Auto-Scaling in the Cloud: Evaluating a Control Based Technique

Brian C. Carroll (Student ID: 13212607)

Department of Computing

DCU, Dublin.
Brian.Carroll26@mail.dcu.ie

ABSTRACT

One of the major benefits of cloud computing is elasticity, that is the ability to provision resources based on workload.
Scaling up resources when load is high to meet service level objectives. Scaling down when resources are underutilised thereby saving on cloud provider costs.
The auto-scaling capabilities offered by cloud providers are in the majority quantitative rule based algorithms [1], dependant on Quality of Service (QoS) and the monitoring thereof. For example, an additional virtual machine (VM) may be introduced to handle increased load when an existing VM CPU metric reaches an 80% threshold, measured over a set period.
In this paper I will provide background on existing scaling techniques and evaluate a fuzzy logic based auto-scaling system which has the advantage over current commercial offerings in that it utilises a qualitative approach to auto-scaling.

1. INTRODUCTION

Cloud elasticity is beneficial for both consumer and provider alike. For the consumer, computing resources can be ramped up on demand or contracted when deemed unnecessary. For the provider, they can maximise operating costs and profit by making efficient use of hardware.
Elasticity should incorporate three design principles [1], 1) Scalability, the requirement of the system to cope with varying workload. 2) Cost efficiency – paying only for what is necessary. 3) Time efficiency – adding/releasing resources as efficiently possible.
Utilisation of cloud computing resources is constructed around a pay-as-you-go model. Costs are mostly based on resource usage hence the importance of using only what you need, akin to turning off lights when not in use.
Moving services to the cloud could be considered as outsourcing and the client must factor in and control any extraneous or hidden costs incurred to monetise the advantages of cloud computing.
Irrespective of economic cost, scaling up computing resources is critical for the cloud services client if they are to maintain their existing customer base and accommodate unexpected traffic.
Auto-scaling issues in most cases can be remediated by careful capacity planning based on historic trending or appropriate communication within the business e.g. marketing informing IT that a web promotion is on the horizon.
Auto-scaling functionality provided by the majority of cloud providers implement a reactive rule-based technique for adding capacity. Although straight forward in theory, it may not be the most effective proposition for the client.
Threshold based auto-scaling poses the following challenges to ensure scaling actions are effective and no overspend is occurred

- The technique is reactive so instantiating a VM when a threshold is reached is not instantaneous. Meanwhile users experience poor application performance.
- The application owner must set thresholds based on a thorough understanding of the application. The application's behaviour may change over time which could have a knock on effect on existing elasticity rules.
- Sensory noise – where the probe experiences varying metrics which could have adverse effects e.g. invoking a scaling action on a spurious measurement

This paper will detail a test environment for evaluating elasticity controllers and will integrate the RobusT2J fuzzy logic controller which uses a feedback control loop to overcome the above challenges [2]
Section 2 presents a brief overview of the various auto-scaling techniques and in particular the rule based strategies used by the majority of cloud service providers.
Section 3 explores the predictive fuzzy logic controller and how it improves on threshold-based rules.
Section 4 provides implementation details for the Cloudstone environment which was used to test the controller.
Section 5 evaluates results and looks at the ADVISE framework which can be used for evaluating cloud service elasticity behaviour.

2. AUTO-SCALING TECHNIQUES

Moore et al. [2] defined auto-scaling solutions as being either reactive: an event occurs which causes the release or addition of a server node or predictive which involves a time series forecaster using historical data. Predictive is deigned to the more accurate and as it takes most providers ten minutes to instantiate a VM, the more efficient.
Loridon-Botrán et al. [1] defined a third model which is a hybrid of reactive and predictive and classified auto-scaling techniques as broadly falling into five categories which are as follows:

1. Static Threshold-based Rules

Typically there are two rules, once for scaling up and another for scaling down. These rules usually have a time variable e.g. bring up an instance if the %processor time > 85 over a 15 minute period. There is usually a cooling down period associated with a rule (post invocation) where a node is not shut down for a defined period.
This is a reactive strategy, an instance is added only when a flag (threshold reached) is raised.

2. Reinforcement Learning

Auto-scaling based on reinforcement learning is a predictive approach to auto-scaling. VM instantiation is predicted via learned behaviour. It makes decisions based on interaction between the auto-scaling agent and the scalable application

3. Queuing Theory

Queuing theory can be utilised to add capacity by analysing and making decisions based on a queue i.e. requests queued at the load balancer.

4. Control Theory

Control systems use a feedback loop by modifying the controller input to influence the normative output. Control theory systems are mostly reactive however they been combined with proactive systems.

5. Time-Series Analysis

Time-series analysis uses historical data to predict future usage

2.1 Cloud provider elasticity strategies

Rule based strategies are employed by the majority of cloud providers. They are built around the monitoring of performance metrics and used to make decisions on acquiring or reducing resource capacity. They are mostly straight forward to implement [3] from a user perspective but some providers have been slow to introduce this functionality, as seen below. An example of the metrics measured to determine elasticity could be as follows [1]:

- Hardware: CPU usage, memory usage, disk I/O, network traffic etc.
- Database services: Active threads, database locks etc.
- OS Processes: Queue length, paging faults etc.
- Web statistics: Response times, number of requests etc.
- Load Balancer: Denied requests, errors etc.

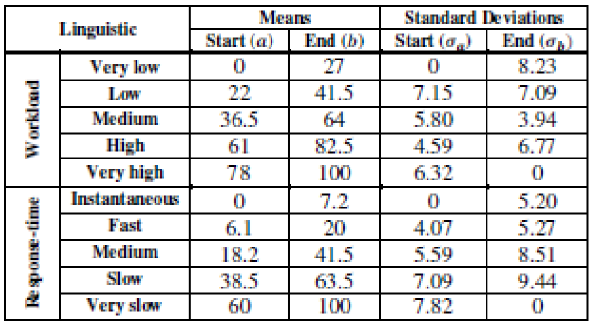
Rackspace recently announced auto-scaling technology [3]. It is implemented via event-based (rule based) or schedule-based technology.
Similarly, the Google cloud platform released documentation on their newly released auto-scaling (Sept 2013) framework [4];
"Each Compute Engine instance runs a local agent that exposes an endpoint for querying current CPU and memory load, which the orchestrator periodically retrieves from all instances…" It makes the current CPU and memory load accessible to the orchestrator, which periodically retrieves this information from all instances. Our "orchestrated" application is the actual business application that the orchestrator watches and manages by scaling VM instances out and back.
Both Amazon EC2 [5] and Microsoft Azure [6] have comparable offerings.

2.2 Rightscale

Rightscale has a novel approach to auto-scaling in that it uses a democratic voting system [8].It is a third party cloud management tool and is delivered via software as a service (SaaS) and is vendor agnostic. This has the advantage of abstracting the management layer from the vendor thereby avoiding vendor lock-in. It manages resources, automates provisioning, configuration and governance across the cloud infrastructure. It works with multiple platforms such as Openstack, AWS EC2, Microsoft Azure, and Cloudstack.
A server array is created and within that array each server has a vote. If a threshold alert is generated for scaling on a particular server in the array, it will cast a vote for scaling up/down. However if it is the only server to cast a vote, the status quo is maintained.
If a quorum of servers vote to scale up or down, an action is then taken.
An advantage of this system is it prevents single server nodes from bringing up an instance and unlike Cloudwatch, the EC2 auto-scaling tool, it differentiates between single server nodes issue and the cluster as a whole.

3. Robus2TScale CONTROLLER

The RobusT2Scale is a hybrid controller which uses fuzzy logic to implement qualitative specifications of elasticity rules. Fuzzy logic systems can better deal with uncertainties due to inexact qualitative values e.g. If response time is *slow* and workload is *high* then scale up an instance.
The controller contains a fuzzy knowledge base or rule base which details information on how to scale the system in relation to assigning qualitative values to linguistic variables (e.g. slow, high). This rule base is based on domain expertise, a survey of 10 cloud computing experts.



Linguistic		Means		Standard Deviations	
		Start (a)	End (b)	Start (σ_a)	End (σ_b)
Workload	Very low	0	27	0	8.23
	Low	22	41.5	7.15	7.09
	Medium	36.5	64	5.80	3.94
	High	61	82.5	4.59	6.77
	Very high	78	100	6.32	0
Response-time	Instantaneous	0	7.2	0	5.20
	Fast	6.1	20	4.07	5.27
	Medium	18.2	41.5	5.59	8.51
	Slow	38.5	63.5	7.09	9.44
	Very slow	60	100	7.82	0

Fig 1. Robus2TScale Scaling logic based on domain expertise

The controller outputs values between -2 and +2 which are used by the actuator to scale virtual machines accordingly.

4. 4. IMPLEMENTATION

4.1 Architecture

To evaluate the RobusT2J controller in a real world environment, the Cloudstone application was used [9]. Cloudstone is a web 2.0 representative application with a Markov-based distributed workload generator with data collection tools.
Cloudstone follows the three-tier web architecture (see fig 1)

- A stateless web tier – Nginx web server
- A stateless application tier – Olio social calendar
- A persistence tier – MySQL and NFS data store

The Cloudstone suite contains the Faban harness driver which benchmarks the Olio Web application.
Scaling is horizontal and performed at the Web/Application service tier. VM's are instantiated depending on controller output.

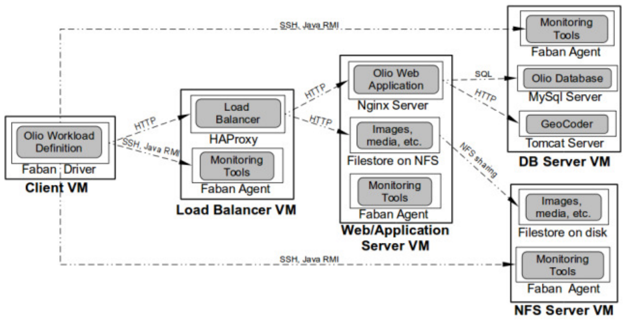


Fig 2. Cloudstone 3-Tier Architecture

4.2 Infrastructure

The test environment was hosted on Amazon AWS, It was important to size the instances correctly to negate bottlenecks elsewhere in the system as scaling was only performed at the Application/Web server tier.

- Web/Application servers - EC2 t2.micro (1 vCPU, 1GB Mem) and the
- Load generator - EC2 m3.large instance (4 vCPU, 8 GB Mem)
- NFS Server – EC2 t2.medium (2 vCPU 4GB Mem)

- Load Balancer - EC2 t2.medium (2 vCPU 4GB Mem)
- Actuator – EC2 t2.micro (1vCPU, 1GB Mem)

Application setup was performed using shell scripts to simplify software installation.
The system incorporates the following components each of which is installed on its own EC2 instance.

- Olio (application server) – A PHP web 2.0 social-events application
- Faban (client) – Load generator, benchmarking framework and reporting tool.
- MySQL (database backend)
- NFS Server – hosts digital images (JPEG)
- Haproxy Load Balancer – Load balances the Olio Application/Web servers
- Actuator – hosts the RobustT2 controller and code for controlling scaling actions

Faban – The Client

There are two main components to Faban [13]:

The Faban driver framework –An API-based framework and component model to help develop new benchmarks rapidly. The driver framework controls the lifecycle of the benchmark run as well as the stochastic model used to simulate users.

- It provides a web interface to manage the benchmark runs and view results
- It automates the collection of system and application statistics.
- It hosts the logs/runtime statistics centrally
- It allows for benchmark queues
- It provides analysis tools for graphing and comparing results

Faban Harness is a tool to automate running of server benchmarks. It also serves as a container to host benchmarks allowing new benchmarks to be deployed in a rapid manner.

- It is driven by declarative workload models
- It provides a workload component model with high level-API's and abstractions
- Time sync between all servers in the software under test
- Provides for throughput-orientated measurements with response time targets

The Load Balancer

Haproxy is used as the load balancer to accommodate session affinity
The Olio application is predominantly stateful and session data is stored in the Web/Application server memory for each user. To alleviate this issue we configure cookie based load balancing whereby the server installs an identifying cookie in the client.

```
cookie PHPSESSID prefix indirect nocache
```

The Database Server

The database backend is MySql It also hosts a geocoding application required for coordinate requests for Olio calendar events.

Olio – The Web/Application Server

The Olio application is a Web 2.0 social calendar [14] which allow users to register a personal profile, create and register to attend events and extend friendship requests.
It mimics a reasonably complex web application with a database backend

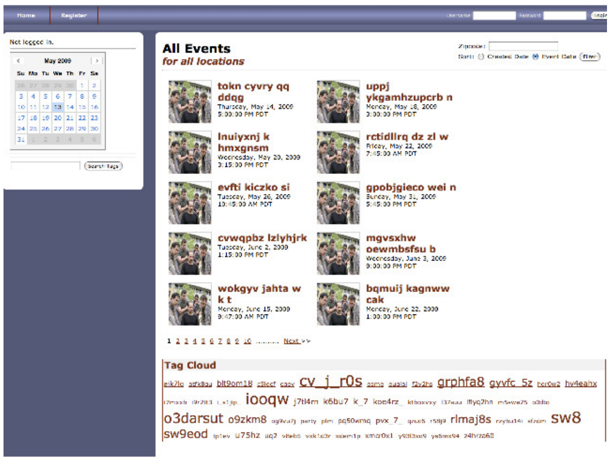


Fig 3. Olio Web Application

The Actuator

The actuator is written in Java. It is made up of four classes which provide the actuator and monitoring sensor logic. The actuator only scales the Web/Application tier.

- A service class used to call the scaling classes
- A addInstance class
- A terminateInstance class
- A parseStats class – parses monitoring data from Faban logs

The actuator was written with the following constraints

- There is never less than 1 Application/Web server and never more than 3.
- If a server is added there is a 90 second delay before it can be terminated or another server added i.e. a cool down period
- If a server is terminated, it will be another 5 minutes before another server can be terminated
- 3 consistent readings must be received prior to a scaling action. This negates the chance of scaling due to spikes.
- Faban stats (workload and response times) are generated and processed at 10 second intervals.

Note: AWS EC2 instances are charged at an hourly rate i.e. if an EC2 instance is shut down 5 minutes after booting, the customer is still charged for the hour.

Integrating RobustT2J

The RobustT2J controller was developed in Mathlab and exported out as a Java class. It required integration into the closed feedback control loop which is made up of the monitoring sensor along with the actuator.

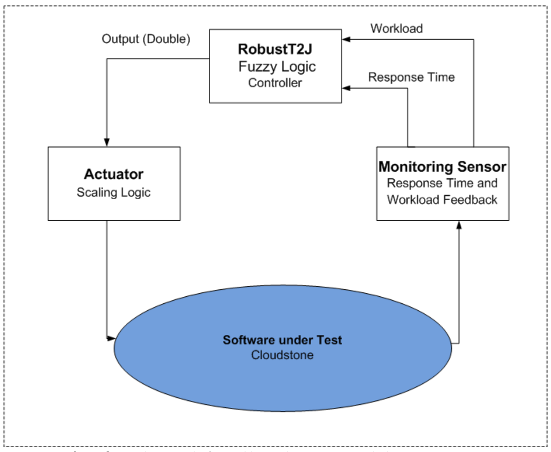


Fig 4. Closed feedback control loop

The controller requires two inputs:

- Workload - number of concurrent users using the system
- Response time – 90th percentile Average of the following operations: Homepage, Login, TagSearch, EventDetail, AddPerson, AddEvent

Both inputs were parsed from the log file which is generated for each Faban run and converted to a percentage based on a service level objective for response time (e.g. 3 seconds to add a person, 2 seconds to retrieve an event detail.) and an approximate number for the maximum number of concurrent users in the load testing (2000 in this environment).
Below is an example of workload and response times extracted from the Faban log file followed by an explanation.

```
Operation response times (HomePage, Login, TagSearch,
EventDetail,PersonDetail, AddPerson, AddEvent)
0.22  0.08  0.36  0.18  0.4  0.3  0.8
Average response time of 7 values (scaled in percent) is
13.37142857142857
Users: 400 Percent: 20%
Robus Returned Value: -1.0029
Number of Instances to Scale: -1
```

Average response time converted to a percentage of average acceptable response time > 13%
User Workload converted to a percentage of max concurrent users> 20%
Output from Robus controller > -1 (rounded)
Scaling action > scale back one instance

RobusT2J processes these values and outputs a number of instances to be scaled which is used by the Actuator to scale up/down instances.
The actuator code also updates the Haproxy configuration file, adding or removing a server to the cluster and restarting the service thereafter.

5. EXPERIMENTAL EVALUATION

The RobusT2J controller was tested with a varying workload over a 25 minute period. Response times will degrade on 3 servers around the 1100 user mark.

Workload applied

400 simulated users for 5 mins
800 simulated users for 5 mins
300 simulated users for 5 mins
1000 simulated users for 5 mins
300 simulated users for 5 mins

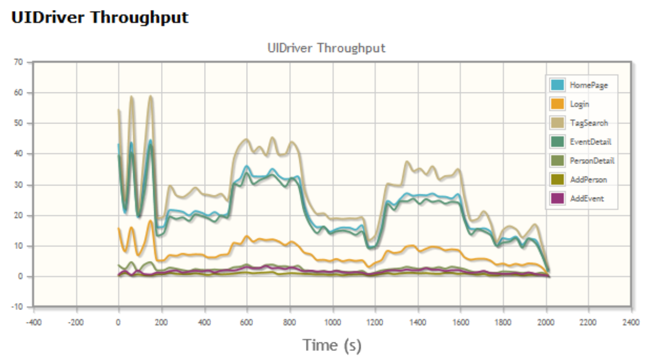


Fig 5. Application Throughput

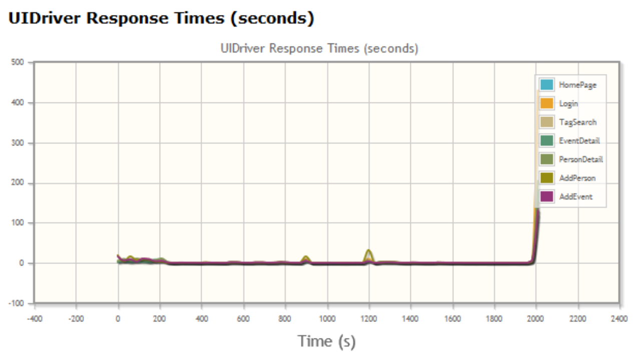


Fig 6. Application Response Times

In Figure 5 we can see the varying throughput
As we can see from fig 6 response times remained consistent with some small exceptions where a new instance was scaled out or released. However the variation is negligible and never exceeds 5 seconds.

5.1 ADVISE Framework

Comparing the benefits of one controller over another is not a simple task. Controllers can be compared on price, see appendix 1, the cost of providing a service without overpaying for hosting it. Efficiency, how many VM's are required while adhering to service level objectives such as acceptable response times.

ADVISE (evAluating clouD serVIce elaSticity bEhavior) [16]
Proposes a solution to the following questions "Which elasticity control processes are the most appropriate for cloud service in a particular situation at runtime? Both cloud customers and providers can benefit from insightful information such as how the addition of a new instance to a cloud service will affect the throughput of the overall deployment and individually on each part of the cloud service. Thus, cloud service elasticity behaviour knowledge under various controls and workloads is of paramount importance to elasticity controllers for improving runtime decision making"
The framework estimates cloud service elasticity behaviour by utilizing different types of information, such as service structure, deployment strategies, and underlying infrastructure dynamics, when applying different external stimuli (e.g., elasticity control processes, scale an instance).

6. CONCLUSION

In this paper we examined different types of auto-scaling techniques and implemented a test environment to allow comparison of different controller mechanisms.
The industry standard for elasticity controller methodologies is threshold based rules however this may not be the most preferable method but will require a technique for comparison which is no easy matter however the ADVISE framework may be of interest here. We also integrated a fuzzy logic controller which mitigates some of the issues associated with rule based elasticity controllers such as sensory noise, a reactive approach based on thresholds and a requirement for a deep knowledge of the application under test.

7. REFERENCES

[1] T. Lorido-Botrán, J. Miguel-Alonso and J. A. Lozano. Auto-scaling Techniques for Elastic Applications in Cloud Environments. *Technical Report EHU-KAT-IK-09-12, University of the Basque Country*, Sept. 2012.

[2] Jamshidi, Pooyan and Ahmad, Aakash and Pahl, Claus (2014) *Autonomic resource provisioning for cloud-based s oftware.* In: 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems SEAMS 2014, 2-3 June 2014, Hyderabad, India. ISBN 978-1-4503-2864-7

[3] Moore, Laura R., Bean, Kathryn, Ellahi, Tariq Transforming Reactive Auto-scaling into Proactive Auto-scaling. SAP Next Business and Technology, SAP (UK) Ltd, Belfast, UK.

[4] Add an auto-scaling group and policy for Amazon EC2 machines http://www.techrepublic.com/blog/the-enterprise-cloud/add-an-auto-scaling-group-and-policy-for-amazon-ec2-machines/5685/ accessed 27/11/13

[5] Auto scaling on the Google Cloud Platform https://cloud.google.com/resources/articles/auto-scaling-on-the-google-cloud-platform accessed 28/11/13

[6] AWS – Auto Scaling Getting Started Guide http://awsdocs.s3.amazonaws.com/AutoScaling/latest/as-gsg.pdf accessed 26/10/13

[7] How to Scale an Application http://www.windowsazure.com/en-us/manage/services/cloud-services/how-to-scale-a-cloud-service/ accessed 26/10/13

[8]Understanding the voting process -Rightscale http://support.rightscale.com/12-Guides/RightScale_101/System_Architecture/RightScale_Alert_System/Alerts_based_on_Voting_Tags/Understanding_the_Voting_Process accessed 27/10/13

[9] Top cloud IaaS providers compared – TechRepublic http://www.techrepublic.com/blog/the-enterprise-cloud/top-cloud-iaas-providers-compared/ accessed 27.11.13

[10] Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware
Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi.
In the 17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), March 2012.

[11] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, A. Fox and D. Patterson – Cloudstone: Multi-platform, multi-language benchmark and measurement tools for Web 2.0 – Cloud Computing and its Applications CCA-08, 2008

[12]Cloudsuite Install http://parsa.epfl.ch/cloudsuite/web.html accessed 12/07/14

[13] Faban Workload creation and testing framework http://faban.org/

[14] Olio Social Calendar https://blogs.oracle.com/klc/entry/olio_java_ee_source_code accessed 13/07/14

[15]Automated CloudStone Setup in Ubuntu VMs https://nikolaygrozev.wordpress.com/2014/05/10/automated-cloudstone-setup-in-ubuntu-vms/ accessed 06/07/14

[16] G. Copil, D.Trihinas, H.Truong, D Moldovan,G. Pallis, S. Dustdar,M.Dikaiakos - ADVISE a Framework for Evaluating Cloud Service Elasticity Behavior, Distributed Systems Group, Vienna University of Technology

APPENDIX

Cloud provider costs

To evaluate the effectiveness of auto-scaling techniques we need to consider the pricing plans of the different cloud vendors and the savings made by adopting non-vendor elasticity controllers.
The prices in Fig1 are based on the AWS small instance (most vendors have similar offerings) with 1.7GB of RAM and 1 CPU running Windows.
EC2 spot prices can possibly be disregarded as they are based on auction bids.

Provider	Variety of Pricing Plans	Costs($)
Established		
Amazon (EC2)	Pay-as-you-go, reserved, spot	66.43
Rackspace	Pay-as-you-go	116.80
GoGrid	Pay-as-you-go, month, semester, year	116.80
Microsoft	Pay-as-you-go, semester, year	65.70
Terremark	Pay-as-you-go	138.90
AT&T	Pay-as-you-go	121.30
Google	Pay-as-you-go	42.42
OpSource	Pay-as-you-go, monthly	95.63
Softlayer	Pay-as-you-go, monthly	182.50
HP	Pay-as-you-go	87.60
Upstarts		
BitRefinery	Monthly	69.90
Lunacloud	Pay-as-you-go	49.92
Nephoscale	Pay-as-you-go, membership	73.00
Tier3	Pay-as-you-go	109.50

Fig 1. Cloud providers pricing comparison August 27th 2013 [8]