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.**

Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 310 (2015) 65–90

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Energy-efficient Checkpointing in High-throughput Cycle-stealing Distributed Systems

Matthew Forshaw[1]

School of Computing Science
Newcastle University
Newcastle upon Tyne, UK

A. Stephen McGough[2]

School of Engineering and Computing Sciences
Durham University
Durham, UK

Nigel Thomas[3]

School of Computing Science
Newcastle University
Newcastle upon Tyne, UK

Abstract

Checkpointing is a fault-tolerance mechanism commonly used in High Throughput Computing (HTC) environments to allow the execution of long-running computational tasks on compute resources subject to hardware or software failures as well as interruptions from resource owners and more important tasks. Until recently many researchers have focused on the performance gains achieved through checkpointing, but now with growing scrutiny of the energy consumption of IT infrastructures it is increasingly important to understand the energy impact of checkpointing within an HTC environment. In this paper we demonstrate through trace-driven simulation of real-world datasets that existing checkpointing strategies are inadequate at maintaining an acceptable level of energy consumption whilst maintaing the performance gains expected with checkpointing. Furthermore, we identify factors important in deciding whether to exploit checkpointing within an HTC environment, and propose novel strategies to curtail the energy consumption of checkpointing approaches whist maintaining the performance benefits.

Keywords: Energy efficiency, Checkpointing, Migration, Fault tolerance, Desktop Grids

[1] Email: m.j.forshaw@newcastle.ac.uk
[2] Email: stephen.mcgough@durham.ac.uk
[3] Email: nigel.thomas@ncl.ac.uk

http://dx.doi.org/10.1016/j.entcs.2014.12.013

1 Introduction

The issue of performance and reliability in cluster computing have been studied extensively over many years [18], resulting in techniques to improve these properties. The issue of cluster *'performability'* is relatively well understood, though until recently little consideration has been given to the energy impact of cluster performability.

High-throughput cycle stealing distributed systems such as HTCondor [23] and BOINC [1] allow organisations to leverage spare capacity on existing infrastructure to undertake valuable computation. These High Throughput Computing (HTC) systems are frequently used to execute large numbers of long-running computational tasks, so are susceptible to interruption due to hardware and software failures. Furthermore, like many organisations we leverage institutional *'multi-use'* clusters comprised of student and staff machines, where jobs may also be interrupted when an interactive user starts to use a machine. Such interruptions lead to the tasks being evicted from the resource, increasing task makespan and wasted energy.

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

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

In recent years attention has turned to the energy consumption of IT infrastructures within organisations. Aggressive power management policies are often employed to reduce the energy impact of institutional clusters, but in doing so these policies severely restrict the computational resources available for high-throughput systems. These policies are often configured to quickly transition servers and end-user cluster machines into low power states after only short idle periods, further compounding the issue of reliability and lowering the availability perceived by applications running in the system.

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

- Evaluate the energy impact of the two checkpoint schemes previously proposed in the literature [38] [31] for a real workload.
- Propose novel checkpoint policies for high-throughput computing environments and evaluate their performance for a real workload in terms of average task makespan and energy consumption.

- Develop a trace-driven simulation environment as a basis for research into energy-efficient fault tolerance approaches for HTC systems.

The rest of the paper is organised as follows. We outline related work in Section 2 and introduce our experimental approach and trace-driven simulation of checkpointing in a high-throughput computing environment using real-world datasets in Section 3. Section 4 describes a number of existing checkpointing strategies, and we propose novel energy- and failure-aware checkpoint strategies. In Section 5 we demonstrate the detrimental effects of the proposed checkpointing policies on energy consumption, motivating the need for an increased understanding of the impact of checkpointing strategies within HTC clusters. Finally we discuss key considerations when adopting checkpointing in HTC clusters in Section 6 and conclude in Section 7.

2 Related Work

2.1 Checkpointing in real-time systems

Previous works in energy-aware checkpointing have primarily focused on real-time systems [41,37,29] subject to strict energy and deadline constraints.

Zhang *et al.* [41] propose an adaptive checkpointing scheme to maximise the probability of satisfying a task's deadline in the presence of k faults, specified by a pre-defined fault tolerance requirement. Energy consumption is then introduced as a secondary optimisation criteria, with Dynamic Voltage Scaling (DVS) employed to maintain a processor in low power state, transitioning to higher frequency operating modes when required to satisfy a task's deadline.

Melhem *et al.* [29] propose a similar approach, employing DVS in the absence of failures to leverage *'slack'* time between a task's deadline and expected completion time, transitioning a processor into a less performant but more energy efficient operating state.

Unsal *et al.* [37] evaluate the energy characteristics of an Application-Level Fault Tolerance (ALFT) scheme, where redundancy and recovery logic is incorporated at the application level, rather than being provided at the system or hardware level and propose a task scheduling heuristic reducing energy consumption by up to 40%.

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

2.2 Checkpointing in HPC

More recently, research has sought to understand the overheads and energy implications of fault tolerance mechanisms, including checkpointing, in anticipation of exascale High-Performance Computing (HPC). Bouguerra *et al.* [6] investigate

the impact of combined proactive and preventative checkpointing schemes in HPC systems, achieving up to a 30% increase in computational efficiency with negligible increase in overheads, but without consideration for its impact on energy consumption.

At exascale, increased frequency of faults are anticipated and energy consumption is a key issue [10]. To this end, Diouri *et al.* explore the energy consumption impact of uncoordinated and coordinated checkpointing protocols on an MPI HPC workload [14], while Mills *et al.* demonstrate energy savings by applying Dynamic Voltage and Frequency Scaling (DVFS) during checkpointing [30].

Further works focus on energy and scalability issues relating to persisting checkpoint images to stable storage. Saito *et al.* [36] consider energy saving when persisting checkpoint images, employing profile-based I/O optimisation to reduce the energy consumption of checkpointing to NAND flash memory by ∼40-67%.

We consider the application of DVS [41,37] and DVFS [30] to reduce the energy consumption of checkpoint operations to be complementary to our approaches.

2.3 Checkpointing in HTC systems

The application of checkpointing in High-Throughput Computing environments and Fine-Grained Cycle Sharing (FGCS) systems is explored extensively in [34,7], though without consideration for its implications for energy consumption.

Aupy *et al.* [2] investigate energy-aware checkpointing strategies in the context of arbitrarily divisible tasks. While divisible tasks encompasses a number of common applications including BLAST sequencing and parallel video processing [40], such tasks represents only a proportion of our workload, and HTC systems do not typically have control over the division of batched tasks.

2.4 Simulation

A number of Grid and Cluster level simulators exist including SimGrid [20], GridSim [8], and OptorSim [4] 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 [9], GreenCloud [19], and MDCSim [22]. However, these do not allow modelling of multi-use clusters with interactive user workloads, nor do they support checkpointing.

Zhou *et al.* [42] propose an extension to the CloudSim [9] framework to support simulation of fault tolerance mechanisms but its codebase has not been made publicly available.

Vieira *et al.* [39] propose ChkSim, a Java-based simulation environment for the evaluation of checkpointing algorithms. The tool focuses on checkpointing approaches for workloads comprising groups of dependent processes communicating with one another across the network, equivalent to an MPI HPC workload. ChkSim focuses on the number of unused checkpoints as its key metric of checkpoint

performance; however it does not assess the impact of checkpointing schemes on energy consumption and may not easily be adapted to model a high-throughput environment and interactive user workloads.

3 Simulation

In this paper, we evaluate the efficacy of existing checkpointing schemes using trace-driven simulation on a real dataset collected during 2010 at Newcastle University [26], comprising details of all job submissions to Newcastle University's HTCondor [23] cluster and interactive user activity for the twelve month period.

3.1 Datasets

In 2010, the Newcastle University HTCondor cluster comprised 1,359 machines from 35 computer clusters. The opening hours of these clusters varied, with some respecting office hours, and others available for use 24 hours a day. Clusters may belong to a particular department within the University and serve a particular subset of users, or may be part of a common area such as the University Library or Students' Union building. Computers within the clusters are replaced on a five-year rolling programme with computers falling into one of three broad categories as outlined in Table 1. Energy consumption values are *'nameplate'* values obtained from manufacturer documentation for the machines provisioned in 2010.

The University has a policy to minimise energy consumption on all computational infrastructure which has been in place for a number of years, governing the provisioning of hardware. Hence the 'Normal' computers have been chosen to be energy efficient. 'High End' computers are provisioned for courses requiring large computational and/or rendering requirements such as CAD or video editing, as such they have higher energy requirements. 'Legacy' computers pre-date the policy of purchasing energy efficient computers and are also the oldest equipment within the cluster. All computers within a cluster are provisioned at the same time and will contain equivalent computing resources. Thus there is a wide variance between clusters within the University but no significant variance within clusters.

Figure 1 shows all HTCondor job submissions for 2010. Our workload comprises primarily batch task submission, with a mean submission rate of 1,454 job submis-

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

Table 1
Computer Types

sions per day. However, the workload exhibits significant variations, with half of the days in the year observing fewer than 65 job submissions, and a number of periods of extremely large batch submission, for example 3rd June 2010 which featured ∼93,000 job submissions. Figure 2 shows the seasonal nature of interactive user activity within these clusters, demonstrating clear differences between weekends and weekdays, as well as term-time and holiday usage.

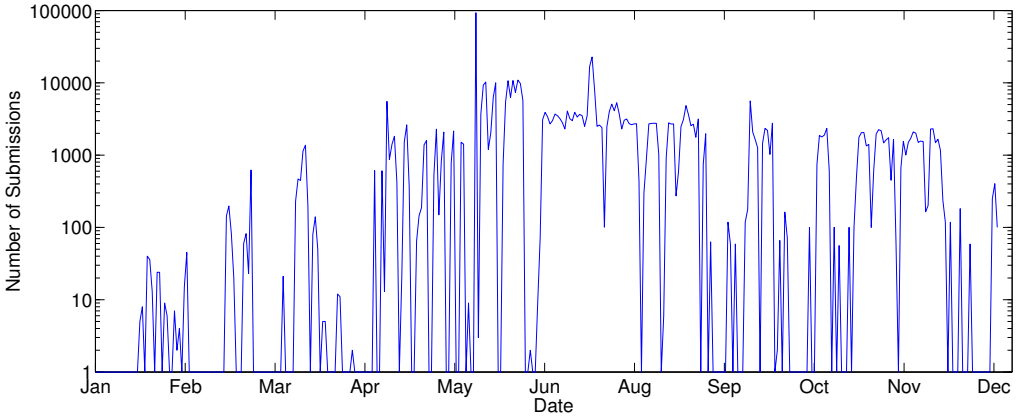


Fig. 1. HTCondor job submissions

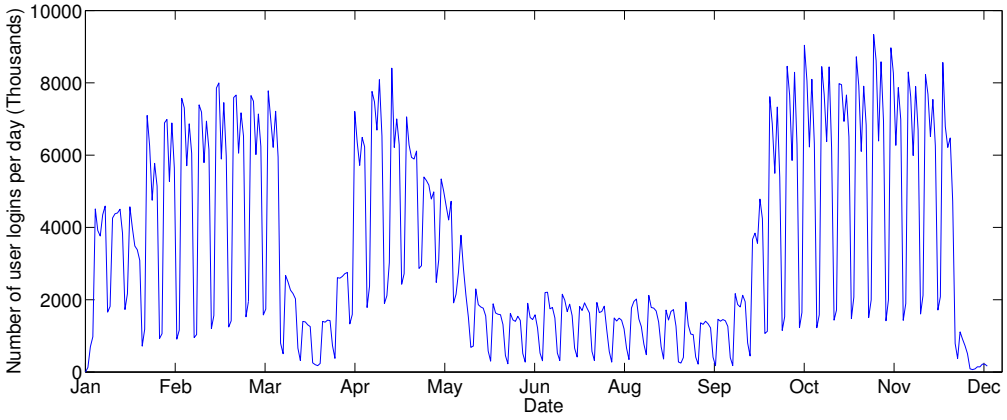


Fig. 2. Interactive user arrivals

3.2 Simulation system

In this work, we extend our trace driven simulation model of a shared resource High Throughput Computing system, based around the HTCondor software [27,25,24]. This Java-based simulation software offers a number of benefits over a measurement approach, allowing us to rapidly evaluate new policy ideas and scheduling decisions in a controlled and repeatable manner, without the need for a costly testing environment, and with isolation from variability introduced by evaluations based on a live

HTCondor environment. As the traffic observed in our environment is highly seasonal, a trace driven simulation approach also allows us to compare policies across various workload and interactive user requirements. The simulation environment is designed in such a way that policies evaluated by simulation may then be easily deployed into a real HTCondor environment [28].

The behaviour of the simulation software is informed by three files, the first describing the policy configuration to use for the simulation, the second a trace log of user access to the computers and the third file a trace log of HTCondor workload. The user trace data indicates login and logout time for the user, and the specific computer that the user occupied. In this paper we do not simulate alterations to this usage pattern. The high-throughput trace data, by contrast, contains only the time that the jobs were submitted, their duration and their memory footprint at time of completion. By interplaying these trace datasets we are able to accurately model the operation of the Newcastle University HTCondor system and computer clusters.

We extend our simulation environment to model the checkpoint model introduced in Section 3.3, and evaluate the impact of enacting various checkpointing policies outlined in Section 4 within the system. While in this work we primarily consider energy consumption and average task makespan, our simulation records numerous additional performance measures, enabling us to evaluate the impact of policies on all areas of the system.

In previous work [27] we investigated the impact of resource allocation strategies on the energy efficiency of high-throughput systems, allocating jobs to resources based on energy efficiency and estimated likelihood of interruption. Throughout this work we consider a random resource allocation strategy as most representative of default policies in many HTC systems. We provide results averaged across multiple simulation runs and report the variability introduced into results as a consequence of this non-deterministic resource allocation.

The introduction of checkpoint and migration strategies to HTC systems exacerbates the issue of wasted execution through the repeated allocation of *'bad'* tasks, those tasks which due to unfulfilled task requirements or faulty operation will never complete [24]. In order to curtail such executions and isolate the impact of checkpointing strategies on the operation of the system, throughout our experiments we bound execution time to a total of 24 hours, which is equivalent to the maximum availability period observed in our HTCondor cluster due to nightly cluster reboots.

Though our simulation environment is designed based on the HTCondor system, our representation of HTC workloads and computational resources are generic, so we believe our results to be easily generalisable to similar high-throughput computing environments.

3.3 Checkpointing and Failure Model

Choi *et al.* [11] present a classification of two types of failures encountered on desktop grid environments: *volatility failures* including machine crashes and unavailability due to network issues, and *interference failures* arising from the volunteer nature

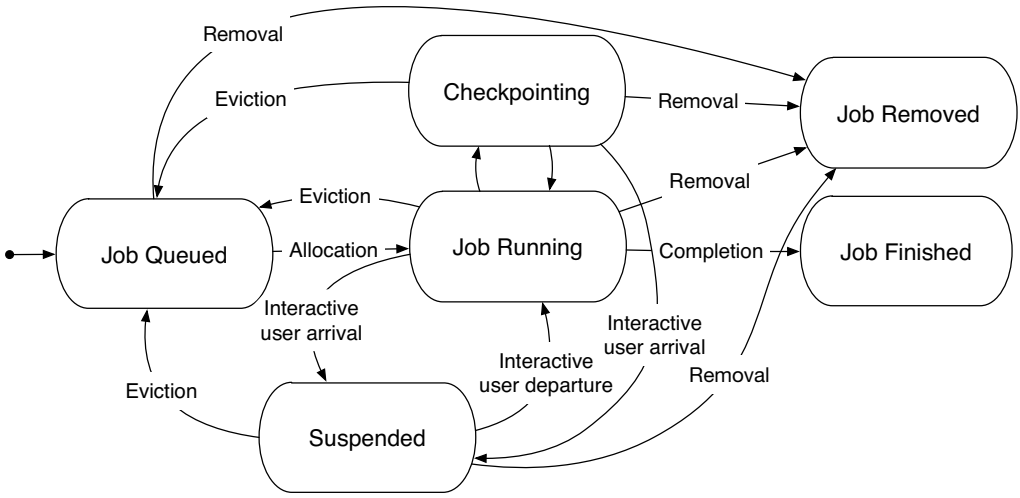


Fig. 3. Job state transition diagram

of the resources. It is these *interference failures* which we consider throughout this work. Furthermore, we consider resource volatility in the form of scheduled nightly reboots for maintenance.

Figure 3 shows the state transition diagram for the execution of a single job in our system in the presence of these failures. Jobs are submitted by users and join a queue prior to being allocated on a resource. Once running, jobs are susceptible to interruption due to interactive users arriving on the resource. Jobs may be evicted immediately, or suspended for a period of time, after which jobs are evicted if the interactive user has not departed. Furthermore, jobs may be manually removed by their owner or a system administrator while in any non-final state.

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

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

3.4 Power model

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 [5] when developing predictive models of energy consumption. These models tend to require significant architecture knowledge and typically were not generalisable to other hardware, nor scalable to entire computer systems. A

strong linear correlation exists between energy consumption and CPU utilisation with works using this as a predictor of energy consumption [15], while others derive linear regression models based on utilisation of CPU, memory and storage subsystems [13,35]. The literature provides models both for single servers [13,35], groups of systems [33,16,15] and virtualised environments [12]

In this work we lack resource utilisation information for the HTC worker nodes, so adopt a power model employing easily obtained *'nameplate'* power consumption values where a machine may belong to one of three operating states as defined in the Advanced Configuration and Power Interface (ACPI) specification [17]; *active* and *idle* (S0), or *sleep* (S3). Table 1 shows the three classes of machines considered in our simulation, and the associated power values in each state.

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

When devising checkpointing strategies we ensure they rely only upon readily available system information and avoid expensive computation, such that they may be easily implemented in a real HTC system. The policies outlined below make use of system information exposed through the HTCondor ClassAd mechanism [32] and other HTC systems, so we consider each of these policies to be realistic.

4 Policies

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

4.1 Baseline policies

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

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

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

4.2 Checkpoint Interval

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

C(n): Each job is checkpointed every n minutes. Hourly checkpointing (**C**(*60*)) is frequently considered in the literature and the HTCondor default strategy equates to **C**(*180*) [38].

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

$$I_{j,\tau} = \begin{cases} n_{open} & \text{if } \exists s_{i,j}, f_{i,j} : s_{i,j} - c_j \leq \tau \leq f_{i,j} - c_j \\ n_{closed} & \text{otherwise} \end{cases} \tag{1}$$

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

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

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

$$i = \begin{cases} m - j_{min} + R & \text{if } j_{min} < (m - t) \\ 60 + (m - j_{min} + R) & \text{otherwise} \end{cases} \tag{2}$$

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

In situations where large batches of jobs are submitted to the system at the same time, this may result in many checkpoints being taken simultaneously. In a real system this could impose significant load on the network and storage nodes. In order to mitigate these potential effects, we introduce a random component in the checkpoint interval R, where R is a random variable uniformly distributed on $[-r, r]$, measured in minutes. As the value of r increases the system will become less susceptible to large numbers of simultaneous checkpoints caused by batch arrivals, but limit the ability of the policy to leverage the minute-in-hour period behaviour in checkpoint scheduling.

Ratio(p): In this policy we place an upper bound on the proportion of execution

time consumed through checkpointing operations. The checkpoint interval i for a given job j is calculated as $i_j = \frac{d_j}{p}$ where d_j is the estimated checkpoint duration for job j, and p the maximum proportion of execution time to be occupied by checkpointing.

StartDelay(n, d): Through preliminary investigation we observe a significant proportion of wasted checkpoints occurred as a result of checkpointing of short-running jobs. While execution time of tasks is not known *a priori* and user estimates have been shown to be inaccurate [3], this policy aims to curtail this waste, applying a start delay d before which a newly allocated task may not be checkpointed, after which tasks are checkpointed every n minutes.

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

$$i_j^n = \begin{cases} a & \text{if } n = 0 \\ ar^{n-1} & \text{if } n \geq 1 \end{cases} \tag{3}$$

where a represents the initial checkpoint interval, r ($r \geq 0$) represents the *'common ratio'* for the sequence. The *'Exponential backoff'* policy proposed by Oliner *et al.* [31] is equivalent to the geometric progression policy where $r = 2$.

4.3 Skip checkpoint policies

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

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

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

$$\begin{cases} (t - c_j) \leq d \;\; \text{if} \;\; \left| \left\{ e_i \middle| e_i \in E \wedge t - w \leq T(e_i) \leq t \right\} \right| \geq l \\ \quad skip \qquad\qquad\qquad\qquad \text{otherwise} \end{cases} \tag{4}$$

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

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

interactive user arrivals to the whole system.

4.4 Proactive migration

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

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

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

5 Results

The impact on average task overhead and energy consumption for *None* and *Opt* policies on average task makespan and energy consumption is shown in Figures 4 and 5 respectively. All results presented are mean values obtained from fifty simulation runs, with error bars signifying 95% confidence interval values.

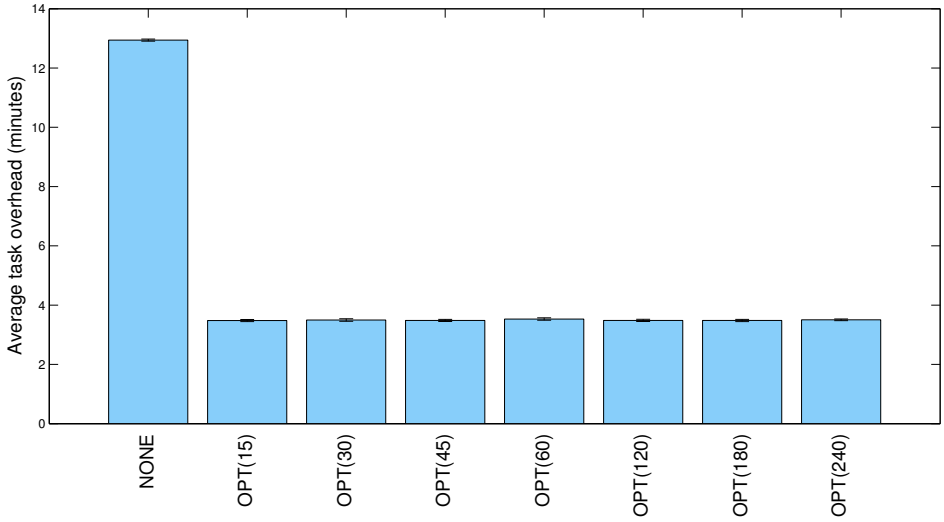


Fig. 4. Average Task Overheads

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

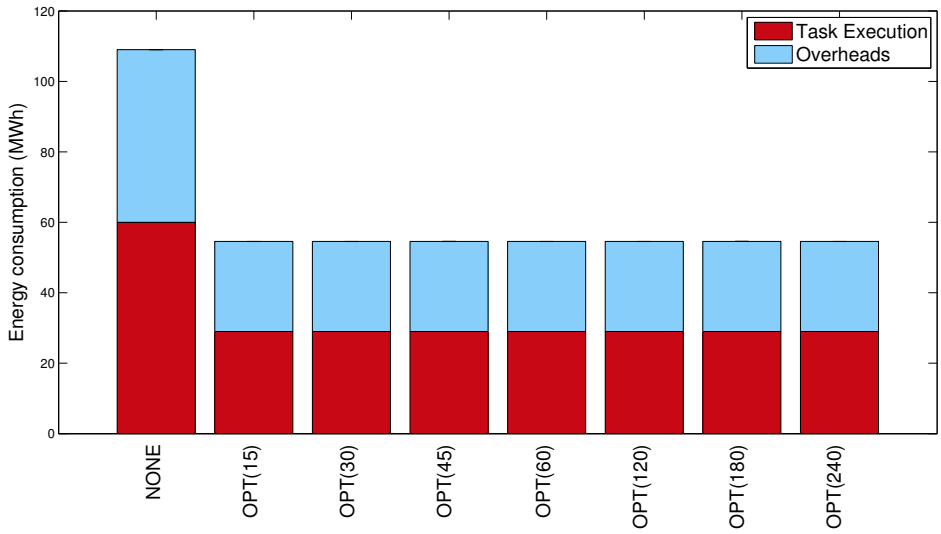


Fig. 5. Energy Consumption

checkpoints is shown to have little impact on the efficacy of checkpointing in the presence of optimal checkpoint interval selection.

5.1 Policy Results

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

$$\text{Benefit} = 1 - \left(\frac{v_x - v_{opt}}{v_{none} - v_{opt}} \right) \tag{5}$$

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

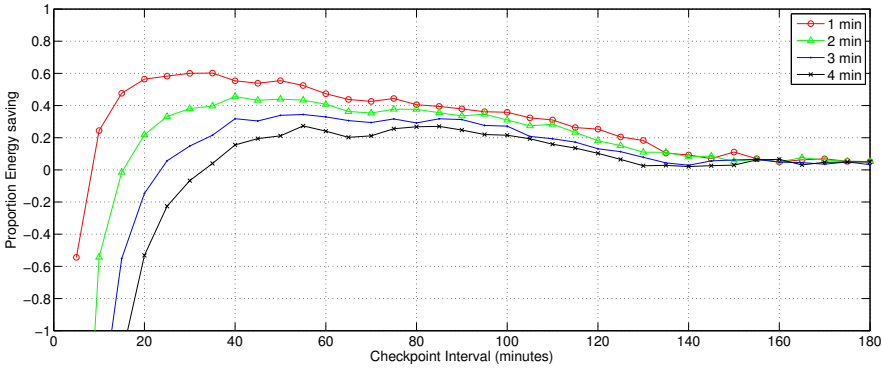
Figures 6a, 6b, and 6c show the impact of the policy on makespan, energy consumption and checkpoint utilisation for our Fixed (periodic) checkpointing policy. Results are shown for checkpoint generation durations ranging from one to four minutes. We observe this policy has the potential to achieve energy and makespan savings which are as much as 60% of optimal when the policy is correctly parameterised. The optimal checkpoint interval is shown to be dependent on the checkpoint duration for the workload, with the optimal interval for one- and four-minute jobs centred around 30 and 55 minutes respectively. In all cases, where a checkpoint interval shorter than 30 minutes are selected performance degrades significantly, with the cost of checkpoint operations exceeding the possible savings, leading to worsening overall performance and energy consumption. As the length of checkpoint intervals increase, the benefits of checkpointing tends towards zero, representing no

checkpointing of jobs. We observe only a small proportion of successfully generated checkpoints are utilised under the Fixed policy, with the time taken to generate checkpoints having negligible impact. Though utilisation rises to approximately 15% for a checkpoint interval of 180 minutes, the benefit of a job resuming from a checkpoint generated that far in the past would be limited. When considering the checkpoint strategies previously considered in the literature, hourly checkpointing ($\mathbf{C}(60)$) delivers good performance dependent on the time required to generate checkpoints for jobs, but we show the HTCondor default of $\mathbf{C}(180)$ [38] to have little benefit for our workload.

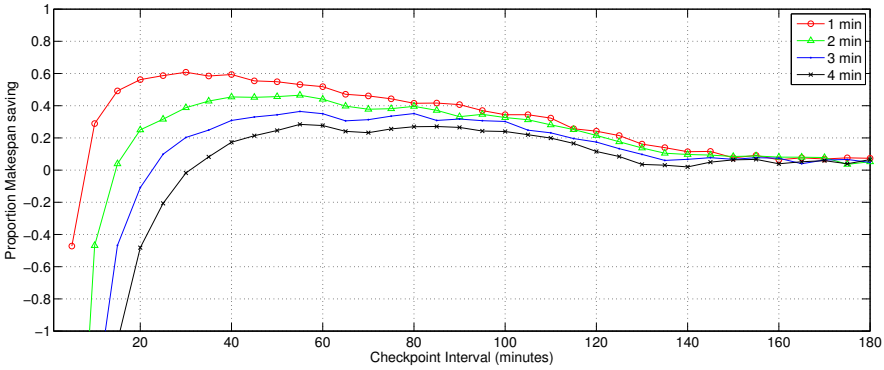
In Figures 7a, 7b and 7c we compare our Fixed periodic scheme with our Scheduled proactive migration policy, both in isolation (SR) and in combination with our ClosedCluster skip policy (CCSR). To aid readability we provide results for each policy for checkpoint durations of one and four minutes. When considering the ClosedCluster policy with Scheduled reboot proactive migration, we observe significant improvements in average task overhead and energy consumption, with the policy outperforming the Fixed periodic checkpointing scheme for all lengths of checkpoint interval. Though the greatest proportional makespan and energy saving is only found to rise from 0.6 for the Fixed periodic scheme to 0.7 for the CCSR scheme, this improvement is observed across a much wider range of checkpoint intervals, making these policies much less susceptible to poor performance due to sub-optimal checkpoint interval selection. Furthermore, we observe a significant increase in the utilisation of checkpoints generated in all cases.

In Figures 8a, 8b, and 8c we present the results of our Geometric policy. Results are shown for a 30 minute checkpoint interval, and varying common ratio parameter r. We find this policy to provide benefits to energy and makespan for all values of r. The best selection of parameter r is dependent on checkpoint duration, as $r \approx 1$ for 1 minute checkpoints, and $r \approx 2$ for 4 minute checkpoints. Furthermore, the selection of this common ratio is dependent on the composition of the HTC workload, with a greater proportion of shorter or longer jobs impacting on the best value to select. An interesting extension of this policy would be to explore the selection of r based on the expected execution time of the workload.

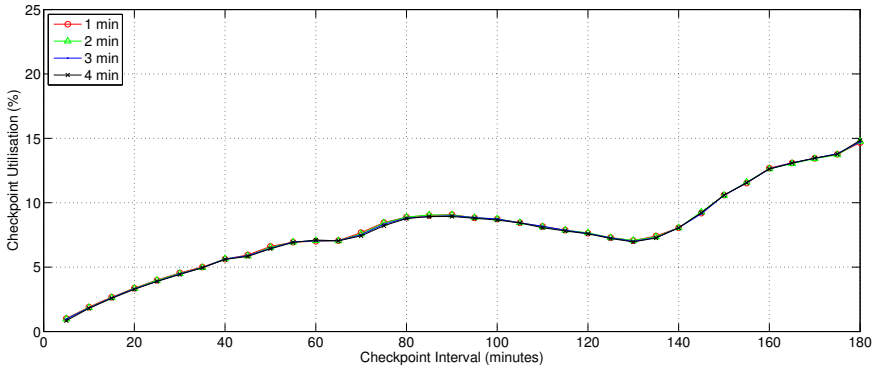
Results of our MinuteInHour policy are shown in Figures 9a, 9b, and 9c. Using knowledge of interactive user activity to inform the placement of checkpoint operations is found to result in an $\approx 20\%$ improvement in energy and makespan saving where $m = 55$ compared to the checkpoints carried out on the hour boundary. We introduce the random component r to prevent large numbers of checkpoints scheduled at the same time, leading to network congestion and increased transfer delays. To exemplify the potential impact of such an adjustment, we show the results for a deliberately conservative value of $r = 5$ minutes. Under this policy, energy and makespan savings are lessened, particularly for the case of four minute checkpoints due to checkpoint operations being deferred towards the hour boundary, increasing their likelihood of interruption. Utilisation remains largely unaffected by the choice of parameter m. In a real system we anticipate a much smaller value of r to be adequate.



(a) The impact of the fixed checkpoint policy on energy consumption.

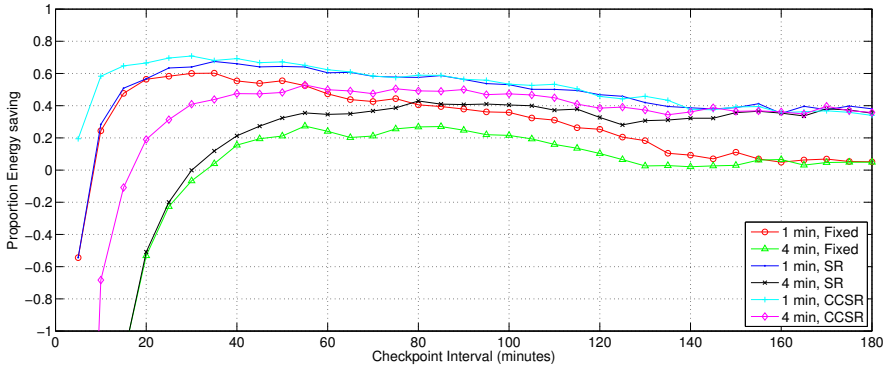


(b) The impact of the fixed checkpoint policy on average task overhead.

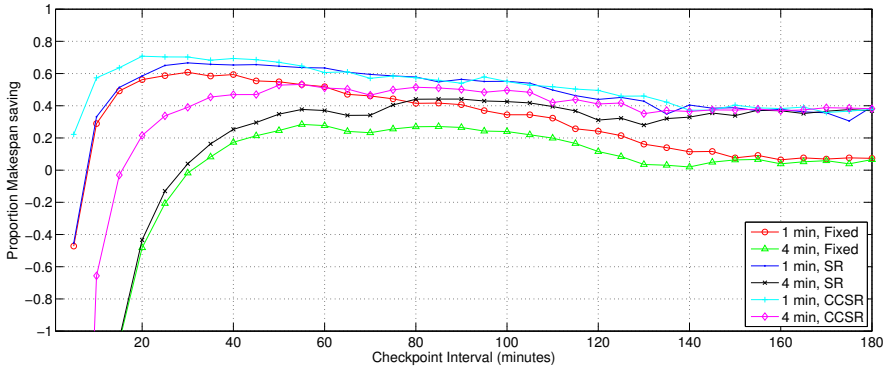


(c) The impact of the fixed checkpoint policy on checkpoint utilisation.

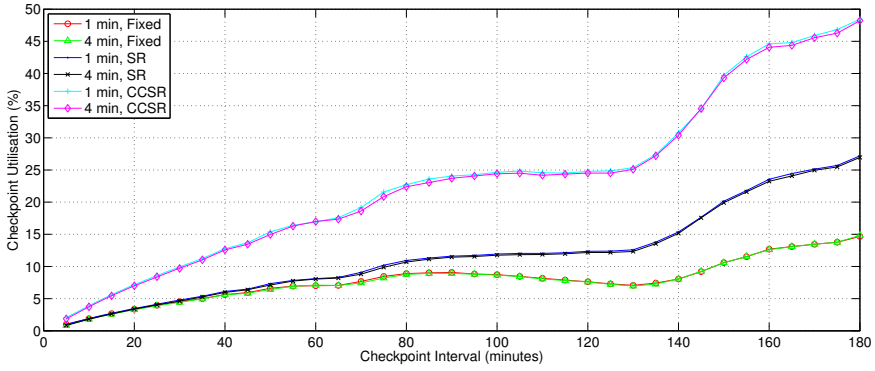
Fig. 6. Fixed checkpoint policy



(a) The impact of the ClosedCluster policy and Scheduled proactive migration on energy consumption.

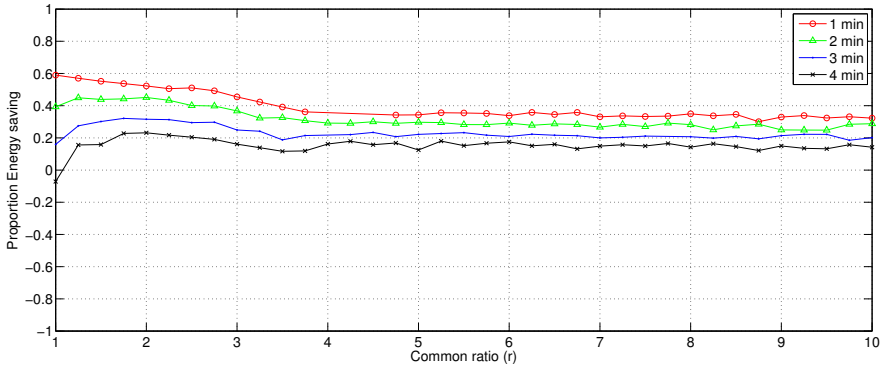


(b) The impact of the ClosedCluster policy and Scheduled proactive migration on average task overhead.

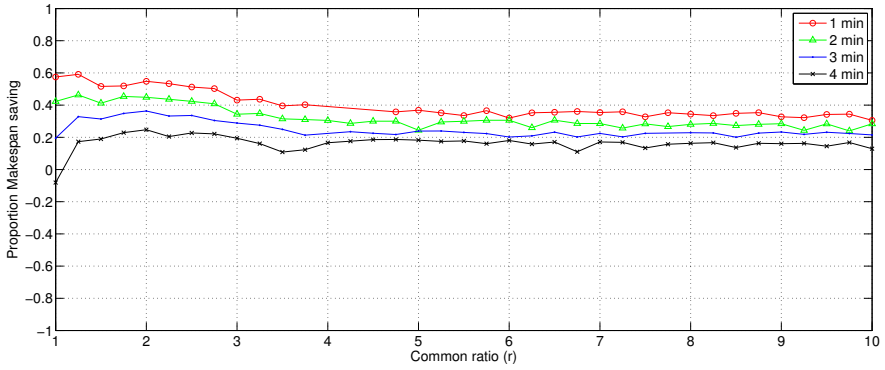


(c) The impact of the ClosedCluster policy and Scheduled proactive migration on checkpoint utilisation.

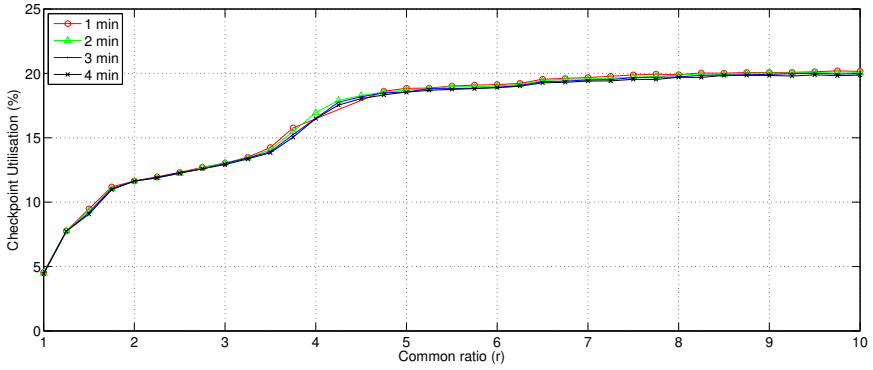
Fig. 7. ClosedCluster policy and Scheduled proactive migration



(a) The impact of the Geometric policy on energy consumption.

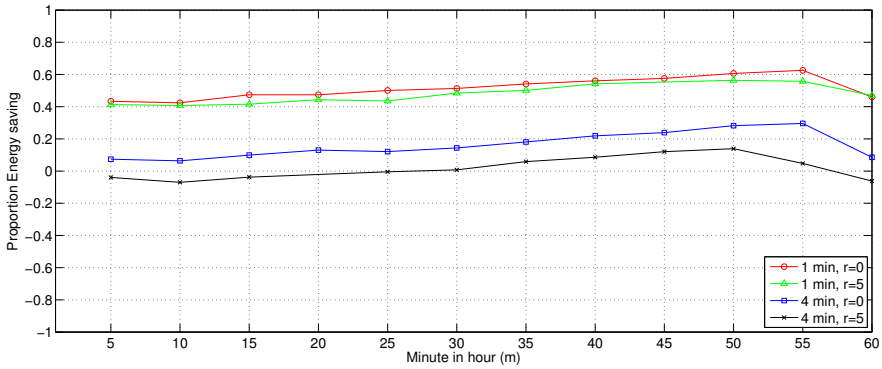


(b) The impact of the Geometric policy on average task overhead.

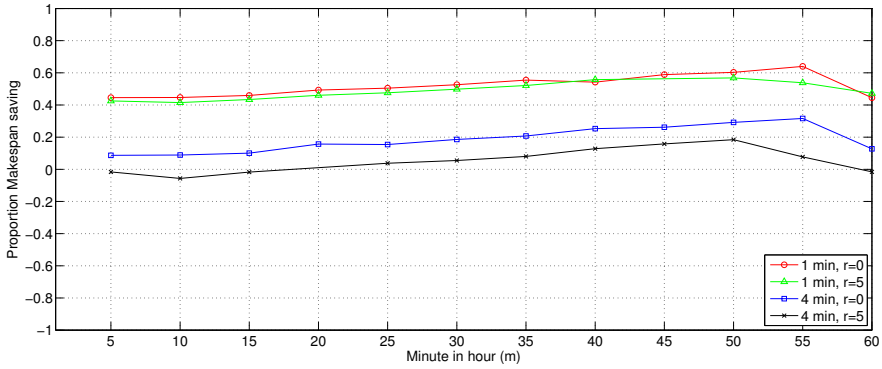


(c) The impact of the Geometric policy on checkpoint utilisation.

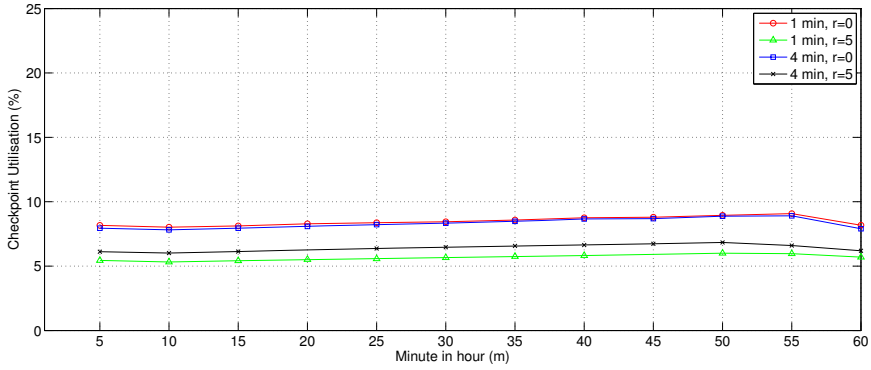
Fig. 8. Geometric policy



(a) The impact of the MinuteInHour policy on energy consumption.



(b) The impact of the MinuteInHour policy on average task overhead.

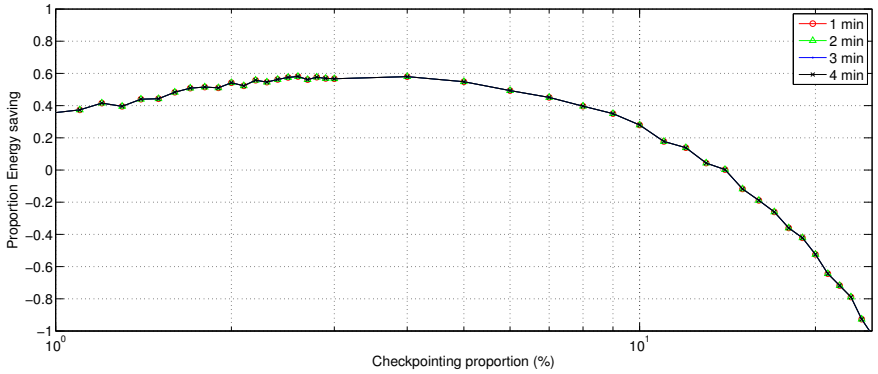


(c) The impact of the MinuteInHour policy on checkpoint utilisation.

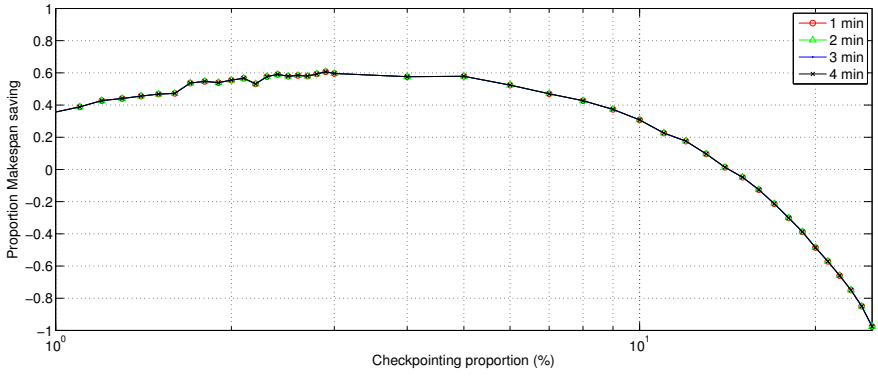
Fig. 9. MinuteInHour policy

Figures 10a, 10b and 10c show the results for the Ratio policy. This policy makes use of estimates of the time required to generate a checkpoint for a given job, and here we demonstrate the policy's ability to deliver equivalent benefits to jobs, irrespective of checkpoint generation duration. We observe the greatest benefit for our workload where checkpointing is configured to take $\sim$ 4% of execution time. Beyond this point, benefits begin to curtail and at $\sim$ 15%, the cost of checkpoint

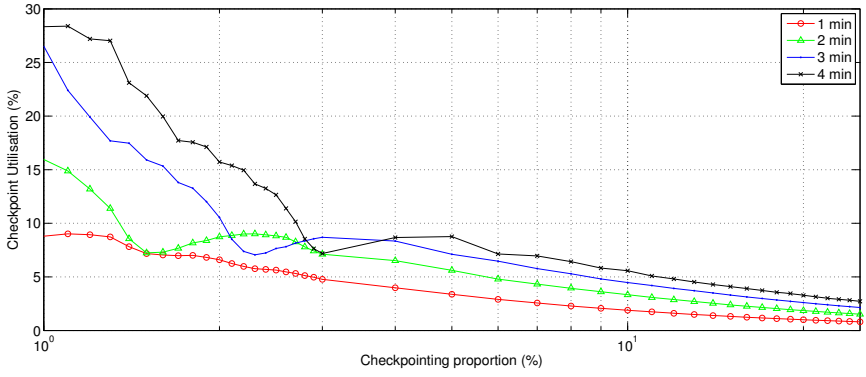
operations exceeds that of lost execution due to interruptions. When considering checkpoint utilisation under the Ratio policy, utilisation falls as the proportion of execution time spent checkpointing (and thus the number of checkpoint operations) increases.



(a) The impact of the Ratio policy on energy consumption.



(b) The impact of the Ratio policy on average task overhead.



(c) The impact of the Ratio policy on checkpoint utilisation.

Fig. 10. Ratio policy

Figures 11a, 11b and 11c show the results for our policy placing a delay on the start of checkpointing for a job. With the exception of $\mathbf{C}(60)$ for one minute

checkpoints, we observe a modest benefit to delaying the start of checkpointing during the first hour of a task's execution. Due to the relatively short execution time of the jobs comprising our workload, results begin to decrease beyond a start delay of $\sim$ 90 minutes, due to the start delay being longer than the execution time of the task. The observable drop in the checkpoint utilisation graph centred at approximately 120 mins is also an artefact of this interaction between task execution time and start delay.

In Figures 12a, 12b and 12c we show results for our Interarrival policy determining the conditions under which a scheduled checkpointing operation should proceed. Each of these results are shown for one minute checkpoint duration, and for sliding windows of length one, ten and twenty minutes. We present results for two variations of the policy, one which enacts checkpoints for a job based on the interactive user arrivals at the cluster where the job is executing, and the other based on interactive user arrivals throughout the entire system. The System-level checkpointing strategy is shown to provide greater improvements to energy consumption and overhead when compared to the Cluster-based approach, despite significantly lower checkpoint utilisation. The results for policies using a one minute sliding window are shown to be more sensitive to selection of interactive user arrival threshold (l) than those with longer window lengths. In both cases the benefits are greatest for small values of l, but we do not find user arrivals in such low quantities to be a sufficiently good predictor of task preemption for our workload.

5.2 Summary

From the results of our preliminary investigation, we note that for periodic checkpointing schemes, checkpoint generation duration is often as important as the checkpointing interval chosen. This highlights the importance of a combined approach between checkpoint scheduling policies and the efficiency of the checkpointing mechanisms themselves.

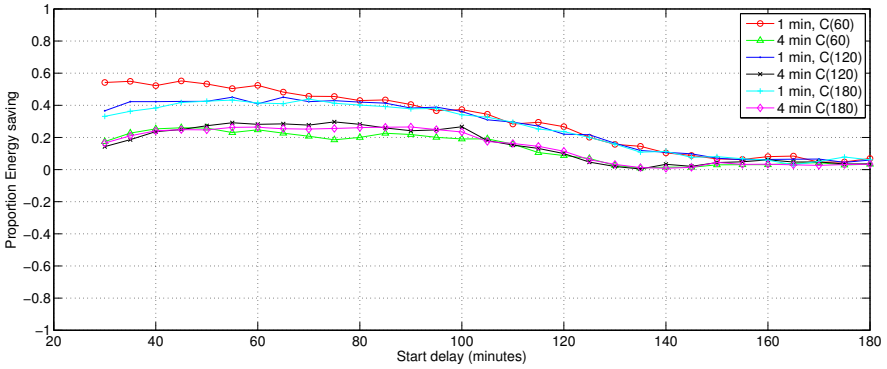
Though we find checkpointing results in significant improvements to task overheads, for many policies including periodic checkpointing, the benefits rely on correct parameterisation of policies. The exploration of approaches to adaptive checkpointing policies with the ability to adapt parameters to the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

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

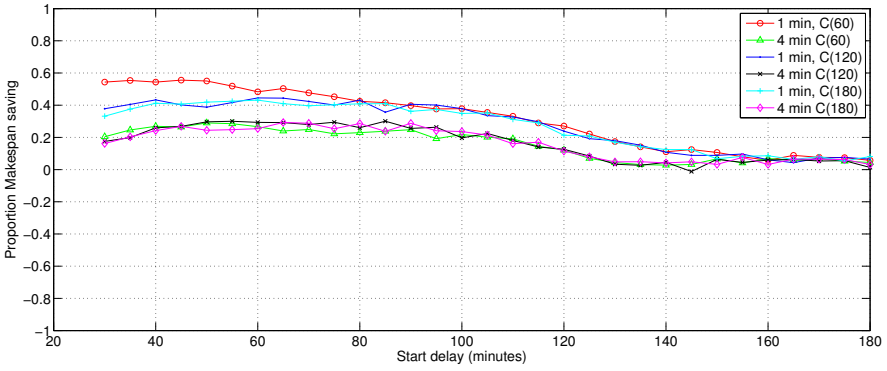
A key finding of this work relates to the effectiveness of load-based measures to govern the operation of a checkpointing scheme. While we found policies leveraging knowledge of scheduled interruptions and periods where clusters will be closed to interactive users, our threshold-based user interarrival policy was not found to offer significant benefits. In a real world system where the collection of such de-

tailed knowledge is non-trivial, simple measures such as cluster opening times and the knowledge of scheduled interruptions seem sufficient in achieving the greatest results.

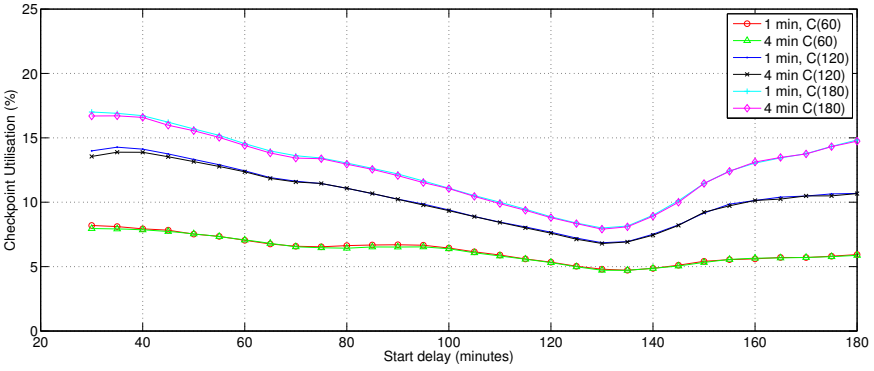
Finally, many of the policies outlined in this paper are not mutually exclusive, and we anticipate a combination of these approaches will yield best results.



(a) The impact of the Start Delay policy on energy consumption.

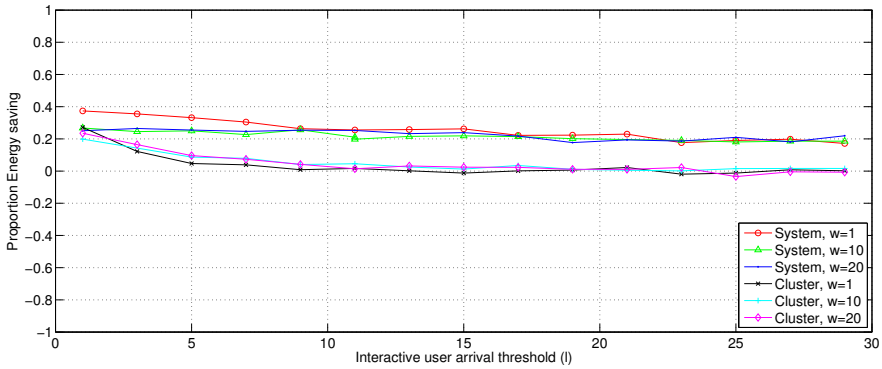


(b) The impact of the Start Delay policy on average task overhead.

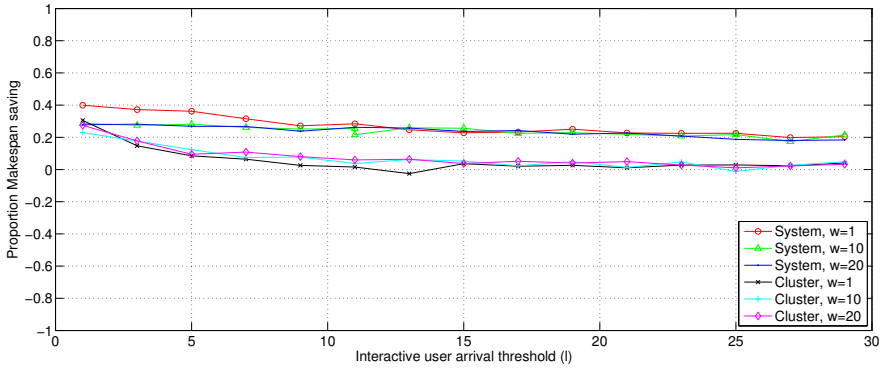


(c) The impact of the Start Delay policy on checkpoint utilisation.

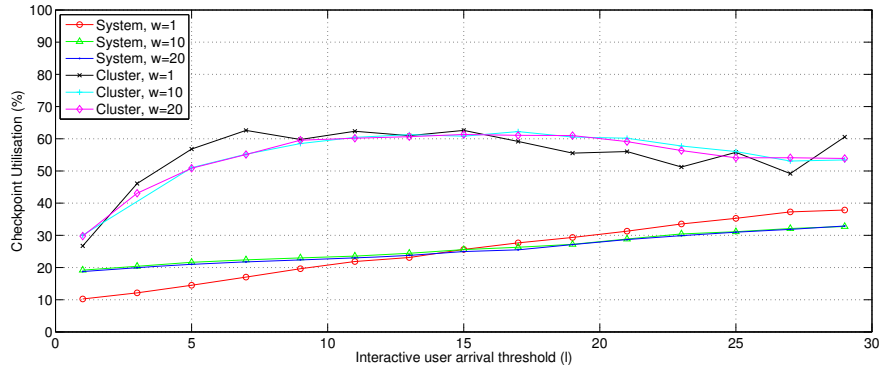
Fig. 11. Start Delay policy



(a) The impact of the Interarrival policy on energy consumption.



(b) The impact of the Interarrival policy on average task overhead.



(c) The impact of the Interarrival policy on checkpoint utilisation.

Fig. 12. Interarrival policy

6 Discussion

In this section, we outline the considerations the administrator of an HTC cluster should make when deciding whether to employ a checkpointing mechanism within their environment. Furthermore, we highlight a number of areas of research interest,

both with respect to energy-efficient checkpointing generally, and also issues specific to the application of these approaches in the context of multi-use clusters.

Operating policies: FGCS systems are typically configured to operate conservatively, with the interactive user of a machine given priority over the HTC workload running on the machine. Historically there was significant potential of interference from an HTC job, degrading performance and responsiveness for interactive users of a system. However, now in multi-core systems, and with the additional separation afforded by virtualisation technologies, the impact of HTC workloads on interactive users has been shown to be negligible [21]. Relaxing operational constrains preventing HTC jobs from running on resources with interactive users not only increases the capacity and throughput of the system, but also offers significant reduction in energy consumption. Our preliminary results demonstrate the energy and performance benefits made possible when leveraging knowledge of scheduled interruptions and user activity, highlighting the benefit of communication between cluster and HTC system administrators. Furthermore, we demonstrate the potential for checkpointing informing the management decisions made at the cluster level. For example, nightly reboots may be staggered to reduce the interference caused by many jobs checkpointing simultaneously, or reboots may be scheduled for shortly after clusters close to interactive users, increasing resource availability.

Workload: The efficacy of checkpointing is largely dependent on cluster workload. Checkpointing is most useful when the execution time of a large proportion of the workload exceeds typical resource mean time to failure (MTTF) or user inter-arrival durations, increasing the likelihood of interruption. Checkpointing in other situations is likely to have a detrimental effect on energy consumption and makespan. Furthermore, some jobs do not support checkpointing, or are unsuitable for checkpointing e.g. those with particularly large application states.

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

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

transfer/storage and instance costs will exacerbate the impact of any checkpoint overheads.

7 Conclusion

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

References

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

[2] Aupy, G., A. Benoit, R. G. Melhem, P. Renaud-Goud and Y. Robert, *Energy-aware checkpointing of divisible tasks with soft or hard deadlines*, CoRR **abs/1302.3720** (2013).

[3] Bailey Lee, C., Y. Schwartzman, J. Hardy and A. Snavely, *Are user runtime estimates inherently inaccurate?*, in: D. Feitelson, L. Rudolph and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, LNCS **3277** (2005), pp. 253–263.

[4] Bell, W. H., D. G. Cameron, L. Capozza, A. P. Millar, K. Stockinger and F. Zini, *Optorsim - a grid simulator for studying dynamic data replication strategies*, International Journal of High Performance Computing Applications (2003).

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

[6] Bouguerra, M., A. Gainaru, L. Gomez, F. Cappello, S. Matsuoka and N. Maruyama, *Improving the computing efficiency of hpc systems using a combination of proactive and preventive checkpointing*, in: *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, 2013, pp. 501–512.

[7] Bouguerra, M., D. Kondo and D. Trystram, *On the Scheduling of Checkpoints in Desktop Grids*, in: *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, CCGrid '13, 2011, pp. 305–313.

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

[9] Calheiros, R. N., R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, *Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms*, Software: Practice and Experience **41** (2011), pp. 23–50.

[10] Cappello, F., A. Geist, B. Gropp, L. Kale, B. Kramer and M. Snir, *Toward exascale resilience*, Int. J. High Perform. Comput. Appl. **23** (2009), pp. 374–388.

[11] Choi, S., M. Baik, C. Hwang, J. Gil and H. Yu, *Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment*, in: *Network Computing and Applications, 2004. (NCA 2004). Proceedings. Third IEEE International Symposium on*, NCA '04, 2004, pp. 366–371.

[12] Dhiman, G., K. Mihic and T. Rosing, *A system for online power prediction in virtualized environments using gaussian mixture models*, in: *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, IEEE, 2010, pp. 807–812.

[13] Economou, D., S. Rivoire, C. Kozyrakis and P. Ranganathan, *Full-system power analysis and modeling for server environments*, International Symposium on Computer Architecture-IEEE, 2006.

[14] El Mehdi Diouri, M., O. Gluck, L. Lefevre and F. Cappello, *Energy considerations in checkpointing and fault tolerance protocols*, in: *Dependable Systems and Networks Workshops (DSN-W), 2012 IEEE/IFIP 42nd International Conference on*, DSN-W '12, 2012, pp. 1–6.

[15] Fan, X., W.-D. Weber and L. A. Barroso, *Power provisioning for a warehouse-sized computer*, , **35**, ACM, 2007, pp. 13–23.

[16] Heath, T., B. Diniz, E. V. Carrera, W. Meira Jr and R. Bianchini, *Energy conservation in heterogeneous server clusters*, in: *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, ACM, 2005, pp. 186–195.

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

[18] Jarvis, S., N. Thomas and A. van Moorsel, *Open issues in grid performability*, International Journal of Simulation and Process Modelling (IJSPM) **5** (2004), pp. 3–12.

[19] Kliazovich, D., P. Bouvry, Y. Audzevich and S. U. Khan, *Greencloud: A packet-level simulator of energy-aware cloud computing data centers*, in: *GLOBECOM*, 2010, pp. 1–5.

[20] Legrand, A. and L. Marchal, *Scheduling distributed applications: The simgrid simulation framework*, in: *In Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid*, 2003, pp. 138–145.

[21] Li, J., A. Deshpande, J. Srinivasan and X. Ma, *Energy and performance impact of aggressive volunteer computing with multi-core computers*, in: *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*, MASCOTS '09, 2009, pp. 1–10.

[22] Lim, S.-H., B. Sharma, G. Nam, E. K. Kim and C. Das, *Mdcsim: A multi-tier data center simulation, platform*, in: *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, 2009, pp. 1–9.

[23] Litzkow, M., M. Livney and M. W. Mutka, *Condor-a hunter of idle workstations*, in: *8th International Conference on Distributed Computing Systems*, ICDCS '88, 1998, pp. 104–111.

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

[25] McGough, A., C. Gerrard, P. Haldane, D. Sharples, D. Swan, P. Robinson, S. Hamlander and S. Wheater, *Intelligent Power Management Over Large Clusters*, in: *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, 2010, pp. 88–95.

[26] McGough, A., C. Gerrard, J. Noble, P. Robinson and S. Wheater, *Analysis of Power-Saving Techniques over a Large Multi-use Cluster*, in: *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, 2011, pp. 364–371.

[27] McGough, A. S., M. Forshaw, C. Gerrard, P. Robinson and S. Wheater, *Analysis of power-saving techniques over a large multi-use cluster with variable workload*, Concurrency and Computation: Practice and Experience **25** (2013), pp. 2501–2522.

[28] McGough, A. S., P. Robinson, C. Gerrard, P. Haldane, S. Hamlander, D. Sharples, D. Swan and S. Wheater, *Intelligent power management over large clusters*, in: *International Conference on Green Computing and Communications (GreenCom2010)*, 2010.

[29] Melhem, R., D. Mosse and E. Elnozahy, *The interplay of power management and fault recovery in real-time systems*, Computers, IEEE Transactions on **53** (2004), pp. 217–231.

[30] Mills, B., R. E. Grant, K. B. Ferreira and R. Riesen, *Evaluating energy savings for checkpoint/restart*, in: *Proceedings of the 1st International Workshop on Energy Efficient Supercomputing*, E2SC '13, 2013, pp. 6:1–6:8.

[31] Oliner, A. J., L. Rudolph and R. K. Sahoo, *Cooperative checkpointing: A robust approach to large-scale systems reliability*, in: *Proceedings of the 20th Annual International Conference on Supercomputing*, ICS '06 (2006), pp. 14–23.

[32] Raman, R., M. Livny and M. Solomon, *Matchmaking: Distributed resource management for high throughput computing*, in: *High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on*, IEEE, 1998, pp. 140–146.

[33] Ranganathan, P., P. Leech, D. Irwin and J. Chase, *Ensemble-level power management for dense blade servers*, , **34**, IEEE Computer Society, 2006, pp. 66–77.

[34] Ren, X., R. Eigenmann and S. Bagchi, *Failure-aware Checkpointing in Fine-grained Cycle Sharing Systems*, in: *Proceedings of the 16th International Symposium on High Performance Distributed Computing*, HPDC '07, 2007, pp. 33–42.

[35] Rivoire, S., P. Ranganathan and C. Kozyrakis, *A comparison of high-level full-system power models.*, HotPower **8** (2008), pp. 3–3.

[36] Saito, T., K. Sato, H. Sato and S. Matsuoka, *Energy-aware I/O Optimization for Checkpoint and Restart on a NAND Flash Memory System*, in: *Proceedings of the 3rd Workshop on Fault-tolerance for HPC at Extreme Scale*, FTXS '13, 2013, pp. 41–48.

[37] Unsal, O. S., I. Koren and C. M. Krishna, *Towards energy-aware software-based fault tolerance in real-time systems*, in: *Low Power Electronics and Design, 2002. ISLPED'02. Proceedings of the 2002 International Symposium on*, 2002, pp. 124–129.

[38] *UW-Madison CS Dept. HTCondor Pool Policies* (2013).
URL http://research.cs.wisc.edu/htcondor/uwcs/policy.html

[39] Vieira, G. M. and L. E. Buzato, *Distributed checkpointing: Analysis and benchmarks*, , **6**, 2006.

[40] Yang, Y. and H. Casanova, *Umr: A multi-round algorithm for scheduling divisible workloads*, in: *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, IEEE, 2003, pp. 9–pp.

[41] Zhang, Y. and K. Chakrabarty, *Energy-aware adaptive checkpointing in embedded real-time systems*, in: *Design, Automation and Test in Europe Conference and Exhibition, 2003*, 2003, pp. 918–923.

[42] Zhou, A., S. Wang, Q. Sun, H. Zou and F. Yang, *Ftcloudsim: A simulation tool for cloud service reliability enhancement mechanisms*, in: *Proceedings Demo & Poster Track of ACM/IFIP/USENIX International Middleware Conference*, MiddlewareDPT '13 (2013), pp. 2:1–2:2.