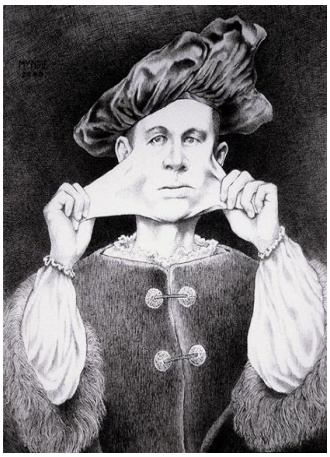
Is your Cloud Elastic Enough? Part 2

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/]

In part 1 of this article [http://www.cmg.org/measureit/issues/mit82/m_82_3.pdf] we 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 BigCo, our first example. In this part we explore the extremes of elasticity for BigCo and introduce Lunch&COB, our second example.

BigCo: Elasticity Alternatives

In order to further understand the benefits of typical cloud elasticity we will explore three elastic provisioning alternatives for BigCo. These are: Worst case (no elasticity), Best case (best practical elasticity that could be achieved given constraints of cloud platform), and Perfect elasticity (best theoretical possible case).

Worst case elasticity has no elasticity mechanism and instead relies on fixed over-provisioning of resources for the 24 hour period. Only if the maximum load is known in advance can sufficient fixed resources can be made available in advance to prevent SLA violations. Depending on the workload, the cost is likely to be higher than using elasticity to manage resources dynamically. If the workload is higher than predicted then the fixed resources will be saturated and the SLA will be violated.

Best case elasticity attempts to predict the most elasticity that can be achieved given the constraints of a given cloud platform. We assume that 4 CPU instances are required for the BigCo application. Because the resources must be increased/decreased in increments of 4 CPUs the resource quantity may not be perfectly matched to the demand. Because of limitations of the cloud monitoring infrastructure, we assume that the minimum rule check period is 1 minute. In the worst case there may be a lag of 1 minute between a load increase and detection which only then triggers a resource increase request. Therefore the main change to predict best case elasticity is that we assume a spin-up time of zero seconds. Zero seconds spin-up may not be achievable in practice, or to do so may change the costing model. For example, some cloud providers provide reserved instance types which

may have a faster spin-up time, but may cost more as there is a fixed cost to reserve an instance for some minimum time period (e.g. 12 months) plus usage costs.

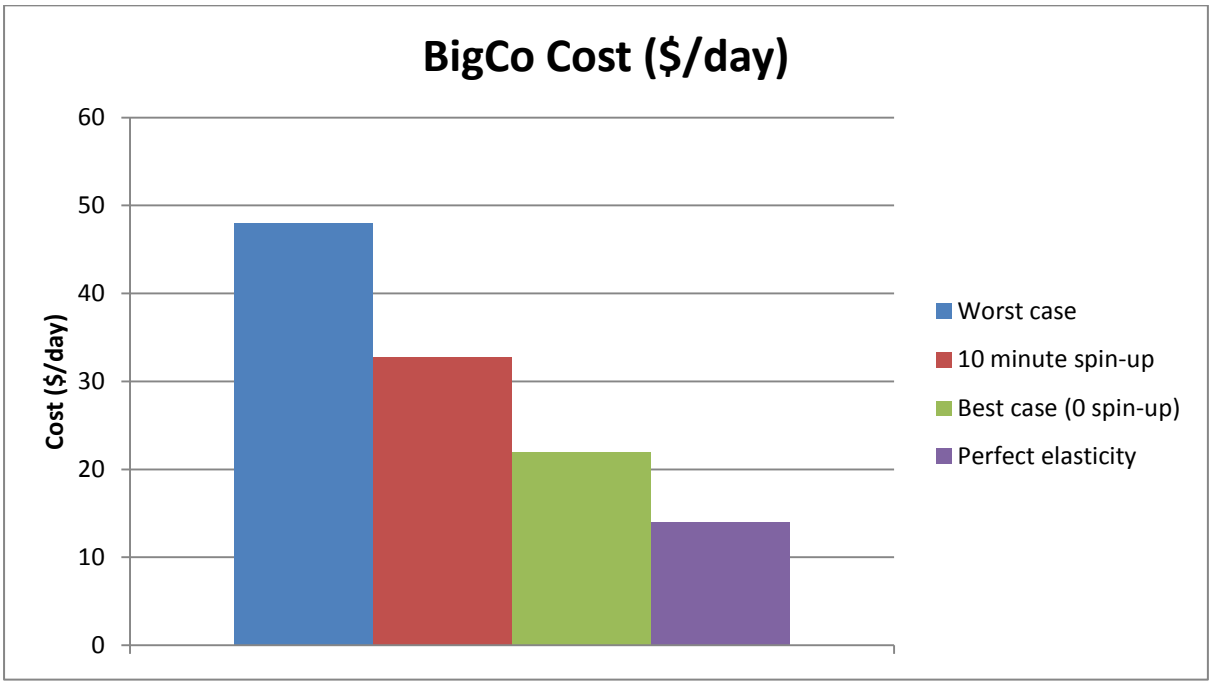
What does a perfectly elastic cloud platform look like? To be perfectly elastic the resources exactly match the demand due to the workload (i.e. the quantity of resources is correct and unlimited), there is no time delay between detecting load changes and changing resourcing levels (resourcing is instantaneous), and you only pay for what you consume (charging is fine-grain consumption based). Even though it is unlikely that any current cloud platforms are perfectly elastic, we can easily model perfect elasticity.

To evaluate the worst case elasticity, our performance model is used to first determine the minimum resources required to satisfy the SLA for the peak load. In this case, 20 CPUs are adequate. The model was then run with fixed 20 CPUs for the entire 24 hour workload and predicted a cost of $48/day.

To evaluate best case elasticity we model the cloud infrastructure with zero spin-up time, and increase threshold = 95% Utilisation and decrease threshold = 50% Utilisation. Higher thresholds can be used and still satisfy the SLA as less headroom is needed due to instances being available instantly. Starting with a single 4-core instance, and allowing instances to be created and destroyed just-in-time, the model predicts a maximum of 20 cores, and a cost of $22/day, just under half (45%) the worst case.

For the worst and best cases we assumed the same default cost model (4 CPU instances charged at 40c/hour or part therefore). However, for perfect elasticity we assume a perfectly elastic cloud platform and an extremely fine-grained cost model which only charges for resources that are actually consumed (by the CPU millisecond, at a rate proportional to the default cost model). The model of perfect elasticity predicts a reduced cost of $14/day.

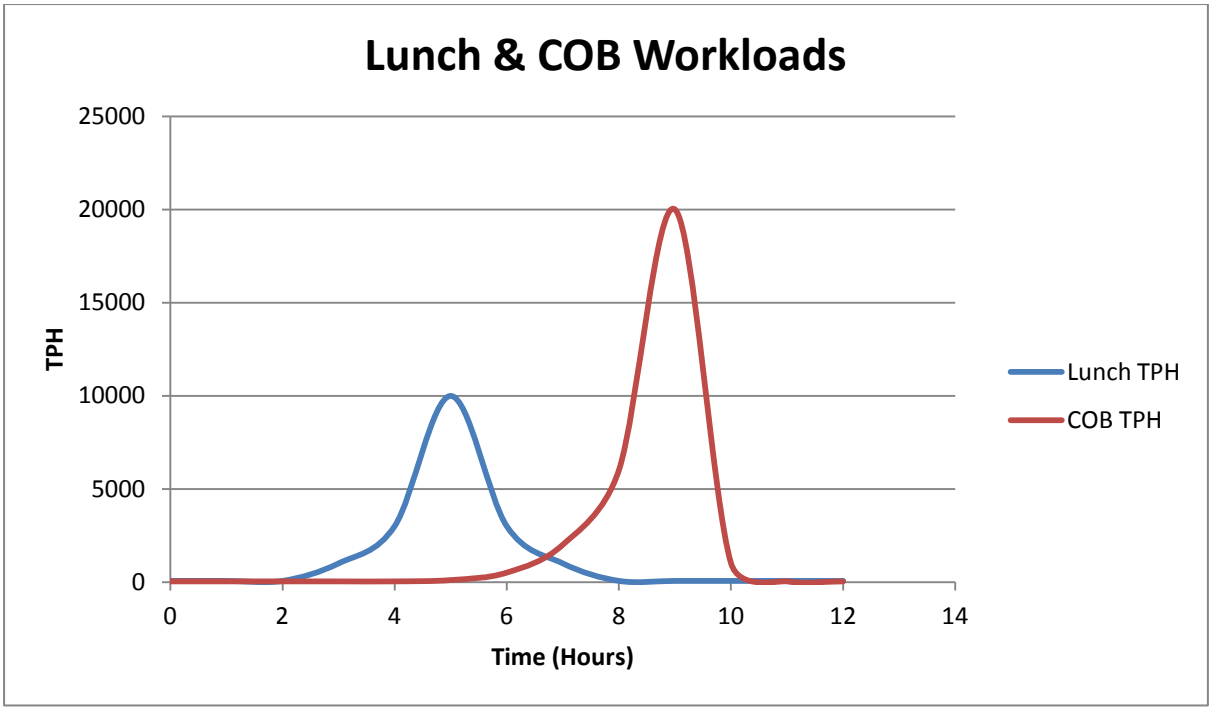
For all of these extremes the SLA is satisfied as 100% of transactions are under the 10 second response time threshold. The following graph shows the cost per day for the all of the elasticity options considered (worst case, 10 minute spin up time from part 1, best case, and perfect).



The next graph shows the cost savings. A 10 minute spin-up results in a cost saving of 32% compared with the worst case, increasing to 54% as spin-up time approaches zero, while perfect elasticity is 71% cheaper.

Lunch&COB Elasticity

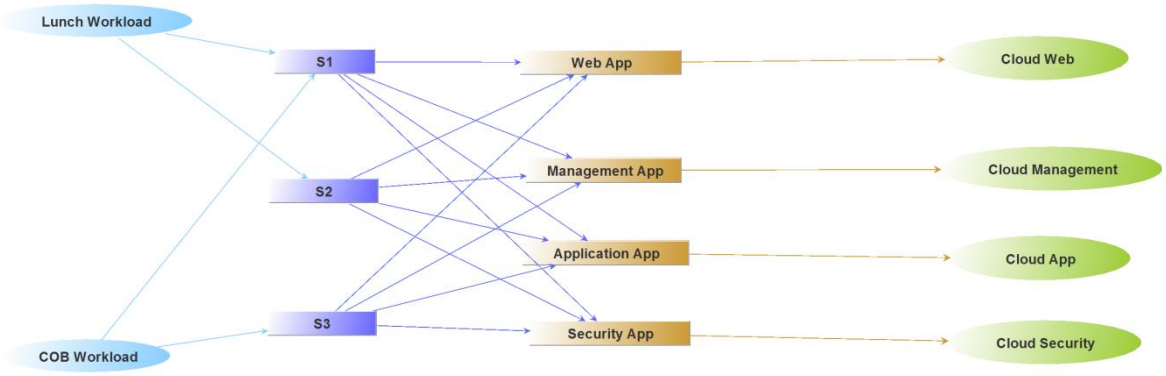
Our second example is modelled on a whole of government Service Oriented Architecture (SOA) application supporting multiple user types including government, business, and citizens. This example was originally modelled and validated on dedicated virtualised hardware, and we have since modelled it on various cloud platforms. It is a distributed application consisting of four distinct application zones (Web, Management, Application, and Security) which are deployed on separate virtual machines so that they can scale independently of each other. The workloads occur during extended business hours (12 hours). Two peaks are expected, one around lunchtime with a peak of 10,000TPH and the highest with a peak of 20,000TPH at close of business (COB) as shown in the following graph.



Since 2007 we have developed a Service Oriented Performance Modelling (SOPM) technology to enable the rapid modelling and prediction of service specific metrics. The approach allows for building, calibrating, validating, and using models directly in terms of concepts relevant to service architectures, including external workloads which consume services, composite and atomic services and their implementation details as workflows, and services deployed to physical resources (including physical and virtual servers, disks, and networks). Models are calibrated with performance data obtained from a variety of sources including documentation (SLAs, software, network, hardware, operational and business), log files, benchmarks and monitoring infrastructure.

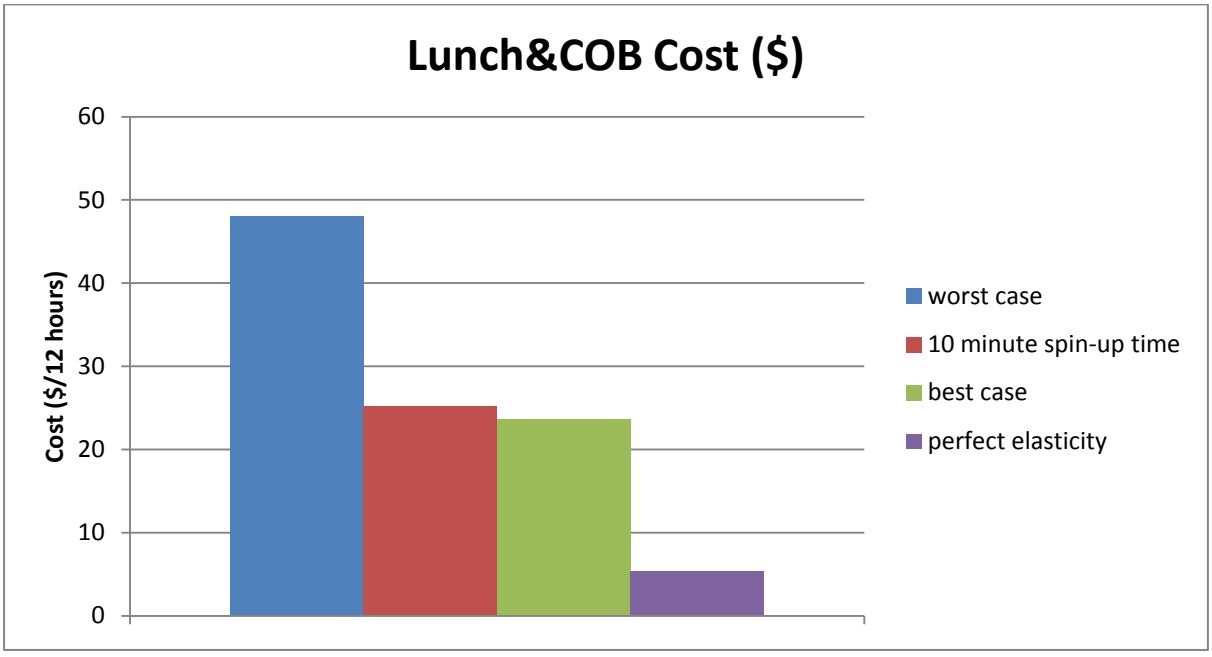
Our tool is model driven and supports a meta-model of the service oriented performance model and GUI-based editing, viewing and animation. To predict the metrics from it, a transformation is made to a run-time version of the model which is simulated dynamically using a discrete event simulator. The impacts of changes can be explored at run-time by manually changing parameters with slider controls or using pre-written scripts. Detailed metrics are available, not just average values, enabling us (e.g.) to produce histograms of response times to compute SLAs. Complex dynamic resourcing models can also be built and solved at run-time with this approach, making it ideal for investigating cloud elasticity which intrinsically has time as a variable. Cost is a secondary metric which is computed from knowledge of the cloud cost model, resource usage, and instance request and termination times.

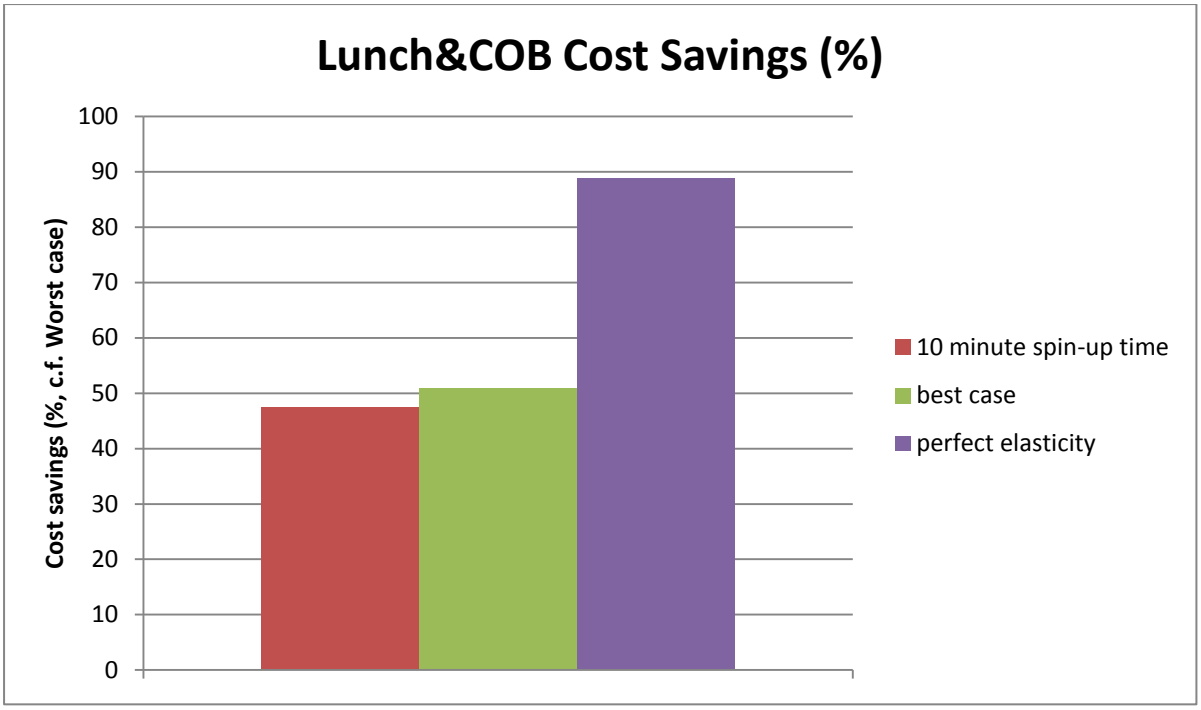
The following screenshot is the high level SOA model view from our performance modelling and simulation tool. The detailed workflows, parameterisation and cloud elasticity mechanisms are at a more detailed level. This view shows the two workloads, Lunch & COB on the left, the external composite services (S1, S2, and S3) that are invoked by the workloads, the implementation of the composite services as workflows calling atomic services deployed on the four application zones, and finally the deployment of the application zones on elastic cloud servers. The workloads impose different demands on each of the application zones, so the resources for each zone must be scaled at different rates. We assume each zone is initially deployed to a single 4 CPU cloud instance.



As with our first example, we initially run the simulation with a default spin-up time of 10 minutes. This proves to be elastic enough for this application and workloads. All transactions are below the 10 second response time SLA threshold, a maximum of 56 CPUs is allocated for the peak workload (20 CPUs for the Web Application Zone, and 12 for each of the other zones), and the cost for 12 hours is estimated to be $25.20. For comparison, we also run the simulation for worst, best and perfect case elasticity. The worst case option with fixed resources for 12 hours costs $48, and the best case costs $23.60. The perfect case is significantly cheaper at $5.37.

The following graphs shows the cost and cost savings for 12 hours for each option. The 10 minute and best case elasticity costs are very similar and represent a cost saving compared with the worst case of up to 50%. The fact that they are very similar is likely to indicate that the elasticity is close to breaking point. However, perfect elasticity is 89% cheaper, suggesting that there is even more room for improvement in the elasticity of cloud platforms for more elastically demanding applications and workloads such as Lunch&COB.





As with the BigCo example we want to determine how much elasticity margin the cloud platform has for this application and workload by increasing the spin-up time in increments of 10 minutes until the SLA (99% of transactions under 10s) is violated. In this case it turns out that the elasticity is close to fully stretched, as a 20 minutes spin-up time results in 5% of transactions exceeded the 10s response time threshold so violating the SLA. The Lunch&COB example is therefore more demanding of elasticity than BigCo and there is little room for an increase in the cloud platform spin-up time before the SLA is violated.

We have demonstrated that it is not just very demanding cloud users such as Yahoo (see quote in article 1) who should be concerned that a 20 minute spin-up time is too long to satisfy their application's elasticity requirements. In part 3 we will explore the elasticity of an even more demanding application, FlashCrowd.