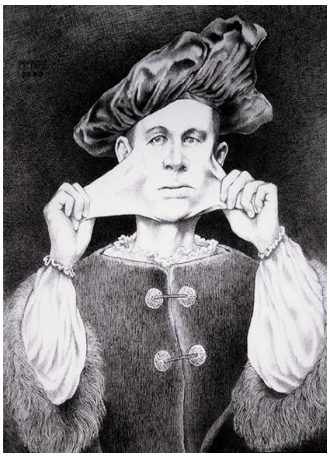
Is your Cloud Elastic Enough? Part 1

Paul Brebner, Senior Researcher, NICTA, Paul.Brebner@nicta.com.au




Paul Brebner is a senior researcher in the e-Government project at National ICT Australia (NICTA, http://nicta.com.au/), based in the Canberra Research Laboratory. His research focuses on distributed systems, software architecture, performance and scalability. He has conducted research and industry engagements on the performance of enterprise middleware, Grid computing, SOA, and cloud computing while working for the CSIRO, UCL, and NICTA. Paul has contributed to the Standard Performance Evaluation Corporation (SPEC) Java committee and currently represents NICTA on the new SPEC Cloud Research Working Group [http://research.spec.org/]

Cloud Elasticity

"My biggest problem is elasticity. VM spin-up time... Ten to 20 minutes is just too long to handle a spike in Yahoo's traffic when big news breaks such as the Japan tsunami or the death of Osama bin Laden or Michael Jackson." [Todd Papaioannou, vice president of Yahoo's cloud architecture, quoted in "Yahoo builds ultimate private cloud", Network World, July 19, 2011, http://www.networkworld.com/news/2011/071911-yahoo-private-cloud.html]

Not everyone may be running an enterprise on the scale of Yahoo, but in order to take advantage of the opportunities provided by cloud computing (and avoid the risks!) it is vital to understand the elasticity characteristics of your applications, workloads and cloud platform. Intrinsic to the definition of cloud computing is that resources can be dynamically increased and decreased on demand, and that charging is consumption based. You can get sufficient resources when needed, but you only pay for what you use. Ideally a cloud platform is infinitely and instantaneously elastic.

An application could be scaled out indefinitely with increasing load, and this could happen as fast as the load increases with no degradation of response times. Resources would be available instantly and the application would be immediately deployed and available for use. A perfectly elastic cloud platform would be ideal for hosting interactive applications with strict response time requirements, and with spiky unpredictable workloads. These are difficult to host on traditional fixed and finite infrastructures as the quantity of resources is not known in advance, and the cost of keeping the resources available for occasional extreme load events is prohibitive.

However, real clouds are not perfectly elastic. If you stretch a rubber band (or your face, see picture) far enough it will break! There will inevitably be a delay between when resources are requested, and when your application is running and available on it. The resource provisioning speed may depend on a number of factors including: the type of cloud platform (e.g. Infrastructure vs. Platform as a Service); the type, cost model, number, size, or speed of resources requested; the availability of spare

resources in the requested region and the demand on the cloud platform from other users; the rate of increase (acceleration) of your workload; and any quotas or limits imposed by the cloud platform or your contract with them.

On a typical IaaS (Infrastructure as a Service) cloud platform the elasticity infrastructure enables auto-scaling of instances for an application with user customised rules which periodically fire to check metric values, make decisions, and then request an action in response (e.g. increasing or decreasing instances by a specified amount). There are typically a number of steps involved in auto-scaling:

- Periodically fire the rules (e.g. every 5 minutes)
- Depending on the metric and threshold (e.g. server utilisation > 80%)
- and the statistic and time period (e.g. average over the last 10 minutes)
- Then execute the resource request actions (e.g. increase instances by 1).

The cloud infrastructure cannot instantly respond to this request as it has to first find and reserve an available server, create a new virtual machine instance on it, deploy the application code and other data onto the new virtual machine and start the application, and include the new instance in a load balancer so it can be accessed externally (e.g. the Amazon Elastic Load Balancer). The time taken for all these steps is often referred to as the instance "spin-up" time. It may also be possible to suspend the rules from firing for some period of time once a rule has fired to allow the requested actions to be completed, to prevent premature rule firing.

There may also be a delay between requesting an instance to be killed to when it is actually terminated. The main impact of this delay is cost rather than performance if instances are charged based on the time they are running for (typically from instance request to instance termination), and some granularity (for IaaS often a minimum of 1 hour or part therefore). If an instance goes over the hour charging boundary before being terminated you will be charged for an extra hour even if you didn't need it.

Different resource types may reduce the spin-up time. For example, some clouds offer reserved (e.g. Amazon) or "always on" (Google AppEngine) instances. Depending on the cloud provider and cost model, these are instances that are paid for up-front for a minimum period of time (e.g. 1 year), but are cheaper to run per hour, and may have a faster spin-up time. However, for the remaining discussion we assume "on-demand" instance types, with a typical (constant) instance spin-up time of 10 minutes. In practice, spin-up time depends on the particular cloud provider and can vary considerably. Consequently, most cloud providers do not have a Service Level Agreement (SLA) for spin-up time.

Cloud Performance Modelling

Our research over the last 4 years has focussed on the performance modelling of Service Oriented Architectures. We have developed a tool and method for SOA performance modelling which has been trialled and validated on a large number of government and non-government enterprise SOAs at different stages of the software development lifecycle. From these engagements we have selected three example applications and workloads that are particularly relevant for evaluating cloud elasticity.

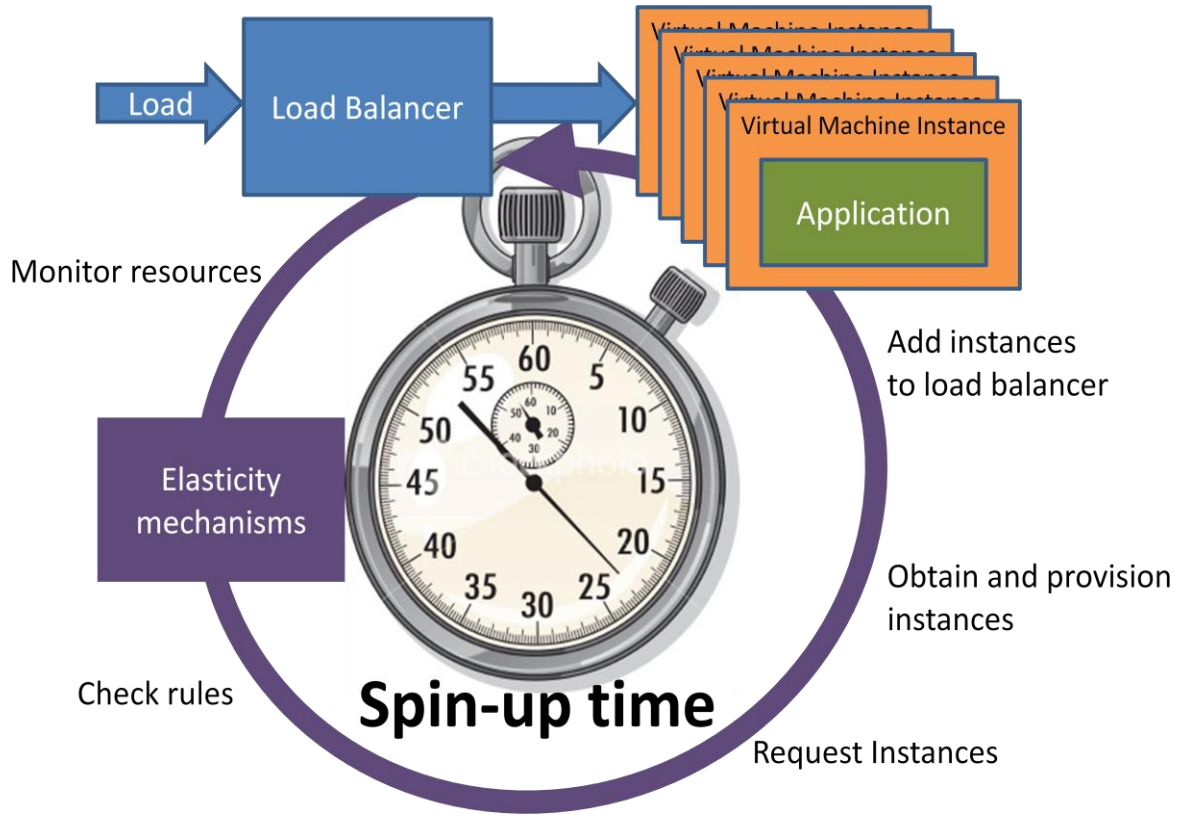
The three example applications are: (1) BigCo, an enterprise application that has a variety of different user types (internal, external business partners, and web and mobile customers), with a workload which gradually increases and then decreases over a 24 hour period (2) Lunch&COB, a whole of government SOA application which is distributed over four zones (separate applications deployed on distinct Virtual Machines), with 2 different workloads, one longer but lower peak at lunch time and another higher but shorter peak near close of business, and (3) FlashCrowd, a web site which typically has a low background load, and then occasionally exhibits a very large spike in demand over

a 1 hour period (representative of a “Flash Crowd”). We present the first example in this article, and the others in part 2.

We have previously modelled and validated these applications and loads for fixed resources to explore the impact of different workloads, for capacity planning, to assist with developing SLAs, and to investigate architectural alternatives. From these experiments we know that these three applications are CPU rather than bandwidth or database limited. We have also modelled some of them with different resourcing models including virtualisation and cloud platforms (e.g. Amazon EC2, Google AppEngine, Microsoft Azure) to predict performance, scalability, cost, and power consumption. [See: http://www.nicta.com.au/pub?id=4428, http://www.nicta.com.au/pub?id=2409].

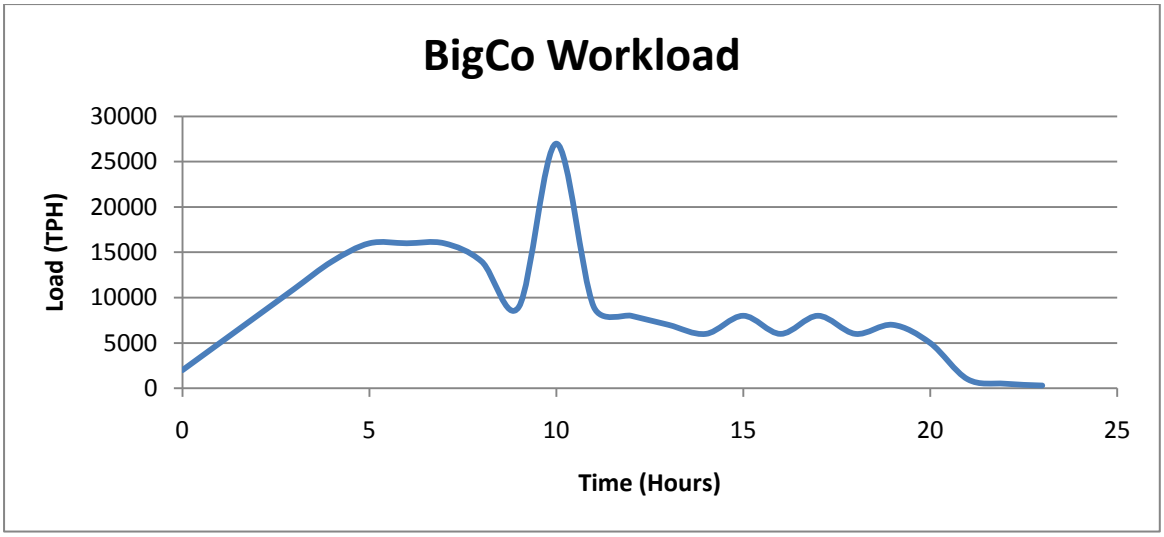
Our service-oriented performance modelling method and tool are also well suited to modelling the impact of the dynamic aspects of systems including workload changes and resourcing elasticity. We have constructed models of these three applications on a generic elastic cloud platform (inspired by Amazon EC2), focussing solely on the CPU resource requirements and costs at present (our previous models have been wider in scope but did not explicitly model elasticity).

The following diagram shows the main components of an elastic compute cloud that are included in our cloud elasticity models: incoming load, elastic load balancer, virtual machines with deployed application, and elasticity mechanisms (monitoring, rules, instance requests, instance provisioning, etc) contributing to the spin-up time.

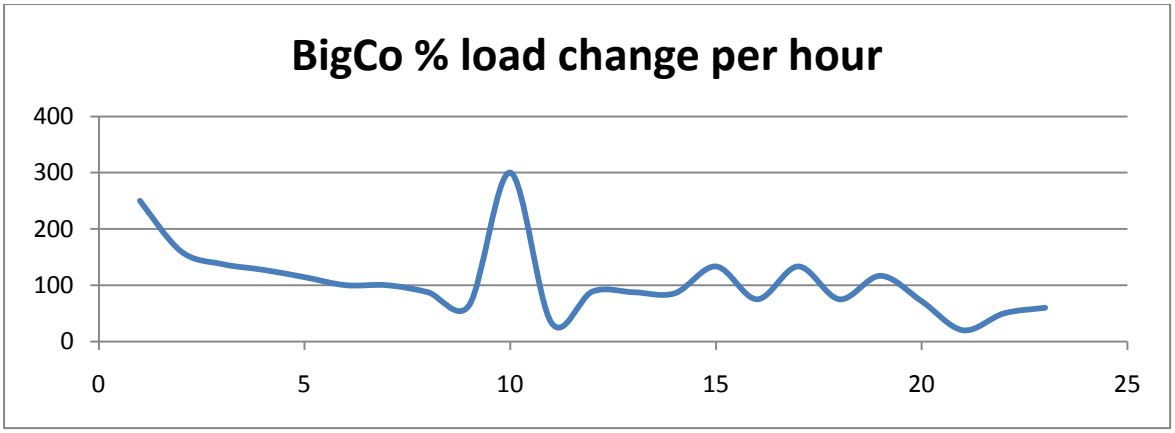


BigCo Elasticity

The workload for the first application, BigCo, runs for 24 hours and represents an observed extreme case of a typical day. During the 24 period the load ranges from a low of 300TPH (Transactions Per Hour) to a high of 27000 TPH, a difference of 9000%.



The next graph shows the percentage load change per hour, and reveals that there are two parts of the workloads that need rapid elasticity: The maximum peak at hour 10 with the maximum rate of increase per hour of 300%, but also the initial hour with 250%.



All three examples chosen are user facing interactive applications with strict response time requirements of a few seconds at most. We assume a SLA of 99% of transactions under 10s response time. In practice a stricter SLA could be required. Even 1% of transactions taking longer than 10s during periods of peak load would result in loss of business and consumer trust.

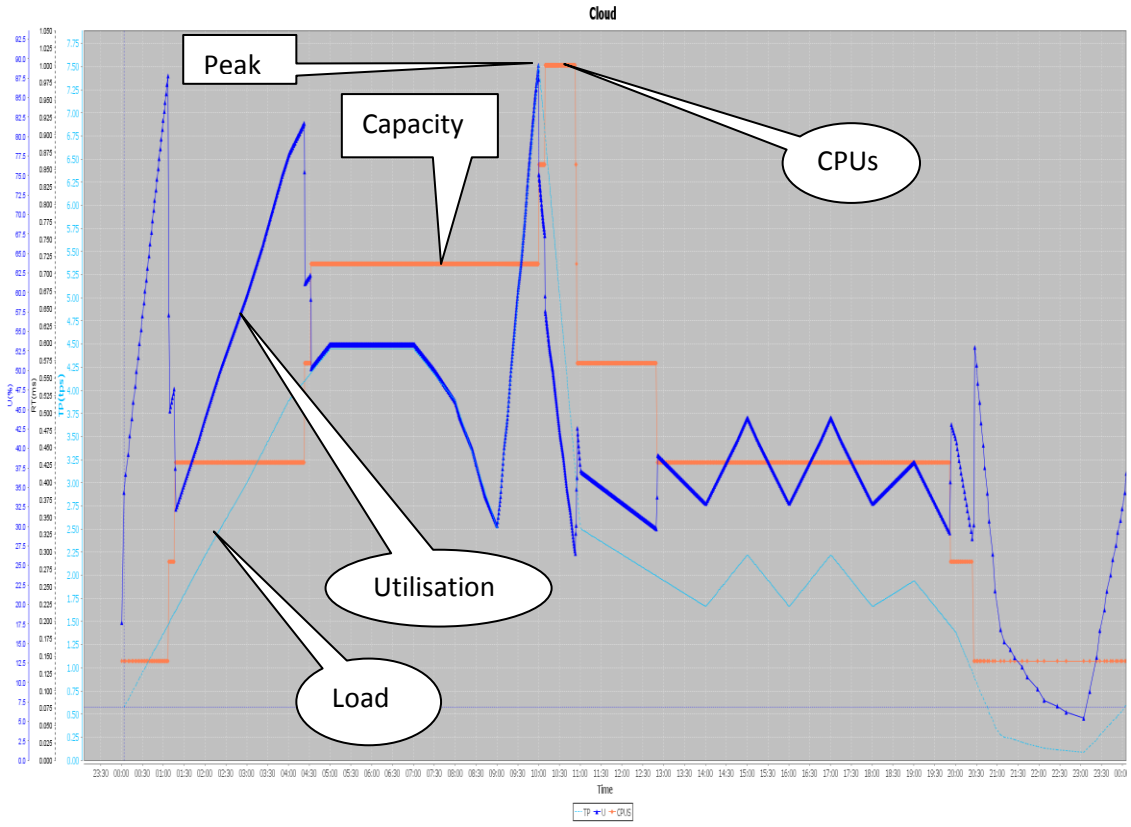
IaaS cloud platforms typically offer a variety of instance sizes, from partial cores (multi-tenancy on shared servers) to multiple cores (single-tenancy on large servers). Due to the minimum application requirements for core speed and other resources (e.g. memory and network bandwidth), we selected a single-tenancy 4 core (approx 2.4GHz Intel core speed) instance type. We assume a cost of 40 cents an hour or part thereof. Instances are charged for a minimum of an hour, and extra hours are charged if the instance is still alive every hour after the start time. Our modelling and simulation tool is used to graph raw performance metrics, and also computes the percentage of transactions under the 10 second response time SLA threshold (% success) and the total estimated cost, for a number of elasticity scenarios. Results with higher percentage success and lower cost are more elastic.

Example Elasticity Predictions

To illustrate the impact of “typical” elasticity characteristics we assume the following default settings for the cloud platform auto-scaling rules and instance spin-up behaviour: Rule check every minute, increase request threshold of 80% average server utilisation computed over 1 minute, instance decrease threshold of 30% server utilisation over 10 minutes, only 1 instance increase/decrease at a

time, and a rule suspension time equal to the expected instance spin-up time. The spin-up time is initially set equal to 10 minutes in the model.

We ran the model with the workload above for a simulated 24 hour period, starting with a single 4-core instance, and allowing instances to be created and destroyed based on these rules and a 10 minute spin-up delay. The model predicts that all response times are under the SLA threshold of 10s (100% successful transactions), a maximum of 24 cores is needed for the peak load, and the cost is $32.40/day. It is clear that the cloud platform is elastic enough for this application and workload combination given these default elasticity characteristics. The following graph shows simulated metrics (Load, Utilisation, CPUs/Cores) for the cloud servers over the 24 hour period. Load is in TPS, CPU Utilisation in percentage of resources used (0-100%), and number of CPUs range from 4-24 (increments of 4 as 4 core servers are used).

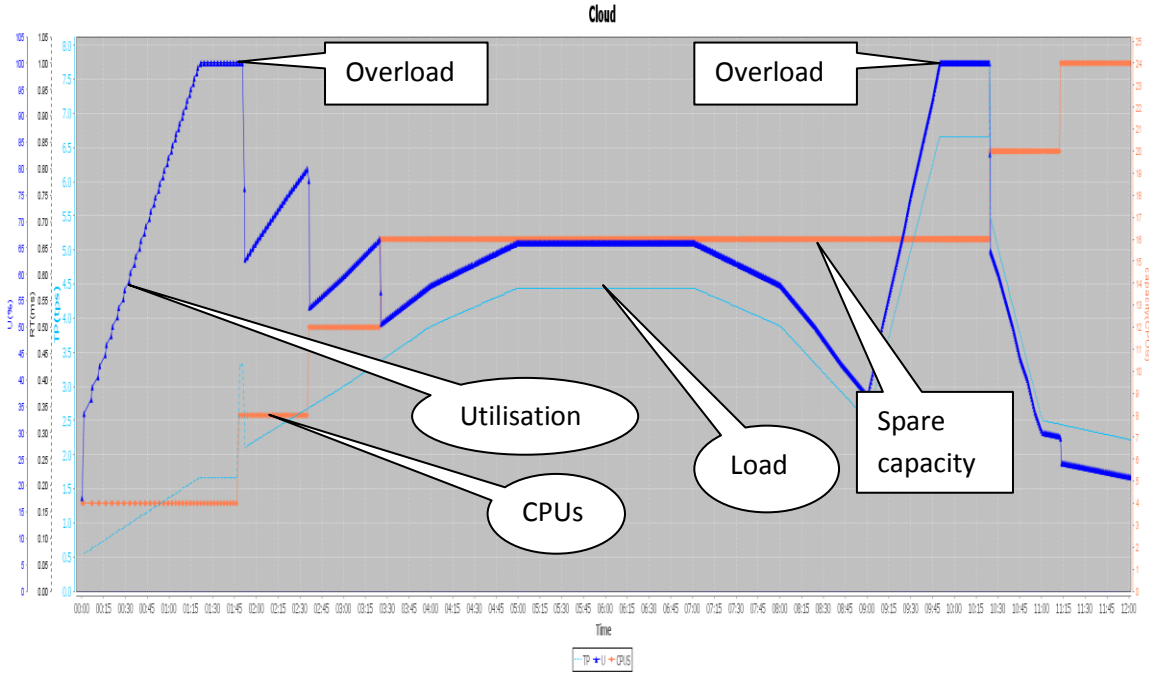


Note that typically the total cost of using the cloud will be higher than this, as different cloud providers charge for different resources in different ways. There may be extra charges per transaction, for data in and out, for data storage, for management, monitoring and load balancing, etc, and based on our previous work these may contribute to approximately 2/3 of the total cost. However, as our focus is only on elasticity at present we ignore these charges.

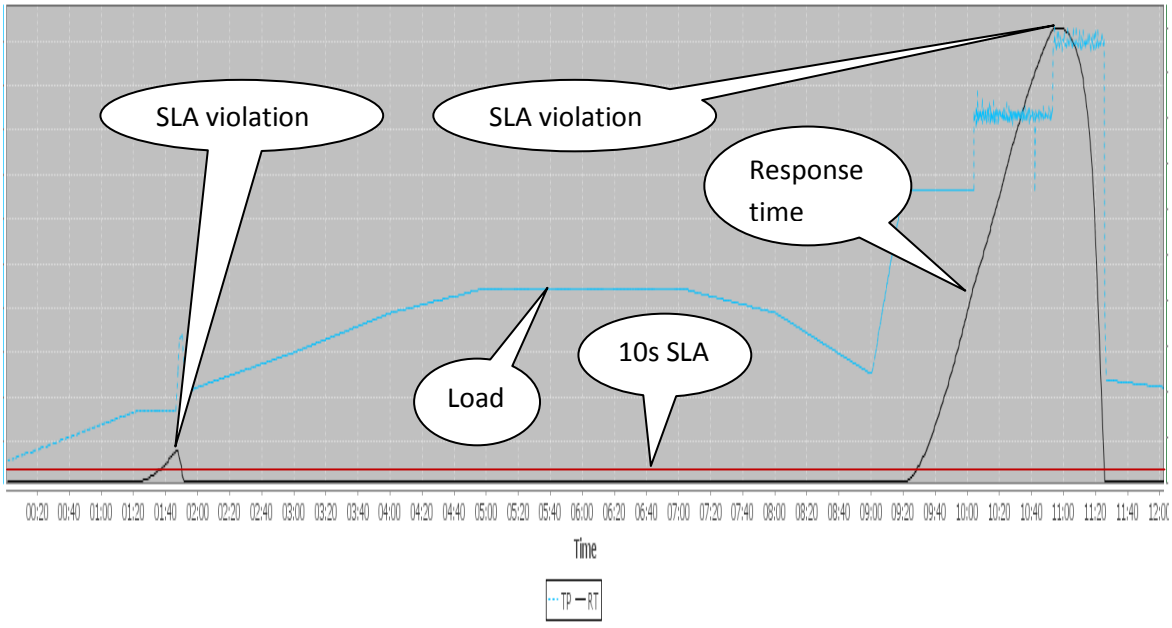
In order to find the breaking point (where the platform is not elastic enough) we increased the spin-up time in increments of 10 minutes until the SLA is violated. Somewhat surprisingly this didn't happen until 50 minutes spin-up time. 50 minutes is longer than any reasonable cloud spin-up time under normal operation, but could be encountered if the cloud platform is overloaded. At 40 minutes spin-up time over 99% of transactions are still under 10s, but by 50 minutes spin-up time only 89% of transactions are successful. The cost is $29.20/day (in general, increasing the spin-up time decreases the cost).

The following graph shows the Load, Utilisation, and available CPUs for the 50 minutes spin-up simulation for the first 12 hours. There are two periods (marked "Overload" where Utilisation is 100%)

where the number of CPUs are not keeping up with demand due to the rapid increase in the load and the excessive spin-up time. In the case of the second peak this did not happen as early as expected due to spare capacity from the previous peak that had not yet been reduced by the instance decrease rule.



The following workload graph shows the load and response times over the same time period. Note the two spikes in response times where elasticity is not sufficient and the 10s SLA (red line) is violated.



In this part of article we have introduced the basic concepts behind cloud elasticity, and used our modelling and simulation approach to illustrate that a typical cloud platform can satisfy the elasticity requirements of our first example application, BigCo. In the continuation of this article we will:

- Explore cost and SLA comparisons for different rules, resourcing approaches, and load scenarios for this application
- Conduct a similar analysis of the Lunch&COB and FlashCrowd applications (with surprising results!)
- Extend the analysis to look at a more realistic deployment scenario using two availability zones (given the recent Amazon outage which impacted applications on a single availability zone).

In the meantime here are some other CMG resources on cloud performance:
http://www.cmg.org/measureit/issues/mit80/m_80_3.pdf
http://www.cmg.org/measureit/issues/mit76/m_76_3.pdf
http://www.cmg.org/conference/cmg2009/awards/9138.pdf