Cloud Performance Benchmark Series

Amazon Elastic Load Balancing (ELB)

Md. Borhan Uddin
Bo He
Radu Sion

ver. 0.5b

1. Overview

Experiments were performed to benchmark the Amazon Elastic Load Balancing (ELB) service. ELB promises to offer fault tolerance of web applications and automatic scaling on Amazon Elastic Compute Cloud (EC2) instances.

ELB distributes incoming application traffic across multiple Amazon EC2 instances in a single or multiple Availability Zone(s). ELB scales its request handling capacity in response to incoming application traffic. It detects unhealthy instances and automatically redistributes traffic to healthy instances.

ELB was benchmarked with HTTP LAMP (Linux, Apache, MySQL, PHP) backend servers using the HTTP performance benchmarking tool httperf. Httperf "is a tool for measuring web server performance. It provides a flexible facility for generating various HTTP workloads and for measuring server performance. The focus of httperf is on providing a robust, high-performance tool that facilitates the construction of both micro- and macro-level benchmarks." (http://code.google.com/p/httperf/)

Four main different aspects of ELB were evaluated:

+ Gain or loss (bottleneck) in throughput due to the deployment of ELB.
+ Responsiveness of ELB to capacity fluctuations in the backend servers (agility)
+ Fairness of ELB traffic distribution among homogeneous backend servers
+ Convergence of ELB in case of failure or (re)start of back-end servers

2. Setup

ELB can be deployed on two different types of Amazon EC2 instances: intra-zone (all EC2 servers, probing machines and ELB are in single "availability zone", Figure 1a), inter-zone (EC2 servers, probing machines and ELB are in multiple availability zones, Figure 1b). ELB can't be configured to operate across multiple EC2 regions, however. It can be configured only as single or multiple availability zone within a given region.

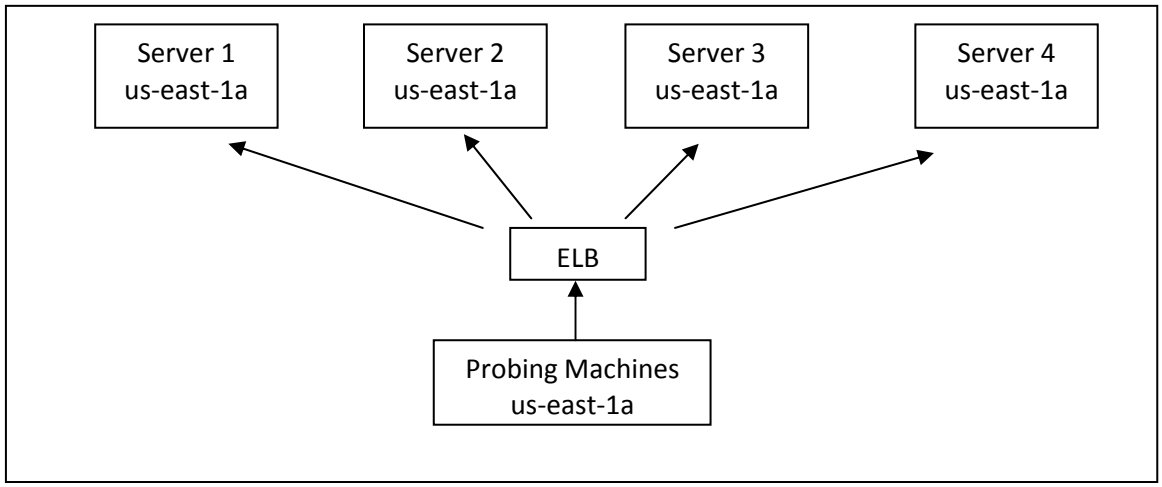

Figure 1a: Intra-zone experimental setup

Four different types of instances were used as back-end servers: small, medium, large and extra-large. Instances varied according to their memory, CPU, I/O and platform configurations.

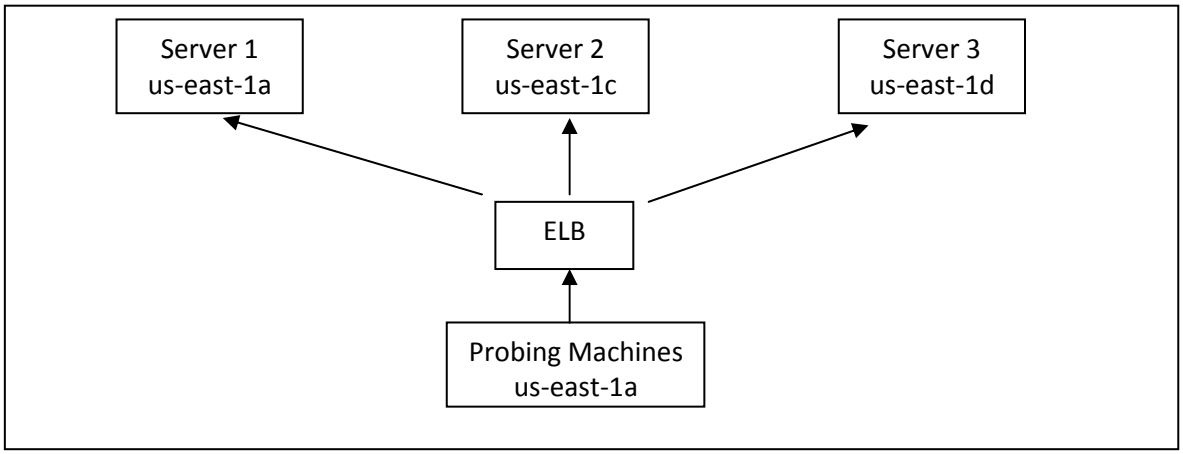

Figure 1b: Inter-zone experimental setup

In an initial setting, LAMP HTTP servers running on medium EC2 instances were registered with an ELB. Then, the ELB was probed from extra-large EC2 instances running Httperf and several custom perl scripts aggregating the results.

The servers' configurations were identical in terms of memory, platform, and allocated EC2 Compute Units. As a reminder, 1 ECU aims to approximate an Intel 1.0-1.2 GHz 2007 Opteron or 2007 Xeon CPU. Processing power is structured as a set of multiple virtual cores (VC). A VC in turn is considered as a collection of ECUs (and has an associated number of ECUs/VC). Thus, the total number of ECU compute units can be computed by

knowing the number of virtual cores and the number of ECUs per virtual core. Table 1 shows the configurations of the EC2 instances:

Instance Size	Memory (GB)	Total ECU (=VC* ECU/VC)	Platform	AMI Id
Small	1.7	1 (=1*1)	Fedora 32-bit	ami-2cb05345
Medium	1.7	5 (=2*2.5)	Fedora 32-bit	ami-2cb05345
Large	7.5	4 (=2*2)	Fedora 64-bit	ami-86db39ef
Extra Large	15	8 (=4*2)	Fedora 64-bit	ami-86db39ef

Table 1: Configurations of the EC2 Instances. (ECU = EC2 Compute Unit, VC= Virtual Core)

ELB configuration parameters are not publicly alterable. The following are some of the interesting ones:

Health Check Interval - time interval that ELB probes the servers to check the health of servers.

Unhealthy Threshold - maximum number of consecutive health check failures allowed, after which the server will be claimed as unhealthy. ELB no longer routes traffic to unhealthy Amazon EC2 instances and instead re-distributes the load across the remaining healthy ones.

Healthy Threshold - minimum number of consecutive successful health checks for a server to be deemed healthy again, after which ELB will start routing traffic to it.

Response Timeout - maximum time allowed for a server to respond to a health check request. If timeout occurs the health check is counted as a failure.

Protocol - health checking can use different protocols (e.g., HTTP, HTTPS, TCP, UDP).

Port number – server port number pinged by health checker.

Path - Server-side target URL for health checks using HTTP.

The default values of these parameters are illustrated in Table 2.

Name	Health Check Interval (Seconds)	Unhealthy Threshold	Healthy Threshold	Response Timeout (Seconds)	Protocol	Port	Path
ELB	30	2	5	5	HTTP	80	/index.php

Table 2: Configurations of Elastic Load Balancing (ELB) service

Four main benchmarking categories were performed: Gain/Loss, Agility/Awareness, Fairness and Convergence.

2.1. Setup for Gain/Loss (bottleneck) testing due to deployment of ELB

a. Throughput

To measure whether deployment of ELB causes any gain (boost-up) or loss (bottleneck) in throughput, we ran the same load with and without deployment of ELB both in intra-zone (Figure 1a) and inter-zone (Figure 1b) setups.

With ELB: For both intra-zone and inter-zone setup, four EC2 medium servers were linked to one ELB, and probed with batteries of requests ranging from 1 to 2^{16} HTTP requests per second, through the ELB. Each battery was run for 3 minutes. To identify any bottlenecks introduced by ELB, logging of requests and responses was done in both the probing machines and in individual servers.

Without ELB: For both intra-zone and inter-zone setup, four medium servers were probed one at a time at rate increasing from 1 to 2^{16} HTTP requests per second. Each battery of requests was run for 3 minutes.

b. Cost:

We thought it would be interesting, in addition to identifying throughput bottlenecks, to also compute the additional ELB-introduced dollar cost per HTTP transaction/connection. Amazon pricing at the time of the experimentation is included here for reference. Figure 2a shows an example hourly pricing chart of Amazon EC2 On-Demand instances. For running EC2-hosted HTTP servers, pricing of both compute instances and the cost of in- and e-gres data transfers need to be considered.

US – N. Virginia	US – N. California	EU – Ireland	APAC – Singapore
Standard On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**	
Small (Default)	$0.085 per hour	$0.12 per hour	
Large	$0.34 per hour	$0.48 per hour	
Extra Large	$0.68 per hour	$0.96 per hour	
High-Memory On-Demand Instances			
Extra Large	$0.50 per hour	$0.62 per hour	
Double Extra Large	$1.20 per hour	$1.44 per hour	
Quadruple Extra Large	$2.40 per hour	$2.88 per hour	
High-CPU On-Demand Instances			
Medium	$0.17 per hour	$0.29 per hour	
Extra Large	$0.68 per hour	$1.16 per hour	

Figure 2a: Pricing (Hourly Rate) of Amazon EC2 On-demand machines (amazon.com)

Furthermore, running HTTP with with ELB incurs two more cost factors, namely the ELB cost and the ELB-related data transfer cost. Figure 2b shows an example hourly pricing chart of Amazon ELB.

Elastic Load Balancing

US – N. Virginia | US – N. California | EU – Ireland | APAC – Singapore

- $0.025 per Elastic Load Balancer-hour (or partial hour)
- $0.008 per GB of data processed by an Elastic Load Balancer

Figure 2b: Pricing (Hourly Rate and Data Processed Rate) of Amazon ELB (amazon.com)

The cost per connection in US microCents is as follows:

Cost per connection without ELB (in microCent) $= (\frac{\text{EC2 Server Hourly rate}}{(\text{Throughput})*3600} +$ (EC2 Server Data Transfer price rate*Amount of Data Transferred per connection))*(10^8)

Cost per connection with ELB (in microCent) $= (\frac{\text{EC2 Server Hourly rate+ELB hourly rate}}{(\text{Throughput})*3600} +$ (EC2 Server Data Transfer price rate + ELB Data Processing price rate)*Amount of Data Transferred per connection)*(10^8)

2.2. Setup for Agility/Awareness Testing

The idea here is to test whether ELB correctly adapts to the capacity of the back-ends. The setup was as follows: the set of back-ends consisted of a small, medium, large and

extra-large HTTP LAMP servers in us-east-1a, all linked to an ELB. The probing machines were also in us-east-1a, sending requests to the ELB at a rate increasing from 1 to 2^{16} requests per seconds. Each battery of requests was run for 3 minutes.

2.3. Setup for Fairness Testing

Similarly, we checked whether ELB is fairly routs traffic to similar back-ends. This was done in two different setups: intra-zone (four HTTP servers and probing machines were located in the same EC2 zone and region, us-east-1a, as in Figure 1a), and inter-zone (one HTTP server in each of us-east-1a, us-east-1c, and us-east-1d, probing machines in us-east-1a, all under the same region, as in Figure 1b).

2.4. Setup for Convergence Testing

Finally, experiments were performed to evaluate the speed and accuracy of ELB health checks. This was achieved partly by programmatically turning back-ends on/off through a set of perl scripts controlled by the probing machines. The identical setting to the one for the intra- zone fairness benchmark was applied. Here both HTTP servers and probing machines were located in the us-east-1a zone.

3. Results

3.1. ELB bottlenecks

a. Throughput

No bottleneck has been observed due to ELB deployment at low to moderate loads. However, **at higher loads, ELB causes significant throughput bottlenecks**. Figure 3a and 3b depict this phenomenon for intra-zone and inter-zone setups respectively.

In the intra-zone setup (Figure 3a) up to a certain load (256 connection requests per second), the observed aggregate throughput of the 4 individual servers without ELB is identical to the throughput of ELB-enabled servers.

Beyond 256 connection requests per second, ELB throughput quickly degrades to under 30% of the non-ELB setup. Once there, overall throughput remains relatively stable for both setups in the intra-zone setup.

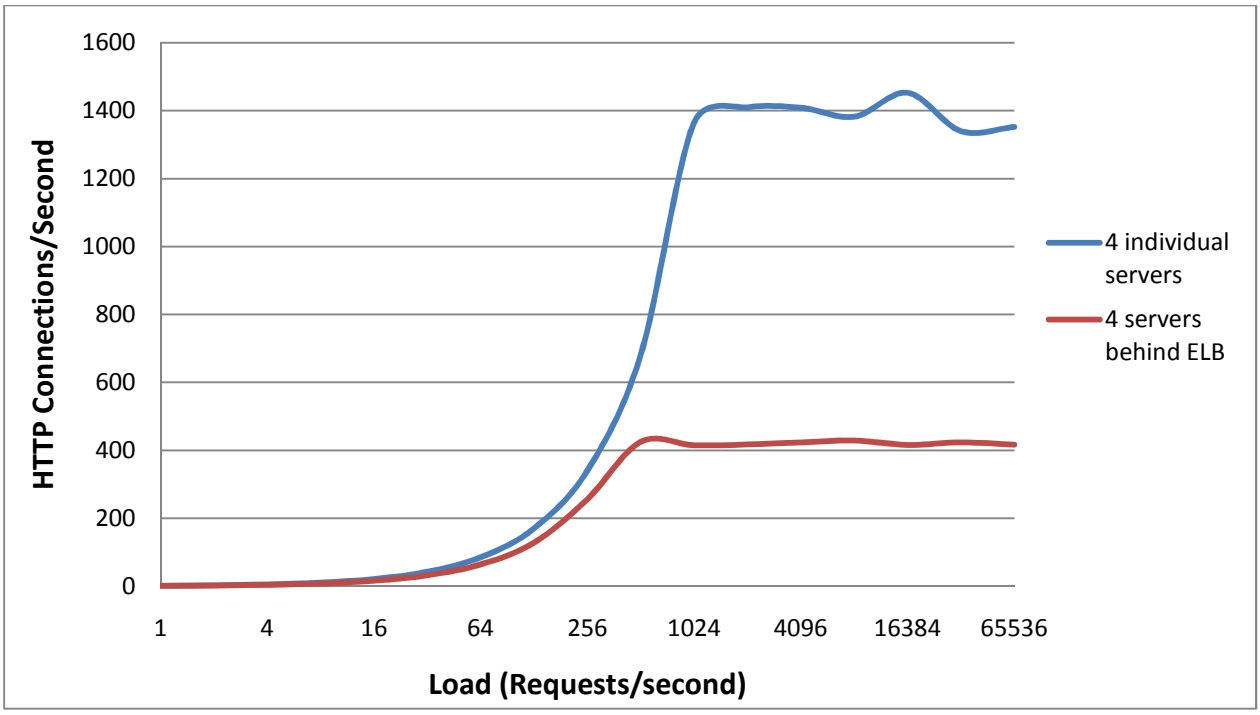

Figure 3a: Gain/Loss due to ELB in intra-zone setup (Indivudual Servers vs. ELB+Servers)

Figure 3b depicts the same phenomenon for inter-zone setup. This time for loads up to 512 connection requests per second, the aggregate throughput of the 3 individual servers without ELB is the same as the ELB throughput. Beyond that limit, ELB throughput drops often to under 25% of the individual servers' aggregate throughput. Moreover, this ELB throughput shows relative instability.

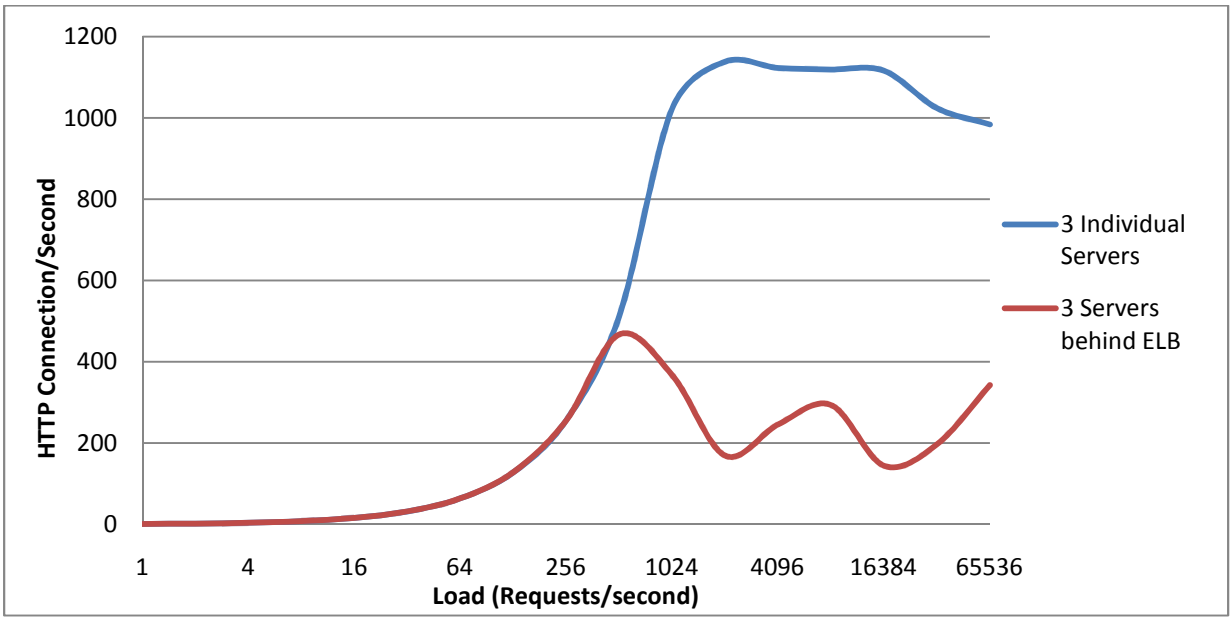

Figure 3b: Gain/Loss due to ELB in inter-zone setup (Servers without ELB vs. Servers with ELB)

Thus, for both intra-zone and inter-zone setups, ELB seems to introduce a significant bottleneck. The bottleneck seems to be more severe in inter-zone setups. The throughput instability at higher load seems to be one of the most major cons in inter-zone setup with ELB.

b. Cost:

Overall per-connection cost for both ELB and non-ELB settings in intra-zone and inter-zone setups is illustrated in microcents in Figures 4a and 4b. In both intra-zone and inter-zone setups, ELB costs always more. At higher loads, the cost per connection seems to be stable for both intra-zone and inter-zone setup with and without ELB. On average, inter-zone setups' per-connection costs are higher than intra-zone setups'.

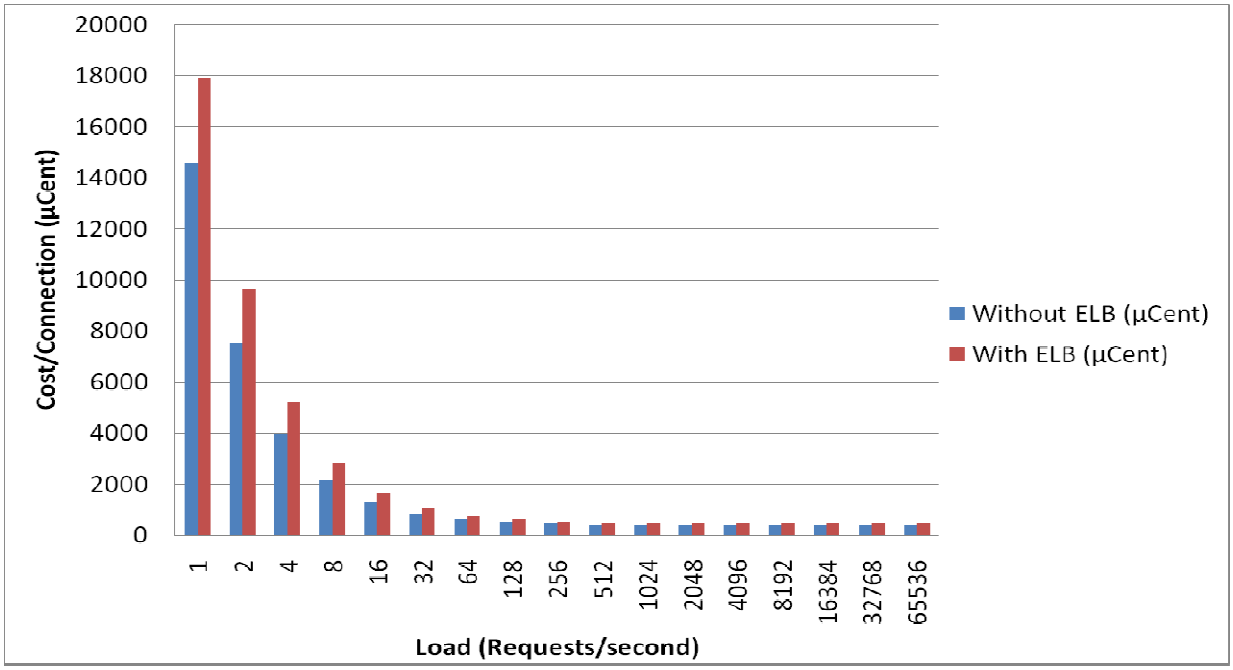

Figure 4a: Cost/connection in intra-zone setup

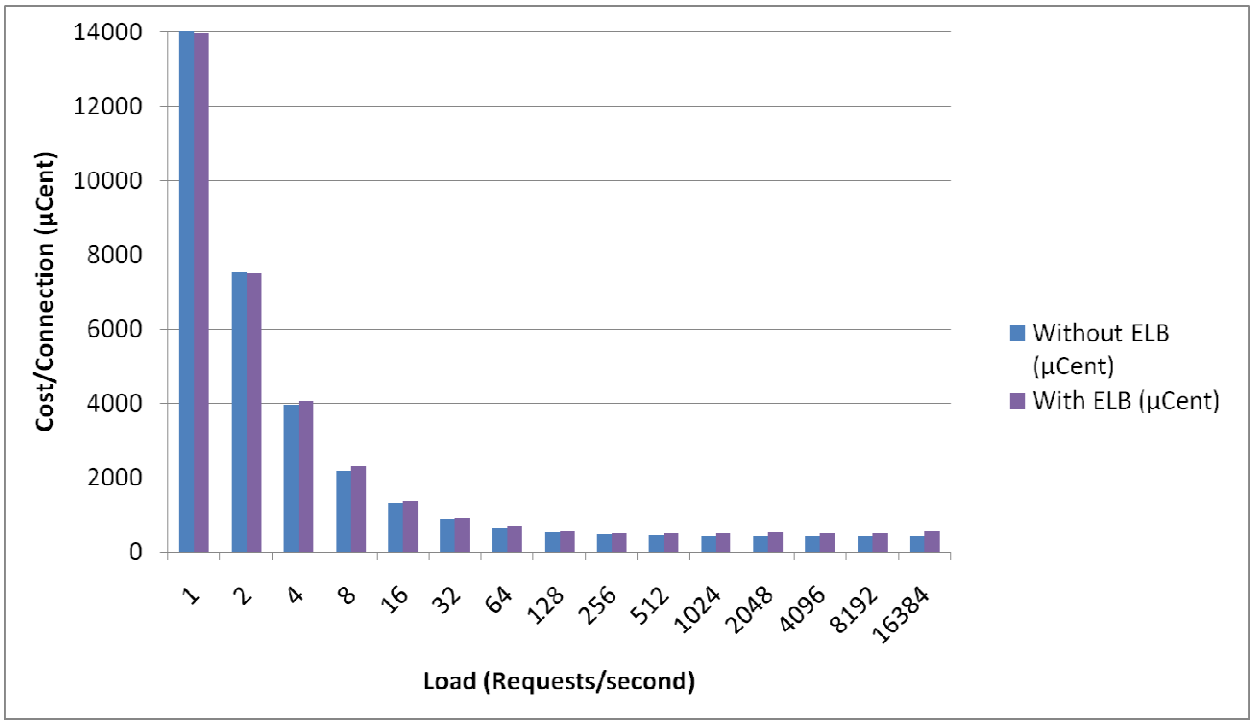

Figure 4b: Cost/connection in inter-zone setup

3.2. Agility (awareness) of ELB

To understand the adaptability of ELB to different configurations, we experimented with heterogeneous back-ends.

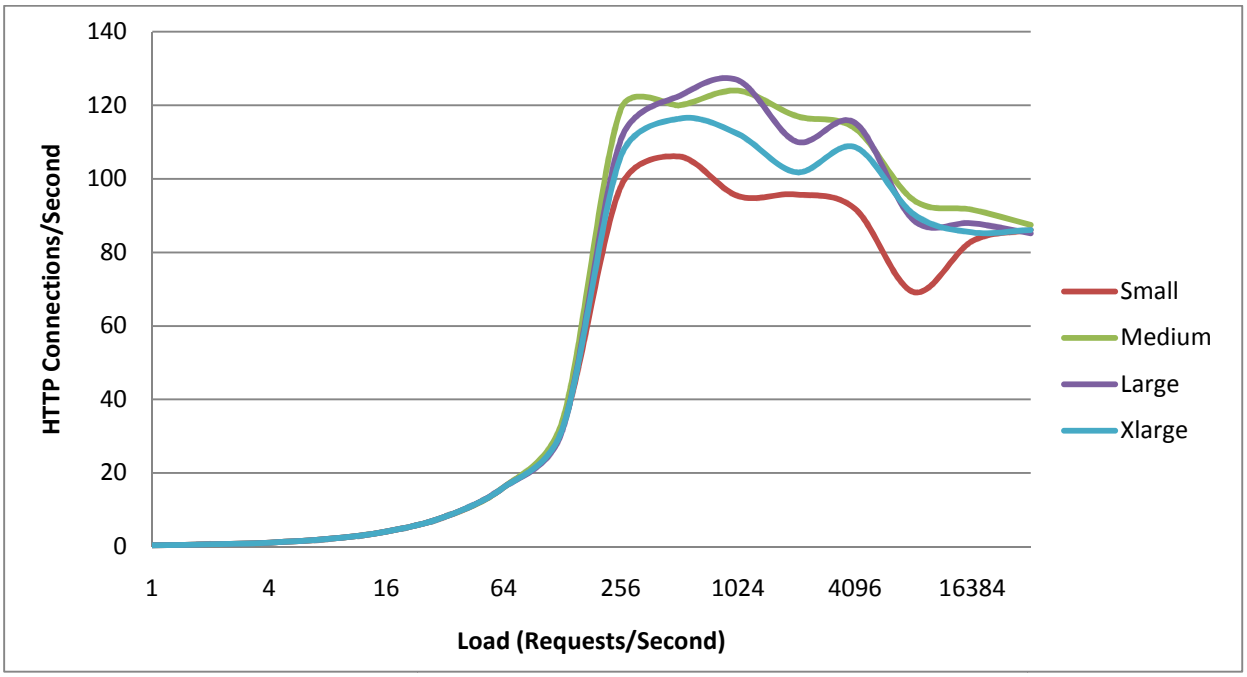

Figure 5: Agility (awareness) of ELB on capacity of back-end servers

We observed that ELB distributes traffic somewhat according to back-end capacity only at high load, and **not at low-moderate load**. Figure 5 depicts this phenomenon. Up to rates of 256 connection requests/second, no major difference between the numbers of requests each server receives can be observed. Beyond that rate, ELB starts to distribute traffic according to capacity. Smaller instances receive a lower number of requests, and larger servers receive higher numbers of requests.

A certain RAM-dependent behavior can also be observed. In this setup, the instances with the most amount of RAM (medium) achieve the highest throughput. This suggests that memory starvation and the triggering of swapping may be main throughput-limiting factor.

3.3. Fairness

In an *intra-zone* setup, no significant variation between ELB-distributed server loads has been observed. The biggest difference was about 5% of total requests received.

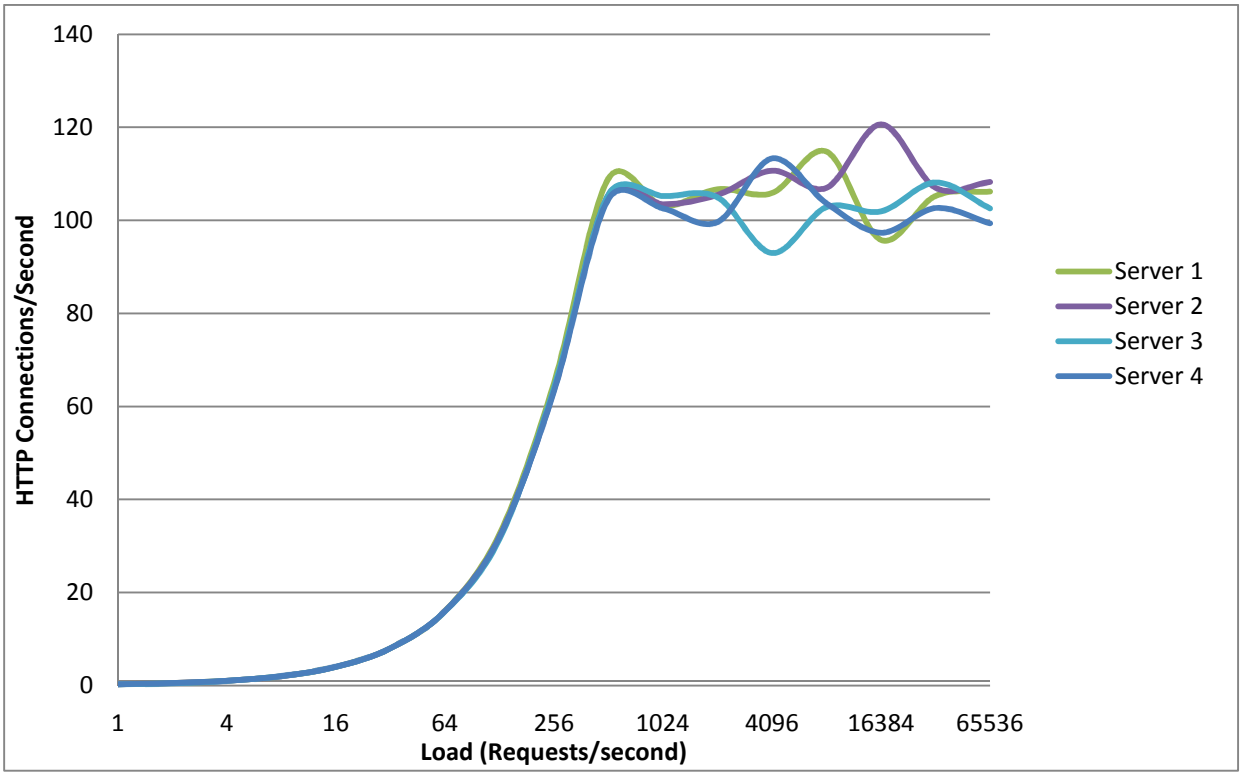

Figure 6: Fairness testing results in intra-zone setup

In the inter-zone setup, the probing machines were located in zone us-east-1a. All HTTP servers were equally distributed in among available zones, one server in each zone: us-east-1a, us-east-1c, and us-east-1d.

Significant differences were observed, most likely due to inter-zone network-related bottlenecks. In fact, the default ELB behavior seems to be to route all traffic to any available server until its capacity is exhausted, and only then route elsewhere. Thus, **in inter-zone setup ELB does not treat individual back-ends fairly**.

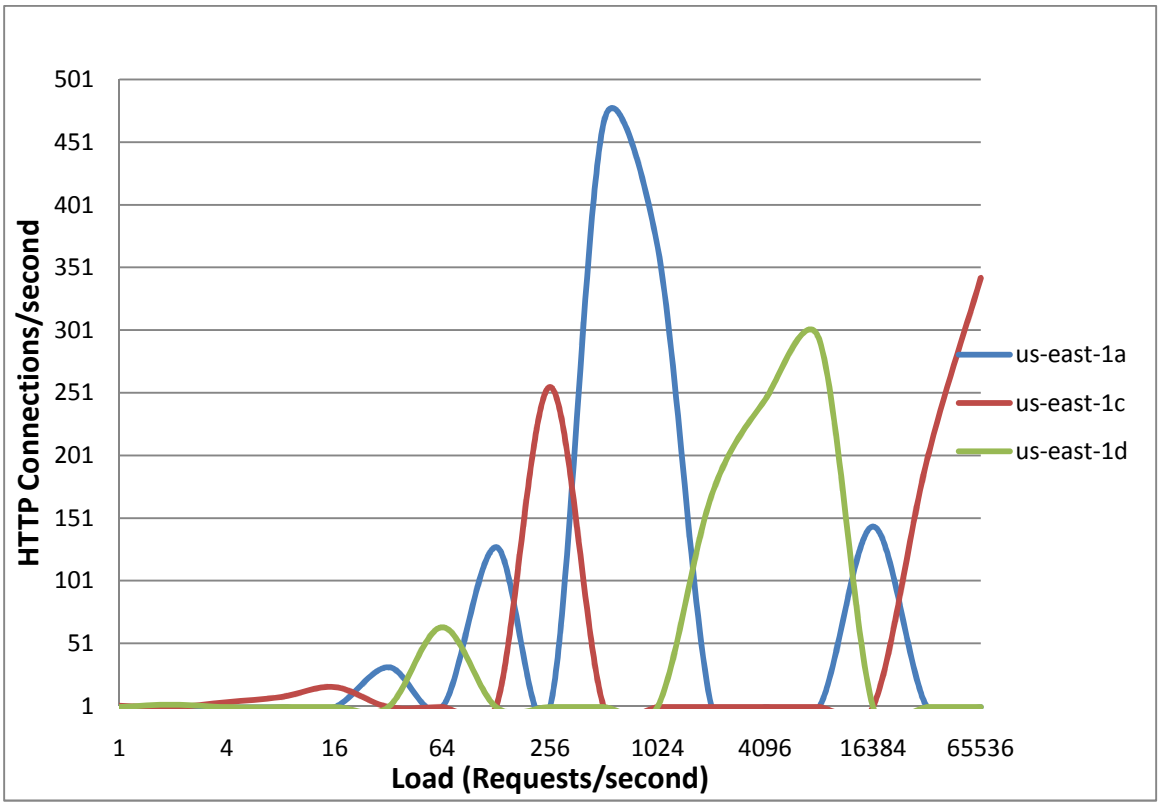

Figure 7: Fairness testing results in inter-zone setup

3.4. Convergence

In this experiment, both the probing machines and back-ends were in the same EC2 zone. Custom throughput monitoring PERL scripts were running on the probing machines, generating load through Httperf at a rate of just under 256 requests per second (experimentally determined optimal load for throughput comparison, see above). The monitoring perl scripts were measuring throughput at ten second intervals. Additional scripts were deployed remotely to back-ends on and off in a cycle: every three minutes, HTTP service in one additional back-end was turned off. After no more HTTP servers were active, every three minutes a new server would be turned on until all servers are on. This benchmark was run in both intra-zone and inter-zone setups.

Figure 8 shows the results for the convergence benchmark in the intra-zone setup. For the first 180 second period, all four servers were up and all of them feature the same throughput. Similarly equal per-back-end throughput can be observed in the 180-360

second (3 servers), 360-480 second (2 servers), and 480-720 second (1 server) intervals. During the 720-900 sec interval none of the servers were up. After that the 4 servers were gradually turned on one by one at 180 second intervals.

We observed that ELB was able to determine back-end health quickly and route traffic accordingly. There is no significant difference between the per-server traffic. In fact, ELB was able to detect unhealthy servers within five seconds, and start redistributing traffic. In intra-zone setups, ELB quickly detects back-end health and converges to fairness.

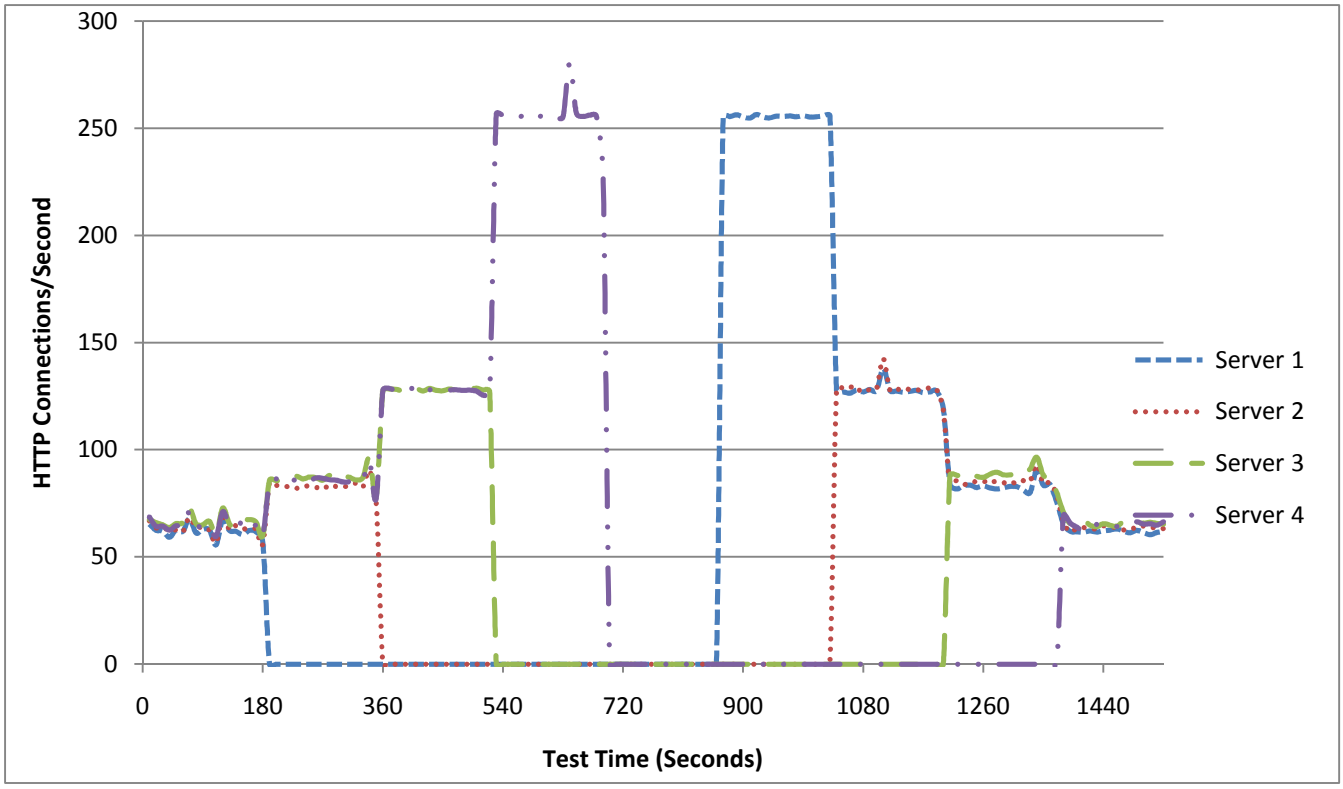

Figure 8: Result of convergence testing in intra-zone setup

However, in inter-zone setup, the result is quite different as shown in Figure 9. While ELB detects the back-end health as quickly as intra-zone setups, it does not converge to a fair distribution, as partially anticipated by the results depicted above in Figure 7.

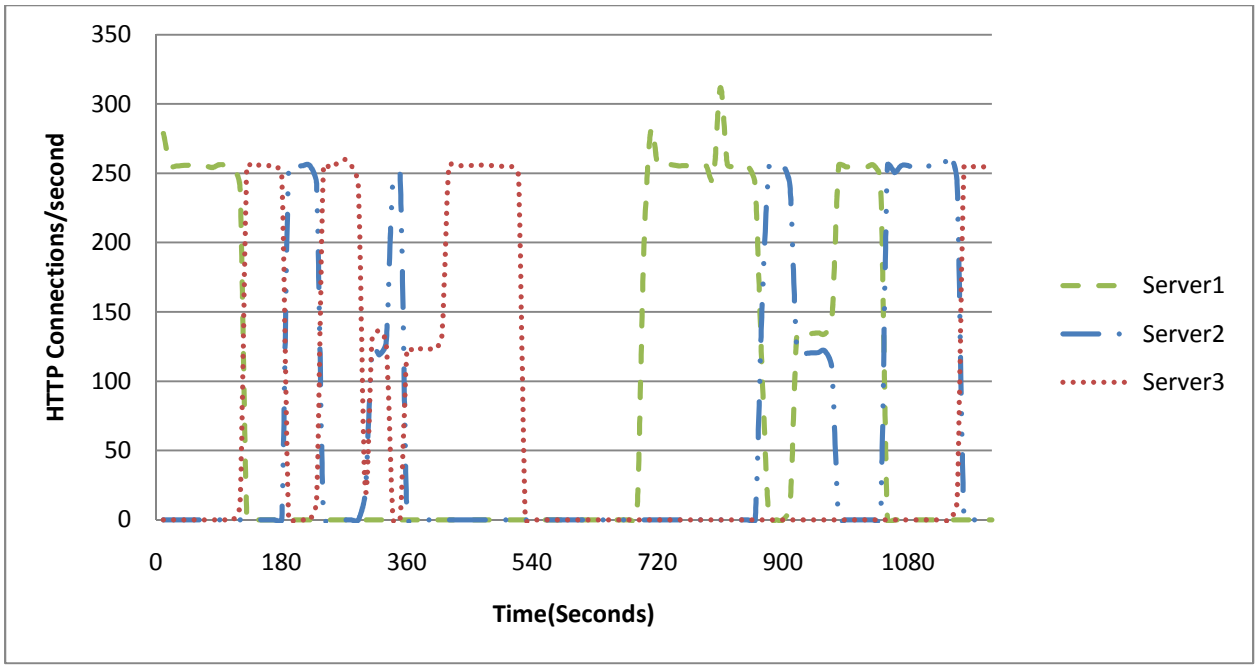

Figure 9: Result of convergence testing in inter-zone setup

4. Conclusions

To provide fault tolerance and automatic scaling, ELB incurs both cost and performance penalties in intra- and inter-zone setups. In inter-zone setups, this penalty seems to be more severe. The penalty is naturally directly proportional to the loads.

Overall, ELB seems to live up to its expectations mostly in intra-zone settings where it seems to deliver agility and fairness. **This does not hold for inter-zone settings**.