will often equate to a CPU load of between 5 and 10%. Although the energy consumption here is typically much lower than active it also maps to ACPI state *'G0 (S0)'*.

Sleep: most of the components in the resource are powered down apart from the RAM which remains active and holds the state of the resource. By keeping the RAM active the resource may resume to an active or idle state without the need to restart the operating system – allowing a much quicker return to active (idle) than would be required if the system were to be fully powered down. This maps to ACPI state *'G1 (S3)'*. As most components are powered down the resource consumes very little energy – normally of the order of 1-2W.

3.2. *Interactive user sessions*

An interactive user session represents a user who sits down in front of a desktop computer and physically logs directly into that computer in order to perform some activity. This can be modelled as a tuple $\langle s_i, c_i, u_i, e_i \rangle$, where:

- s_i is the user login timestamp and e_i is the corresponding logout timestamp
- c_i is the unique name of the resource that was used – either an IP address or hostname
- u_i is a hash of the username.

By storing the username as a hash we can help protect the anonymity of the user, while still allowing correlations to be derived from the use of the system by a particular user.

An ordered set of all interactive sessions can then be used to replay the interactive user activity across the different compute resources within an organisation. We assume here that we cannot affect the actions of the interactive users within the system.

3.3. Cluster

A group of resources which share the same specifications, are located in the same physical space, were provisioned at the same time and are managed through the same management policies, is referred to as a 'cluster'. If these are dedicated resources located in a machine room then other energy impacts such as cooling can be taken into account using Power Usage Effectiveness (PUE) [51]. However, if the resources are in an open location – such as an open plan office – PUE cannot be, by definition, applied. In open locations we model an energy impact due to the environment in a similar way to PUE. Though as heat energy from computers may have a beneficial impact on the environment – reducing the amount of heating required – the value can be above or below one.

The management policies for a 'cluster' determine the behaviour of the resources. This includes: at what times the resources should be rebooted, at what times high throughput jobs can be run, the times at which the 'cluster' is open for interactive users, are high throughput jobs allowed to run at the same time as interactive users are logged in, the length of time before an idle resource will transition into the sleep state, or the length of time before an idle resource becomes available for high throughput jobs. Management policies are defined by a specific time-frame over which they are active, such as the day(s) of the week and the hours of the day when they apply. In addition to the normal resource management policies it is also possible to model *'special'* policy sets which alter the normal management policies. For example this may be to cater for bank holidays, or the closure of a cluster for maintenance/upgrades.

3.4. HTC Job

The workload of the HTC system is comprised of a set of high throughput jobs. Jobs may be submitted independently or together as part of a 'batch' submitted at the same time. We define a job by the tuple $\langle j_i, b_i, q_i, d_i, h_i, e_i, u_i, o_i \rangle$, where:

- j_i – the unique identifier of job i (or batch of jobs)
- b_i – the unique identifier of a job within a batch (if present)
- q_i – the timestamp of the job submission event
- d_i – the duration observed in the original system for job i
- h_i – the hash of the username of the user who submitted the job
- e_i – the final result state of the high throughput job (either 'success' or 'terminated'). If a job was terminated (result state e_i equals 'terminated') then d_i is the timestamp of when the request to terminate the job was submitted
- u_i represents the size of the input data transferred to the resource prior to the job running
- o_i represents the size of the result data transferred back from the resource on completion of the job.

This represents the core elements which all HTC systems can provide. It should be noted that although many HTC systems support many more elements as these are not common to all HTC systems they cannot be considered as part of this 'generic' HTC simulation system.

The state transitions for a high throughput job is depicted in Figure 3. When a job is submitted into the system it will initially be placed into the queued state. However, if an appropriate compute resource is available at this time the job can be allocated to that resource and move immediately to the running state. If no appropriate resource is available then the job will remain in the queued state until an appropriate resource becomes available. Ideally once a job enters the running state it will remain in this state until completed, at which stage it will transition into the job finished state. However, if an interactive user logs into the computer (and high throughput jobs cannot run at the same time as interactive users) then the job will initially be placed into the the suspended state – where the job will receive no compute cycles. This prevents the loss of job execution when an interactive user logs in for only a short (in the order of minutes) time and allows the job to continue once the user logs out. If the suspension time exceeds a pre-determined threshold then the job will be evicted from the resource and re-enter the queue. If the system supports checkpointing and it is enabled then at regular intervals (defined by the checkpoint policy) the state of the job will be checkpointed. If a checkpoint job is evicted then, when allocated to a new compute resource, it can resume from the last checkpointed state. Note that this will require time to transmit the checkpoint image to the new compute resource. At any point a job may be terminated – either by the system administrator or the user who submitted the job – in which case the job will move to the Job Removed state and execution terminated on any resource currently running the job.

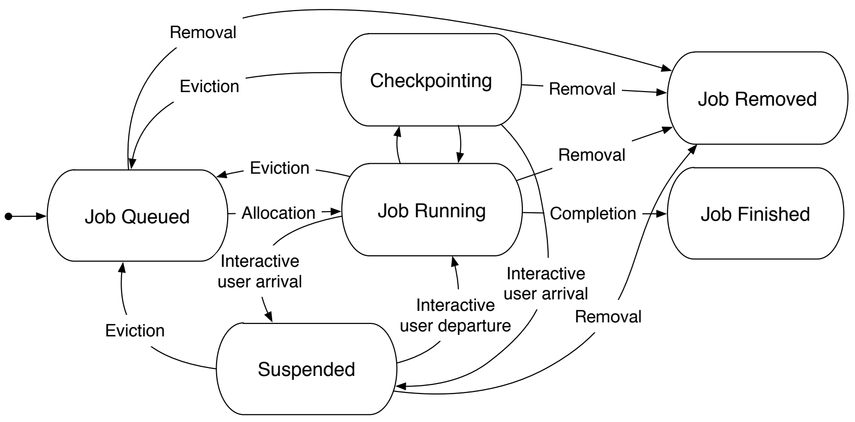

Figure 3. Job state transition diagram for the HTC system

4. HTC-SIM IMPLEMENTATION DISCUSSION

Here we discuss the implementation of the HTC-Sim system along with the modular nature of our system which allows the easy integration of alternative policies within the system. We conclude this section with a discussion of the metrics for user impact, energy, cost and environmental implications.

4.1. Simulation Model

The HTC-Sim operates as a standard ordered event queue simulation system developed as a Java based application. The main objects within the simulation are those entities defined within Section 3, namely Compute Resources, Clusters, Interactive Users, and HTC Jobs along with a entity representing the HTC management system. As these entities may have multiple future events within the event queue, for example a HTC job may have an event for the time the job would complete but also a time when the job will be terminated by the submitter, rather than placing the entity itself in the event queue a proxy object is placed in the queue which invokes the real object. In the case of our two HTC job events above these would be `JobFinishEvent` and `JobKillEvent` respectively. These will then call the appropriate code within the HTC job object.

4.2. Conceptual Model Architecture

Figure 4 represents the conceptual architecture of HTC-Sim. Each unshaded box within the figure represents a different service within the simulation. Each of these can be directly mapped to functionality commonly found within a HTC management system. The shaded boxes represent metric collection points of which there is only one instance, of each type, within the simulation. Solid arrows represent information flow between services whilst hollow arrows represent information flow to the metrics collectors.

Each of the services within the architecture is implemented as a skeleton service which offers only the common actions of that service – for example the Interactive User Model provides a mechanism to feed interactive user tuples into the Interactive User Management service. Any functionality which is specific to a particular realisation of that service is provided through a “pluggable” extension interface – the plugin “Policy” element. Continuing our example two pluggable policy extensions exist for the Interactive User Model – those of trace driven tuples or to generate a synthetic log of user tuples based on a set of user arrival, departure and computer selection statistical models. For each pluggable interface within the conceptual architecture we provide at least one implementation – this provides the functionality equivalent to how the HTCondor system operated in Newcastle University in 2010.

A full description of each pluggable policy interface and all implementations is beyond the space available here. Instead we present some examples in Section 4.5 along signposting in the rest of this paper where pluggable policy implementations are discussed. The selection of the pluggable implementation to use is through a common configuration script. Which comprises of name / value pairs. Elements within this script fall into one of three categories:

- **Core Simulation configuration**: relating to configuration parameters relating to the core simulation. Including start and end time for the simulation and metrics to collect.

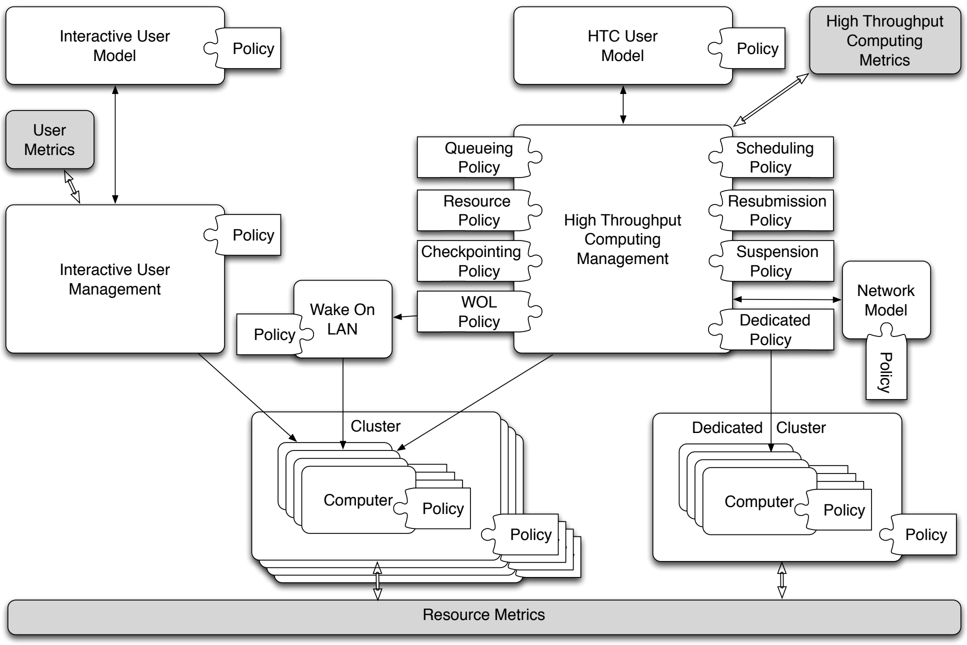

Figure 4. Conceptual Architecture for HTC-Sim

- **Pluggable Policy implementation**: to run for a given simulation. This is achieved through a name representing the pluggable interface associated with a value giving the fully qualified name of the Java class implementing this pluggable functionality. By default if no policy implementation is given then the simulation will default back to the implementation providing the functionality equivalent to HTCondor from Newcastle University in 2010.

- **Pluggable Policy implementation configuration**: provides a mechanism for configurations to the pluggable policy implementations. To prevent name-space clashes names must start with the fully qualified name of the Java Class implementing the policy. The remaining part of the name and the value are as required by the Pluggable Policy implementation.

4.3. *Convention over configuration*

The configuration of HTC-Sim follows the *convention over configuration* paradigm, by providing sensible default policies and operating behaviours. This aids the simplicity of configuring the system, where a practitioner need only customise the particular configurations they wish to modify from the default behaviour of the system. Therefore it should be noted that the simulation may be invoked with an empty configuration file. In which case all default policy implementations, with their default configurations will be executed.

4.4. *Executing Ensembles of simulations*

In many situations, in order to identify the optimal configuration of the system, it is desirable to run a number of simulations with varying configuration files. In order to simplify this process a common configuration file can be specified which contains alternative choices. It is then possible to iterate through all of the possible combinations just by specifying the common configuration file and which iteration is to be executed for this run.

We provide the following iteration logics:

- $\{v_1, v_2, ..., v_n\}$: a set of alternate values which may be used with a specific name element. Each v_n will be run as a separate simulation.

- $\{v_b; v_s; v_e\}$: allows sets of numerical alternative values to be specified by stating a start value (v_b), step size (v_s) and end value (v_e).

- $\{[nv?T,F]\}$: a selection option indicating that if a specific name / value pair (nv) exists within the document then the value represented within the T clause should be added into the configuration file otherwise the F clause will be added.

If multiple value sets exist within the configuration file all possible combinations will be produced.

4.5. *Pluggable Policy framework examples*

We exemplify a number of Pluggable Policy interfaces used within our simulation framework allowing us to simulate different HTC systems and to quickly evaluate different policies within existing systems.

4.5.1. *Policy – Job Descriptions*

Job Description reader: (cf. HTC User Model Policy in Figure 4) Our simulation model consumes tuples of elements describing jobs as discussed in Section 3.4. However, it is desirable to evaluate policies for a wider range of HTC systems, and for workload traces available in the literature [52, 53, 54, 55]. The Interactive User Model requests job tuples from the Pluggable Policy implementation in time-stamp order. We have developed a workload translator to support the use of workload traces in Grid Workload Format (GWF) [52], as well as supporting HTCondor workload traces.

4.5.2. Policy decisions for HTC The operating behaviour of HTC systems are governed by a number of common policy decisions. Here we discuss some of those which have already been developed in HTC-Sim:

Resource Allocation: (cf. Resource Policy in Figure 4) An HTC system must decide upon which compute resource to allocate a job awaiting execution. The HTC system must select the most appropriate candidates from a selection of avialable resources, in order to optimise the required performance, energy and QoS metrics. Many approaches exist including random allocation, reduced energy consumption, reduced likelihood of eviction, or fastest execution time [56, 57, 58] – discussed further in Section 5.5. The Pluggable Policy interface provides a simple interface which a developer can use to implement their own scheduler. The interface provides link-in points for selecting a computer given the current state of the system, and informs the scheduler that jobs have been terminated, evicted or completed.

Job resubmission: (cf. Resubmission Policy in Figure 4) In a system where jobs can be evicted through activities outside of its own control (reboots and interactive user logins) there is a need to decide if a job should be resubmitted or not. This is non-trivial as a job may exhibit multiple evictions due to a number of factors; the job may be 'broken' and will never complete, or these multiple evictions might just indicate that the job has been unfortunate in its previous allocation to resources [58, 59]. Once a job is evicted the system will invoke the Pluggable Policy providing information about the job, the time it was running, the reason for eviction (interactive user login or computer reboot), the number of times the job has previously been evicted and the amount of time it has run for previously. From this the implementation can return whether to allow the job to be resubmitted or whether to consider it a bad job and terminate it.

Reboots (deferral): (cf. Cluster Policy in Figure 4) The infrastructure on which many Desktop Grids are installed have nightly reboot policies allowing for system clean-up and software updates. Given that HTC workloads tend to have the greatest chance of running to completion throughout the night, particularly true for longer-running jobs, the ability for HTC jobs to defer these reboots can significantly improve the chance of jobs completing [56, 58].

Suspension: (cf. Suspension Policy in Figure 4) In the event of an HTC job being interrupted during its execution, the ability to suspend jobs offers great potential for 'saving' the effort already exerted on a job. Selecting an appropriate suspension timeout, after which the HTC job is abandoned and commences execution on a different computing node, represents a challenging trade-off. 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 [56, 58].

Checkpointing: (cf. Checkpointing Policy in Figure 4) Checkpointing allows period snapshots of the state of jobs throughout their execution to be persisted. In the event of an eviction, this allows the job to resume execution from the last available checkpoint, thus aleviating the time and energy associated with re-running execution which would otherwise have been lost. However, as the process of checkpointing consumes both time and energy – copying checkpoint images around the network – a careful balance is required to minimise energy consumption [58, 60, 61].

4.5.3. Policy decisions for Infrastructure A number of infrastructure-level policy decisions govern the power management and availability of resources to the HTC system [56] (implemented through the Cluster Policy in Figure 4) including:

Time before HTC usage: As soon as a computing resources becomes idle it is a potential target for HTC work. However, in busy multi-use clusters a user logout could be quickly followed by another user logging in, resulting in a job eviction. Therefore allowing some time between user logouts and HTC use is desirable to reduce the likelihood of job evictions early in their execution.

Time to sleep: Considerable energy savings may be sought by applying aggressive power management strategies, such as sending resources to sleep as soon as they become idle. However, this impacts on the execution time of HTC jobs, as the resources need to return from the sleeping state before execution may commence. This is exacerbated if resources are required to be idle for a certain amount of time before they can be used for HTC work.

Waking up resources: If the HTC system lacks the ability to wake up resources on demand, then this can lead to resource starvation once the resources have gone to sleep. Likewise if an HTC system can wake up computers then this leads to potentially increased energy consumption, so it is important for such policies to provision resources carefully to balance the trade-off between energy consumption and performance.

Allow HTC usage: At busy times of day, specific clusters may exhibit particularly high levels of job eviction. In these situations, it may be desirable to disable HTC workload on specific clusters, steering the workload towards quieter clusters where the jobs are more likely to complete execution successfully. Furthermore, institutional policies often seek to disable the use of HTC resources during special circumstances such as computer-based exams.

4.6. Metrics

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

Performance: We evaluate the performance of proposed policies using three metrics, namely average job overhead, slowdown [62] and bounded-slowdown [63].

Average job overhead is defined as the difference in time between the job entering and departing the system, and the actual job execution time. Average job overhead is defined as follows:

$$\frac{\sum_j^{j \in J}(f_j - s_j - d_j)}{|J|} \tag{2}$$

for a set of jobs J, where q_j is the arrival time for job j, f_j is the departure time, and d_j is the job duration. Here q_j and d_j are as defined in the job tuple.

Slowdown [62] is a metric providing the overhead for job j, normalised by the job runtime, and is defined as follows:

$$O_j = \frac{f_j - q_j}{d_j}. \tag{3}$$

Furthermore, we can define the average Slowdown for a set of jobs:

$$\frac{\sum_j^{j \in J} O_j}{|J|}. \tag{4}$$

It should be noted that as each job slowdown is related to the actual execution time for that particular job the average slowdown cannot be directly compared with average overhead.

A limitation of this slowdown metric is the disproportional representation of very short running jobs. To address this we also report the average "bounded-slowdown" metric [63], under which slowdown for jobs with short runtimes is measured against an interactive threshold τ, and is defined as

$$\frac{\sum_{j}^{j\in J} \max\left\{O'_j, 1\right\}}{|J|} \tag{5}$$

where:

$$O'_j = \frac{f_j - q_j}{\max\{d_j, \tau\}}. \tag{6}$$

Energy consumption: HTC-Sim is capable of reporting energy consumption metrics at various levels of granularity; at per-computer, cluster and system levels. Reported results may be further broken down by operating state of resources e.g. sleep, idle, HTC and/or interactive users. The total energy consumption is calculated as follows:

$$\sum_{c=0}^{n}\sum_{p=0}^{m} t_{c,p}E_{c,p} \tag{7}$$

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 and $E_{c,p}$ is the energy consumption rate of computer c in state p.

Power Usage Effectiveness (PUE) [51] values for environments such as data centres and machine rooms may be utilised, 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 be legitimately applied to desktop machines based in users' clusters due to the multi-use nature of the environment in which the machines reside, and variations introduced by user occupancy.

We further model for each policy the proportional energy increase, which represents the energy consumption of a job within our system relative to the lowest possible energy consumed in servicing the job. This may be defined as:

$$Q_j = \frac{\sum_{k}^{k\in A}(f_{j,k} - s_{j,k}) \cdot E_{j,k}}{d_j \cdot E_{opt}} \tag{8}$$

where A is the set of all attempts to run job j, $f_{j,k}$ is the finish time of invocation k of job j on a compute resource, $s_{j,k}$ is the matching start time, $E_{j,k}$ is the energy consumption rate for the compute resource used for that attempt and E_{opt} is the energy consumed by the most energy-efficient computer within the system. Note that here we only consider jobs which do eventually complete and that only one of the elements in set A represents a successful completion – the last one.

We can then compute the average proportional energy increase for a set of jobs J as:

$$\frac{\sum_{j}^{j\in J} Q_j}{|J|} \tag{9}$$

Good jobs terminated: In order to minimise the amount of energy consumed by jobs which will never terminate (due to a bug within the code or incorrect configuration) we may choose to limit the maximum number of resubmission attempts. In doing such this may lead to a policy which terminates a good job which has been unfortunate to receive too many evictions. Under this situation we seek to record the number of good jobs which have been incorrectly terminated and minimise the number of such circumstances. This reduction can be achieved by looking at the conditions under which jobs have been evicted [59] or providing dedicated resources for jobs which have been evicted [64].

Data transfer: often represents a significant overhead to HTC jobs. This is particularly evident for jobs which depend on large datasets as input, or when using checkpointing. HTC-Sim 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* [65] bandwidth measurement tool was used to ascertain the peak bandwidth available between cluster machines and an average value of 94.75 MBits/s was used within the HTCondor simulations.

Cluster utilisation and throughput: Many operating policies, such as fault tolerance and replication, have the potential for significant impact on throughput and overall cluster utilisation. HTC-Sim is capable of reporting measures of average and peak utilisation/throughout, both in terms of the HTC workload in isolation, and also including interactive user load.

Cost and environmental impact: It is insufficient to evaluate energy consumption and performance of policies without also considering the implications of policy on operating cost. We model electricity cost per kWh, and a carbon emissions charge for each kilogram of CO_2 produced [66] (currently £16 per metric tonne in the UK). These values 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. We have in previous work extended the energy model to account for additional costs including hardware and network infrastructure [67].

Thus, the total operating cost C for set of resources r is calculated as:

$$C(r) = \sum_{r=0} u_r * p_r + \frac{u_r}{1000e_r} * t_r \tag{10}$$

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.

5. CASE STUDY: MODELLING A HTCONDOR DEPLOYMENT

We present here a validation of the HTC-Sim simulation environment by modelling the Newcastle University HTCondor deployment and evaluating the impact of a number of resource selection policies. As part of the discussion we describe how the trace-logs for both the HTCondor system and the interactive users can be obtained and cleaned such that they can be used within the HTC-Sim system. We exemplify this through the use of twelve months of trace data obtained during 2010 [†].

5.1. HTCondor pool at Newcastle University

The HTConodor pool comprised of $\sim$1400 desktop computers (resources) which were distributed among 35 clusters located around the campus. The characteristics (management polices) for these clusters varied depending on cluster location and expected student presence within those locations. For example the opening hours of clusters reflected whether the location within the university would be open for other reasons – clusters located deep within a particular department often adhered to office hours, whilst clusters located within the library or the Students' Union were often available twenty-four hours per day. The university adopted a rolling four-year programme for replacing computers. This lead to computers falling into one of three broad categories as listed in Table II.

[†]In this work we leverage HTC workload and interactive user logs collected throughout 2010. These traces are indicative of current system usage and analysis that has been ongoing since 2010.

Type	Cores	Speed	Power Consumption		
			Active (P_{peak})	*Idle* (P_{idle})	*Sleep*
Normal	2	∼3Ghz	57W	40W	2W
High End	4	∼3Ghz	114W	67W	3W
Legacy	2	∼2Ghz	100-180W	50-80W	4W

Table II. Computer Types

It should be noted that the University, by 2010, had enacted a policy of choosing energy efficient computers. Hence, all but the 'Legacy' systems, which pre-date this policy, were low-energy systems in comparison to their performance. 'Normal' computers are targeted for normal student use such as word processing, email reading and web browsing, whilst 'High End' computers are purchased for the more computationally intense student work – such as engineering CAD work or video editing – which in general lead to higher energy utilisation. Due to purchasing policy entire clusters would be replaced en-mass leading to clusters of homogeneous resources. This leads to a situation where there is little variance of hardware within a cluster though there may be significant variance between clusters. The sizes of clusters varied dramatically from clusters of just ten computers to clusters of over one hundred computers. Some large open-access areas contained multiple clusters – with each cluster being managed and upgraded separately.

Power consumption rates are provided for operating states equivalent to those detailed in Section 3.1. The trace logs we were able to capture for 2010 lack any record of the utilisation of the resources (desktop computers) – c in Equation 1. Therefore we were unable to employ the linear interpolation model. Instead, we assumed that the energy consumption of the resources were those provided by the manufacturer *'nameplate'* power consumption values as depicted in Table II. Where an active computer (interactive user or HTCondor job) consumed P_{peak} watts, while the same computer in idle or sleep mode consumed P_{idle} or P_{sleep} respectively.

As the primary use of these computer clusters was for the use by students their configuration were tailored to their preferences. The desktop computers ran a Microsoft Windows-based operating system – as demanded by the students. Unfortunately the Windows based version of HTCondor lacks the ability to perform checkpointing and migration of jobs. However, a small number of computers (primarily within Computing Science) were Linux based – providing ∼5% of the HTCondor pool – which provided the capabilities of checkpointing. Central management required the computers to reboot nightly between 3am and 5am. This was performed to reset the computers for the following day – restoring any transient crashed services, along with installing new software and patches. Computers located within 24-hour clusters which had an interactive user at the time of reboot would have the reboot postponed until the user logged out.

5.2. *Specifics of HTCondor*

Jobs within a HTCondor system are described using ClassAds [68]. Each ClassAd consists of a number of name / value pairs – one per line of a text document – which holds all the data about the given job. Although a ClassAd contains many name / value pairs – the Newcastle HTCondor system provides more than fifty per job – there are only nine name / value pairs which are required for HTC-Sim. A mapping of ClassAd names to their equivalent job characteristics – as identified in Section 3.4 – is presented in Table III. Although the value of `JobStatus` can be any of the values zero through four, the only values which a job can finally end in are those of *'4'* for compleated jobs and *'3'* for jobs terminated by the user or system administrator. It should be noted that the value of d_i does not take into account any time the job was in a suspend state. This can be rectified by subtracting the value of `CumulativeSuspensionTime`.

The matching of resources to jobs within HTCondor is performed by the 'Matchmaker'. The Matchmaker inspects all resource ClassAds and all job ClassAds and attempts to find the most compatible pairings. In order to achieve this it considers two name / value pairs within the ClassAd documents – those of `Rank` and `Requirements`. Requirements provides a boolean set of requirements which must evaluate to true for the match to be considered possible. This can include such things as minimum memory requirements, the required operating system or the existence of pre-installed software. By contrast Rank (which exists in both resource and job ClassAds) is used to order those matches which satisfy the Requirements and evaluates to a number – where higher values are considered better. As the `Requirements` and `Rank` name / value pairs within our trace-log was almost completely unused [5] and our intention here is to evaluate the energy consumption under different policy sets we are not using these for the purpose of job to resource matching – thus default HTCondor matching devolves to random resource selection. However, the resource selection policy within HTC-Sim could easily be extended to take `Requirements` and `Rank` into account.

5.3. Preparation of the User logs

Access to the desktop computers for interactive users at Newcastle University is managed through a central Managed Desktop Service (MDS) where historical records are stored in a backend database. Queries can be run against the database to identify all login and logout events in 2010. Unfortunately, these queries produce separate outputs for login and logouts and, due to the fact that user home space is mounted as part of the login procedure, contain duplicate records – both in the form of identical time stamps and records which vary by only a few milliseconds. Further the results of the queries were not produced in chronological order. We have developed a small tool which is capable of identifying the duplicates within the query results, re-ordering the login and logouts into chronological order and then matching the appropriate login and logout events. This problem if further compounded in the cases where an appropriate login and logout pair cannot be identified. This was a consequence of a computer crashing whilst a user was logged in or the user powering down the computer via the mains switch – leading to a login without matching logout. These scenarios were deemed safe to ignore as they accounted for less than 0.1% of the dataset.

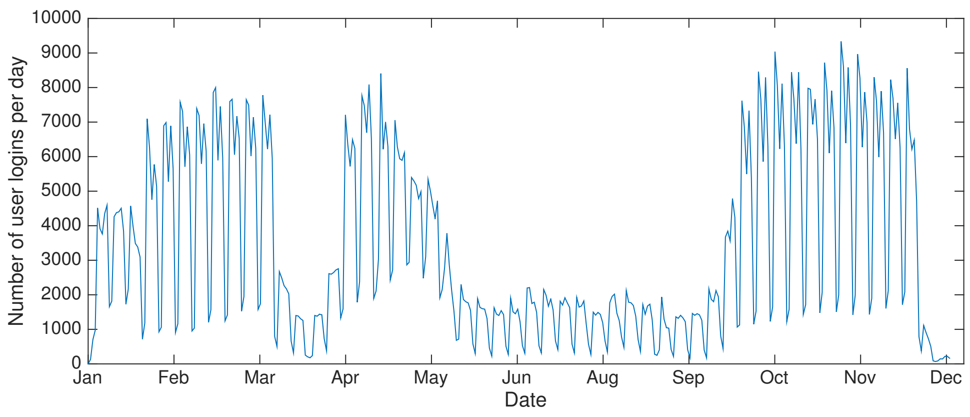

Figure 5. Interactive user logins per day for 2010

Job characteristic	Tuple term	HTCondor parameter or expression
Job identifier	j_i	`ClusterId`
Batch identifier	b_i	`ProcId`
Submission time	q_i	`QDate`
Job duration	d_i	`EnteredCurrentStatus-JobCurrentStartDate`
Owner	h_i	`Owner`
Result state	e_i	`JobStatus`
Data transfer in	u_i	`BytesSent`
Data transfer out	o_i	`BytesRecvd`

Table III. Job Characteristics to HTCondor mappings

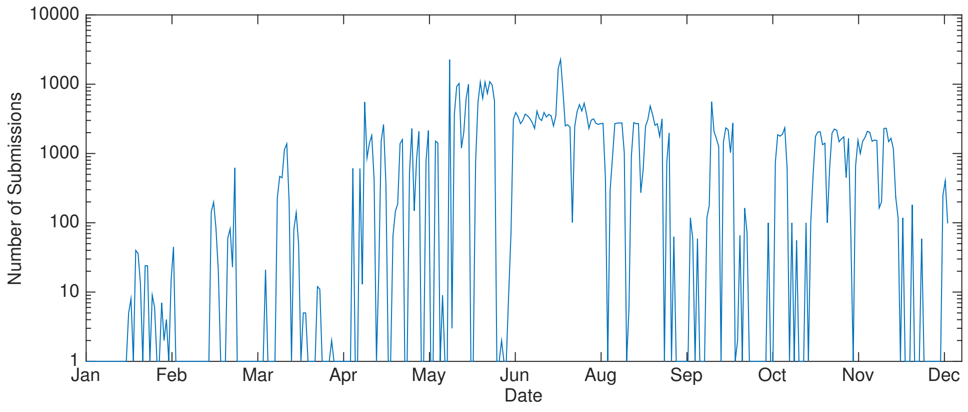

Figure 6. HTCondor workload trace for 2010

The number of user logins per day during 2010 is shown in Figure 5. This represents 1,229,820 login events. From this figure it is possible to identify the weekly usage patterns – less logins at the weekend, along with the termly usage patterns – the three terms can be easily identified along with the christmas, easter and summer breaks. As the MDS system does not capture resource utilisation we assume here that when an interactive user is logged in the resource is being used at 100% utilisation. Although this is clearly an overestimate we justify it by remarking that our goal here is to determine ways to reduce energy consumption through HTC jobs within the system and that the interactive users provide a constant background energy consumption which we cannot affect.

5.4. Preparation of the HTCondor logs

Once a job has finished within HTCondor – i.e. either completed or been terminated – its ClassAd is archived within the history log. These ClassAds can be retrieved through the `condor_history -long` command. In general HTCondor is normally configured to only keep the previous N jobs which have finished (where N is a configurable value). Also, in terms of simplicity for HTCondor the ordering of these records is based on completion times of the job rather than submission time. Regular capturing of the history log can be used to overcome the first issue – though this will lead to duplicate ClassAds held in consecutive captures. The second issue can be overcome by post-processing the ClassAds to reorder them by submission time. This functionality, along with functionality to remove duplicate records, is provided by a small tool. Once prepared the ordered ClassAds can be fed into HTC-Sim via the HTCondor Pluggable Policy implementation.

The number of HTCondor jobs submitted per day in 2010 is shown in Figure 6. A total of 561,851 jobs were submitted over the whole year with a mean submission rate of 1,454 jobs submitted each day. Unlike the interactive user logins per day there is no clearly visible pattern within this data.

Further information about the execution of a HTCondor job, which is normally only made available to the submitter of the job, is available within the HTCondor system. This contains information such as periodic recording of he memory and disk usage of the running job, a complete log of all resources that the job was allocated to (not just the last allocation), along with records of individual job suspensions and checkpoints. Modification to the configuration of the central HTCondor configuration script allows this information to be collected centrally for all jobs. The listing presented in Listing 1 shows the configurations required to achieve this. We have been centrally collecting this level of job statistics since December 2012.

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

Listing 1: HTCondor configuration options to enable centralised usage collection.

Our trace logs were generated through Condor version 6.6. However, our tool for log reordering and duplicate removal, along with the ability to run our logs through HTC-Sim remain compatible through to the current version of HTCondor – currently version 8.4.2.

As HTCondor logs contain personal information in the form of usernames, executable command names, parameters and paths to files it is often difficult to share logs between organisations. To aid with this we provide tooling to anonymise and sanitise log files. ClassAd name / value pairs which could provide useful insights to understanding usage patterns such as usernames and executable names are replaced by hashes allowing analysis without access to the personal data.

Figure 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 observe significant differences in the levels of interactive user usage between the clusters. The 'GLOBE' cluster, only available to the staff and students of a particular department, and located far from main thoroughfairs, exhibits interactive user usage of only 0.35%. Conversely, the 'SIDE' cluster is a popular 24-hour cluster with 36.17% interactive user occupancy in 2010. Similarly, the 'TARN' and 'WEAR' clusters are located in the main University library and are subject to significant footfall, were occupied for 31.05% and 31.18% of the year respectively. The 'TEES' cluster, also located in the main University library, receives only 11.4% utilisation due to its basement location and 08:30-17:30 opening hours relative to 'TARN' and 'WEAR' which were open 08:00-21:30. These inter-cluster differences highlight the importance of resource allocation policies which are aware of interactive users to reduce energy waste and maintain performance. When considering the HTC workload, we may observe that the offered workload to our system in 2010 results in very low system utilisation (12%).

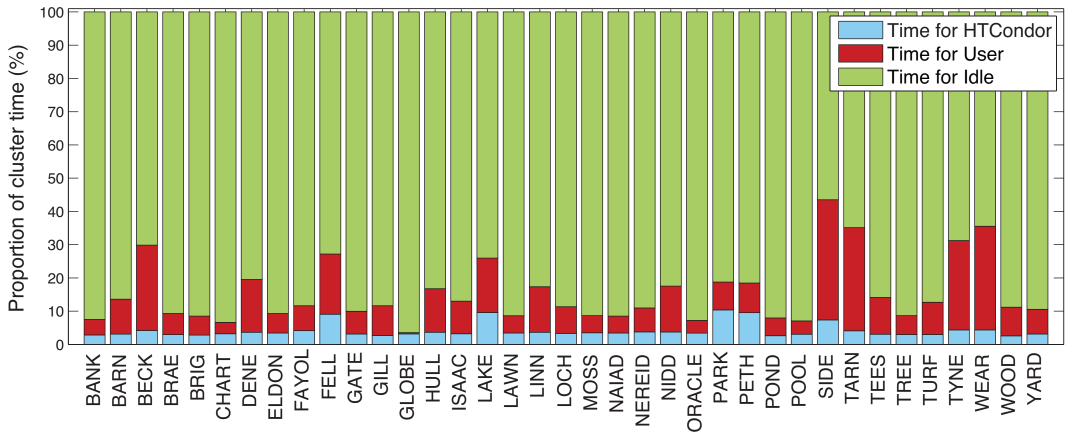

Figure 7. Proportion of cluster time used by interactive users and HTCondor

Figure 8 shows the probability that a job of length x hours will complete given that it is submitted during hour y of the day [57]. 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 therefore 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 completing successfully falls between 3am and 8am. By using Figure 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 (re)submission. The prediction of task completion time for our institutional workload is explored in greater detail by Bradley *et al* [69].

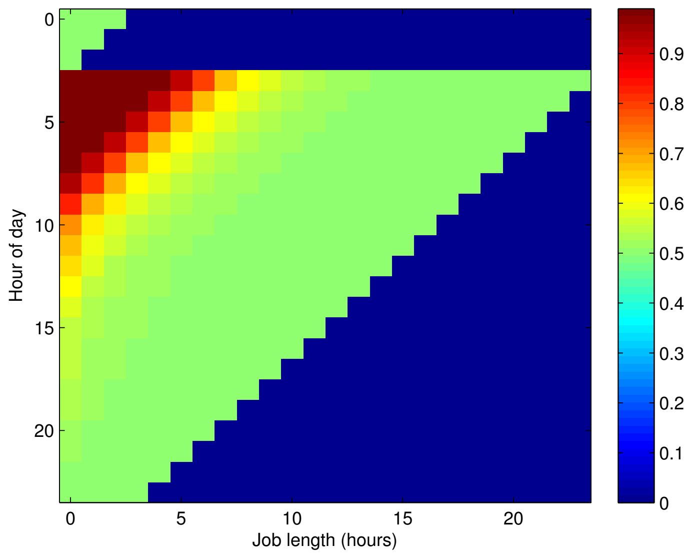

Figure 8. Heat map showing the probability of successful job completion given job duration and submission time

5.5. *Example of evaluating policies: Energy-aware resource allocation*

We exemplify here how HTC-Sim can be used to evaluate the energy-efficiency of a number of resource allocation strategies (cf. Resource Policy in Figure 4). Each of these strategies can be compared in terms of the metrics of average job overhead, slowdown, bounded-slowdown and total energy consumption.

S1 HTCondor default: this degrades to a random resource selection policy which favours computers which are powered up. We include this as the policy which was enacted on our infrastructure during 2010, and as a commonly used default set by the HTCondor system. We see this as a non-energy saving policy and expect all policies to do better than this.

S2 Target the most energy efficient computers. This process is an approximation to the efficiency of a job, as different computers will handle different computational jobs 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.

S3(*i*) Targeting jobs towards computers which have the least interactive user activity, this can be ranked by: *a)* the greatest average inter-arrival time between users and *b)* the lowest number of interactive users within a given time period (e.g. one day). Our interactive user workload traces demonstrate that computers placed in locations frequented by students tend to have short durations between interactive users. By contrast, computers in less popular locations typically observe much greater durations between interactive users. Computer usage can also be affected by cluster 'opening hours'. It is therefore 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 future. However, we can look for general trends in the usage patterns of computers from historical evidence and use this to inform our resource selection decisions.

S4 Target clusters which are currently closed for use by interactive users. Each cluster has pre-defined opening and closing hours. Here we propose seleting computers in closed clusters which have the greatest amount of time remaining until the cluster reopens, thus minimising the likelihood of the job being evicted by the arrival of an interactive user.

S5(i) Target jobs towards clusters with the smallest amount of interactive user activity, this can be ranked by: *a)* the lowest sum of interactive user durations, *b)* the smallest mean of interactive user duration.

S6 Targeting jobs towards clusters which have the lowest number of interactive user arrivals over the last Δ minutes. This can be determined by:

$$\min_{c \in C} \left\{ |E_{c,t,\Delta}| \right\} \tag{11}$$

where $E_{c,t,\Delta}$ is the set of user logins to computers in cluster c in the time window $[t - \Delta, t)$, C is the set of all clusters and t is the current time. This policy was first introduced in [70] as an extension of resource allocation strategies first explored in [56].

5.6. *Analysis of job length on slowdown and energy impact*

We analyse here the impact on job length on the slowdown and proportional energy increase for jobs run through our HTC-Sim configured for HTCondor. For these results we analyse slowdown and proportional energy increase against the default resource selection policy (**S1**). In each case we have grouped together all jobs which had an actual execution time (just the time for running the job, ignoring any data transfer times) within the same hour-long interval (e.g. 0 to 1 hours is interval 1). Note that we have ignored all results for intervals over 14 hours in length, as each of these intervals contained less than five jobs, making any statistical analysis meaningless. It should also be noted that we have ignored all jobs here which fail to complete as these have no meaningful interpretation of actual execution time.

Figure 9 shows the average bounded slowdown of jobs for the default policy (**S1**), as defined in Equation 5 with parameter $\tau = 60$ minutes to ameliorate the disproportional impact of very short running jobs on slowdown results. Error bars depict standard deviation for each bar, and the number above each bar refers to the number of jobs falling within each interval; for example, there are 8580 jobs in the 2 to 3 hour interval. The figure shows that short-running (less than two hours) jobs have significantly higher slowdowns than longer-running jobs (between four and ten hours). The minimum slowdown occurs when the actual job execution time is in the range 6-7 hours. This impact on short-running jobs is due to the overheads of running the job through an HTC system – short intervals of waiting for deployment along with time to transfer data to (and back from) remote computers can become significant in comparison to the short run-times. Likewise for longer running jobs, although the overheads of the system (in comparison to the actual execution time) become a less dominant factor the impact of the job being evicted and needing to be re-run on a different resource starts to dominate. A similar scenario can be observed in Figure 10 which depicts the proportional energy increase for the same selection policy – both of these figures sharing similar shapes. The higher proportions visible in Figure 10 can be attributed to the fact that here we assume that the 'best' case will be when the job is run on the lowest power computer available, whilst for slowdown the choice of computer was irrelevant.

This 'optimal' job execution length (6 to 7 hours) would appear to go against the perceived wisdom within the HTCondor community[‡] where it is assumed that jobs of two hours or less are 'optimal'. This can be explained through two justifications. First, although the slowdown for short (less than one hour) jobs can be significant in comparison to the actual execution time of the job, the amount of time wasted can still be relatively short – in the order of a few minutes, which is often not observed by the user. A second explanation can be seen from Figure 11 in which the number of times a job is submitted and fails to complete is compared with the actual job execution time. Here we can see that there is a minima for average resubmissions – for 0 to 1 hour jobs. Most HTC users

[‡]As observed from delegates at the annual HTCondor user conference, HTCondor Week, 2010-2012.

think only of the number of resubmission attempts when determining the ‘optimal’ job execution length, and one may assume that they see 2 to 3 and 3 to 4 hour jobs having much higher average resubmission counts and see that 0 to 1 hour is the minima value. It should be noted here that for the 13 to 14 hour interval the true number of average resubmission attempts was 106 with the figure being clipped to make the other bars easier to observe.

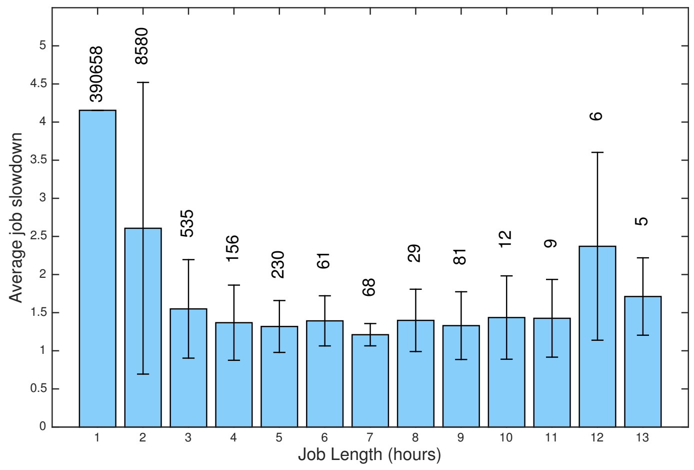

Figure 9. Average slowdown results from exemplar policy

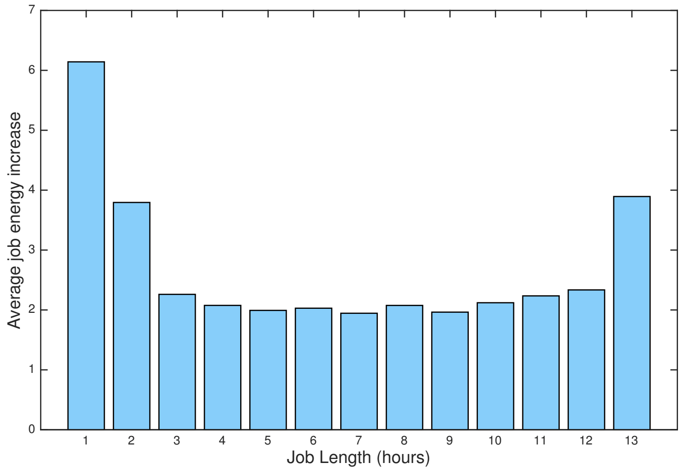

Figure 10. Energy consumption against job length

5.7. Analysis of Energy-aware resource allocation policies

The average overheads, as defined by Equation 2, for our policies S1 through S6 are presented in Figure 12. There is a large detrimental impact on overheads observed with policy **S3** (target clusters with least interactive user activity). The overhead impact of policy **S6** (target jobs towards clusters with low user arrivals over previous Δ minutes) is higher than the remaining policies. The remaining policies have similar overheads with **S2** having the best value. Unfortunately the size of the sliding window for policy **S6** seems to have no effect on the overheads observed and the overheads are significantly larger than those for **S5** which assumes perfect knowledge. The energy consumption, as defined in Equation 7, for the different policies is shown in Figure 13. Policy **S2** shows the lowest

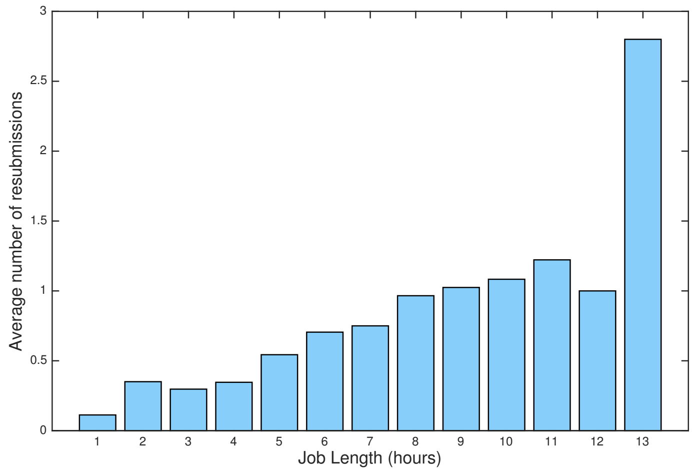

Figure 11. Average number of resubmissions against job length

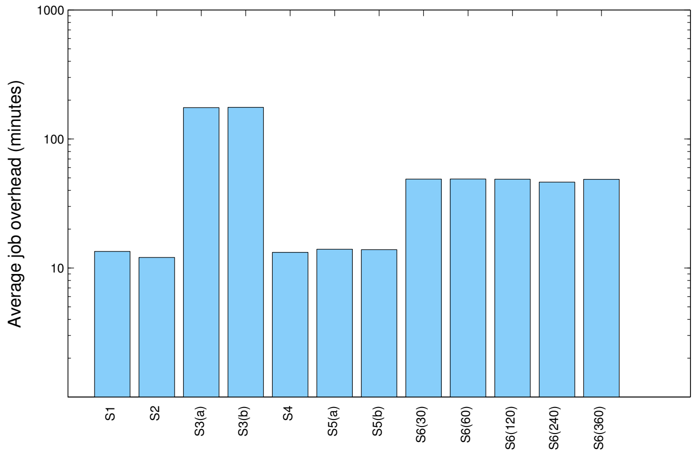

Figure 12. Overhead results from exemplar policy

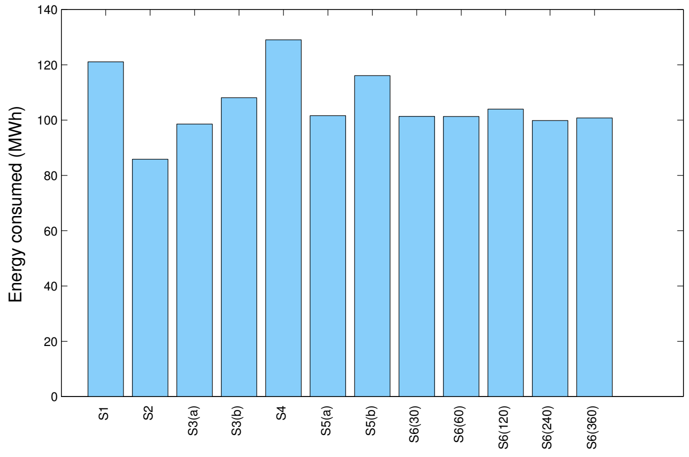

Figure 13. Energy consumption results from exemplar policy

energy consumption, this coupled with the low overheads from Figure 12 would make this the best policy overall to adopt.

Figure 14 shows the relative difference in slowdown for each policy in comparison to policy **S1**. Relative difference is calculated as $\frac{v_{S1}-v_{Sx}}{v_{S1}} \times 100$ where v_{Sx} is the slowdown for the policy being evaluated, and v_{S1} is the slowdown for policy **S1** against which the other policy is compared. We report both average slowdown, and also bounded-slowdown (BSD) with interactive threshold τ of one, ten, 30 and 60 minutes. The number of jobs in our workload which are encompassed by each of these values of τ, such that $d_j \leq \tau$ where d_j is the job duration d_j, is 135,945 for $\tau = 1$, 206,131 for $\tau = 10$, 308,268 for $\tau = 30$ and 390,658 for $\tau = 60$.

Considering these slowdown metrics offers additional insights into the operation of the policies, revealing whether policies provide equitable treatment to HTC jobs of varying durations, or whether policies have a particularly positive or detrimental effect on jobs of a particular length. As discussed in Section 4.6, since each job slowdown is related to the actual execution time for that particular job the average slowdown cannot be directly compared with average overhead, and two policies may exhibit very similar average overhead values but slowdown metrics may vary significantly.

We observe that each of the policies we consider – with the exception of **S4** – are capable of achieving reductions in energy consumption while incurring only a minimal increase of average job slowdown. In the cases of policies **S2**, **S5(a)** and **S5(b)** we observe that the policy has a positive impact in slowdown particularly for long-running tasks within our system. Policies **S5(a)** and **S5(b)** target clusters less susceptible to interruptions by interactive users. While short-running jobs are generally unaffected by this decision, these policies are particularly beneficial for long-running tasks which would otherwise struggle to complete execution on a cluster with greater interactive user activity.

In Figures 15 and 16 we demonstrate the use of HTC-Sim to reason over infrastructure decisions. We explore the impact of removing clusters from the HTCondor pool on both energy consumption and average job overhead. In doing so we simulate a situation where the operator of a student clusters withdraws these from use by the HTC system. We remove clusters in descending order of interactive user activity, removing the *'busiest'* clusters first. We see in Figure 15 that the removal of clusters offers modest benefits for energy consumption as it reduces the amount of energy consumed by idle resources. However, in Figure 16 we see that up to seven clusters may be removed from HTC operation while incurring modest increase in average job overhead, but further reductions begin to severely restrict the computational resources available to the HTC system.

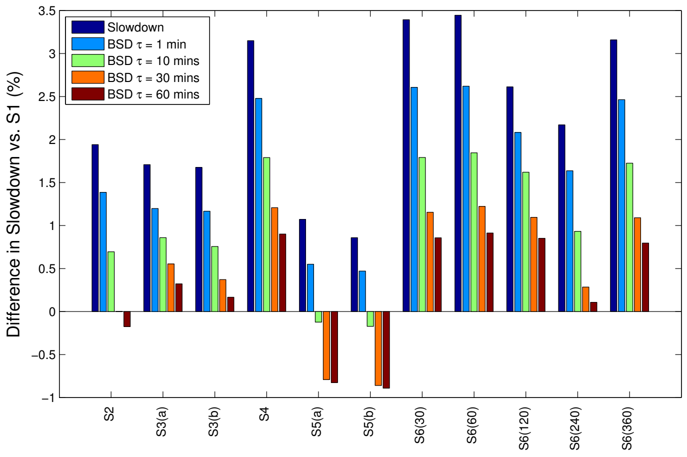

Figure 14. Difference in slowdown compared to policy **S1**

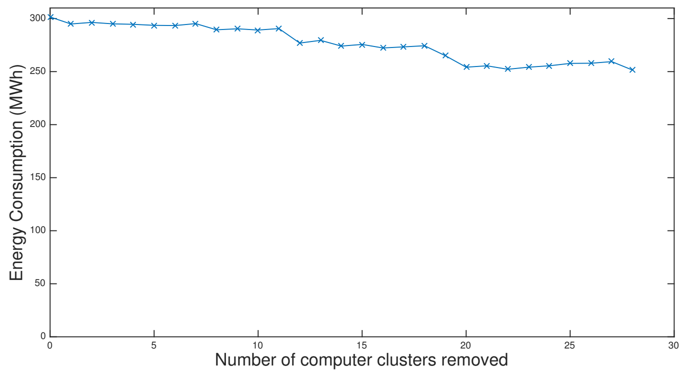

Figure 15. Impact of reducing cluster availability on energy consumption

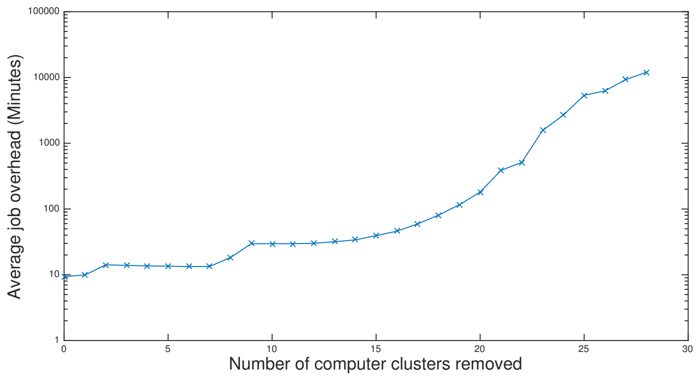

Figure 16. Impact of reducing cluster availability on average job overhead

6. PERFORMANCE EVALUATION

Here we evaluate the performance of our simulation software and justify its applicability to arbitrary sized HTC data sets. We evaluate this in terms of the wall-clock time to run the simulation and the maximum (peak) memory footprint. The timing for a simulation and the memory footprint will be a direct consequence of the policy set being evaluated.

In our evaluation we present figures for simulations based on baseline policy **S1** and policy **S4** which leverages historical information of resource availability to steer jobs to clusters which are closed, or are soon to close. We seek to explore the impact of these policy decisions on the execution time and maximum memory consumption of the simulation.

Each simulation was run on a machine with a 2.50GHz quad-core Intel Core i7 processor with 16GB 1600MHz DDR3 memory and a 500GB PCIe-based Samsung SSD storage, running the OS X Yosemite 10.10.3 operating system. Results are based on fifteen simulation runs to reduce random effects.

Running our real historical trace log of HTCondor workload requires an average of 2:41 minutes. Running the simulation without the HTCondor requires 0:46 minutes, whist running without interactive users (representing a dedicated cluster) requires 2:24 minutes. Note that you cannot sum these two times to give the overall simulation time due to simulation bookkeeping and the processing of cluster events such as computer reboots and clusters opening and closing. The memory footprint for these simulations are 876MB, 280MB and 845MB respectively. The higher memory footprint from the HTCondor-only simulation is 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 [56] – over ten times our real

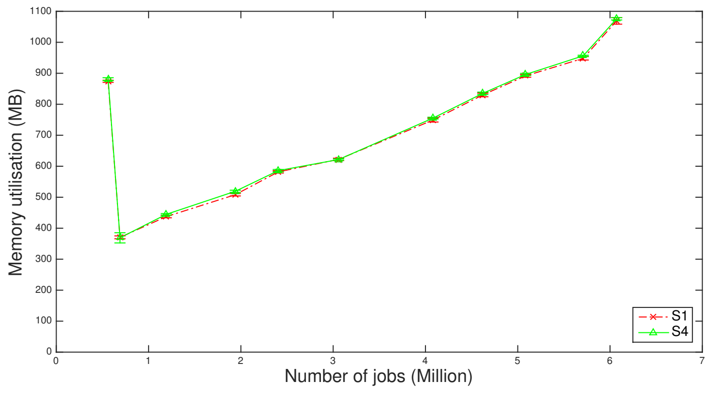

Figure 17. Maximum memory footprint

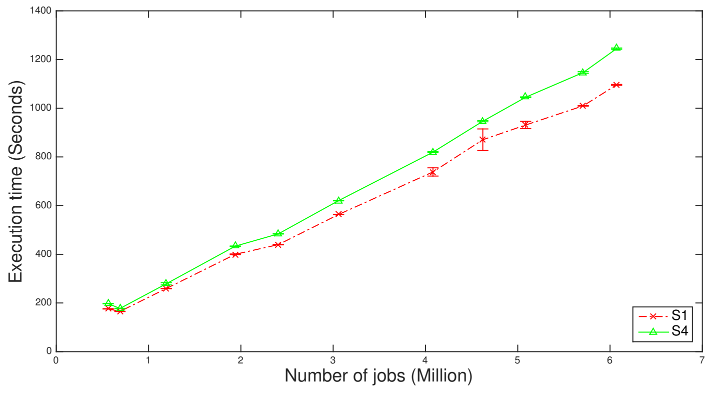

Figure 18. Execution time

workload (∼six million jobs). Figures 17 and 18 show the memory footprint and execution times respectively for both our original simulation and synthetic trace logs under policies **S1** and **S4**. Error bars show the standard deviation between based on fifteen simulation runs.

Figure 17 shows our real dataset to consume 874MB of memory for policy **S1** and 881MB for policy **S4**. We observe a significant reduction in memory utilisation between our real dataset and our first synthetic workload. This is a consequence of the characteristics of the real and synthetic workloads, with the real workload featuring a large spike of approximately 93,000 jobs in May 2010, while such extreme bursty behaviours are not observed in our synthetic workloads. This effect is also observable in Figure 18 as a modest reduction in execution time between the two workloads. Memory utilisation is shown to scale linearly with the size of workload, and is similar for both policies for all workloads. Figure 18 shows execution time to scale linearly with the size of workload. We observe that for smaller workloads that policy **S4** has similar execution time to policy **S1**, as the size of workload increases, the execution time required to evaluate policy **S4** increases more quickly.

7. PRIOR USE OF HTC-SIM

HTC-Sim has been under active development for the past four years, and has seen extensive use as a tool to evaluate the energy consumption of HTC workload based at Newcastle University. Initial work [56] investigated different resource selection algorithms (cf. Resource Policy in Figure 4), simple polices to handle jobs which are repeatedly evicted from resources (cf. Resubmission Policy) and polices for determining power management policies in clusters (cf. Cluster Policies).

The most significant energy consumption within a Desktop Grid HTC cluster can generally be attributed to jobs which are evicted due to interactive users taking control back from a resource. This problem is exacerbated further when jobs are incapable of completing due to excessive execution time or faulty hardware / software, leading to a situation where jobs are perpetually reallocated between computers – wasting time and energy. We term those jobs which were evicted as *'miscreant'* jobs. HTC-Sim was used to evaluate a number of candidate polices which sought reduce wasted energy in these circumstances [59]. We evaluated miscreant policies based on last execution time, total number of evictions and the reasons for job eviction (cf. Resubmission Policy). These policies offered a potential 50% saving of the energy for a HTC cluster when only considering evictions due to resource reboots. We are now evaluating two different approaches to energy saving to further reduce the impact of miscreant jobs – those of checkpointing and migration of jobs to different resources (cf. Checkpointing Policy) and more intelligent job placement using Reinforcement Learning [71] (cf. Resource Policy).

Checkpointing and Migration are fault-tolerance mechanisms commonly used in HTC systems. These allow the execution of long-running computational jobs on compute resources subject to hardware or software failures, as well as interruptions from resource owners and more important jobs. Checkpointing allows the application state of jobs to be periodically persisted to stable storage. In the scenario where a job is evicted by an interactive user arrival, the availability of a checkpoint saves both execution time, as the jobs do not need to restart from the beginning, and energy as effort is not expended repeating the previously performed work. However, the act of checkpointing itself impacts performance and energy, so careful balancing is required when determining how often checkpointing should be performed. Performing too many checkpoints will waste time and energy in performing checkpoints which will not be used – including time and energy required to move checkpoints to a new resource. Conversely, checkpointing too infrequently will require more work to be repeated. We demonstrate, using real-world datasets, that existing checkpointing strategies are inadequate at maintaining an acceptable level of energy consumption whilst retaining the performance gains expected. We identify factors important in determining whether to exploit checkpointing within an HTC environment, and propose novel strategies to curtail the energy consumption of checkpoint approaches while maintaining the performance benefits [58, 60, 61].

Significant energy can be saved by placing jobs onto a resource less likely to be used by an interactive user before the job completes. However, this is not possible to compute *a priori* as the times when an interactive user will login, nor the execution times for jobs, can be known at the time of resource selection. Analysis of our interactive user trace logs shows a high degree of daily and weekly seasonality, with significant variation also observed between different clusters. We evaluate the use of Reinforcement Learning (as a Resource Policy) in order to determine when and where jobs should be allocated within the institution. In doing so we seek to minimise energy consumption and reduce the number of evictions. Through this work we have shown that it is possible to save up to 53% of the wasted energy within an institution [57].

A modified version of HTC-Sim was used for evaluating the cost implications for running work on the cloud [67] – using the same HTC workload trace. These two simulation models were then combined in order to compare the cost of running workloads on the Cloud versus an existing Desktop Grid comprising non-dedicated resources.

HTC systems operating on non-dedicated computers (so called *'volunteer computing'*) offers a compelling opportunity to perform significant computation without the need to invest in computational resources. However, such systems are typically configured to prioritise the interactive user of the computer in situations where there is resource contention between users and HTC jobs. In [72] we relax some of the common policies used in management of computers in large organisations in order to evaluate if alternative policies which may have a minor impact on the primary users of the computers could save enough energy to make this impact tolerable. We focus on our scenario of institutional clusters, augmenting reboot policies to reduce their impact on HTC workloads, and introducing a scheme whereby student users arriving to a cluster are directed away from computers currently servicing HTC jobs.

Alongside the development of HTC-Sim, we have been developing an extension to the job scheduling (cf. Scheduling and Queueing Policy) and management system within HTCondor [1], allowing us to deploy a number of our developed polices [5]. Our simulation software is allowing us to build up confidence in our policy sets before deploying them to a production cluster.

8. FUTURE WORK

Computational grids may be used to execute jobs belonging to one of a number of categories [73]; Bag-of-task, Message Passing Interface (MPI), and Workflow. In this work we consider a Bag-of-tasks workload [74], comprising multiple independent jobs with no communication among each other. By contrast, MPI workloads are required 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, likewise the loss of one resource will cause the whole MPI application to terminate. We have a strong desire to broaden the applicability of HTC-Sim to other contexts, and in further work we shall extend HTC-Sim to other paradigms in large-scale distributed computing which would benefit from our simulation support. We shall extend our simulation environment to include; a) support MPI workloads spanning multiple compute nodes, b) incorporate support for Workflow workloads, e.g. by modelling the functioning of the Directed Acyclic Graph Manager (DAGMan) [75], the meta-scheduler used by HTCondor to handle the dependencies between Workflow jobs. Our preliminary work in this area is presented in [76]. This would allow the extension of existing DAG-based application mapping techniques [77, 78, 79] to be evaluated for energy consumption in the context of multi-use clusters. An overview of the workflow support of grid systems is available in [80].

In our previous experimentation, at most one HTC job may execute on a computer at a time. We are confident that further energy savings may be sought through the consolidating of multiple HTC jobs onto the same computer. We are not able to explore this at this time as our HTC workload trace lacks the resource consumption information required to determine which consolidation possibilities would be feasible. We see this as an important area of future work and are working to obtain the required trace data to reason over consolidation policies.

Dynamic Voltage and Frequency Scaling (DVFS) is commonly used within the literature to reduce energy consumption in HTC and HPC systems [81, 82, 83]. To date we consider many of these works to be complementary to our own but have been unable to evaluate them due to a lack of performance knowledge in our trace datasets. We see the exploration of DVFS policies within our context of an HTC system operating within a multi-use cluster of great interest, and shall extend our model of our compute resources to facilitate DVFS.

Finally, we see value in a combined approach whereby simulation-based studies facilitated by HTC-Sim may be verified against real infrastructure. However, in situations where energy consumption is considered, the required monitoring infrastructure involves often prohibitive capital expense. Furthermore, the deployment of invasive in-situ monitoring within a production environment is non-trivial. One promising possibility is in emerging systems which include energy monitoring deployed as standard. This is often at an aggregate level, e.g. a smart meter per cluster room, or measurement infrastructure at a per-rack level in a datacenter setting. While this would not provide per-machine energy consumption measurements, this would not be a major limitation when evaluating policies governing the operation of large-scale computing environments comprising many hundreds or thousands of computers. Furthermore, existing approaches to energy disaggregation [84] could be applied in these contexts to provide greater insights into individual computer usage within these environments. We see these as compelling areas of future work, and ones whose results could easily be combined with the existing HTC-Sim framework.

9. CONCLUSIONS

We have presented our Java-based trace-driven simulation, HTC-Sim, for simulating High Throughput Computing systems comprising both dedicated resources and multi-use resources shared with interactive users. We have presented the core model of the simulation along with discussion of implementation, the trace logs required and the methods needed to obtain such logs. We demonstrate how a set of resource selection policies for HTCondor can be tested using the simulation software. Although 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 performance impact of HTC-Sim both in terms of memory footprint and execution time. We demonstrate, through the use of synthetic trace logs, that the simulation software scales linearly in both memory and execution time with the number of simulated jobs; hence, we consider HTC-Sim a suitable tool to simulation large workloads. We go further to analyse the impact of slowdown and proportional energy impact on different job execution times and are able to show that, against commonly held beliefs, the 'best' execution times for an HTC job in our environment is around 6 to 7 hours in length.

Although HTC-Sim is not developed for normal users – we see it as a tool which would be used by either system administrators or researchers – it possesses a low barrier to entry. The software itself can be executed from a Java .jar file with an empty configuration file, thus allowing immediate execution of the software. Only if a user wishes to evaluate alternate configurations would they need to start learning about how to write a configuration file or develop their own Pluggable Policy implementations. As we supply the software with a number of alternate policy implementations users can try out many alternative cluster configurations without the need to develop their own code. This is in contrast to many large-scale computer simulation systems which require programmatic development from the outset.

We now continue to investigate more advanced policies to further reduce the energy consumption of HTC systems whilst minimising overheads. We are also extending HTC-Sim to model federation of HTC systems both on owned resources and resources obtained through Cloud bursting.

ACKNOWLEDGEMENT

We thank the authors of previous papers culminating in HTC-Sim. Newcastle University: Ben Allen, Clive Gerrard, Paul Haldane, Jonathan Noble, Dave Sharples & Dan Swan; Arjuna Technologies Ltd: Sindre Hamlander & Stuart Wheater; Red Hat Inc: Paul Robinson.

REFERENCES

1. Litzkow M, Livney M, Mutka MW. Condor-a hunter of idle workstations. ICDCS '88, 1998; 104–111.
2. Anderson DP. Boinc: A system for public-resource computing and storage. IEEE, 2004; 4–10.
3. Jarvis S, Thomas N, van Moorsel A. Open issues in grid performability. *IJSSSP* 2004; **5**(5):3–12.
4. Bertoldi P, Anatasiu B. Electricity Consumption and Efficiency Trends in European Union Status Report 2009.
5. McGough A, Gerrard C, Haldane P, Sharples D, Swan D, Robinson P, Hamlander S, Wheater S. Intelligent Power Management Over Large Clusters. *CPSCom*, 2010; 88–95.
6. BonFIRE Consortium. Bonfire (homepage) 2014. URL http://www.bonfire-project.eu/.
7. Kavoussanakis K, Hume A, Martrat J, Ragusa C, Gienger M, Campowsky K, Seghbroeck GV, Vázquez C, Velayos C, Gittler F. BonFIRE: the clouds and services testbed. *CloudCom 2013*, vol. 2, IEEE, 2013; 321–326.
8. Li B, Li J, Huai J, Wo T, Li Q, Zhong L. Enacloud: An energy-saving application live placement approach for cloud computing environments. *CLOUD 2009*, IEEE, 2009; 17–24.
9. Beloglazov A, Buyya R. OpenStack Neat: A framework for dynamic consolidation of virtual machines in OpenStack clouds–A blueprint. *Cloud Computing and Distributed Systems (CLOUDS) Laboratory* 2012; .
10. Naicken S, Livingston B, Basu A, Rodhetbhai S, Wakeman I, Chalmers D. The state of peer-to-peer simulators and simulations. *SIGCOMM Comput. Commun. Rev.* Mar 2007; **37**(2):95–98, doi:10.1145/1232919.1232932.
11. Evans NS, Grothoff C. Beyond simulation: Large-scale distributed emulation of p2p protocols. CSET'11, USENIX Association, 2011; 4–4.
12. Hibler M, Ricci R, Stoller L, Duerig J, Guruprasad S, Stack T, Webb K, Lepreau J. Large-scale Virtualization in the Emulab Network Testbed. *USENIX Annual Technical Conference*, 2008; 113–128.
13. Carrera EV, Pinheiro E, Bianchini R. Conserving disk energy in network servers. *Proceedings of the 17th annual international conference on Supercomputing*, ACM, 2003; 86–97.

14. Legrand A, Marchal L. Scheduling distributed applications: The simgrid simulation framework. *CCGrid*, 2003; 138–145.
15. Buyya R, Murshed M. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *CCPE* 2002; **14**(13):1175–1220.
16. Bell WH, Cameron DG, Capozza L, Millar AP, Stockinger K, Zini F. Optorsim - a grid simulator for studying dynamic data replication strategies. *IJHPCA* 2003; .
17. Buyya R, Ranjan R, Calheiros RN. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. *HPCS'09*, IEEE, 2009; 1–11.
18. Kliazovich D, Bouvry P, Audzevich Y, Khan SU. Greencloud: A packet-level simulator of energy-aware cloud computing data centers. *GLOBECOM*, 2010; 1–5.
19. Lim SH, Sharma B, Nam G, Kim EK, Das C. MDCSim: A multi-tier data center simulation, platform. *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, 2009.
20. Kliazovich D, Bouvry P, Khan SU. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing* 2012; **62**(3):1263–1283.
21. Méndez V, García F. Sicogrid: A complete grid simulator for scheduling and algorithmical research, with emergent artificial intelligence data algorithms.
22. Howell F, McNab R. SimJava: A discrete event simulation library for java. *Simulation Series* 1998; **30**:51–56.
23. Schwetman H. CSIM: a C-based process-oriented simulation language. *Proceedings of the 18th conference on Winter simulation*, ACM, 1986; 387–396.
24. Leijen D, Meijer E. 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.
25. Bucy JS, Schindler J, Schlosser SW, Ganger GR. The disksim simulation environment version 4.0 reference manual (cmu-pdl-08-101). *Parallel Data Laboratory* 2008; :26.
26. Minartz T, Kunkel J, Ludwig T. Simulation of power consumption of energy efficient cluster hardware. *Computer Science - Research and Development* 2010; **25**:165–175.
27. Verma A, Ahuja P, Neogi A. Power-aware dynamic placement of hpc applications. *Proceedings of the 22nd annual international conference on Supercomputing*, ACM, 2008; 175–184.
28. Terzopoulos G, Karatza HD. Dynamic voltage scaling scheduling on power-aware clusters under power constraints. *Distributed Simulation and Real Time Applications (DS-RT), 2013 IEEE/ACM 17th International Symposium on*, IEEE, 2013; 72–78.
29. Niemi T, Kommeri J, Happonen K, Klem J, Hameri AP. Improving energy-efficiency of grid computing clusters. *Advances in Grid and Pervasive Computing*, *LNCS*, vol. 5529. 2009; 110–118.
30. Ponciano L, Brasileiro F. On the impact of energy-saving strategies in opportunistic grids. *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, IEEE, 2010; 282–289.
31. Zikos S, Karatza HD. Performance and energy aware cluster-level scheduling of compute-intensive jobs with unknown service times. *Simulation Modelling Practice and Theory* 2011; **19**(1):239–250.
32. Faria I, Dantas M, Capretz MA, Higashino W. Network and Energy-Aware Resource Selection Model for Opportunistic Grids. *Convergence of Distributed Clouds, Grids and their Management (CDCGM) track of WETICE, 2014 IEEE 23rd International Workshop on*, 2014.
33. Da Silva DP, Cirne W, Brasileiro FV. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. *Euro-Par 2003 Parallel Processing*. Springer, 2003; 169–180.
34. Aupy G, Benoit A, Melhem RG, Renaud-Goud P, Robert Y. Energy-aware checkpointing of divisible tasks with soft or hard deadlines. *CoRR* 2013; **abs/1302.3720**.
35. Bellosa F. The benefits of event: driven energy accounting in power-sensitive systems. *Proceedings of the 9th workshop on ACM SIGOPS European workshop*, ACM, 2000; 37–42.
36. Singh K, Bhadauria M, McKee SA. Real time power estimation and thread scheduling via performance counters. *ACM SIGARCH Computer Architecture News* 2009; **37**(2):46–55.
37. Zedlewski J, Sobti S, Garg N, Zheng F, Krishnamurthy A, Wang RY. Modeling hard-disk power consumption. *FAST*, vol. 3, 2003; 217–230.
38. Brooks D, Tiwari V, Martonosi M. *Wattch: a framework for architectural-level power analysis and optimizations*, vol. 28. ACM, 2000.
39. Gurumurthi S, Sivasubramaniam A, Irwin MJ, Vijaykrishnan N, Kandemir M. Using complete machine simulation for software power estimation: The softwatt approach. *IEEE HPCA*, IEEE, 2002; 141–150.
40. Argollo E, Falcón A, Faraboschi P, Monchiero M, Ortega D. Cotson: infrastructure for full system simulation. *ACM SIGOPS Operating Systems Review* 2009; **43**(1):52–61.
41. Fan X, Weber WD, Barroso LA. Power provisioning for a warehouse-sized computer. *ACM SIGARCH Computer Architecture News*, vol. 35, ACM, 2007; 13–23.
42. Economou D, Rivoire S, Kozyrakis C, Ranganathan P. Full-system power analysis and modeling for server environments. International Symposium on Computer Architecture-IEEE, 2006.
43. Davis JD, Rivoire S, Goldszmidt M, Ardestani EK. 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.
44. Davis JD, Rivoire S, Goldszmidt M, Ardestani EK. Accounting for variability in large-scale cluster power models. Exascale Evaluation and Research Techniques Workshop (EXERT), 2011.
45. Meisner D, Wenisch TF. Peak power modeling for data center servers with switched-mode power supplies. *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, IEEE, 2010; 319–324.
46. Lange KD. Identifying Shades of Green: The SPECpower Benchmarks. *IEEE Computer* 2009; **42**(3):95–97.
47. Standard Performance Evaluation Corporation (SPEC). Spec virt_sc2013 (homepage) 2014. URL http://www.spec.org/virt_sc2013/.
48. Poess M, Nambiar RO, Vaid K, Stephens Jr JM, Huppler K, Haines E. Energy benchmarks: a detailed analysis. *e-Energy 2010*, ACM, 2010; 131–140.

49. SPEC. SPECpower_ssj2008. http://www.spec.org/power_ssj2008/.
50. Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd and Toshiba Corporation. ACPI Specification. http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf.
51. Belady C, Rawson A, Pfleuger J, Cader T. Green grid data center power efficiency metrics: PUE and DCiE. *Technical Report*, Green Grid 2008.
52. Iosup A, Li H, Jan M, Anoep S, Dumitrescu C, Wolters L, Epema DH. The grid workloads archive. *Future Generation Computer Systems* 2008; **24**(7):672–686.
53. Archive TGW. http://gwa.ewi.tudelft.nl/ 2014.
54. http://www.cs.huji.ac.il/labs/parallel/workload/.
55. Feitelson DG, Tsafrir D, Krakov D. Experience with using the Parallel Workloads Archive. *J. Parallel & Distributed Comput.* Oct 2014; **74**(10):2967–2982, doi:10.1016/j.jpdc.2014.06.013.
56. McGough AS, Forshaw M, Gerrard C, Robinson P, Wheater S. Analysis of power-saving techniques over a large multi-use cluster with variable workload. *Concurrency and Computation: Practice and Experience* 2013; **25**(18):2501–2522. URL http://dx.doi.org/10.1002/cpe.3082.
57. McGough AS, Forshaw M. Reduction of wasted energy in a volunteer computing system through reinforcement learning. *Sustainable Computing: Informatics and Systems* 2014; .
58. Forshaw M. Operating policies for energy efficient large scale computing. PhD Thesis, Newcastle University, UK 2015.
59. McGough A, Forshaw M, Gerrard C, Wheater S. Reducing the number of miscreant tasks executions in a multi-use cluster. *Cloud and Green Computing (CGC), 2012 Second International Conference on*, 2012; 296–303, doi: 10.1109/CGC.2012.111.
60. Forshaw M, McGough AS, Thomas N. On energy-efficient checkpointing in high-throughput cycle-stealing distributed systems. *3rd International Conference on Smart Grids and Green IT Systems (SMARTGREENS)*, 2014.
61. Forshaw M, McGough AS, Thomas N. Energy-efficient checkpointing in high-throughput cycle-stealing distributed systems. *Electronic Notes in Theoretical Computer Science* 2015; **310**:65–90.
62. Feitelson DG. Metrics for parallel job scheduling and their convergence. *Job Scheduling Strategies for Parallel Processing*, Springer, 2001; 188–205.
63. Feitelson DG, Rudolph L, Schwiegelshohn U, Sevcik KC, Wong P. Theory and practice in parallel job scheduling. *Job scheduling strategies for parallel processing*, Springer, 1997; 1–34.
64. McGough A, Gerrard C, Noble J, Robinson P, Wheater S. Analysis of Power-Saving Techniques over a Large Multi-use Cluster. *IEEE DASC 2011*, 2011; 364–371, doi:10.1109/DASC.2011.78.
65. Sourceforge project. The iperf project. http://iperf.sourceforge.net/.
66. Department of Energy and Climate Change, UK Gov. CRC Energy Efficiency Scheme Order: Table of Conversion Factors 2013/14 2014; .
67. McGough AS, Forshaw M, Gerrard C, Wheater S, Allen B, Robinson P. Comparison of a cost-effective virtual cloud cluster with an existing campus cluster. *Future Generation Computer Systems* 2014; .
68. Solomon M. The ClassAd Language Reference Manual. *Computer Sciences Department, University of Wisconsin, Madison, WI, Oct* 2003; .
69. Bradley JT, Forshaw M, Stefanek A, Thomas N. Time-inhomogeneous population models of a cycle-stealing distributed system. *29th Annual UK Performance Engineering Workshop (UKPEW) 2013*, 2013; 8–13.
70. Forshaw M, Thomas N, McGough S. Trace-driven simulation for energy consumption in high throughput computing systems. *18th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2014.
71. Sutton R, Barto A. *Reinforcement Learning: An Introduction*. 1998.
72. Forshaw M, McGough AS. Flipping the priority: effects of prioritising htc jobs on energy consumption in a multi-use cluster. *Proceedings of the 8th International Conference on Simulation Tools and Techniques*, 2015; 357–364.
73. Rahman M, Ranjan R, Buyya R, Benatallah B. A taxonomy and survey on autonomic management of applications in grid computing environments. *Concurrency and computation: practice and experience* 2011; **23**(16):1990–2019.
74. Cirne W, Brasileiro F, Sauvé J, Andrade N, Paranhos D, Santos-Neto E, Medeiros R. Grid computing for bag of tasks applications. *Proc. of the 3rd IFIP Conference on E-Commerce, E-Business and EGovernment*, Citeseer, 2003.
75. Tannenbaum T, Wright D, Miller K, Livny M. Condor: a distributed job scheduler. *Beowulf cluster computing with Linux*, MIT press, 2001; 307–350.
76. McGough AS, Forshaw M. Energy-aware simulation of workflow execution in high throughput computing systems. *19th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2015.
77. He L, Jarvis SA, Spooner DP, Bacigalupo D, Tan G, Nudd GR. Mapping dag-based applications to multiclusters with background workload. 2005; 855–862 Vol. 2.
78. He L, Jarvis SA, Spooner DP, Nudd GR. Performance evaluation of scheduling applications with dag topologies on multiclusters with independent local schedulers. *IPDPS*, 2006.
79. He L, Jarvis SA, Spooner DP, Nudd GR. Dynamic, capability-driven scheduling of dag-based real-time jobs in heterogeneous clusters. *IJHPCN* 2004; **2**(2/3/4):165–177.
80. Yu J, Buyya R. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing* 2005; **3**(3-4):171–200.
81. Zhang Y, Chakrabarty K. Energy-aware adaptive checkpointing in embedded real-time systems. *Design, Automation and Test in Europe Conference and Exhibition, 2003*, 2003; 918–923, doi:10.1109/DATE.2003.1253723.
82. Unsal OS, Koren I, Krishna CM. Towards energy-aware software-based fault tolerance in real-time systems. *ISLPED 2002*, 2002; 124–129.
83. Mills B, Grant RE, Ferreira KB, Riesen R. Evaluating energy savings for checkpoint/restart. *Proceedings of the 1st International Workshop on Energy Efficient Supercomputing*, E2SC '13, ACM, 2013; 6:1–6:8.
84. Batra N, Kelly J, Parson O, Dutta H, Knottenbelt W, Rogers A, Singh A, Srivastava M. Nilmtk: An open source toolkit for non-intrusive load monitoring. *ACM e-Energy*, 2014; 265–276.