Comparison of a cost-effective virtual Cloud cluster with an existing campus cluster

A. Stephen McGough[1a], Matthew Forshaw[b], Clive Gerrard[b], Stuart Wheater[c], Ben Allen[b], Paul Robinson[d]

[a]*Durham University,*
Durham, DH1 3LE, United Kingdom
stephen.mcgough@durham.ac.uk
[b]*Newcastle University,*
Newcastle upon Tyne, NE1 7RU, United Kingdom
{m.j.forshaw, ben.allen}@ncl.ac.uk
[c]*Arjuna Technologies Ltd, Newcastle upon Tyne, United Kingdom*
stuart.wheater@arjuna.com
[d]*Red Hat Inc., Newcastle upon Tyne, United Kingdom*
paul.robinson@redhat.com

Abstract

The Cloud provides impartial access to computer services on a pay-per-use basis, a fact that has encouraged many researchers to adopt the Cloud for the processing of large computational jobs and data storage. It has been used in the past for single research endeavours or as a mechanism for coping with excessive load on conventional computational resources (clusters). In this paper we investigate, through the use of simulation, the applicability of running an entire computer cluster on the Cloud. We investigate a number of policy decisions which can be applied to such a virtual cluster to reduce the running cost and the effect these policies have on the users of the cluster. We go further to compare the cost of running the same workload both on the Cloud and on an existing campus cluster of non-dedicated resources.

Keywords: Cloud, Economic, Simulation

[1]Work carried out whilst based at Newcastle University.

1. Introduction

Cloud Computing [**?**] provides a new model for computational processing and data storage removing many of the access barriers to large-scale computing (often referred to as High Throughput Computing (HTC)) by eliminating the need for capital expenditure on large private infrastructures. Instead users pay only for the computational power or data space they use – more than they could afford to buy though enough to meet their immediate needs from an apparently infinite (henceforth we just say infinite) pool of resources – transferring capital expenditure to operational cost. This allows the user to work in-spite of local resource availability. Large collections of resources can be provisioned in a short period of time, quicker than many organisations can offer, for a relatively small operational outlay, and at a fraction of the capital cost. This approach has been used in scenarios with significant temporal variation in requirements, alternating between periods of little (or no) activity to periods of high activity and jobs which require low data transfers, to mitigate the data transfer times and costs.

Traditionally many organisations such as universities or companies have provided HTC through a dedicated centralised cluster of computers, where capital expenditure is committed to a fixed number of computational resources and data storage. This has the advantage of economies of scale as most users of the HTC facility will not need full access to the facility at the same time. The size of such a facility is dominated by two factors: the anticipated load on the cluster and the available budget. The aim is to provision enough resources to deal with all but the exceptional load scenarios placed on the resources. The exceptional load is dealt with either by failing to achieve the required level of Quality of Service or by outsourcing work, for example to a Cloud provider [**?** **?**]. Excess jobs which cannot be handled on local resources are sent to a (public) Cloud for execution – thus allowing the owners to temporarily increase the size of their own cluster.

Here we explore an alternative use case – moving the entire cluster onto the Cloud. We investigate a number of polices which can be applied over an existing HTC management service for determining the number of Cloud instances which should make up the virtual Cloud cluster. We further investigate whether there are advantages in all HTC users within an organisation sharing resources to help reduce costs.

We evaluate our approach through the use of two metrics: the financial cost of using the Cloud (based on the number of hours consumed along with

data transfer charges) and the impact on job overheads. We define overhead as the difference between the total time a job spends within the system and the actual execution time for the job, a more formal definition for overhead is given in Section 3. The overheads include both the time to upload and download data to the Cloud along with any other delays incurred from using the Cloud. This data transfer also has implications for the cost of using the Cloud as most Cloud providers charge for data transfers.

We use a trace-driven simulation [**?**], using trace logs from the HTCondor [**?**] (formerly called Condor) desktop cluster based at Newcastle University [**? ?**], to evaluate the effectiveness of our approach. In order to evaluate our policies more thoroughly we have generated a number of synthetic trace logs based on increasing the number of users submitting work into the HTC cluster. These synthetic loads represent approximately one to five times the workload from our real logs, allowing for evaluation of our policies under greater workload. Using just the submission times for jobs to the cluster, their execution times and the data ingress/egress volumes allows us to submit jobs into the simulated Cloud cluster where jobs will either receive service immediately, if virtual computational instances (referred to here as *instances*) are idle, or enter a queue awaiting execution. A Policy can then be enacted to determine if (and when) a new Cloud instance should be started or unused instances terminated. As the main focus of this paper is a comparative evaluation of a number of policies we do not concern ourselves with how users would have changed their usage patterns on the Cloud, instead using these trace-logs for comparison only – real deployment would almost certainly alter usage patterns. We acknowledge here that the execution times of workload on the Cloud would vary in comparison with the execution times observed on our local desktop cluster. However, our aim here is to compare the different polices for optimising our use of the Cloud hence we do not take this variation into account. Further, Gillam et al. [**?**] observe over 100% variation in performance of Cloud instances advertised as being the same thus making any scaling process highly inaccurate.

An alternative approach used by many organisations is to make use of their existing computational resources for a secondary purpose, thus exploiting the idle time on these computers for HTC workload. However, as computers are used by the HTC system speculatively, computational work may need to be sacrificed in the case when the user requires his/her computer. This has the advantage that although these resources are no longer dedicated for the processing of computational workload it does allow the organisation

to make use of a large collection of computers for little (if any) capital expense. This form of desktop cluster, often referred to as a desktop grid, can therefore be seen as an alternative to using the Cloud.

We have previously shown that ∼120MWh of energy was consumed in 2010 to power the Newcastle HTCondor desktop cluster [**?**]. This being made up from ∼43MWh from good HTCondor work which completed and ∼77MWh from bad HTCondor work which didn't complete. In order to fairly compare the use of a desktop cluster with the Cloud we additionally factor in the other charges which would be required for running this service, those of staff costs, carbon emissions and dedicated server costs.

We see our key contributions from this work as being:

- An evaluation of the feasibility and cost of moving an entire HTC cluster into the Cloud based around real trace logs and trace logs generated from synthetic users.
- An evaluation of a number of policies for minimising the cost of using the Cloud for HTC workload along with the effect that this will have on the overheads observed by the user.
- A comparison of the cost implications of running large HTC workloads on a Cloud as opposed to using a non-dedicated HTC desktop cluster.

The rest of this paper is set out as follows. Section 2 discusses related research to the work we propose. In Section 3 we describe in more detail the cluster we are modelling. We present a number of policies for optimising the cost for using the Cloud in Section 4 along with the perceived benefits of these policies. The simulation environment is described in Section 5 with the simulation results being presented in Section 6 where we also compare the cost of using the Cloud to the cost, in terms of both energy and hardware, for using the campus based cluster at Newcastle when executing the same workload. Finally our conclusions are presented in Section 7.

2. Related Work

There is currently great interest in Cloud Computing [**?**]. This has led to a number of investigations into the applicability of the Cloud as a tool for aiding users in their work. A number of simulation approaches to model the benefits of Cloud computing have been performed. Deelman [**?**

] evaluated the cost of using Amazon's Elastic Compute Cloud (EC2) [**?**] and Amazon's Simple Storage Service (S3) [**?**] to service the requirements of a single scientific application. Here we seek to service the requirements of multiple users and multiple applications.

De Assuncao [**?**] proposed the use of Cloud computing to extend existing clusters to deal with exceptional load. This work was further extended by Mattess [**?**] by proposing the use of Amazon Spot Instances, supply-and-demand driven pricing of instances, to further reduce the cost of Cloud Bursting. Our approach differs to these in the sense that we seek to deploy our entire cluster to the Cloud. The approach of using Spot Instances, however, could easily be included in our approach and would allow for the same cost reduction as proposed by Mattess. Van den Bossche et al. [**?**] uses Binary Integer Programming to select which workflows should be burst to the Cloud. This approach is computationally expensive to determine the optimal approach and does not address the issue of when to terminate instances. To address the computational expense Van de Bossche et al. extend their work by developing scheduling algorithms for bag-of-tasks applications in hybrid cloud environments [**?**]. It may be naively assumed that our approach here is no more than the degenerative case with no local resources. However, these papers discuss when Cloud resources should be brought in, whilst our work discusses how to best manage the starting/termination of instances. These two approaches can therefore be seen as complementary.

Marshall [**?**] proposes policies for how to extend the number of Cloud instances to use along with simulations of a small number of short running synthetic jobs to evaluate overhead times. Here we use a full trace log containing over half a million real jobs, along with synthetic traces derived from this real log, and evaluate for both overhead and Cloud cost including the effect on overheads of data transfer and the cost for the data transfer.

Palankar [**?**] showed the criticality of data locality in the Cloud. In this work we take into account the effects of uploading and downloading data from the Cloud without storing data there. This gives us an upper limit on the data transfer cost. We see that moving our data to the Cloud and keeping it there will help to reduce the data locality problem and associated data transfer costs.

Additional functionality such as Amazon CloudWatch [**?**] allow instances to be brought up and down dependant on the characteristics of existing instances that are being used. The approaches we propose could be built into such a system.

2.1. Cost of Clouds

Lampe et al. [**?**] propose an exact mathematical model for computing the most optimal placement of work onto the Cloud for cost minimisation based on Binary Integer Programming [**?**] though conclude that this approach is non-tractable when the number of jobs is larger than 20. They go further to propose a heuristic for approximating this minimisation based on the knapsack problem [**?**]. However, in both cases they only consider the cost of using the Cloud and not the impact this will have on the overheads.

Byun et al. [**?**] propose an architecture for computing workflows on the Cloud. They compute the minimum number of Cloud instances required to complete the workflow within a pre-determined time interval using an approach of Balanced Time Scheduling [**?**]. This allocates a fixed number of Cloud instances for the duration of the workflow. This differs from our approach where the number of Cloud instances changes dynamically throughout and we do not limit ourselves to the execution of a single workflow.

Kondo et al. [**?**] evaluate the cost benefits for using a volunteer computing environment such as BOINC [**?**] over running the same workload in the Cloud. As they are not responsible for the costs of computation on the volunteer computers their approach shows that the use of the Cloud quickly becomes more expensive. However, they do conclude that the use of the Cloud for providing the central resources would be appropriate. Our work is similar in that we are not responsible for the cost of local resources which are provisioned primarily for other purposes, though our work differs in that our workload is much more heterogeneous.

Koch et al. [**?**] evaluated three approaches to allocating resources within an educational environment in order to minimise cost whilst maintaining Quality of Service (QoS), namely: resource pre-allocation based on peak demands; reactive resource allocation based on current demand; and proactive resource allocation that considers workload characteristics and parameters of the domain. They concluded that the workload aware proactive approach is the best for meeting QoS for a minimum cost. This is similar to our work though we do not assume knowledge of the workload.

3. Cloud Cluster Model

We adopt the Cloud model used by many providers (e.g. Amazon EC2 [**?**], Microsoft Azure [**?**] and RackSpace [**?**]) allowing users to deploy virtual

machine images onto servers owned by the provider – referred to as Infrastructure as a Service (IaaS) [**?**]. Figure 1 illustrates our basic architecture in which users submit job descriptions to the cluster based job management system such as HTCondor [**?**], PBS [**?**] or (Sun) Grid Engine [**?**] along with a number of files required to perform the job and details of files to be staged back after completion of the job. A Job Management Service is used to allocate these jobs to a dynamic pool of instances within the Cloud. Like many organisations Newcastle University does not provide a shared file system for its HTC users. We thus assume that the Cloud alternative works in the same manner. We acknowledge that storing data on the Cloud could help alleviate transfer times and costs. However, as we do not possess information as to the contents of files within our logs we cannot identify which files would be appropriate to keep in the Cloud. We would expect that keeping files in the Cloud to be beneficial. Although illustrated here as a single entity the job management service may consist of multiple entities allowing balancing of job submissions and data transfers to and from the Cloud.

We seek here not to replace the existing HTC management system but rather to augment it with the ability to add and remove computers to our virtual cluster in the Cloud. Policy is able to decide when extra instances need to be provisioned and when instances can be removed. These policy decisions are based on the current state of the cluster and the perceived future state of the cluster. Additional software is required to increase the number of Cloud instances, when required, and terminate these when no longer needed – the policy component in Figure 1. Instances within the Cloud cluster can be in one of three states with interactions illustrated in Figure 2:

- **Unallocated**: these are the potential Cloud instances not currently under contract – (effectively) an infinite set. The Job Management Service can 'hire' such an instance to run a job, placing it in the Active state.

- **Active**: the instance is 'hired' by the cluster and is currently servicing a job. On job completion the instance will enter the idle state.

- **Idle**: the instance is 'hired' by the cluster but not currently servicing a job. The instance will become active if the cluster allocates a job before the end of its billing period otherwise it will be released into the unallocated state.

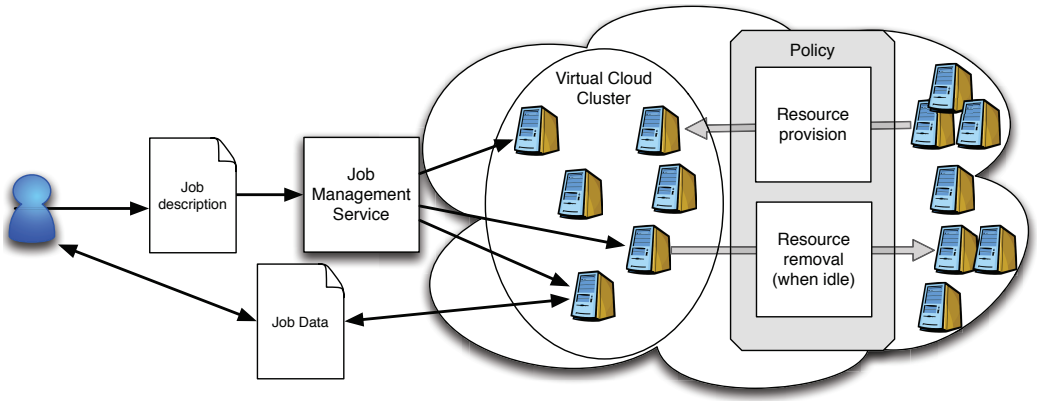

Figure 1: The Cloud cluster architecture

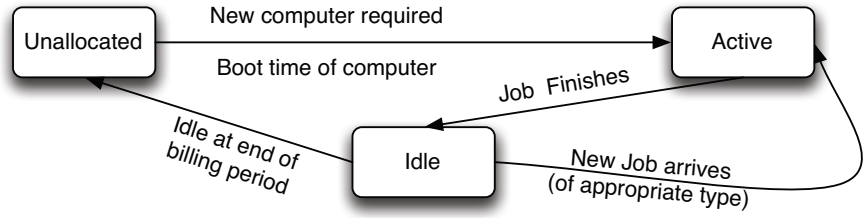

Figure 2: The state diagram for Instances

We assume here that instances only run a single job at a time. As an instance incurs the same charge irrespective of when it is terminated within a billing period it is always kept 'hired' until the end of this period – increasing the chance of there being an idle instance available when a job arrives. Instances can either be provisioned for all users within a cluster or only a specific user. If provisioned for all users then the instance will accept new jobs from any user, whilst instances provisioned for a given user will only accept new jobs from that specific user. Although accepting jobs from only a specific user will in general reduce the utilisation of Cloud instances this may be desirable for security reasons.

Jobs are first matched against idle instances capable of accepting jobs from that user, receiving continuous service from the active instance until completion when the instance will become idle. Jobs arriving to find no 'idle' instances capable of servicing them will cause a new instance to be provisioned, requiring time for the operating system and middleware to start, before running the job.

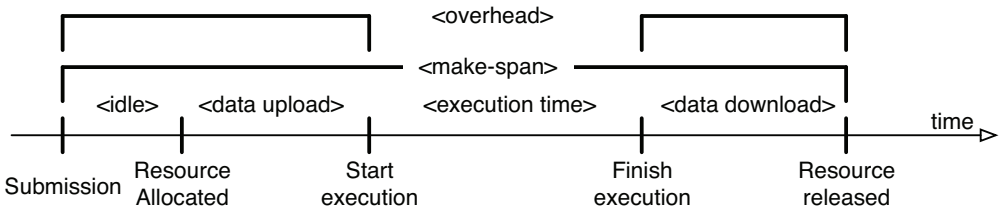

Figure 3: Timeline for a job

Figure 3 illustrates the timeline for a job executing within the Cloud cluster. A job will first encounter an idle period before being allocated to a resource. This idle period could be due to policy decisions within the cluster or time waiting for a resource to become available for use. Once a resource is allocated then the data for this job will be uploaded to the instance before the job itself executes. On completion of execution data is staged back before the resource is placed into the idle state either to be allocated to a new job or released from the cluster at the end of its current charging interval. Figure 3 indicates the two time intervals which make up the overheads for a job along with the time interval of the make-span. It is assumed that all data will be staged from and to a computer within the home campus through a high-speed network link with appropriate capacity for storing returned results.

3.1. Cloud Cost Model

In terms of the work that we seek to perform on the Cloud charges will apply for data upload and download from the Cloud along with charges for the amount of time that Cloud computers are actively 'hired'. As our current HTC desktop cluster does not support data storage we omit this from our costing model, though note that using Cloud storage would have a cost implication from the storage charges and a cost benefit from removing the need to upload and download all data. The general equation for our Cloud charging model is given in equation 1:

$$cost = \sum_{m=1}^{M}(\sum_{i=1}^{I} h_{m,i}r_i + \sum_{j=1}^{J} u_{m,j}s_j + \sum_{k=1}^{K} d_{m,k}e_k) \quad (1)$$

where M is the number of billing periods over which we are modelling – for most Cloud providers a billing period is one month in length, I is the number of instance types – here we flatten out different charging rates for the same instance type into separate instance charging types to allow us to enumerate

them, J and K are the number of cost categories for data ingress and egress respectively. We define $h_{m,i}$ as the number of computation instances rented (typically measured in hours) from the provider during month m, for instance charging type i with a charge of r_i per unit. Similarly for data ingress $u_{m,j}$ is the number of data transfer units (normally measured in GB) in billing period m in charging category j charge at s_j. Conversely $d_{m,k}$ represents the number of egress data transfer units during billing period m in charging category k charged at rate e_k. Although this leads to a large number of potential charging categories most Cloud providers do not use them all. For example most Cloud providers do not have separate charging models for the same instance type and do not charge for data ingress. In order to match the resources used within our desktop cluster we have selected a Cloud instance type which closely resembles the performance of one of our normal desktop computers – this is approximately equivalent to an EC2 c1.medium instance which is currently charged at $0.145 per hour. We acknowledge that memory and CPU characteristics could be used to select more appropriate resources within the Cloud. However, our trace logs do not include enough information to allow us to determine this.

Billing is typically by the hour with partly used hours incurring a full hour charge. The start of a billing period varies between providers. Some, including Amazon and Azure [**? ? ?**], have charged from the start of the wall-clock hour in which the instance was started – billing from 7pm for an instance started at 7:58pm – whilst others charge from the time the instance was started. For clarity we refer to the former case as *wall-clock charging* and the latter as *exact charging*. Although the user of a Cloud cannot select which form of charging they will receive, except for choosing a different provider, we attempt to show here the impact of such charging polices. It should be noted that although other billing intervals exist our results are not invalidated by the use of shorter (or longer) periods, they merely alter the severity of the impacts that we seek to mitigate.

3.2. Desktop cluster cost model

We have extended our desktop cluster based simulation for HTCondor [**?**] to take account of the data transfer times along with a measurement of the proportion of time that each cluster within the university is used by HTCondor. The results for running the simulation for our current HTCondor setup at Newcastle are shown in Figure 4. Each of the different clusters is represented as a separate bar (along the X-axis) with each bar representing the

proportion of time that computers within the cluster are used for HTCondor, interactive users or in an idle/powered down state. From these scaled proportions, along with the resource prices (see Table 1) which depicts the average prices paid for hardware over the last four years, we can compute an estimate for the hardware cost that HTCondor *could* be accountable for. This allows us to provide a realistic cost for how much the University could consider when evaluating the cost of using HTCondor rather than just factoring in the raw cost of the electricity. We base our cost model around that proposed in [**?**].

We estimate the cost of providing technical support for the HTCondor desktop cluster to be less than half a day of effort per week – ∼$6,750 per annum. The capital expense of a server to manage the cluster would have cost $5,142, however, as this would be needed for both the Cloud cluster and the desktop cluster we have ignored it in our calculations. Charges incurred for carbon emission taxes are £12(∼$17.95) per metric ton with ∼0.541kg of CO_2 being generated per KWh. We do not factor in here space charges for the computers, as the computer space is provided for student use, nor do we factor in repair costs as the computers will be repaired as part of the general cluster room management. Network charges are based on the proportion of time computers within a particular cluster were used for HTCondor. Thus our annual charge for the desktop cluster will be:

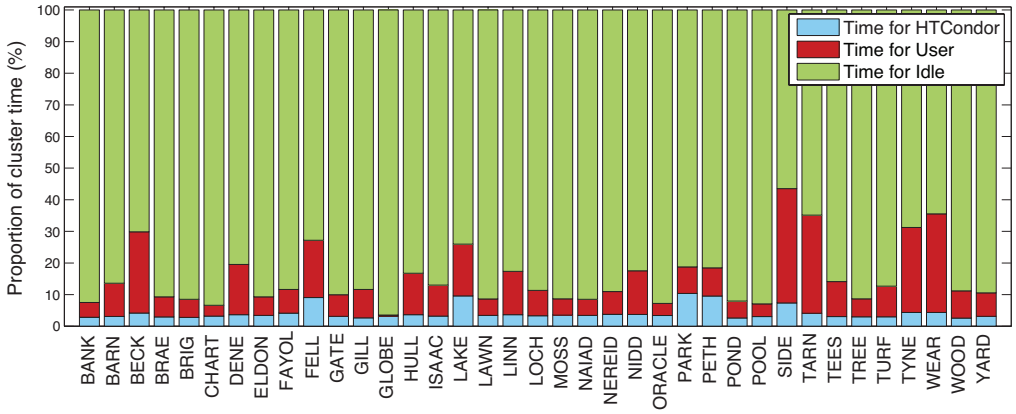

Figure 4: Proportion of cluster time used by interactive users and HTCondor

Table 1: Scaled hardware costings for clusters

Hardware type	**Average price**	**Years of service**	**Cost per year**
Normal computer	\$785	4	\$196.25
High spec computer	\$1,100	4	\$275.00
Switch (24 port)	\$1,570	10	\$157.00

$$cost = \$6,750 + \sum_{c=1}^{C}(hardware_c + carbon_c + energy_c) \tag{2}$$

where C is the number of clusters in the university, and hardware cost ($hardware_c$), CO_2 tax ($carbon_c$) and energy cost ($energy_c$) are defined as follows:

$$hardware_c = \rho_c(p_c n_c + \lceil \frac{n_c}{24} \rceil s) \tag{3}$$

where ρ_c is the proportion of the cluster which was used by HTCondor during the year, p_c is the cost per computer divided by the number of years of service, n_c the number of computers within the cluster and s the cost of a single 24 port switch. It should be noted that for ease of management the University use the same 24 port switches across the campus, further savings would be possible by varying the switch type and port count for each cluster.

$$carbon_c = \frac{TPt_c e_c}{1000} \tag{4}$$

where T is the CO_2 tax rate (\$17.95 per metric ton in 2010), P the mass of CO_2 produced per KWh (0.541 Kg), t_c the total time used by HTCondor in the cluster over the whole year and e_c the energy consumption rate for the computers whilst active.

$$energy_c = t_c e_c \epsilon \tag{5}$$

where ϵ is the energy cost (per KWh) and t_c, e_c are as defined above. It should be noted here that the cost for the operating system is part of the price of each computer and that no other charges are incurred for the use of

software as all work performed either used free software or software written by the users.

4. Policy

In this section we discuss a number of policies which can be applied to a Cloud based cluster aimed at reducing the number of hours consumed by the cluster but still allowing it to successfully complete all jobs. In each case we indicate how the policy could be realised and how we would expect the Cloud cluster to be affected.

P1: Limiting the maximum number of Cloud instances: Although the Cloud offers an (apparently) infinite availability of instances each provider has a threshold over which prior approval is required for more resources – EC2 is initially restricted to 20 instances per region, giving an overall limit of 200 instances. However, this limit can easily be increased through a simple email to Amazon. Limiting instances helps prevent starting an excessive number of instances when users submit large numbers of short jobs, leading to low utilisation and high cost.

Jobs arriving to find no instances in the 'idle' state can either cause a new instance to be started, provided that the instance limit has not been reached, or be placed into a queue of pending jobs. Pending jobs are serviced in a FCFS manner as instances become 'idle'. This will reduce the number of hours consumed by the cluster at the expense of increased average overhead.

P2: Merging of different users' jobs: Each Cloud instance is capable of running a single job at a time. We can either restrict an instance to only accept jobs from one particular user – the user that the instance was started for – or allow it to accept jobs from any user. Restricting jobs to a single user could provide a greater level of security for the running jobs. Alternatively, allowing multiple users to share each instance allows us to determine whether there is an advantage in bringing the workload of multiple users together as opposed to allowing each user to use the Cloud independently. Allowing users to share Cloud instances could help reduce costs as fewer instances will be required and reduce overheads as jobs are more likely to discover usable idle instances. As the current cluster shares resources we are not reducing the available security.

This policy can be implemented by having one central pool of Cloud instances with jobs being allocated to any 'idle' instance. This does, however,

have the complexity of how to sub-charge for these 'shared' instances. Equation 6 can be used to evaluate sub-charges once the instance has terminated:

$$Cost_i = up\frac{\sum_{j=1}^{N_i} e_{i,j}}{\sum_{k=1}^{M}(\sum_{j=1}^{N_k} e_{k,j})} \tag{6}$$

where u is the number of time units that the instance was active, p is the unit price per hour, N_i is the number of jobs from user i, M is the number of users and $e_{i,j}$ is the execution time for the j'th job from user i. Thus each user's cost is based on the proportion of the overall time the user was active on the instance relative to all other users on this instance.

P3: Instance keep-alive: It has been shown that the time for an instance to initialise is around two minutes and can be as high as 3.25 minutes [**?**]. Further, before an instance can start accepting jobs it must communicate its presence to the HTC system which can take a further couple of minutes. This adds extra time to the overheads for jobs arriving when no instances are idle. This policy allows an idle instance at the end of a billing period to remain 'hired' for the next period with probability $f(p)$, thus allowing it to serve new incoming jobs more quickly. It is difficult to predict a priori whether a job will arrive within a given period, we therefore identify three alternative policies for deciding if an instance should remain alive and define $f(p)$ for each:

- **Fixed:** instances will be kept alive with probability $f(p) = p$.
- **Idle:** instances will be kept alive with a probability proportional to the number of currently idle instances: $f(p) = (1 - \frac{i}{t})p$ where i is the number of Cloud instances which are idle at the decision time and t is the total number of Cloud instances at this time.
- **Load:** instances will be kept alive with a probability proportional to the current load on the system: $f(p) = \frac{\int_{t-T}^{t} u_i \ di}{\int_{t-T}^{t} a_i \ di}p$ where t is the decision time, T is the interval over which we are evaluating the load, u_i the number of active Cloud instances at time i and a_i the total number of Cloud instances we have hired at time i.

This would have a benefit if jobs were expected to arrive within the extended time-frame. To prevent a half-life decay an instance which is 'idle' for a full charging interval will always terminate. This policy may have more

impact on the overheads than on the cost saving, as an arriving job is more likely to find an 'idle' instance. The cost may go up due to instances running when no jobs are present.

P4: Delaying the start of instances: This policy, like P1, aims to reduce the impact of short running jobs. Arriving jobs which cannot be allocated to an 'idle' instance are queued. If the job fails to obtain an instance within t minutes then a new instance will be created. This helps the overall cost for using the Cloud by reducing the chance of instances being brought up for short-running jobs. The average overhead will go up due to the extra waiting time.

P5: Removing the delay on starting an instance: Policy P4 can be slow to react when large numbers of jobs are submitted. This throttling can be removed while the queue size exceeds a given proportion (r) of the maximum instance count. Although this is expected to increase the cost of using the Cloud it should reduce the average overhead.

P6: Waiting for the start of the next hour: Where a Cloud provider adopts a wall-clock charging model it may not be economical to start an instance just before the end of an hour. Jobs arriving within b minutes of the end of an hour are delayed until the start of the next hour. Although this will increase the overheads of the job it should decrease the cost.

5. Simulation Environment

Our simulations use the trace logs from the HTCondor high-throughput cluster at Newcastle University [**?** **?**]. The university HTC service comprised 1,359 student accessible desktop computers, running Microsoft Windows XP, located in 35 cluster rooms spread around campus. Computers were replaced on a four year rolling cycle a whole cluster at a time.

Figure 5 depicts the profile for the 409,479 successful jobs executed in 2010 by 21 unique users requiring ∼487,068 hours to execute. Jobs which were terminated before completing by the submitting user have not been included in this simulation due to their lack of execution time. We also plot the job submission profile of synthetic workload S1 for comparison.

In order to evaluate our polices over a wider range of workloads we have used the synthetic logs generated as part of [**?**]. Here we use synthetic trace logs derived from individual users each of whom are assumed to be generating job submissions in a bursty manner – intervals of no job submissions interleaved with intervals of job submissions. Table 2 illustrates the

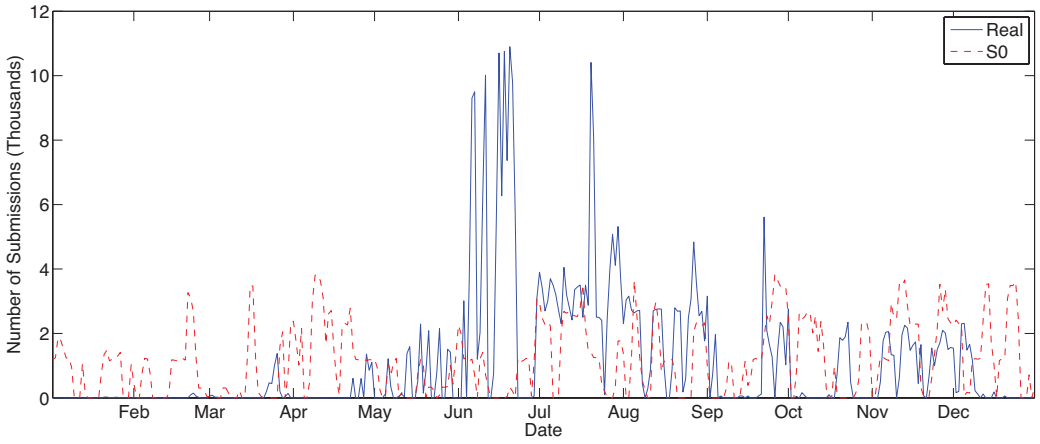

Figure 5: Profile of job submissions

number of jobs and the workload generated from each of the synthetic trace logs along with the real trace log. It should be noted here that the synthetic workloads are roughly indicative of how a system with increased load may perform, but we recognise that they are not completely representative of how real logs may look.

In order to compute the time requirements for data transfer, bandwidth tests were conducted between computers on the Amazon EC2 (US East Northern Virginia Region) cluster and a server within the university. The *iperf* bandwidth testing software [**?**] was used for this purpose with the results illustrated in Figure 6. This figure illustrates capturing the network bandwidth every half hour between Monday May 20^{th} 2013 and Tuesday May 28^{th} 2013 based on the GMT time zone. There is a clear day and night pattern to this data, although there are a number of outlying points. Bandwidth seems to be greatest during the early hours (GMT) with the up-

Table 2: Statistics on the real and synthetic trace logs

Log	Real	S1	S2	S3	S4	S5
Total Jobs	409,479	508,883	909,929	1,405,463	1,742,130	2,212,209
Workload	107,699	66,469	114,872	184,992	232,472	292,770

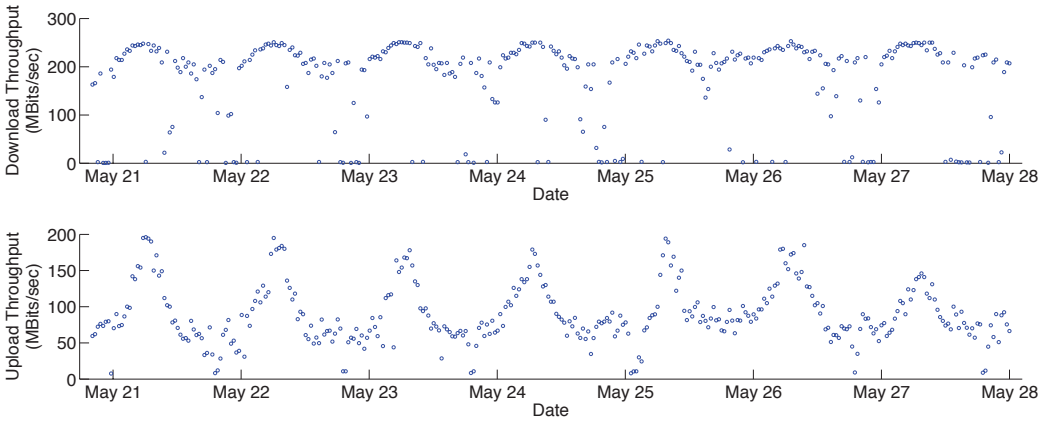

Figure 6: Upload and download speeds for the Cloud

load speeds showing the greatest variation. A full analysis and modelling of this variation in bandwidth is beyond the scope of this paper and we simply use the average bandwidth values from our test period for our simulations, those being an upload speed of 90.08MBits/s (11,811bytes/ms) and download speed of 174.88MBits/s (22,925bytes/ms). It should be noted that the largest data transfer observed in the data set was 903MB with our transfer experiments running for five minutes reaching up to 9.4GB of data transfer. It should also be noted that these are maximum bandwidth potentials for the connections; real use is likely to be less, thus these give a lower estimate on data transfer times.

Similar experiments were conducted to determine the bandwidth for the local cluster. These are not reproduced here as they indicated that the bandwidth averaged 94.75MBits/s (12,419 bytes/ms). This value was consistent over tests conducted from 1Kbyte up to and including 1Gbyte and in both directions. Although the download speed from Amazon exceeded this value this is consistent with our network topology. Within the university cluster room computers are connected to switches at 100MBits/s. These switches are then connected back to the main university machine room at 1GBits/s. However, our link to the outside world is a 20GBits/s connection allowing potential transfer between ourselves and Cloud providers at greater rates than to cluster room computers.

6. Simulations and Results

6.1. Cloud optimisation

We evaluate our policies in order to assess an optimal set of policies for our Cloud cluster. These evaluations could be performed on different cluster data and we believe that the conclusions from this work will be applicable to other similar clusters. To help exemplify the generality of our policies we demonstrate them against both our real data set and five synthetic data sets. As we are combining here the cost for both using resources on the Cloud and the data transfer costs we need to convert these into monetary values. For the purpose of this paper we are adopting the pricing policy from Amazon as of 30th June 2013. We assume here all jobs could be run on a single Linux based instance – for Amazon we choose c1.medium instances, with the current cost per instance hour of $0.145. Data ingress is free for Amazon (and most Cloud providers) with the first 1GB of egress also being free. We use the costs for Amazon data transfers: first 1GB is free, next 10TB is charged at $0.12 per GB, the next 40TB is charged at $0.09 per GB and data egress in excess of this level is charged at $0.07 per GB.

6.1.1. Baseline Results

Table 3 shows the results under the assumption of infinite instance availability. However, to minimise the cost in using the Cloud idle instances are terminated at the end of their charging period. Exact charging gives a significant decrease in hours consumed over wall-clock charging. This equates to ∼30,000 hours for the real log, a difference of ∼12%. This difference remains

Table 3: Baseline results for an infinite size Cloud Cluster

Log	**Hours Consumed**		**Average**	**Cost**	
	Exact	**Wall Clock**	**Overhead**	**Exact**	**Wall Clock**
R	209457	238956	14.84s	$43512.51	$47789.86
S1	100792	114403	14.59s	$29946.03	$31919.63
S2	178387	203603	16.27s	$54646.92	$58303.24
S3	281321	319877	13.61s	$76506.08	$82096.69
S4	353456	402049	14.58s	$97967.88	$105013.87
S5	444837	506192	13.98s	$120921.13	$129817.52

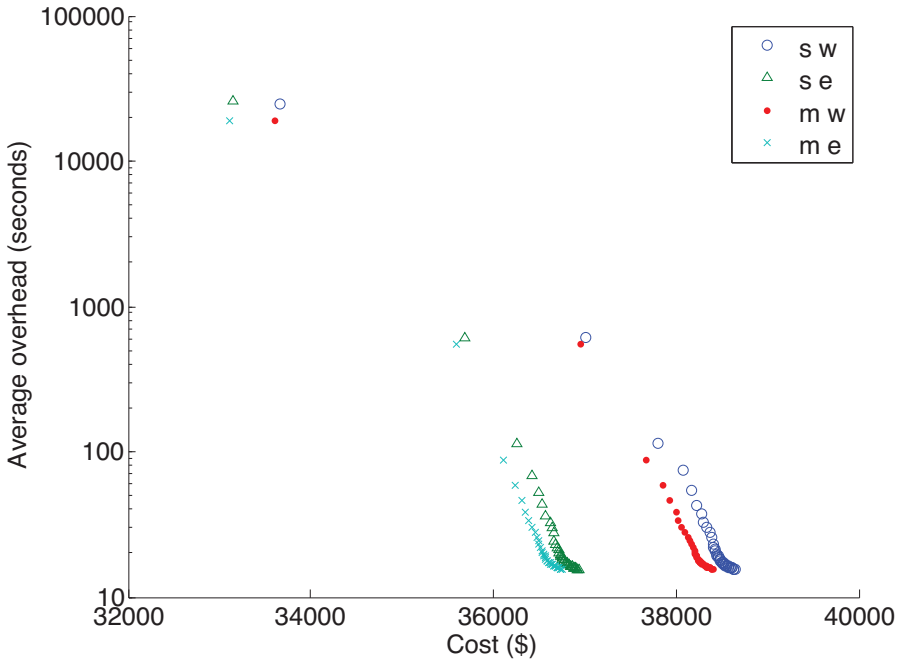

Figure 7: Effect of varying the maximum instance count on Cloud cost and Overhead

consistent across all synthetic data sets. The overhead is identical for both wall clock and exact charging in these cases as boot time is zero with the overhead just representing the data transfer times.

The cost for using the Cloud comprises of a cost for using the instances and a cost for data transfer. The data transfer cost is around $16,534.47 for the real data set, though this figure varies slightly due to policy decisions. Changes which cause the transfer times to move between months will vary the volume of data transferred in each of the months.

The following key letters are used to indicate the Cloud pricing model and user merging policy (P2) in the following graphs: **w** - wall-clock charging, **e** - exact charging, **m** - jobs can run on any instance, **s** - jobs can only be run on instances allocated to a particular user.

6.1.2. Limiting the maximum of Cloud instances and merging users jobs

Figure 7 illustrates the effect of policy P1 for the real workload. In this graph we plot cost against overhead. The four pricing models are represented using different colours and symbol shapes as follows: blue circles represent separate users with wall-clock charging, green triangles represent separate

users with exact charging, red dots represent merged users with wall-clock charging and cyan crosses represent merged users with exact charging. The cheapest (and greatest overhead) points on each set represent a maximum instance limit of 50, whilst the most expensive (and lowest overhead) represents the maximum instance limit of 2000, with all other instance limits being in increasing steps of 50 between these two points. This suggests in order to minimise the overhead the limit on maximum instances should be as high as possible, whilst if the intention is to minimise cost then the lowest value for maximum instance count should be selected. However, it is not possible to minimise both cost and overhead at the same time. A user of the Cloud will need to select their own preference for overhead against cost.

Exact charging remains more cost effective than wall-clock charging and gives a larger financial benefit than allowing different users to access the same cloud instances (P2). Although the user of a Cloud service cannot select between wall-clock and exact charging this shows that it has the potential for a significant increase in revenue for Cloud providers. This equates to around 10% increase in overall cost between exact and wall-clock charging over the entire set of real jobs. Whilst the difference between allowing only single users or multiple users to access the same instance only yields approximately a 1.2% reduction in cost. The difference between overheads for wall-clock and exact charging is negligible. For small values of maximum instances there is a significant improvement in using merging of users ($\sim$10% for a limit of 100 instances). However, this becomes less significant as the maximum number of instances increases. This is caused by the fact at low maximum instance counts a job may arrive to find the maximum number of instances active, though none able to accept it due to being dedicated to a different user. Whilst for merging of users jobs any free instance can accept the job. However, when the maximum instance count becomes large then a job arriving under a non-merging policy can normally just start up a new dedicated instance.

Figure 8 shows the rate of cost-overhead benefit B_c when increasing the maximum number of instances (c):

$$B_c = \begin{cases} -(\frac{C_c - C_{c-1}}{O_c - O_{c-1}}) & \text{if } O_c \neq O_{c-1} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where O_c is the overhead for a maximum instance count of c and C_c is the cost for the same instance count.

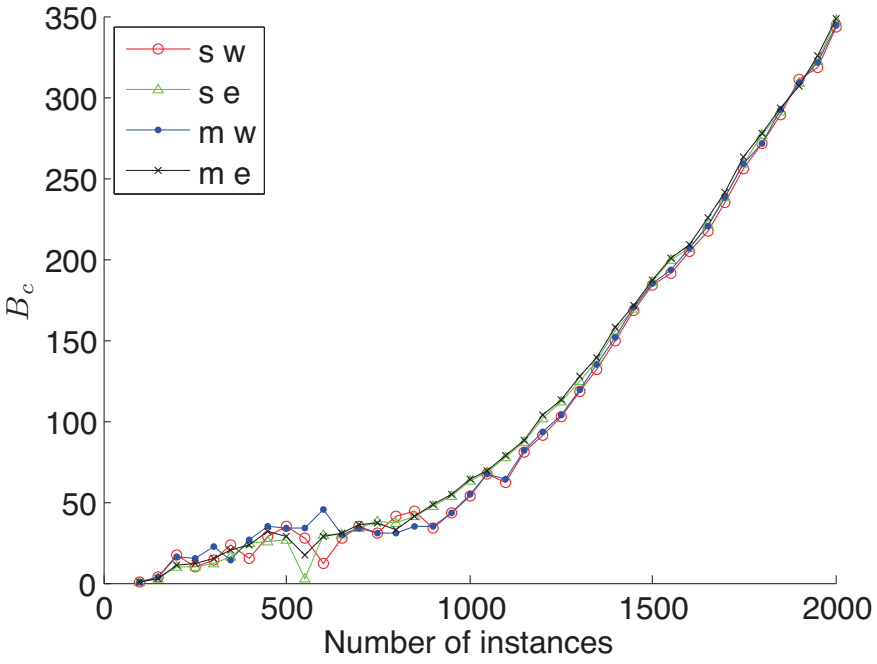

Figure 8: Benefit of increasing the maximum instance count

For small maximum instance limits c this leads to relatively low values of B_c indicating that the rate of reduction in overhead is large for a small rate of increase in cost. However, when c exceeds ∼900 instances B_c starts to increase much quicker indicating that to achieve an equivalent decrease in overhead requires a much larger increase in cost. This is a consequence of the fact that c represents the maximum number of Cloud instances which can be rented rather than an actual number rented. As c increases there will be fewer points within the simulation where this number of Cloud instances will be required with far fewer instances required for most of the time. Thus to handle these peaks many more Cloud instances are required. If the glut is short then the extra instances will not be well utilised leading to high cost for low benefit.

For the rest of the results here we fix the maximum number of Cloud instances at 500 as this gives a representative balance between overhead and cost.

The impact of merging jobs by different users (P2) on cost is shown in Figure 9, results for overhead are omitted as they show no perceivable impact.

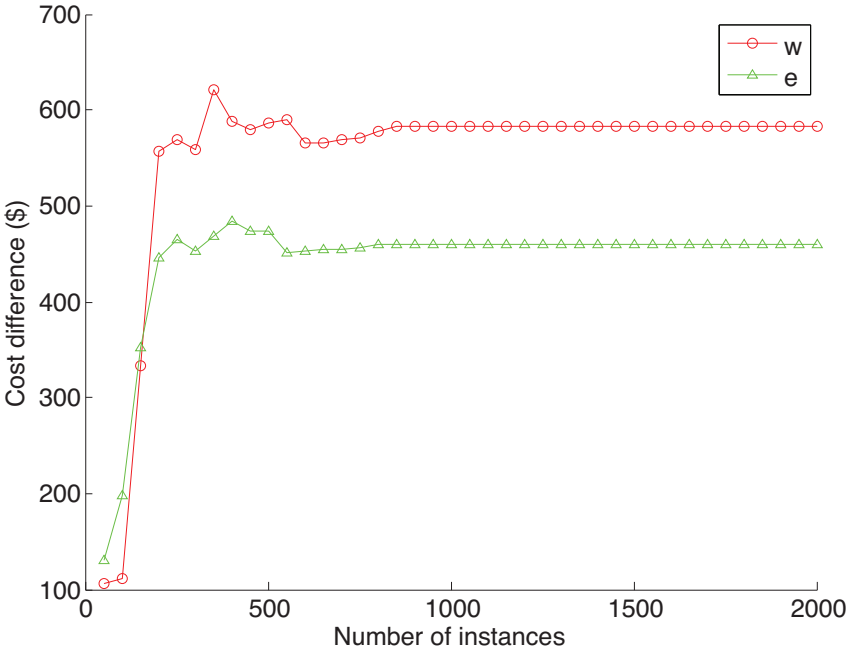

Figure 9: Cost difference between merging and not merging workloads

For very small maximum instance counts (50) there is no perceivable benefit to merging users. This is most likely due to the fact that all instances are active and merging cannot find idle instances that separate users cannot access. However, this quickly increases and stabilises at around $450 for exact charging and $575 for wall-clock charging. Both lines becoming flat after ∼900 instances suggests that all possibility of exploiting the effects of merging have been exhausted by this point. We would expect that as the workload increases so too will the gap between the costs for merged and unmerged due to the larger number of hours of Cloud use. This is borne out for the real and synthetic logs with the percentage increase of cost for separate user instances to merged user instances (at a maximum of 500 instances) – Real: 1.15%, S1: 2.78%, S2: 5.11%, S3: 7.80%, S4: 8.92% and S5: 9.93%. This indicates that the opportunity to exploit the merging of users' jobs increases as the workload increases.

6.1.3. Keeping Cloud Instances Alive

In Figure 10 we investigate the effect of start-up time for instances and whether it is beneficial to keep instances 'idle' in the absence of jobs (P3). We

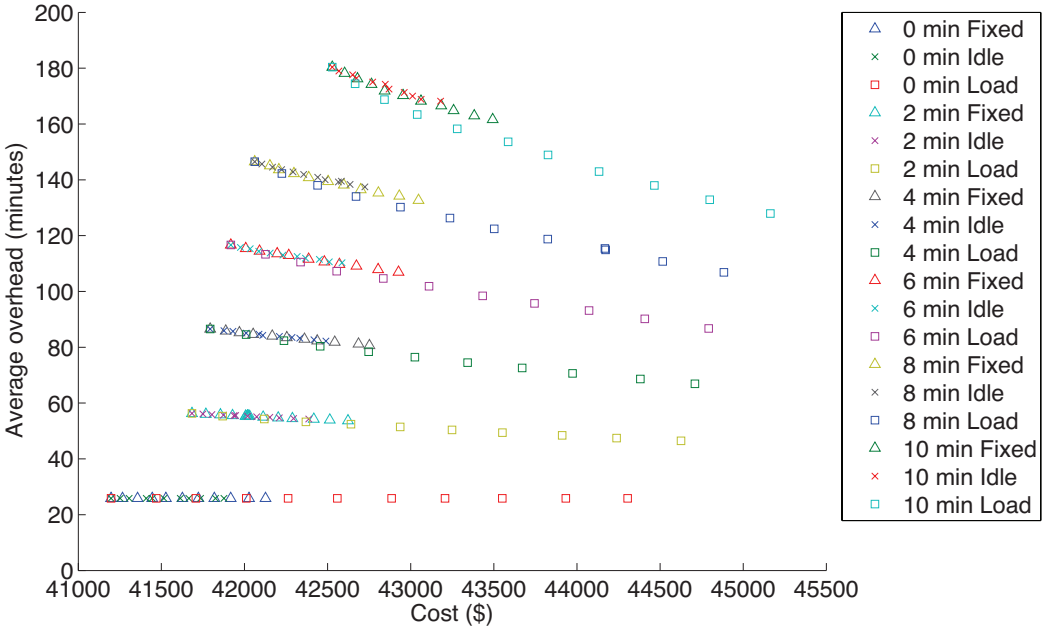

Figure 10: Varying the boot time and chance of keep-alive on average overhead and cost

again produce a scatter plot of cost against overheads. Assigning different symbols to each keep-alive policy: triangles for the fixed policy, crosses for the idle policy and squares for the load policy. For each combination of keep-alive policy and boot time the value of p ranges from zero at the furthest left symbol, progressing in steps of 0.05 to a value of 0.5 for the furthest right symbol. For the case of zero boot time this leads to a horizontal line, for each of the three policies, as there is no benefit to overhead by having an instance already running. However, for all other cases there is a small improvement to overhead by increasing the chance of keep-alive – about 12 seconds (6.6%) for the idle policy, about 18 seconds (10%) for the fixed policy and about 57 seconds (31.7%) for the load policy all for the case of a ten minute boot time. Though the increase in cost of running idle instances would seem less – ∼$600 (1.3%), ∼$1,000 (2.2%) and ∼$2,700 (5.8%) respectively. However, if the boot time is less the benefit becomes far less. The idle and fixed policy seem to follow the same curve, though with the idle points having less range, whilst the load policy shows a greater rate of decrease in overhead with respect to cost and a much larger range of costs. Therefore if cost saving is the primary goal then not using this policy makes sense, whilst if overheads

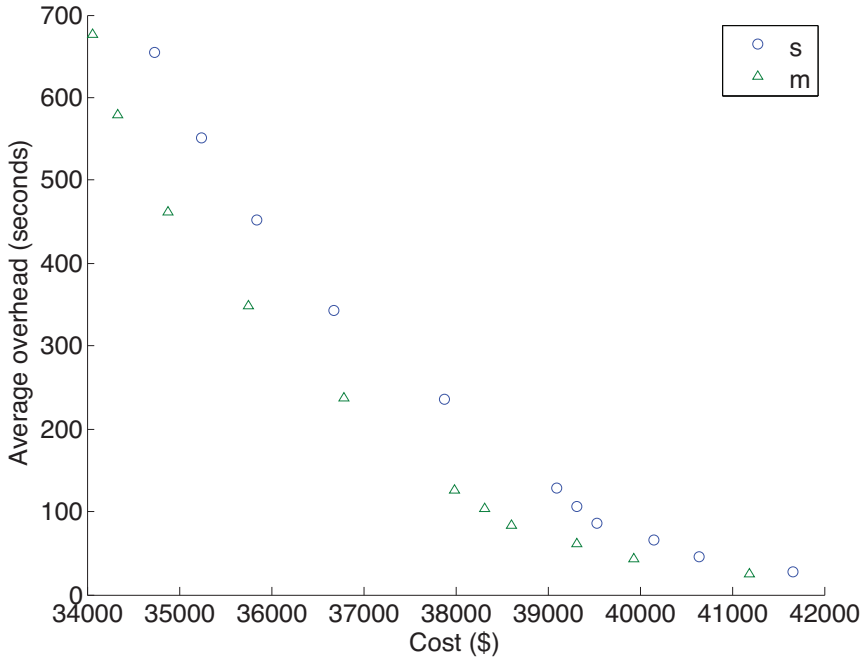

Figure 11: Varying max instance delay on average overhead and cost

are a concern and boot times are high then using the load policy would make the best sense.

6.1.4. Delaying the Start of Cloud Instances

Policy P4 is evaluated in Figure 11 in which we vary the maximum delay time for starting a new Cloud instance, in an attempt to reduce the hours consumed. Here we plot only for instances which can accept jobs from multiple users (green triangles) and instances which can only accept jobs from a single user (blue circles). A delay limit of zero minutes is the symbol furthest to the right, whist a delay limit of 30 minutes is the symbol furthest to the left with the increase in the delay for the first five symbols on the right being one minute and the remaining ones being spaced out by five minutes. As we increase the maximum delay the hours consumed decrease but the average overhead increases. As these two characteristics are conflicting in their demands it is necessary to balance maximum delay against increases in overhead. The reduction in cost is slightly more pronounced for smaller values of maximum delay whilst the overhead is almost linear which would suggest that small values for maximum job delay are appropriate.

For Figure 12 we investigate policy P5 in which we remove the delay on starting new instances (P4) when there is a high influx of jobs to the Cloud cluster. In aid of clarity, we display results for only merged jobs under exact charging, but similar trends are also observed under wall-clock charging, though under wall-clock charging overhead values fall sharply and converge to the same low value. Again we use different colours and symbols to represent different numbers of minutes delay – blue circles for one minute delay, green triangles for 10 minutes, red dots for 15 minutes, cyan crosses for 20 minutes, pink squares for 25 minutes and light green diamonds for 30 minutes. In this case the size of the queue at which delays are abandoned, as a proportion of the maximum number of instances allowed, increases from 0% on the far right to 50% at the far left in steps of 5%.

For all policies a capping of 0 instances gives the same value – this is essentially degrading the policy back to a maximum n instance policy (P1). As we increase the capping (size of the queue over which we abandon the use of delaying job start) we both decrease the cost but increase the overheads. This is due to the fact that jobs arriving when few jobs are queued will

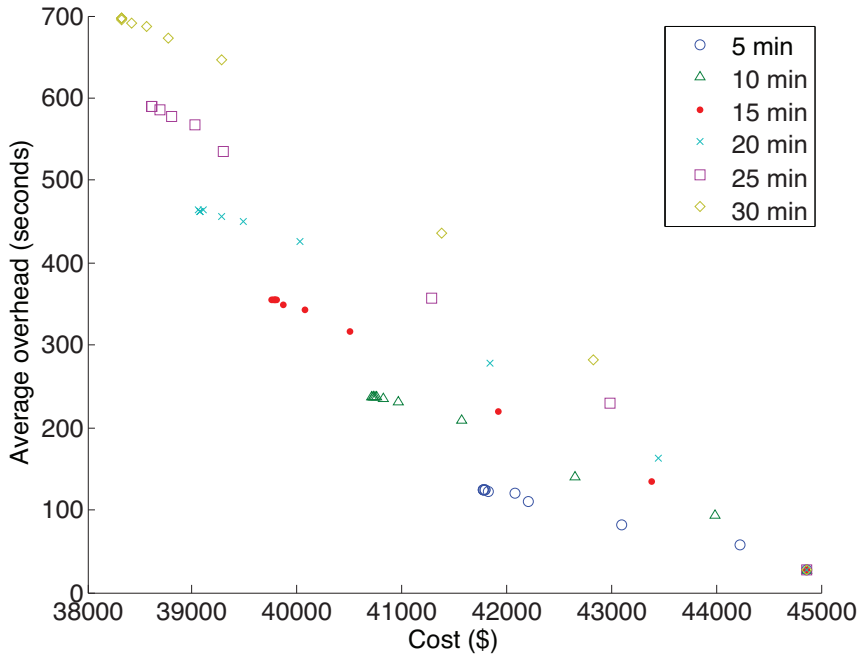

Figure 12: Varying max job delay and job delay capping on average overhead and cost

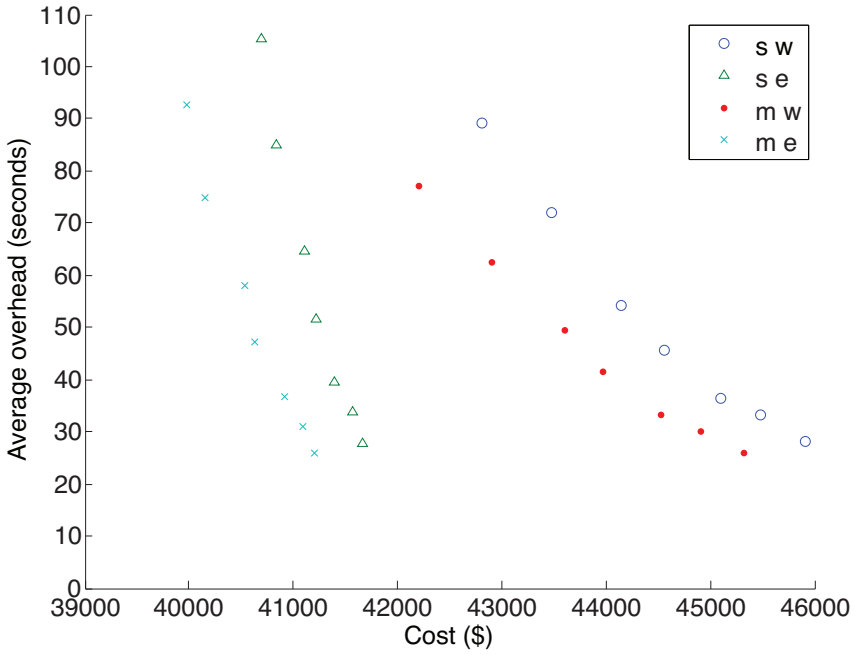

Figure 13: Varying max delay to next hour capping on average overhead and cost

be delayed – triggering an increase in overhead. But as the jobs are now potentially more bunched together then they require less instance hours to complete. This policy gives lower overheads than the equivalent delaying jobs only policy (P4) as would be expected. For example in the five minute delay policy overheads are reduced by between 2.4 and 79.5%, though cost is increased by between 10 and 18%. For the 30 minute delay case this changes the overhead to have an increase of 3.1% when abandonment is 50% of cloud instances down to a decrease of 96% when abandonment is turned off in conjunction with a 12.5 to 31.7% for increase in cost. Therefore if cost is the driving factor then this policy is not as optimal as the delay only policy, whilst if overhead is the key requirement then using a maximum instance only policy would be better than this.

We explore the effect of delaying starting up new instances until the start of the next wall-clock hour (P6) in Figure 13. Our scatter plot contains four colour / symbol combinations: blue circle for separate users wall-clock, green triangle for separate users exact charging, red dot for merged wall-clock and cyan cross for merged exact charging. The number of minutes delay until

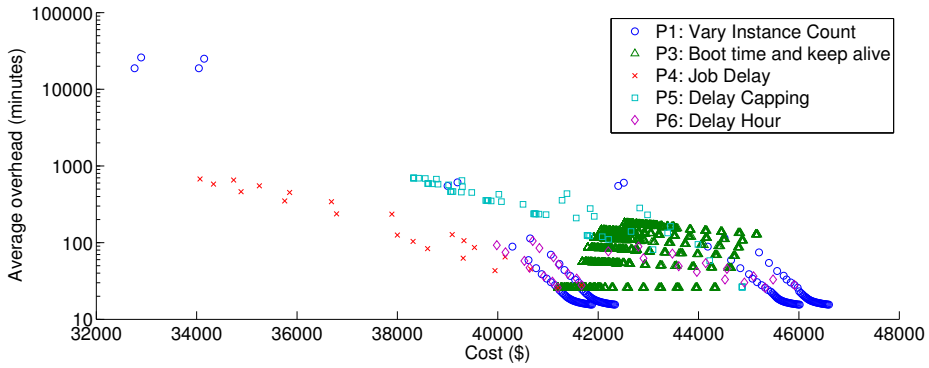

Figure 14: Comparison of all Policies

the next hour decreases for each point from right to left with the furthest right value representing no delay and furthest left representing half an hour delay with each point representing an decrease in delay capping of 5 minutes. Each transition between symbols represents a change in delay of five minutes. In all cases increasing the delay until next hour reduces cost but increases overhead. The wall-clock charging model shows the greatest decrease in cost - this is because the charge for a partial hour (starting an instance part way through a billing period) is now removed in favour of renting full hours. This gives $\sim$7% cost saving. Exact charging shows some cost reduction ($\sim$2.5%) due to this policy degrading in this case to a delay start of instance policy. In all cases the overheads increase by a similar, large, proportion. Thus unless cost is the primary concern this policy would not seem good.

6.2. Overall Evaluation

Here we compare the effectiveness of each policy set that we have proposed. Figure 14 shows all of the different polices together on the same scatter graph. Note that in this case each policy is shown with a different colour / symbol. As has been stated earlier it is not possible to minimise both cost and overheads at the same time. If the desire here is to minimise the cost of using the Cloud then having policy P1 and keeping the maximum number of instances very small. However, this will lead to excessively high overheads – $\sim 2,600$ seconds. Though using the job delay policy (P4) achieves nearly the same reduction in cost but with over an order of magnitude reduction in overheads (down to $\sim$ 675 seconds). This is still far higher than the best case of 15 seconds, though if cost is most important then this is a sensible

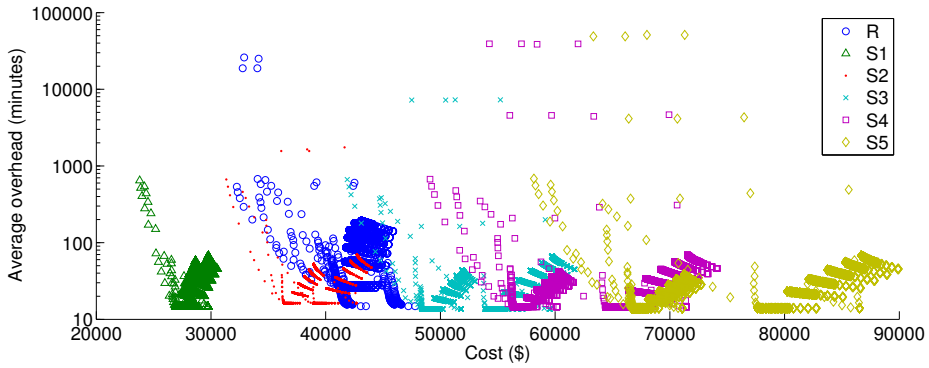

Figure 15: Comparison of all Policies over Synthetic Workloads

compromise. If overhead is the driving concern then having the highest cap on instances is the most sensible option – even better is to have no cap at all – although this leads to the greatest cost.

Policy P2: Merging different users workload. This has shown to give an improvement, though this is not highlighted in Figure 14. P3 – instance keep-alive also does not show its benefit in this scatter plot as the boot time will make it in general more expensive and higher overheads than the zero boot time cases. Though in cases where boot time is an issue it may be prudent to use it. Policy P5 unfortunately shows no situation in which it is best. Although it drops overheads slightly it is always more expensive than the non capped equivalent (P4). Likewise for policy P6 this is always outperformed by policy P4 in terms of cost and showing very similar overhead. Although not a policy there is a clear benefit, in all cases in using exact charging over wall clock charging.

Figure 15 shows the different policies now for each of the different workloads. In this scatter plot the different workloads are each represented with a different colour / symbol. Although this does not distinguish between the different polices it does highlight that the different workloads show similar trends in terms of their scatter. The synthetic workloads shows a more acute change of slope at the the lower left corner for the max instance count policy (P1) – a consequence of the synthetic workload not exhibiting the extreme submission events as seen in the real trace log (see for example June to August in Figure 5). Though each synthetic workload changes the overall cost of running work in the cloud – to be expected as they vary the number of jobs

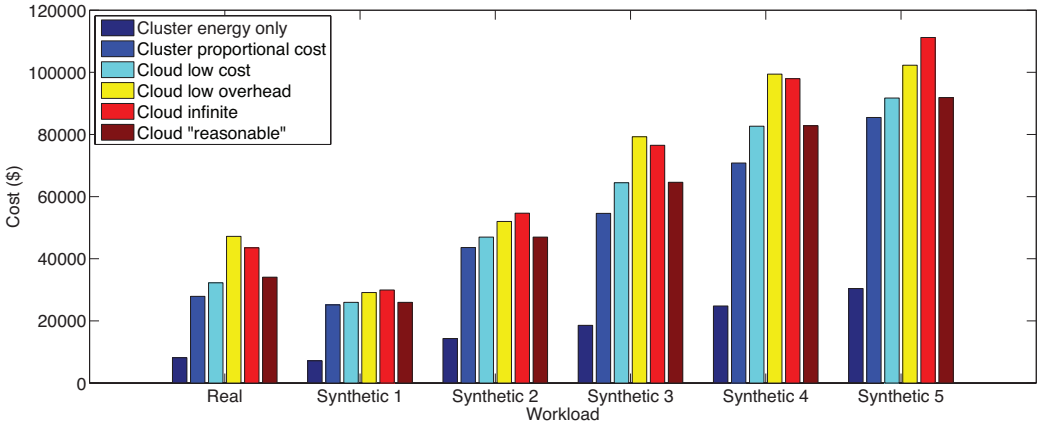

Figure 16: Cost comparison between campus cluster and cloud cluster

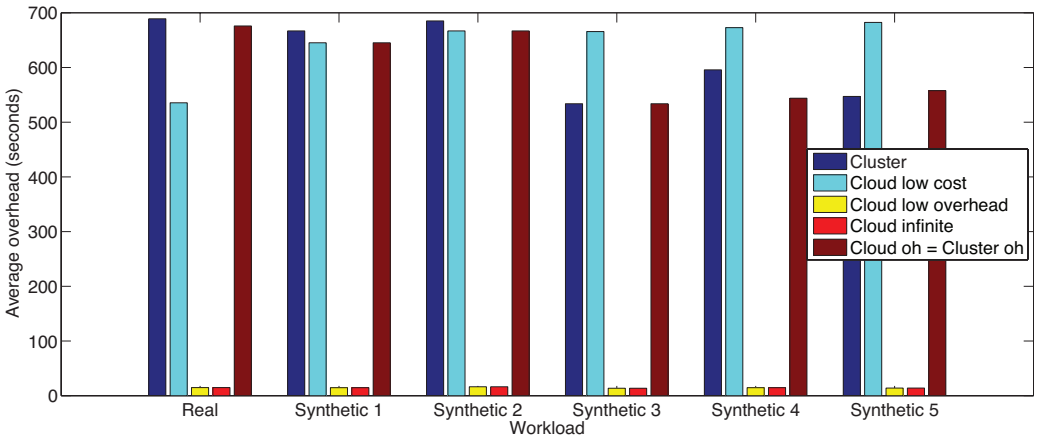

Figure 17: Overhead comparison between campus cluster and cloud cluster

to run – they show high similarity in overheads. This should be expected due to the high capacity of resource availability on the Cloud.

6.3. Comparison with Campus Cluster

Here, in Figures 16 and 17, we compare our existing HTCondor cluster with a virtual HTCondor cluster running in the Cloud in terms of cost and overheads for job execution. In these figures we evaluate two local cluster cost models, only charging for energy used and a proportion of the cluster build cost (equation 2), labelled *Cluster energy only* and *Cluster proportional cost* respectively. These were computed from our simulation model of HTCondor

[**?**] and interactive users in 2010 along with the same job logs used in the rest of this paper. Note that in Figure 17 we only show a single value here for the *cluster* as both options give the same overhead. For the Cloud we show four different results:

- *Cloud low cost* – representing the lowest cost which we were able to achieve through our simulation by varying the parameters described above.
- *Cloud low overhead* – representing the lowest overhead which we were able to achieve through our simulation by varying the parameters described above.
- *Cloud infinite* – allowing an unbounded maximum instance count.
- *Cloud same overhead* – The parameter selection which gives the most equivalent overhead to that observed in the University cluster.

We have evaluated the power consumption, for the HTCondor system, to be ∼43MWh for our real workload, which at an energy cost of $0.189 (£0.12) per KWh equates to $8,153.12 (including a CO_2 tax of £774.33), clearly making the university HTCondor resources more economic when compared to all the Cloud options we have obtained using our approaches. However, this is at the expense of a far higher overhead (over 46 times larger) than *Cloud low overhead* or *Cloud infinite*. This is a consequence of work needing to be evicted from the HTCondor due to the interactive users and delays whilst waiting for computers to become available to HTCondor. This gives a clear selection criteria over whether to use an internal cluster or the Cloud, preferring the Cloud in cases where overhead is of concern, or Quality of Service requirements are high, whilst preferring the cluster if cost is the driving factor. This effect holds for the synthetic workloads.

If we take a proportion of the cluster cost into account (using the individual cluster proportions and the costing values in Table 1 – *Cluster proportional cost*) this increases the cost for running work on the HTCondor system to $27,895.78 for the real workload. This is still cheaper than all Cloud options (with the cheapest being $32,255.58). Selecting a Cloud policy set which closely matches the overheads of the local cluster (*Cloud same overhead*) offers little over the cheapest cloud offering, often being the same or a higher cost. However, again the driving motivation to use the Cloud

would be the significant reduction in overheads. This holds for the synthetic workloads. It should be noted that the cost for computers in the university is high due to our desire to have reliable equipment and the requirement to have above average components such as graphics cards. A dedicated cluster could 'reasonably' be put together for a lower price.

The cost of data egress from the Cloud – data ingress was free during 2010 – is substantial \$13,141.24 ($\sim \frac{1}{3}$). This highlights the criticality of selecting the most appropriate work to run on the Cloud – work with little egress requirements being much more cost-effective to run on the Cloud. We have no evidence within our logs to indicate that output from one job is used as input to subsequent jobs therefore we can only assume that all output was required by the users for further local work. Ingress of data does not impose a financial cost, however, it does impose a time penalty for the data transfer. Storage of data within the Cloud – say on Amazon S3[**?**] would be an alternative, though this imposes an additional charge. Therefore we can only say that if we could reduce the data egress by $\sim 66\%$ then this would make the *Cloud low cost* option comparable with the *Cluster proportional cost* option – though with significantly better overhead. A more detailed analysis of such data storage policies would require knowledge of the contents of the files transferred and is beyond the scope of this paper.

The overhead difference between *Cloud infinite* and *Cloud low overhead* is negligible, however, for the real workload case the cost is slightly higher (by \$3,687.79). This being a consequence of the policies used. Unfortunately this is not apparent with all of the synthetic workloads due to the exact policy set used.

It should be noted that the use of Reserved Instances and Spot Instances could be used to bring the Cloud price down further. However, in the interest of fairness if we were to allow checkpoint and migration within the Cloud we should also allow for a checkpoint and migration process within the Cluster. Below we give some simple indications of how the cost could be affected, with a full analysis of both the Cloud and cluster under checkpointing and migration policies forming the basis for future research.

We assume here for simplicity that both the Cloud and the Cluster are equipped with a perfect checkpoint and migration system. This requires zero time to checkpoint and is always capable of checkpointing immediately before an eviction (we call losing a spot instance an eviction). In order to bring the price of the cloud down to the \$8,153.12 *cluster energy only* case we would need an overall average spot price of less than \$0.0618 per hour – achievable

as spot prices have been as low as \$0.012 per hour. If perfect checkpointing were to be used on the Cluster this would bring energy consumption costs down to $\sim$ \$6,319.77 ($\sim$33MWh of electricity plus CO_2 tax) thus requiring an average spot instance price no more than \$0.048 per hour, which would still seem reasonable. However, as data transfer costs are \$13,141.24 this would make it impossible to provide Cloud instances cheap enough to match this unless we can reduce data egress. If we instead assume a spot price of \$0.012 per hour this would cost \$1,581.88, leaving \$4,737.89 for data transfer. Therefore a decrease of data egress (either through better selections of files to return or through the use of compression where possible) of $\sim$36% would allow for equivalent costs.

Likewise if the proportional cost of the cluster were taken into account then the spot price would need to be on average less than \$0.197 – achievable as normal instances are cheaper. With Data transfer factored in this would need to be on average less than \$0.097. Data egress reductions could be used to improve this value.

Reserved instances allow you to pre-pay for accessing resources in the future. For example you may wish to pre-pay for n reserved instances over the next one or three years, in which case you may simultaneously use up to n instances during this time-frame at the lower hourly cost with instances in excess of this being charged at the standard rate. This may prove beneficial if your expected utilisation over this period is high enough to make the savings from lower instance prices greater than the up-front cost of taking the reserved instances – Amazon claim that you need to have at least 11% utilisation of your n instances to make this worthwhile [**?**].

A full analysis of the potential savings through using reserved instances is beyond the scope of this paper. However, we present here some initial findings on how much could be saved for the workloads presented here. We use the same cloud cost of \$0.145 per instance hour for normal cloud use and \$0.032 per instance hour for reserved instances with an up-front charge of \$405 for one year. Note that we do not include the ingress/egress charges here as this is effectively a constant value. In the case of our real workload it would be possible to reduce the overall cost of using the cloud by \$4,943.70 (21.3%) by using 20 reserved instances. By contrast for a much larger synthetic workload (Synthetic 5, 2,212,209 jobs) this would allow us to save \$15,115 (30.1%) by purchasing 46 reserved instances. This is a consequence of the larger synthetic workload having a more even spread of load over the year and being more likely to require instances for a higher proportion of the time

– leading to higher levels of utilisation.

7. Conclusions

In this paper we have demonstrated through the use of simulation how a cluster can be deployed completely on the Cloud. We have demonstrated how policies over provisioning of instances can effect the overall cost of using the Cloud and the consequence this has on average overhead for users jobs. All of these policies have the potential to decrease the cost of using the Cloud at the expense of increasing the overhead. These metrics impose conflicting demands, preventing them from both being optimised at the same time. It is therefore important to weigh up these two considerations in order to select an optimal policy set for a given Cloud cluster.

The policy of merging jobs from different users (P2) provides a benefit for overhead at no appreciable increase in cost. All other policies can provide a cost benefit, though at the expense of higher average overheads. All the presented policies have the potential to be used together thus increasing the potential gain though it is not possible to optimise both metrics at the same time hence a choice needs to be made as to the relative importance of these metrics. As the policies affect when to start up instances and how long to wait before doing so a merging of the policies would require one policy to take precedence over another. For example delaying jobs for at least ten minutes (P4) unless they are within twenty minutes of the start of the next hour (P6).

Although the Newcastle cluster is currently free it does have drawbacks: non-dedicated resources, imposed operating system and high overheads for jobs. If electricity charges were introduced – 43MWh would currently equate to $8,153.12 which is far cheaper than performing the same work on the Cloud. This is maintained as the workload increases. However, this lower cost comes with the downside of a far greater overhead on the work to be performed – 46 times longer than in the Cloud. Thus a user needs to determine if overheads or cost are most crucial to their work. Likewise the cost of the Cloud can easily become dominated by data transfer costs – in our logs we see $\sim \frac{1}{3}$ of the cost coming from data transfer. This shows the criticality of managing data transfers when working with the Cloud. Careful management of data transfer is essential if the Cloud is to compete on cost with a local cluster. Thus in general the Cloud offers a good solution in the case where prompt turnarounds and Quality of Service are important, whilst the local

cluster is good for cases of high data transfer demands and cost is the most significant factor.

If we factor into our local cluster charge a proportion of the cluster cost equivalent to the amount of work performed on it then the distinction between Cloud and cluster becomes small. The cluster just managing to beat the Cloud on cost. However, this reduction in cost difference coupled with the vast reduction in overheads is likely to make the Cloud more favourable to many. In neither case are we optimising the system. For the Cloud a checkpointing system along with the use of spot instances and better data transfer management could help significantly. Whilst in the cluster the use of checkpoint and migration could allow more efficient use of the resources. Although both systems have the potential for improvement the Cloud has the larger potential.