**CSC326**

**Lab 3 Report**

**Development Phase 3**

Group #: 4

Names: Anny Kim Ly (997581718)

Kevin Gumba (997585117)

**Section 1: Benchmarking**

1) Methodologies used to benchmark your web application

The Apache benchmarking tool was used to evaluate the performance of the web server. Performance is measured through the maximum number of connections and maximum number of requests that can be handled. CPU and memory usage was also tracked during the performance runs. As well, Google PageSpeed Insights was used to check for performance optimization methods.

2) Report the statistics collected from benchmarking

*a) Maximum number of Connections*

Our current server can handle 27 connections before connections are dropped. The connections are increased until a maximum number of connection is found. The following pages shows cpu and memory usages during connection and data retrieved while running apache benchmarking software.

Data 1 shows the successful performance results from running the apache benchmark using 27 concurrent requests with 27 requests in total, thus using 27 connections. Data 2 shows that the experiment using 28 connections, with one request per concurrent request, failed before completion. The connection to the web server gets reset after 27 requests thus the maximum number of connections is determined to be 27.

*b) Maximum number of requests that can be handled*

Our current server can go over 9 000 requests before the server crashes. The test is left running overnight to collect the results.

Data 3 shows the successful results from running the apache benchmark using 100000 requests.

*c) CPU and Memory Usage*

Data 4 and Data 5 shows the CPU and memory usage during the apache benchmarking experiments. Please note x axis is in seconds.

As the amount of users request the website, the EC2 memory decreases over time. However, CPU becomes idle at certain times.

Data 4 shows server receiving 100000 requests. Evident by the graphs, more memory is used as more uses requests to access the website. However, without setting an amount of connections (no -c), CPU is kept near idle almost all the time.

Data 5 showsrunning apache with -n 2000 -c 10. As more connections are used, more CPU is used in order to handle the amount of requests. More memory is used as well during the time of requests.

**Data 1:** Max 27 connections

ab -n 27 -c 27 http://ec2-54-225-76-58.compute-1.amazonaws.com/

This is ApacheBench, Version 2.3 <$Revision: 655654 $>

Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ec2-54-225-76-58.compute-1.amazonaws.com (be patient).....done

Server Software: WSGIServer/0.1

Server Hostname: ec2-54-225-76-58.compute-1.amazonaws.com

Server Port: 80

Document Path: /

Document Length: 5362 bytes

Concurrency Level: 27

Time taken for tests: 1.604 seconds

Complete requests: 27

Failed requests: 0

Write errors: 0

Total transferred: 148959 bytes

HTML transferred: 144774 bytes

Requests per second: 16.83 [#/sec] (mean)

Time per request: 1603.828 [ms] (mean)

Time per request: 59.401 [ms] (mean, across all concurrent requests)

Transfer rate: 90.70 [Kbytes/sec] received

Connection Times (ms)

min mean[+/-sd] median max

Connect: 17 17 0.3 17 18

Processing: 19 600 590.6 468 1586

Waiting: 18 599 590.6 467 1585

Total: 36 617 590.9 485 1603

Percentage of the requests served within a certain time (ms)

50% 265

66% 708

75% 712

80% 1597

90% 1601

95% 1602

98% 1603

99% 1603

100% 1603 (longest request)

**Data 2:** Lost connection at 28

ab -n 28 -c 28 http://ec2-54-225-76-58.compute-1.amazonaws.com/

This is ApacheBench, Version 2.3 <$Revision: 655654 $>

Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ec2-54-225-76-58.compute-1.amazonaws.com (be patient)...apr\_socket\_recv: Connection reset by peer (104)

Total of 27 requests completed

**Data 3:** Running with 100000 requests

ab -n 100000 http://ec2-54-225-76-58.compute-1.amazonaws.com/

This is ApacheBench, Version 2.3 <$Revision: 655654 $>

Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ec2-54-225-76-58.compute-1.amazonaws.com (be patient)

Completed 10000 requests

Completed 20000 requests

Completed 30000 requests

Completed 40000 requests

Completed 50000 requests

Completed 60000 requests

Completed 70000 requests

Completed 80000 requests

Completed 90000 requests

Completed 100000 requests

Finished 100000 requests

Server Software: WSGIServer/0.1

Server Hostname: ec2-54-225-76-58.compute-1.amazonaws.com

Server Port: 80

Document Path: /

Document Length: 5362 bytes

Concurrency Level: 1

Time taken for tests: 3671.039 seconds

Complete requests: 100000

Failed requests: 0

Write errors: 0

Total transferred: 551700000 bytes

HTML transferred: 536200000 bytes

Requests per second: 27.24 [#/sec] (mean)

Time per request: 36.710 [ms] (mean)

Time per request: 36.710 [ms] (mean, across all concurrent requests)

Transfer rate: 146.76 [Kbytes/sec] received

Connection Times (ms)

min mean[+/-sd] median max

Connect: 16 18 15.8 17 1023

Processing: 18 19 6.5 18 678

Waiting: 17 18 6.5 18 677

Total: 34 37 17.1 35 1042

Percentage of the requests served within a certain time (ms)

50% 35

66% 35

75% 36

80% 36

90% 36

95% 37

98% 60

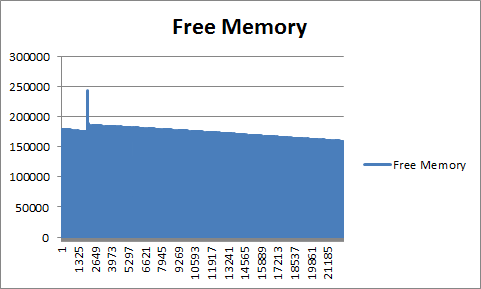
99% 63

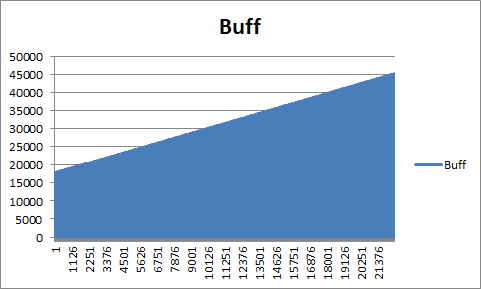
100% 1042 (longest request)

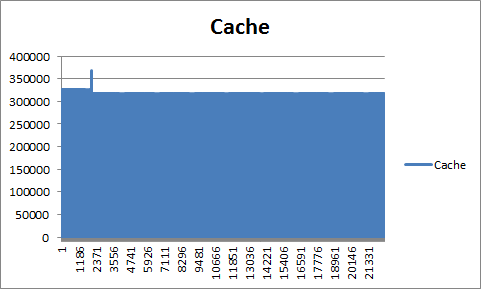
**Data 4:** Running with 100000 requests

Note: X axis is in seconds.

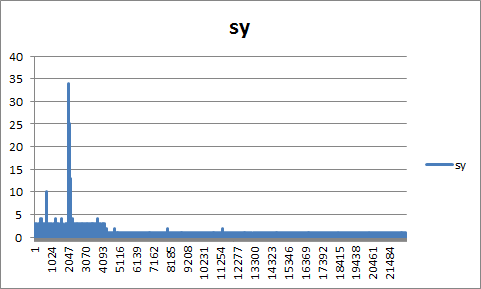
**Memory Usage**



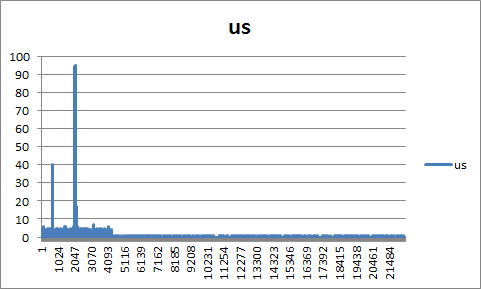




**CPU Usage**



% of time CPU spent in kernel mode

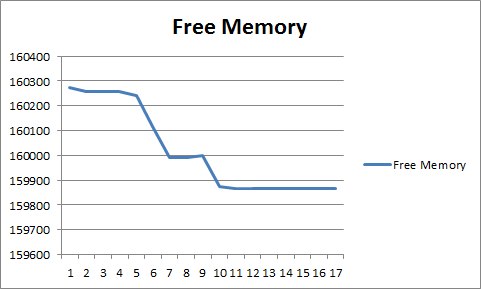


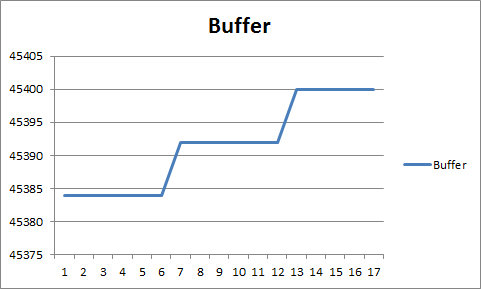
% of time CPU spent in user mode

**Data 5:** Running -n 2000 -c 10

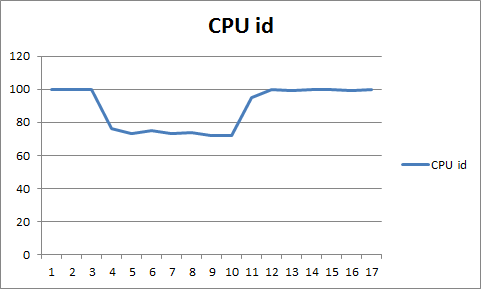
Note: X axis is in seconds.

**Memory Usage**

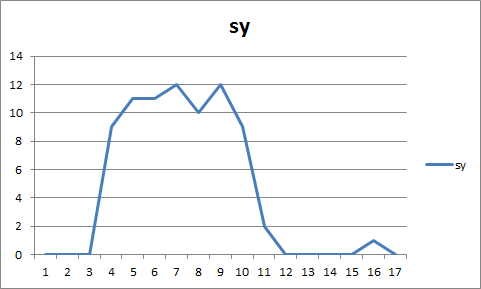




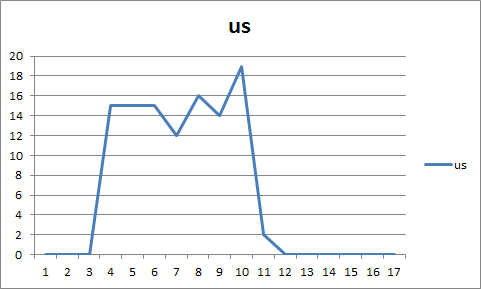
**CPU Usage**



% of time CPU is idle



% of time CPU spent in kernel mode



% of time CPU spent in user mode

**Section 2: Frontend (by Anny Ly)**

1.0 Design Decisions

*1.0.1 Homepage*

The homepage of the search engine only displays the name and logo, as well as the Google+ sign-in button. After the user signs in to Google+ and allows the search engine application access, the search bar and submit button is then displayed. The Google+ sign-in button is then replaced by a log out button. The Google+ Sign In mechanism is done with Javascript.

*1.0.2 Results Page*

On the results page, a log out button is displayed at the top of the webpage for users to sign out of their Google+ account. Once they are signed out, they will be redirected to the home page and not allowed to access to the search engine feature until they sign in again.

*1.0.3 Error 403 Page*

If the user tries to access a web page without signing in to their Google accounts, they will be redirected to the error 403 page that includes a button that can return them to the homepage.

1.1 Instructions to Run Code

**Note:** Please keep **pop-ups disabled** when accessing the search engine.

Please run “python crawler.py” in the Backend folder first before running SpaceExplorer.py. A dbFile.db must first be created in order to access persistent data.

To run the program, simply type “python <file name>.py” in the command line to start up the search webpage. In this case the file name is “SpaceExplorer.py”, thus do “python Space Explorer.py”.

On execution of the file, the home page can be accessed through “[http://ec2-54-225-76-58.compute-1.amazonaws.com](http://ec2-54-225-76-58.compute-1.amazonaws.com/)”. The user will first be presented with a login page consisting of a Google+ sign in button and the name and logo of the search engine. Once signed in, a simple interface to submit a keyword or phrase will appear. A log out button can also be accessed at the top of the webpage. Once the “search” button is clicked, the user will be redirected to the results page. There, they can then find a results table with a list of URLs ordered by their page rank along with additional detail such as title and description. Pagination will occur if there are more than 10 results. The user can also sign out using the log out button at the top of the results page.

**Section 3: Backend (by Kevin Gumba)**

2.0 Design Decisions

*2.0.1 Elastic IP Address*

The site is located at <http://ec2-54-225-76-58.compute-1.amazonaws.com/>

The server is run on an ongoing basis using the screen command. A screen is created inside the instance, the contents on the installPackage.tar.gz is placed inside and and installed, and afterwards detached. Using the screen command on Ubuntu allows for reattachment later or termination if the search engine is needed to be taken down. The public IP address is 54.225.76.58 and is accessed using the spaceKey.pem.

*2.0.2 installPackage.tar.gz*

The tar file containing the search engine is extracted using command tar ­zxvf installPackage.tar.gz

The tar file installPackage contains 6 main files:

* BeautifulSoup.py - Aids crawler.py in parsing web pages
* bottle.py - Aids L3SpaceExplorer.py in running a server
* crawler.py - Back end code that data handles and stores web page information
* install.sh - Installs and runs search engine
* dbFile.db - Allows the server to have default values.
* L3SpaceExplorer.py - Front end code that creates user interface
* urls.txt - Contains sites where crawler.py shall crawl first

Run command tar ­zxvf installPackage.tar.gz inside the instance to extract the files.

*2.0.3 install.sh*

The search engine server is installed in an unmodified Ubuntu AMI by running bash install.sh

The shell script allows running the crawler, the allowing permission for the front end L3SpaceExplorer.py to use restricted ports, and to finally run the front end code.

Rather than importing files via shell script, files were downloaded and placed into the zip file. This makes sure that the correct version of each source is correct in exchange for increase memory needed to store in the tar file.

Run command bash install.sh once tar file is extracted.

**Data 6:** aws ec2 describe-instances

Reservations": [

{

"OwnerId": "125137181039",

"ReservationId": "r-831be0e5",

"Groups": [

{

"GroupName": "SpaceCadet",

"GroupId": "sg-833878e8"

}

],

"Instances": [

{

"Monitoring": {

"State": "disabled"

},

"PublicDnsName": "ec2-54-225-76-58.compute-1.amazonaws.com",

"RootDeviceType": "ebs",

"State": {

"Code": 16,

"Name": "running"

},

"EbsOptimized": false,

"LaunchTime": "2013-11-06T18:37:26.000Z",

"PublicIpAddress": "54.225.76.58",

"PrivateIpAddress": "10.147.176.235",

"ProductCodes": [],

"StateTransitionReason": null,

"InstanceId": "i-1791a072",

"ImageId": "ami-d0f89fb9",

"PrivateDnsName": "ip-10-147-176-235.ec2.internal",

"KeyName": "spaceKey",

"SecurityGroups": [

{

"GroupName": "SpaceCadet",

"GroupId": "sg-833878e8"

}

],

"ClientToken": null,

"InstanceType": "t1.micro",

"NetworkInterfaces": [],

"Placement": {

"Tenancy": "default",

"GroupName": null,

"AvailabilityZone": "us-east-1b"

},

"Hypervisor": "xen",

"BlockDeviceMappings": [

{

"DeviceName": "/dev/sda1",

"Ebs": {

"Status": "attached",

"DeleteOnTermination": true,

"VolumeId": "vol-879bfcc4",

"AttachTime": "2013-11-06T18:37:30.000Z"

}

}

],

"Architecture": "x86\_64",

"KernelId": "aki-88aa75e1",

"RootDeviceName": "/dev/sda1",

"VirtualizationType": "paravirtual",

"AmiLaunchIndex": 0

}

]

}

]

}