Business Requirement [C]

# BR [C].1

- Created ec2 t2.xlarge VM with instance id i-0b57bc05c6c0bf0b

- Navigate to Security Groups on EC2 Management Console and add inbound custom TCP rule 80 for launch-wizard-3

- Install apache (package managed on all 3 web servers)

sudo su

apt-get install apache2

apt install rsplib-legacy-wrappers

a2enmod headers

- Install on the load-balancer the following

sudo su

a2enmod proxy

a2enmod proxy\_http

a2enmod proxy\_ajp

a2enmod rewrite

a2enmod deflate

a2enmod headers

a2enmod proxy\_balancer

a2enmod proxy\_connect

a2enmod proxy\_html

a2enmod lbmethod\_byrequests

sudo service apache2 restart

- Add the following config to /etc/apache2/sites-enabled/000-default.conf to setup load balancing

vi /etc/apache2/sites-enabled/000-default.conf

#BR(C).1: Load Balancer

ProxyRequests off

<Proxy balancer://mycluster>

# Web Server A

BalancerMember http://54.153.248.3:8080

# Web Server B

BalancerMember http://54.206.2.135:8050

# mod\_lbmethod\_byrequests

ProxySet lbmethod=byrequests

</Proxy>

<Location /balancer-manager>

SetHandler balancer-manager

</Location>

ProxyPass /balancer-manager !

ProxyPass / balancer://mycluster/

- Created Load Balancer

- Navigate to Load Balancing on EC2 Management Console and create Load Balancer.

- Named Load Balancer s3563242 and s3563242-lb for security group name.

- Changed Ping path from /index.html to / under Configure Health Check

- Added EC2 instanced for Web Server A and Web Server B

- DNS name for Load Balancer is s3563242-1984617293.ap-southeast-2.elb.amazonaws.com

# BR [C].2

The final loadfactor used were:

Web Server A: 1

Web Server B: 2

My reasons for choosing this are highlighted in BR [C].3

# BR [C].3

- Refer to Assignment Specs

*“ Your cloud engineer has recommended that you create a new vm(“Load Testing” vm) using a “m4.xlarge” ec2 instance as your load testing server. ”*

- Created ec2 m4.xlarge VM with instance id i-079f2af01e3044581

sudo su

sudo DEBIAN\_FRONTEND=noninteractive apt-get –y update

sudo DEBIAN\_FRONTEND=noninteractive apt-get -y install apache2-utils

sudo DEBIAN\_FRONTEND=noninteractive apt-get -y install httperf

## Initial load tests run based off LoadFactor setup in BR [C].3

Here the loadfactors chosen were:

Web Server A: 1

Web Server B: 2

Web Server B to work 2 times harder than Web Server A as it is larger instance. i.e. Web Server B is t2.large and Web Server A is t2.micro thus can handle more requests.

- Refer to Assignment Specs

*“ Based off estimates from business, your load tests need to allow for*

*/load/cpu.php has a concurrency of 10 & test 150 requests.*

*/load/io.php has a concurrency of 10 & test 200 requests. “*

I next tested the /load/io.php.script (this script was run 3 times)

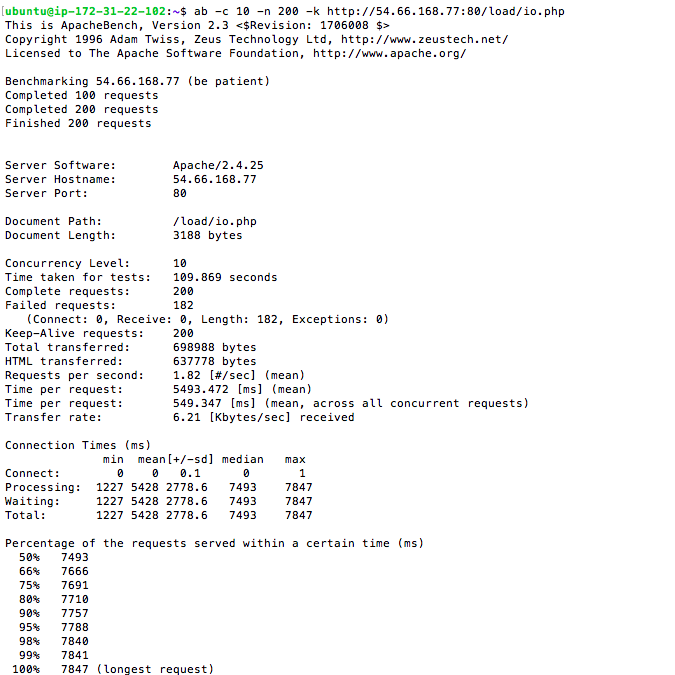
ab –c 10 –n 200 –k http://54.66.168.77:80/load/io.php

I next tested the /load/cpu.php script(this script was also run 3 times)

ab –c 10 –n 150 –k http://54.66.168.77:80/load/cpu.php

Test Results 1

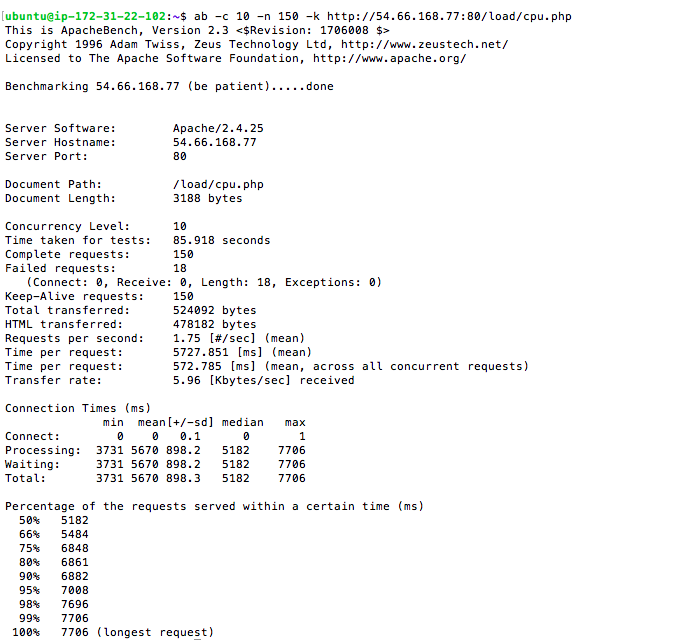
io.php



cpu.php

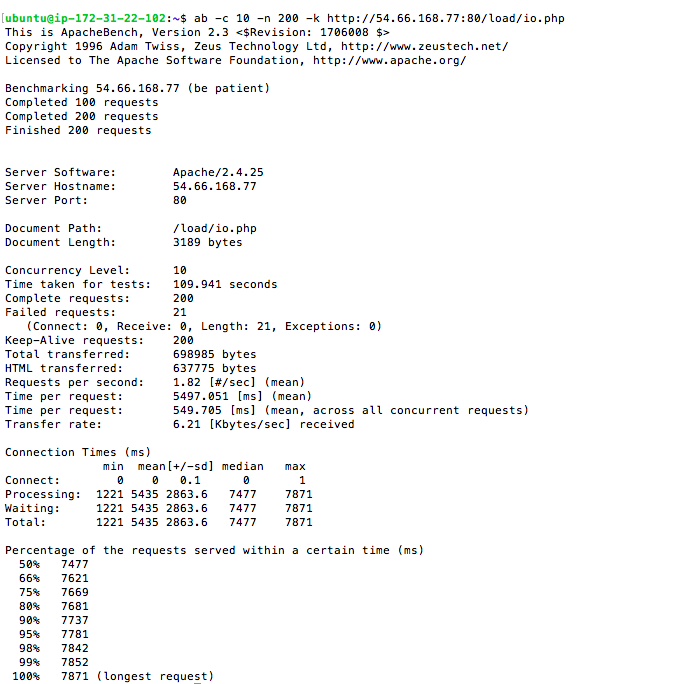
Test Results 1

cpu.php



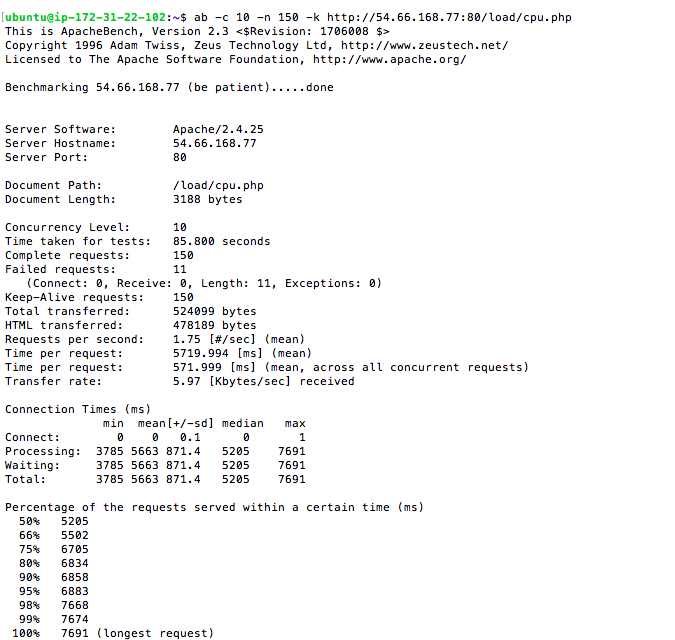
Test Results 2

io.php



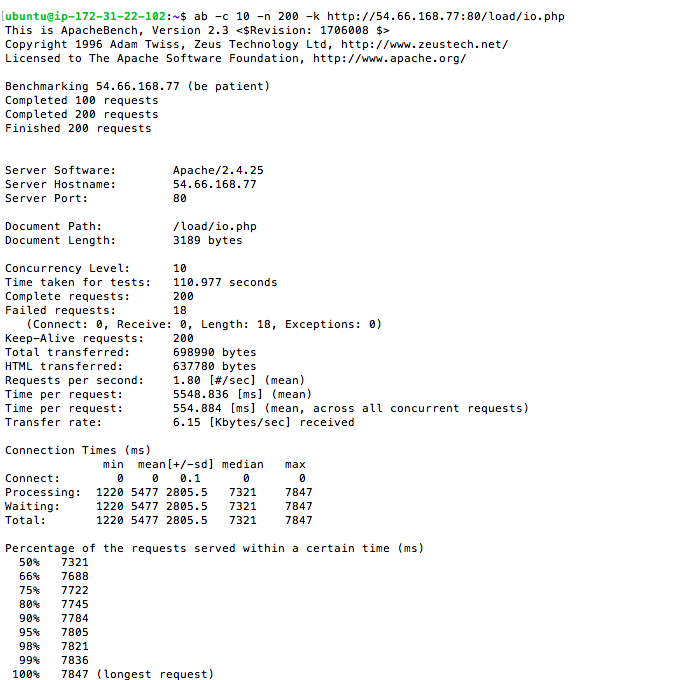
Test Results 2

cpu.php



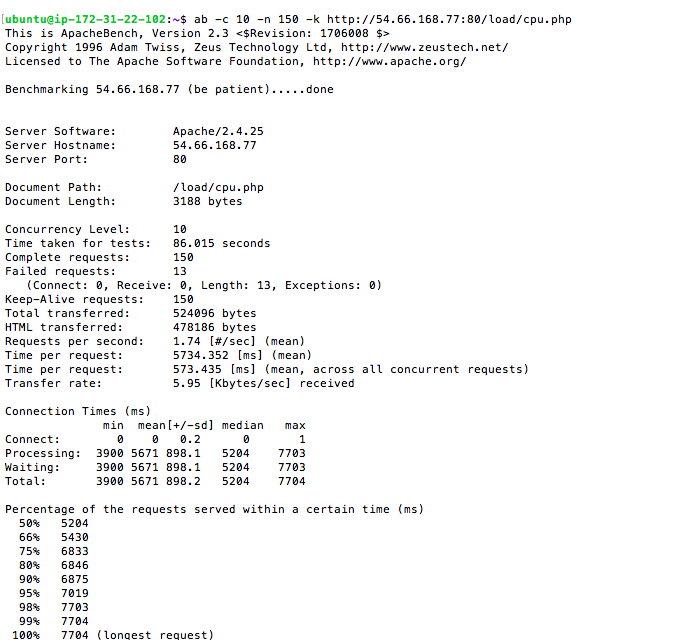
Test Results 3

io.php



Test Results 3

cpu.php



## 

Based off the above tests the following results were received:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| All Times Are In Seconds .. | | | | |
| **Figure** | **Test 1** | **Test 2** | **Test 3** | **Average** |
| Time Taken for io.php | 110 | 110 | 111 | 110.33333 |
| Time Taken for cpu.php | 86 | 86 | 86 | 86 |
| Final Number Used (just average) |  |  |  | 98.166665 |
|  |  |  |  |  |
|  |  |  |  |  |

So the result for loadfactors shown below was: 98.17 secs

Web Server A: 1

Web Server B: 2

## Round 2 load tests run based off NEW LoadFactor

Here the loadfactors chosen were:

Web Server A: 1

Web Server B: 5

Web Server B to work 5 times harder than Web Server A as it is larger instance. i.e. Web Server B is t2.large and Web Server A is t2.micro thus can handle more requests.

- Refer to Assignment Specs

*“ Based off estimates from business, your load tests need to allow for*

*/load/cpu.php has a concurrency of 10 & test 150 requests.*

*/load/io.php has a concurrency of 10 & test 200 requests. “*

I next tested the /load/io.php.script (this script was run 3 times)

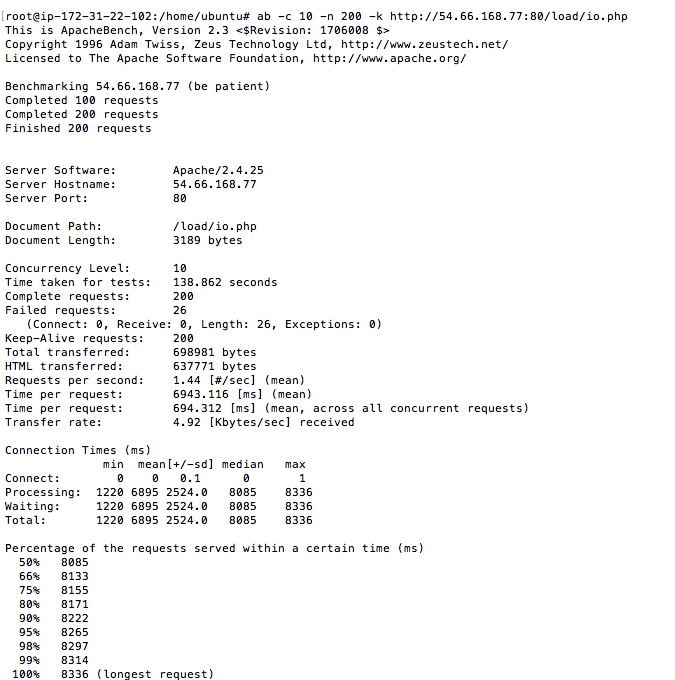
ab –c 10 –n 200 –k http://54.66.168.77:80/load/io.php

I next tested the /load/cpu.php script(this script was also run 3 times)

ab –c 10 –n 150 –k http://54.66.168.77:80/load/cpu.php

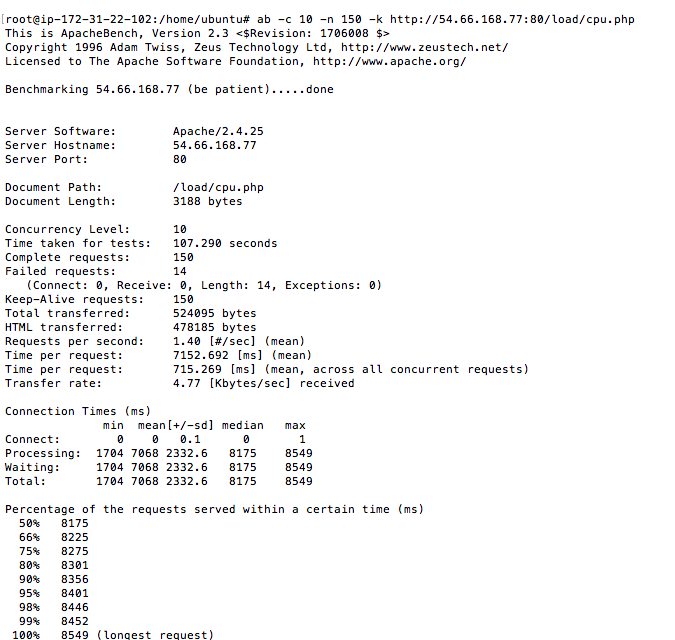
Test Results 1

io.php



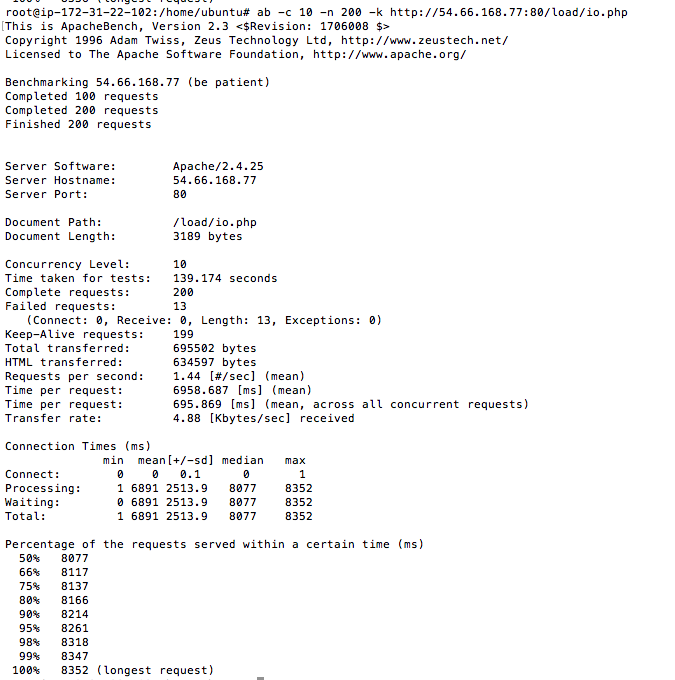
Test Results 1

cpu.php



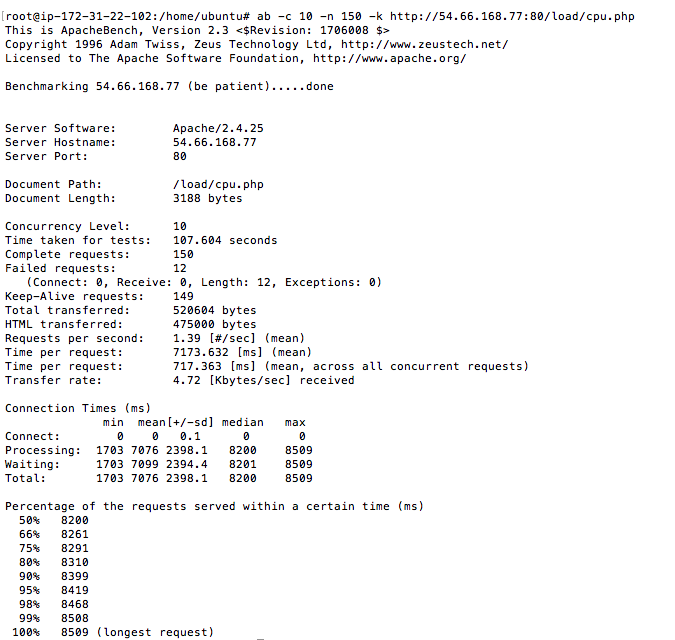
Test Results 2

io.php



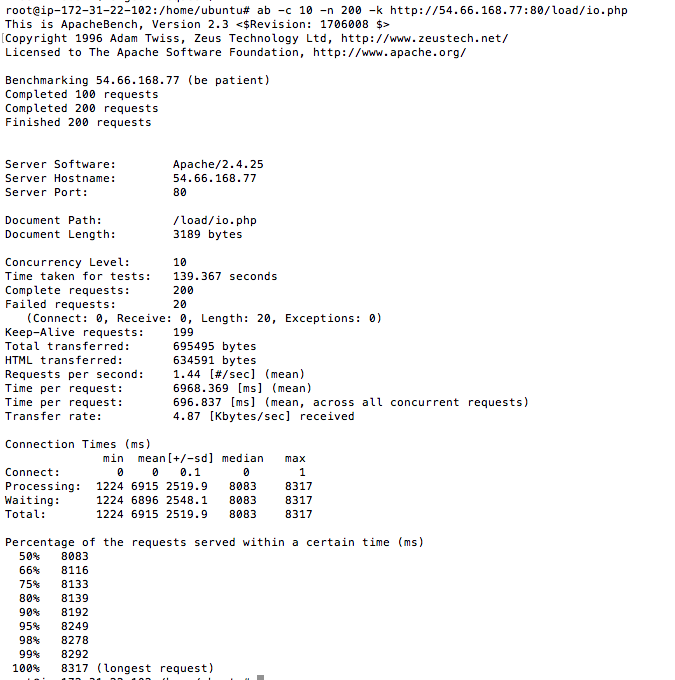
Test Results 2

cpu.php



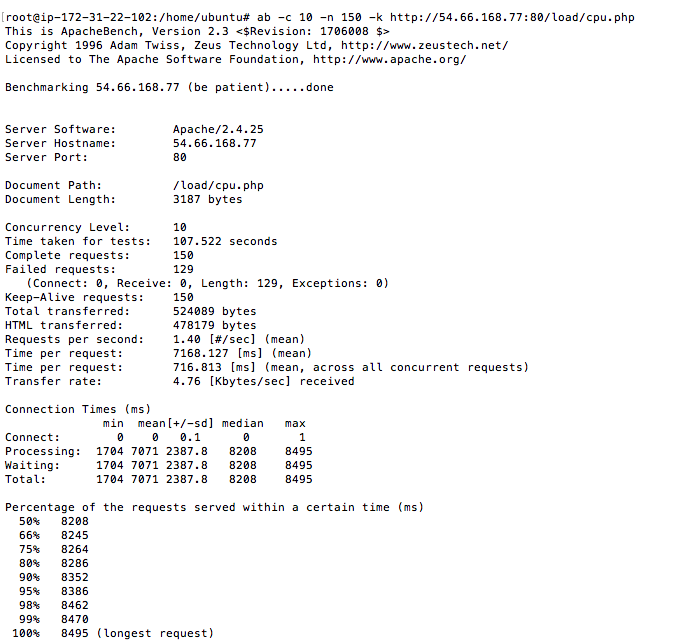
Test Results 3

io.php



Test Results 3

cpu.php



Based off the above tests the following results were received:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| All Times Are In Seconds .. | | | | |
| **Figure** | **Test 1** | **Test 2** | **Test 3** | **Average** |
| Time Taken for io.php | 139 | 139 | 139 | 139 |
| Time Taken for cpu.php | 107 | 108 | 108 | 107.66667 |
| Final Number Used (just average) |  |  |  | 123.33333 |
|  |  |  |  |  |
|  |  |  |  |  |

So the result for loadfactors shown below was: 123.33 secs

Web Server A: 1

Web Server B: 5

Final Result:

Based on the above tests, we can deduce that the following loadfactors yield the best performance:

Web Server A: 1

Web Server B: 2

Most optimum load factor is the one that gives you the best performance.

I would choose load factor 1 and 2 as they responded much faster than 1 and 5.