Developing a cost-effective virtual cluster on the Cloud

A. Stephen McGough

School of Computing Science, Newcastle University,
Newcastle upon Tyne, NE1 7RU United Kingdom
`http://www.cs.ncl.ac.uk/people/Stephen.McGough`

Abstract. The Cloud provides highly democratic 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 tasks and data storage. This has been used in the past for single research endeavours or as 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 made over such a virtual cluster to reduce the running cost and the effect these policies have on the users of the cluster.

1 Introduction

Cloud Computing [4] provides a new model for computational processing and data storage removing many of the access barriers to large-scale computing by eliminating the need for capital expenditure on large private infrastructures. Instead a user can 'rent' computational power or data space on a short-term basis – more than they could afford to buy though enough to meet their immediate needs – transferring expense to an operational cost. This approach tends to work best in scenarios with significant temporal variation in requirements – alternating between periods of little (or no) activity to periods of high activity.

This is in contrast to conventional resources available within organisations such as Universities or Companies – often in the form of a cluster of computers. Here capital expenditure is outlaid on a fixed number of computational resources and data storage. The size is dominated by two factors: the available budget, and the anticipated load on the cluster. The aim is to provision enough resources to deal with all but the exceptional load scenarios placed on the resources.

Like many institutions Newcastle University provides a computational cluster for researchers. This has the advantage of economy of scale – researchers share resources allowing each access to more than they could individually afford. Although researchers loose exclusive access to resources this is not seen as a problem as few utilise the resources 24/7. The Newcastle cluster is formed from the student access computers located around the campus. This has two disadvantages for cluster users: computers can be lost from the cluster at any time due to students logging in, no choice on operating system – student access mandates that computers run Windows, whilst most cluster users prefer Linux.

We have previously shown that ∼120MWh of energy was consumed in 2010 to power the Newcastle Condor cluster [14]. Although additional capital expenses for computer hardware and operational costs for computer maintenance exist as these computers are primarily for other purposes we do not currently take account of these. As the University is investigating the use of low powered thin clients for student access and direct charging for the energy used in the cluster this could lead to alternative approaches becoming more favourable in the future.

The advent of the Cloud, which removes capital cost and provides apparently infinite resources, has given researchers with a new way to work – often in-spite of local resource availability. Large collections of resources can be provisioned in a short period of time, quicker than many institutions can offer, for a relatively small operational outlay, a fraction of the capital cost. A second approach, Cloud Bursting, has emerged where owners of clusters have exploited the Cloud to cope with excessive demand which exceeds the resources available in-house.

Here we explore an alternative use case – moving the entire cluster onto the Cloud. Investigating if the economy of scale benefits of a conventional cluster map onto a virtual cluster and the effectiveness of policies, applied to the virtual cluster, in terms of cost savings and impact on the cluster users. We evaluate the cost of using the Cloud in terms of the hours consumed on the Cloud and the impact on the cluster users as the effect on the average make-span for their jobs. Defining make-span as the time between job submission and job completion.

We use a high level trace-driven simulation [7], using trace logs from the Condor cluster [10] based at Newcastle University [13, 14], to evaluate the effectiveness of our approach. Using just the submission times for jobs to the cluster and their execution times allows us to submit jobs into the simulated Cloud cluster where jobs will either receive service immediately, if virtual computational instances (refereed to here as *instances*) are idle, or enter a queue awaiting execution otherwise. 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 to comparatively evaluate a number of policies we do not concern ourselves with the appropriateness of these trace logs, using them only for comparison – real deployment would almost certainly alter usage patterns.

We adopt the Cloud model used by many providers (e.g. Amazon's EC2 [2]) allowing users to deploy virtual machine images onto servers owned by the provider – referred to as Infrastructure as a Service (IaaS) [18]. 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 charge from the start of the wall-clock hour in which the instance was invoked – billing from 7pm for an instance stated at 7:58pm – whilst others charge from the time the instance was invoked [9]. For clarity we refer to the former case as *wall-clock charging* and the latter as *exact charging.* 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.

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 that 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. Finally our conclusions are presented in Section 7.

2 Related Work

There is currently great interest in Cloud Computing [4]. This has lead to a number of investigations into the applicability of the Cloud as a tool for aiding researchers in their work. A number of simulation approaches to model the benefits of Cloud computing have been performed. Deelman [8] evaluated the cost of using Amazon's Elastic Compute Cloud (EC2) [2] and Amazon's Simple Storage Service (S3) [3] to service the requirements of a single scientific application. Here we seek to service the requirements of multiple users and multiple applications.

de Assuncao [5] proposed the use of Cloud computing to extend existing clusters to deal with exceptional load. This work was further extended by Mattess [12] 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 [6] uses Binary Integer Programming to select which workflows should be bursted to the Cloud. This approach is computationally expensive to determine the optimal approach and does not address the issue of when to terminate instances. It may be naively assumed that the 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 optimally manage the invocation / termination of instances. These two approaches can therefore be seen as complementary.

Marshall [11] 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 and evaluate for both overhead and Cloud cost.

Palankar [16] showed the criticality of data locality in the Cloud. We see that moving our data to the Cloud will help to reduce the data locality problem.

Additional functionality such as Amazon CloudWatch [1] allow instances to be brought up and down dependant on the characteristics of existing instances. The approaches we propose can be built into such a system.

3 Cloud Cluster Model

We discuss the general Cloud Cluster architecture that we are modelling. Each individual user is able to submit jobs to the Cluster at any time. A Job Manage-

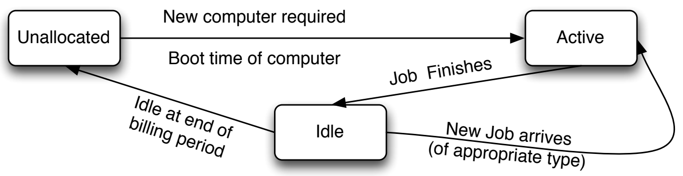

Fig. 1. The state diagram for Instances

ment Service is used to deploy these jobs to dynamic pool of instances within the Cloud. This can be one of the many existing Cluster management tool such as Condor [10], PBS [17] or (Sun) Grid Engine [15]. Additional software is required to allow the cluster to add Cloud instances, when required, and terminate these when no longer needed. Instances within the Cloud Cluster can be seen as being in one of three states with interactions illustrated in Figure 1:

– **Unallocated**: those potential Cloud instances not currently under contract of the cluster – (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 for a user. 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 it's billing period otherwise it will be released into the unallocated state.

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 when a job arrives. Instances can either be provisioned for all users within a cluster or only a specific user.

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.

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 in order 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 number of Cloud instances: Although the Cloud offers (apparently) infinite availability each provider has thresholds over which prior approval is required for more resources – EC2 is restricted to 20 instances per region, giving an overall limit of 200 instances. Limiting instances helps prevent excessive instance consumption when users submit large numbers of short jobs.

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

P2: Merging of different user's jobs: Allowing users to share Cloud instances could help reduce costs as less instances will be required and reduce make-span as jobs are more likely to discover usable idle instances. As the current cluster shares resources we are not reducing the security available to the user.

This can be implemented by having one central pool of Cloud instances with jobs being allocated to any 'idle' instance. This does, however, bring in the complexity of how to sub-charge for these 'shared' hours of Cloud usage. This can be done after an instance has been terminated using the following equation:

$$Cost_i = hours \times price \times \frac{\sum_{j=1}^{N_i} execution\_time_{i,j}}{\sum_{k=1}^{M}(\sum_{j=1}^{N_k} execution\_time_{k,j})} \qquad (1)$$

Where $hours$ is the number of hours the instance was active, $price$ is the unit price per hour, N_i is the number of jobs from source i, M is the number of sources and $execution\_time_{i,j}$ is the execution time for the j'th job from source i. Thus each source's cost is based on the proportion of the overall time the source was active on the instance relative to all sources on this instance.

P3: Instance keep-alive: Experimentation has shown the time for an instance to initialise and start accepting jobs can range from 1 to 15 minutes, with high values being detrimental to overheads. This policy allows idle instances at the end of a billing period to remain 'hired' for the next period with probability p. To prevent a half-life decay an instance which is 'idle' for a full hour will always terminate. This policy many have a more impact on the make-span 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 make-span 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 make-span.

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 make-span of the job it should decrease the cost to the Cloud.

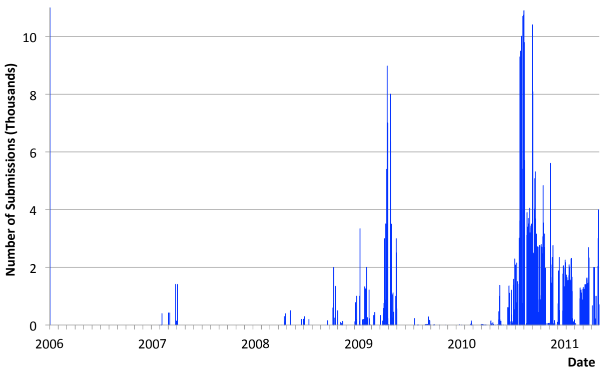

Fig. 2. Profile of job submissions

5 Simulation Environment

Our simulations are based on trace logs from the Condor high-throughput cluster at Newcastle University [13, 14]. The 1359 student access desktops, running Microsoft Windows XP, were replaced on a four year rolling cycle. As the main focus of our work is the comparison of different polices for reducing the cost of using the Cloud we ignore the differences between local and Cloud performance and assume the Cloud execution time will match the original execution time.

Figure 2 depicts the profile for the 574,701 successful jobs made between 13th October 2005 to 13th March 2011 by 21 unique users requiring 228,688 hours to execute. Jobs which were terminated before completing by the submitting user have not been used for this simulation due to their lack of execution time.

6 Simulations and Results

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. As the cost per hour of different providers varies and even temporally within a provider we quote all values here in hours consumed. A simple multiplication of this value by the current hourly rate will yield the real cost.

Table 1 shows the results under the assumption of infinite instance availability. Exact charging gives a significant decrease in hours consumed over wall-

Table 1. Baseline results for an infinite size Cloud Cluster

Charge Type	Hours Consumed	Average make-span
Exact charging	401,981	24.65 minutes
Wall-clock charging	472,571	24.75 minutes

clock charging. This equates to 97,000 hours or ∼23.3 minutes for each instance started. The make-span is almost identical with the discrepancy attributable to wall-clock instances (in general) powering down before exact charged instances, thus arriving jobs are less likely to find idle instances. The rest of the results are computed relative to the exact charge case to exemplify the relative benefits.

The following key letters are used to indicate the Cloud pricing model and source merging policy (P2) in the following graphs: **h** - wall-clock charging, **w** - exact charging, **m** - jobs can run on any instance, **s** - jobs can only be run on instances allocated to its own source. If present the number is the amount of time (in minutes) relative the experiment being performed.

Figures 3 and 4 exemplify policy P1. Increasing the maximum instances increases the hours consumed but reduces the average make-span. Exact charging remains much more optimal than wall-clock charging. The impact of merging jobs by different users (P2) appears to have only a marginal effect (1.3% ∼5,000 hours) on hours consumed, and no perceivable impact on make-span – a consequence of the cluster users working at different times, if more users were active at

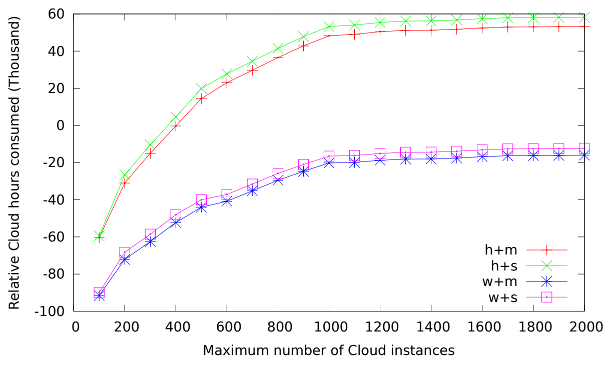

Fig. 3. Varying the maximum instance count on Cloud hours consumed

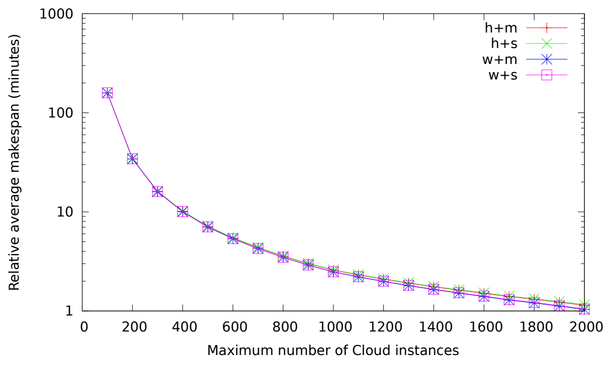

Fig. 4. Effect of varying the maximum instance count on average make-span

the same time this could lead to a significant reduction in hours. All subsequent experiments have been performed with a maximum of 500 instances.

In figures 5 and 6 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). As the startup time of instances increases so too does the number of hours consumed and the average make-span. Only for start-up times in excess of ten minutes is there a perceivable benefit to the make-span in increasing the chance of an instance remaining 'hired', though the hours consumed increase almost linearly as we increase the chance of an instance remaining 'hired'. Therefore using a policy to keep instances 'hired' in the absence of jobs only makes sense for boot times over ten minutes and with a probability of only around 10-15%.

Policy P4 is evaluated in figures 7 and 8 in which we vary the maximum delay time, for starting a new Cloud instance, in an attempt to reduce the hours consumed. As we increase the maximum delay the hours consumed decreases but the average make-span increases. As these two characteristics are inversely proportional it is necessary to balance maximum delay against increases in make-

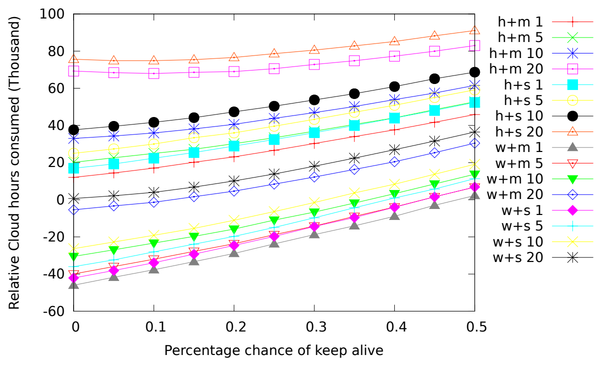

Fig. 5. Varying the boot time and chance of keep-alive on hours consumed

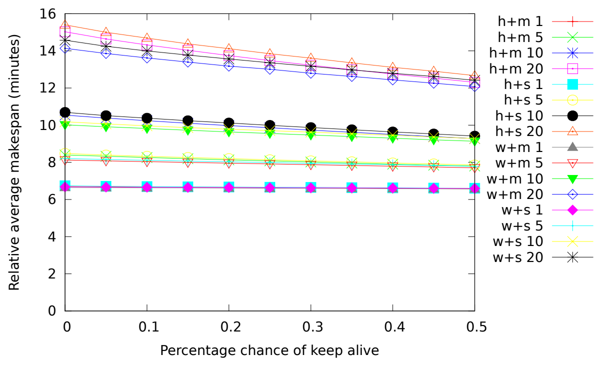

Fig. 6. Varying the boot time and chance of keep-alive on average make-span

span. The reduction in hours consumed is slightly more pronounced for smaller values of maximum delay whilst the make-span is almost linear which would suggest that a small value for maximum job delay would be appropriate.

For figures 9 and 10 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. Here there is a clear distinction between the policies for merging or not merging different users jobs. For the hours consumed if the policy is not to merge sources then there is a benefit of having a 5% threshold on removing the delay to starting resources. However, increasing this threshold has no further impact. If merging sources then the improvement isn't immediate though it does become better than the non-merged approach at around 10-20% capping. For make-span the non-merged policy reaches a maximum at 5% threshold whilst the merged policy approaches this as the threshold increases to 50%. Thus if used a capping of over 5% for the non-merged approach and around 5-15% for merged sources.

We explore the effect of delaying starting up new instances till the start of the next wall-clock hour (P6) in figures 11 and 12. The number of hours consumed

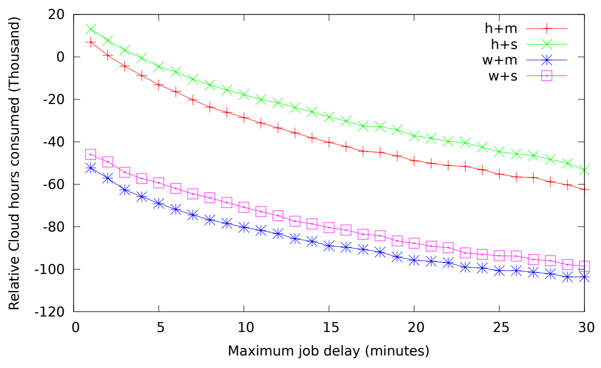

Fig. 7. Varying max instance delay on hours consumed

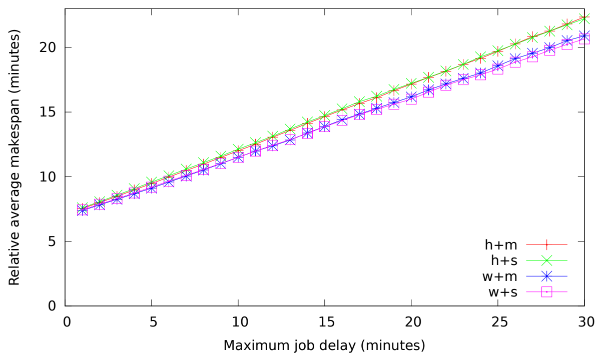

Fig. 8. Varying max instance delay on average make-span

decreases as we increase the number of minutes before the start of an hour. This is most significant for the cases of Cloud instances with wall-clock charging. The exact charging model also shows this reduction as we are producing a variation of policy P4 in which the maximum delay on instance creation is variable. When we look at the average make-span the value does increase, but only slowly, rising by only two minutes over the half hour range. Thus unless make-span is the overriding concern then this policy should be used with a high value.

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 make-span for users jobs. All of these policies have the potential to decrease the cost of using the Cloud at the expense

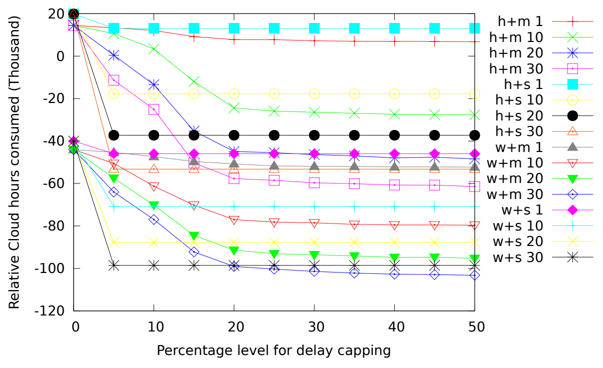

Fig. 9. Varying max job delay and job delay capping on hours consumed

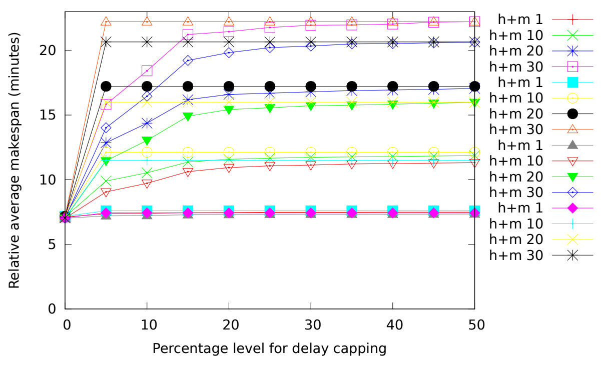

Fig. 10. Varying max job delay and job delay capping on average make-span

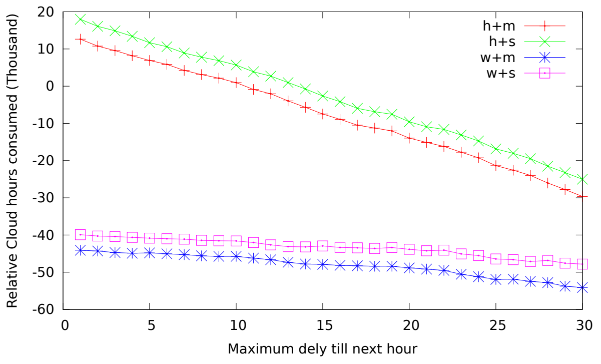

Fig. 11. Varying max delay to next hour on hours consumed

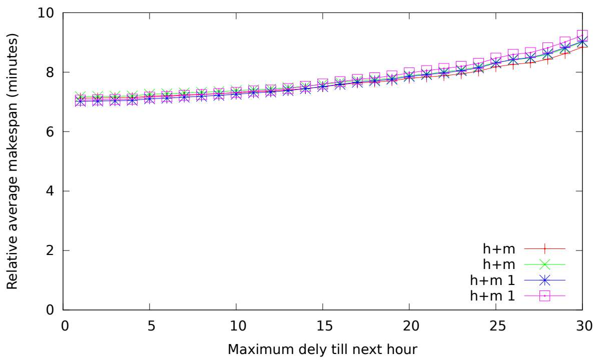

Fig. 12. Varying max delay to next hour capping on average make-span

of increasing the make-span. It is therefore important to weigh up these two considerations in order to select an optimal policy set for a given Cloud cluster.

The policies of delaying the start of instances (P4) and delaying the start of instances to the next hour (P6) appear to have the biggest impact on cost of using the Cloud with least impact on the job make-span. Especially in the latter case for resources with wall-clock charging. All the presented policies have the potential to be used together thus increasing the potential gain. As the policies effect 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 and imposed operating system. If electricity charges were introduced – 120MWh would currently equate to 335,000 hours on Amazon – although the cost of working in-house would still be cheeper the cumulative benefits for working on the Cloud would make it appear a much better option.

References

1. Amazon Web Services: CloudWatch. http://aws.amazon.com/cloudwatch/
2. Amazon Web Services: Elastic Compute Cloud. http://aws.amazon.com/ec2/
3. Amazon Web Services: Simple Storage Service. http://aws.amazon.com/s3/
4. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM 53(4), 50–58 (Apr 2010)
5. de Assuncao, M.D., di Costanzo, A., Buyya, R.: Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In: Proceedings of the 18th ACM international symposium on High performance distributed computing. pp. 141–150. HPDC '09, ACM, New York, NY, USA (2009)
6. Van den Bossche, R., Vanmechelen, K., Broeckhove, J.: Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. pp. 228 –235 (july 2010)
7. Cheng, P.S.: Trace-driven system modeling. IBM Systems Journal 8(4), 280 –289 (1969)
8. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The cost of doing science on the cloud: the montage example. In: Proceedings of the 2008 ACM/IEEE conference on Supercomputing. pp. 50:1–50:12. SC '08, IEEE Press, Piscataway, NJ, USA (2008)
9. Li, B., O'Loughlin, J., Gillam, L.: Fair benchmarking for cloud computing systems. Poster presentation, Cloud Workshop, UK All Hands Meeting (2011)
10. Litzkow, M., Livney, M., Mutka, M.W.: Condor-a hunter of idle workstations. In: 8th International Conference on Distributed Computing Systems. pp. 104–111 (1998)
11. Marshall, P., Keahey, K., Freeman, T.: Elastic site: Using clouds to elastically extend site resources. In: Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on. pp. 43 –52 (may 2010)
12. Mattess, M., Vecchiola, C., Buyya, R.: Managing peak loads by leasing cloud infrastructure services from a spot market. In: Proceedings of the 2010 IEEE 12th International Conference on High Performance Computing and Communications. pp. 180–188. HPCC '10, IEEE Computer Society, Washington, DC, USA (2010)
13. McGough, A.S., Robinson, P., Gerrard, C., Haldane, P., Hamlander, S., Sharples, D., Swan, D., Wheater, S.: Intelligent power management over large clusters. In: International Conference on Green Computing and Communications (GreenCom2010) (2010)
14. McGough, A.S., Gerrard, C., Noble, J., Robinson, P., Wheater, S.: Analysis of power-saving techniques over a large multi-use cluster. In: International Conference on Cloud and Green Computing (CGC2011) (2011)
15. Oracle: (Sun) Grid Engine. http://www.oracle.com/technetwork/oem/grid-engine-166852.html
16. Palankar, M.R., Iamnitchi, A., Ripeanu, M., Garfinkel, S.: Amazon s3 for science grids: a viable solution? In: Proceedings of the 2008 international workshop on Data-aware distributed computing. pp. 55–64. DADC '08, ACM, New York, NY, USA (2008)
17. Veridian Systems: Portable Batch Systems. http://www.openpbs.org
18. Youseff, L., Butrico, M., Da Silva, D.: Toward a unified ontology of cloud computing. In: Grid Computing Environments Workshop (GCE '08). pp. 1 –10 (2008)