Is your Cloud Elastic Enough? Performance Modeling the Elasticity of Infrastructure as a Service (IaaS) Cloud Applications

Paul Brebner
NICTA/ANU
Canberra
Australia

Paul.Brebner@nicta.com.au

ABSTRACT

Elasticity, the ability to rapidly scale resources up and down on demand, is an essential feature of public cloud platforms. However, it is difficult to understand the elasticity requirements of a given application and workload, and if the elasticity provided by a cloud provider will meet those requirements. We introduce the elasticity mechanisms of a typical Infrastructure as a Service (IaaS) cloud platform (inspired by Amazon EC2). We have enhanced our Service Oriented Performance Modeling method and tool to model and predict the elasticity characteristics of three realistic applications and workloads on this cloud platform. We compare the pay-as-you-go instance costs and end-user response time service level agreements for different elasticity scenarios. The model is also able to predict the elasticity requirements (in terms of the maximum instance spin-up time) for the three applications. We conclude with an analysis of the results.

Categories and Subject Descriptors
C.4 [Performance of Systems]: Modeling Techniques

General Terms
Performance

Keywords
Cloud, IaaS, Elasticity

1. 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 [1].

Not everyone is running an enterprise on the scale of Yahoo, but in order to take advantage of the opportunities provided by cloud computing it is vital to understand the elasticity characteristics of applications, workloads and cloud platforms. Intrinsic to the definition of cloud computing is that resources are dynamically increased and decreased on demand, and that charging is consumption based.

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. There will inevitably be a delay between when resources are requested, and when the 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 the workload; and any quotas or limits imposed by the cloud platform or the contract with them.

On a typical (e.g. Amazon EC2) 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 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, 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. 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.

1.1 Cloud Elasticity Modeling

Our research since 2007 has focused on the performance modeling of Service Oriented Architectures. We have developed a tool and method for SOA performance modeling which has been trialed 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 have previously modeled 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 and evolution [10][11][12][13][14]. From these experiments we know that these three applications are CPU rather than network or database limited. We have also modeled different resourcing models including virtualisation and cloud platforms (e.g. Amazon EC2, Google AppEngine, Microsoft Azure) to predict performance, scalability, cost, and power consumption [8][9]. Other related research has explored related cloud elasticity issues including cloud modeling prediction, and cloud elasticity architecture [2][3][4][5][6][7]. 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. Complex dynamic resourcing models can 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 start and stop times.

We have constructed models of these three applications on a generic elastic cloud platform (inspired by Amazon EC2), focusing solely on the CPU resource requirements, SLAs and costs. Figure 1 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.

1.2 Elasticity Scenarios

We explore the following elasticity scenarios for the examples.

Default (10 minute spin-up time). 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, 1 instance increase/decrease at a time, and a rule suspension time equal to the instance spin-up time, which is set to 10 minutes by default.

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.

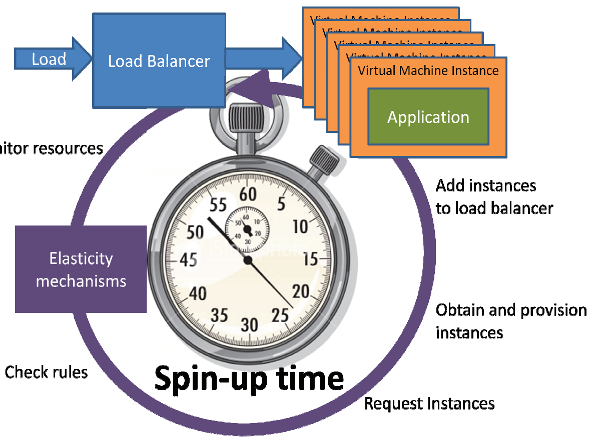

Figure 1 Cloud Elasticity Architecture

Best case elasticity attempts to predict the most elasticity that can be achieved given the constraints of a cloud platform, but assuming zero spin-up time. The minimum rule check period is 1 minute, and that the charging model is hourly. Increase threshold is increased to 95% Utilisation and decrease threshold is increased to 50% Utilisation. Higher thresholds can be used and still satisfy the SLA as less headroom is needed due to instances being available instantly.

To be **perfectly elastic** the resources exactly match the demand (no more or less), 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 cloud platforms are perfectly elastic, we can model it by assuming 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 pro rata to the default cost model. Table 1 summarizes the elasticity scenarios (columns) in terms of the resources (fixed or elastic), spin-up times, and charging settings.

We also predict the **Elasticity Break point.** In order to find the breaking point (where the platform is not elastic enough) the spin-up time is increased until the SLA is violated (or decreased if 10 minutes is too long).

Table 1 Elasticity Scenarios

	Worst	10 min	Best	Perfect
Resources	Fixed	Elastic	Elastic	Elastic
Spin-up	In advance	10 min	0 min	0 min
Charging	Hourly	Hourly	Hourly	Millisecond

2. Examples Elasticity Predictions

2.1 Example 1: BigCo

The workload for the first application, BigCo, runs for 24 hours and represents an observed extreme case of a typical day (Figure 2). During the 24 period the load ranges from a low of 300TPH (Transactions Per Hour) to a high of 27,000 TPH.

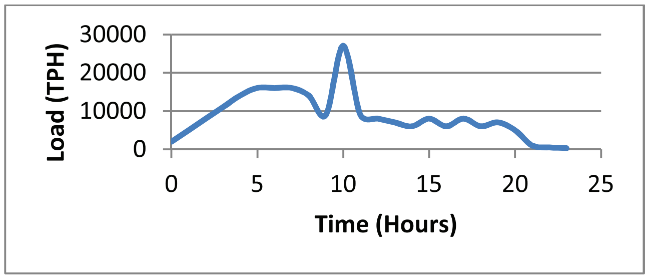

Figure 2 BigCo Workload

All three examples are user facing interactive applications with strict response time requirements of a few seconds at most. The SLA is 99% of transactions taking less than 10 seconds. 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 with a cost of 40 cents an hour or part thereof. We ran the model with the 24 hour workload for all the elasticity scenarios. Figure 5 shows the total running cost, and Figure 6 shows the elasticity breaking point (50 minutes). 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.

2.2 Example 2: Lunch&COB

Our second example is modeled on a government Service Oriented Architecture (SOA) application supporting multiple user types including government, business, and citizens. This example was originally modeled and validated on dedicated hardware, and we have since modeled it on various cloud platforms. It is a distributed application consisting of four distinct application zones which are deployed on separate virtual machines. The workloads occur during 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 Figure 4. The workloads impose different demands on each of the application zones, so the resources for each zone must be scaled at different rates.

The breaking point is 20 minutes when the SLA is violated (Figure 6). The Lunch&COB example is more demanding than BigCo and there is little room for an increase in the cloud platform spin-up time before the SLA is violated. Figure 5 shows the cost for 12 hours for the scenarios. 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 indicates 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.

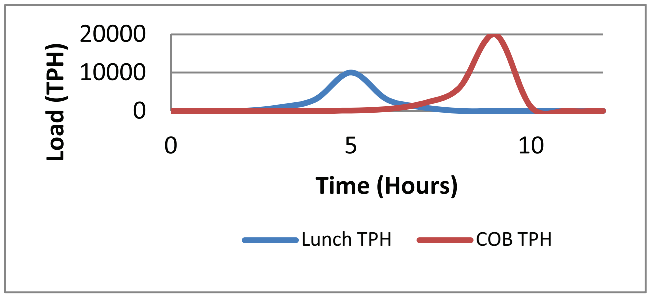

Figure 3 Lunch&COB Workloads

2.3 Example 3: FlashCrowd

The 3rd example is modeled on an emergency web service application which was designed for infrequent use when the main web site was in danger of overloading due to an exceptional load spike caused by an emergency situation. The workload is expected to increase within 30 minutes from the normal peak load of 5TPS to a maximum of 470TPS (a factor of 92), and then drop back to normal very quickly, the load spike lasting for 1 hour. The complete workload runs for 2 hours with over 1.25 million service calls in this period (Figure 4).

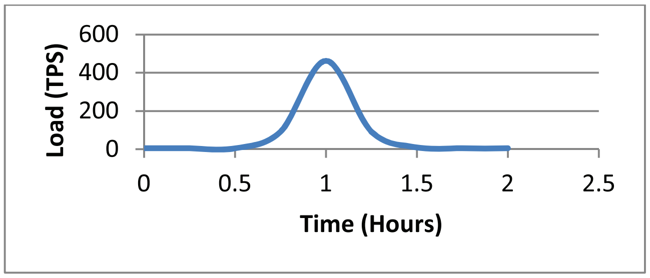

Figure 4 FlashCrowd Workload

FlashCrowd has more demanding elasticity requirements than the preivious examples, as a 10-minute spin-up time cannot provision CPUs fast enough, and the system saturates and catastrophically fails the SLA. In fact, the model predicts that 5 minutes is the longest possible spin-up time. However, in order to achieve the SLA with 5 minutes spin up it is necessary to request *30 instances* at a time, rather than just 1 at a time. Unlike the 10 minute default spin-up time scenario, the worst, perfect and best case elasticity scenarios all satisfy the SLA. Figure 5 shows total costs for 2 hours for scenarios, with the default case using a 5 minute spin-up time. Figure 6 compares the elasticity breaking points for all the examples.

3. Observations

More cores are needed for the peak loads using elasticity compared with fixed resources. For example, for FlashCrowd 356 cores were needed with 5 minute spin-up compared with 272 for fixed. This is significantly higher, and will impose more demand on the shared cloud infrastructure, and potentially more cost for the user, particularly if the elasticity rules result in significantly more instances being requested than actually needed.

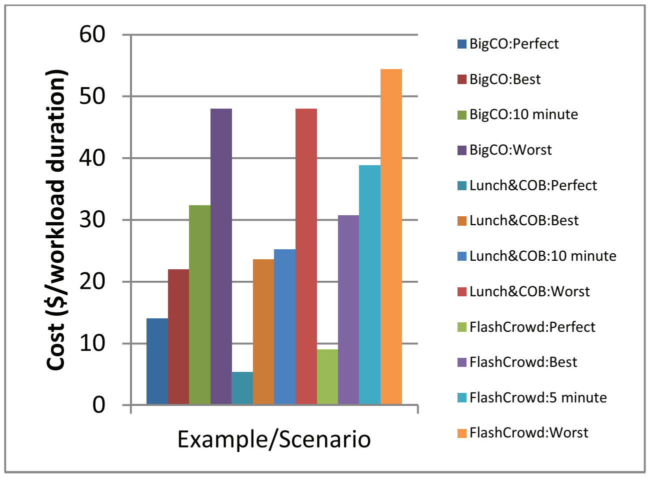

Figure 5 Example/Scenario costs ($/workload duration)

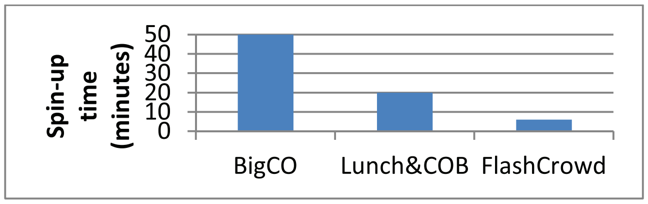

Figure 6 Elasticity Breaking points

The approach of predicting both cost and response time SLAs for applications deployed on cloud platforms allows for both comparison of different elasticity scenarios, and determining if the platform is elastic enough.

Using perfect elasticity as the base line, the best case scenarios were 1.5 to 4.5 times more expensive, the 10/5 minute spin-up scenarios were 2 to 5 times more expensive, and the worst case (fixed) scenarios were 3 to 10 times more expensive. This indicates that dynamic resourcing shows consistent cost advantages over fixed resourcing, realistic (10/5 minute) spin-up times were not significantly more than best case (zero minute), and that reducing spin-up times to zero (best case) without also reducing the granularity of the charging model (perfect case) will not significantly reduce costs. However, all the cloud elasticity options, even the fixed instance scenarios, are relatively cheap.

A 10 minute spin-up time was sufficiently elastic for 2 out of the 3 example applications, but the 3rd example required a spin-up time of 5 minutes and knowledge of the workload so that a larger number of instances could be requested. This suggests that for many applications the current elasticity (spin-up times and elasticity mechanisms) may be adequate, but for more demanding applications improvements in cloud elasticity technology are needed. Computing the SLAs for given elasticity settings enables us to determine if the SLA is satisfied, and to find the longest spin-up time that satisfies the SLA, thereby answering the question "Is a cloud platform elastic enough for a given application and workload?" Modeling elasticity prior to deployment enables cloud platforms to be evaluated before investing the effort porting the application to a selected platform, with increased confidence in cost and performance.

4. ACKNOWLEDGMENTS

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

5. REFERENCES

[1] Julie Bort, "Yahoo builds ultimate private cloud", Network World, July 19, 2011.

[2] Amy Spellmann, Richard Gimarc, Mark Preston, "Leveraging the Cloud for Green IT: Predicting the Energy, Cost and Performance of Cloud Computing" , CMG '09 Conference

[3] Urszula Herman-Izycka, "Flash Crowd Prediction", Master's Thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2006

[4] Nilabja Roy, Abhishek Dubey, Aniruddha Gokhale, "Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting," IEEE 4th International Conference on Cloud Computing 2011, 500-507.

[5] Clovis Chapman, Wolfgang Emmerich, Fermín Galán Márquez, Stuart Clayman, Alex Galis: "Software architecture definition for on-demand cloud provisioning", HPDC '10. ACM, New York, NY, USA, 61-72.

[6] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, Rajkumar Buyya: "CloudSim: a toolkit for modeling and simulation of cloud computing environments", Softw., Pract. Exper. 41(1): 23-50 (2011)

[7] Luis M. Vaquero, Luis Rodero-Merino, and Rajkumar Buyya. "Dynamically scaling applications in the cloud", SIGCOMM Comput. Commun. Rev. Volume 41, Issue 1, 2011, 45-52.

[8] Brebner, P., O'Brien, L, Gray, J., "Performance modeling power consumption and carbon emissions for Server Virtualization of Service Oriented Architectures (SOAs)". EDOCW 2009. 13th. 92-99.

[9] Paul Brebner, Anna Liu: "Performance and Cost Assessment of Cloud Services", ICSOC Workshops 2010: 39-50.

[10] Brebner, P. C. 2008. "Performance modeling for service oriented architectures", ICSE Companion '08. ACM, New York, NY, 953-954.

[11] Paul Brebner, Liam O'Brien, Jon Gray. "Performance Modeling for e-Government Service Oriented Architectures (SOAs)", ASWEC Conference Proceedings (Perth, March, 2008), 130-138.

[12] Brebner, P. 2009. "Service-Oriented Performance Modeling the MULE Enterprise Service Bus (ESB) Loan Broker Application", SEAA 2009. IEEE Computer Society, Washington, DC, 404-411.

[13] Paul Brebner, Liam O'Brien, Jon Gray: "Performance modeling evolving Enterprise Service Oriented Architectures", WICSA/ECSA 2009: 71-80.

[14] Paul C. Brebner. "Real-world performance modeling of enterprise service oriented architectures: delivering business value with complexity and constraints". ICPE '11. ACM, New York, NY, USA, 85-96.