# DISTRIBUTED SYSTEMS – LAB 5 REPORT

Distributed Systems – Lab 3 Web Application & Performance

Submitted: November 24, 2014
By: Devan Shah 100428864
Submitted to: Weina Ma

# PART I: Setup for development

7. Can you explain the observations Step 6? Use the source code lab-web/sample/HelloWorld.java in your explanation.

## Answer:

http://localhost:4567/hello

When this link was accessed "Hello World." was displayed in the web browser. This is displayed with the use code in HelloWorld.java, this code is using Spark to listen for requests on the default localhost port. The string "Hello World." is displayed due the multil-setp get blocks which has assigned /hello to display "Hello World." when called from http://localhost:4567/. Following is the code snippet from the HelloWorld.java that handles the URL path “/hello”:

```
get ( "/hello", ( req, resp ) ->
{
   return "Hello World.";
} );
```

http://localhost:4567/add?a=3.1415&b=10.0

When this link was accessed "3.1415 + 10.0 = 13.1415" was displayed in the web browser. The HelloWorld.java code uses a request response structure using Spark to attach on the localhost:4567 and listen/respond to the get structures defined in the java code. In this case using "/add" to perform a computation of 2 numbers. The code uses request to get the perms that are passed in with the /add and perform an add operator on the 2 parameters and display the results with the question also. The way that the parameters are passed in it makes it easy for spark to pick up on the parameters based on the name, for example: a=3.1415 in this case spark knows that a will have a value of 3.1415 the same goes for b and multiple parameters can be passed in with the use of & operation. Following is the code snippet from the HelloWorld.java that handles the URL path “/add” and the parameters that are passed with the “/add” URL:

```
get("/add", (req, resp) -> {
   String message;

   try {
      String a = req.queryParams("a");
      String b = req.queryParams("b");
      if(a == null) {
         message = "The \"a\" parameter is missing.";
      } else if (b == null) {
         message = "The \"b\" parameter is missing.";
      } else {
         float result = Float.parseFloat(a) + Float.parseFloat(b);
         message = a + " + " + b + " = " + result;
      }
   } catch(Exception e) {
      message = e.toString();
   }
   return message;
});
```

http://localhost:4567/

When this link was accessed "404 Not found The requested route [/] has not been mapped in Spark" was displayed in the web browser. This is displayed because there are no routes for the root "/". This is the default message that is printed by Spark in the case that they are not mapped. This would have to be implemented by adding code into HelloWorld.java get (“/”. (request, response) -> { });.

8. Based on **HelloWorld,** write your own Web application, call it **LabWeb,** to support a few more URLs. You are free to design URL path name and their functionalities. Make sure at least one reads from the file system.

## Answer:

Implemented the following URL path and their functionalities:

Some general things

http://localhost:4567/hello/devan
http://localhost:4567/say/hello/to/devan

Looking at protected sites with authentication and un-authentication

http://localhost:4567/protected/hello
http://localhost:4567/authenticate
http://localhost:4567/protected/hello
http://localhost:4567/unauthenticate
http://localhost:4567/protected/hello

Reading a file form the computer's file system

http://localhost:4567/Lab_5_Report.txt

Included in the submission is a zip file containing the source code in file LabWeb.java and other necessary files compile.cmd, Lab_5_Report.txt and run.cmd. The same steps can be followed to run the program and access the links. Please make sure that file “Lab_5_Report.txt” is inside the same directory where the program is running from.

## PART II: Benchmarking using ab.exe

2. Rank the response times of each of your URL paths of your Web application from the slowest to the fastest.

### Answer:

Number of requests sent for the below was 500.

| URL Path | Response Time |
|---|---|
| http://localhost:4567/Lab_5_Report.txt (5Kb file) | 0.340021 seconds |
| http://localhost:4567/protected/hello | 0.150003 seconds |
| http://localhost:4567/add?a=3.1415&b=10.0 | 0.130008 seconds |
| http://localhost:4567/ | 0.120006 seconds |
| http://localhost:4567/hello | 0.111007 seconds |
| http://localhost:4567/authenticate | 0.111005 seconds |
| http://localhost:4567/unauthenticate | 0.111002 seconds |

3. Explain the performance differences you see.

### Answer:

The performance difference that can be seen is that the more complicated the request is the longer it takes to perform the request. As seen the most complicated request of reading a file is taking the longest. There are multiple of URL operation that took close to the same amount of time, this is due to the fact that the complexity is close to the same. While some took longer was because the operations that were being performed when URL was accessed were little more complex.

5. What does concurrent sessions do to the response time?

### Answer:

Concurrent sessions decrease the response time up to the point where any more concurrent sessions would cause the response time to increase. The reason that the response time would increase at one point is because there is only one server that is accommodating for all the client requests, using concurrent sessions initially response time would be low but would increase over time as the number of concurrent sessions is to increase. This is the case because with the use of concurrent sessions the server would prepare the requests concurrently but the requests can only be handled one at a time.

6. By increasing the concurrency 1, 2, 3, 4, 5, 6, …, if we want to achieve a response time of 5 sec or less, what is the **maximum** number of concurrent sessions your Web application can support?

## Answer:

To accomplish this task the my "Lab_5_Report.txt" file had to be increased in size to about 234.14MB to reach a threshold that is close to 5 seconds per request. Following are the statistic of multiple 1 to 1 ratios between number of requests and the number of concurrency.

| number of requests | number of concurrency | URL Path | Response Time per Request |
|---|---|---|---|
| 3 | 0 | http://localhost:4567/Lab_5_Report.txt | 1.4534 seconds |
| 4 | 4 | http://localhost:4567/Lab_5_Report.txt | 2.3636 seconds |
| 5 | 5 | http://localhost:4567/Lab_5_Report.txt | 3.9253 seconds |
| **6** | **6** | **http://localhost:4567/Lab_5_Report.txt** | **4.8342 seconds** |
| 7 | 7 | http://localhost:4567/Lab_5_Report.txt | 5.4214 seconds |
| 8 | 8 | http://localhost:4567/Lab_5_Report.txt | 5.8343 seconds |

From the table above we can see that with the file size increased to 234.14MB the response time increases as there is more data to display. Also this allows to measure a more understandable reading of the effect of concurrency and the factor that would cause the most issues. **Therefore we see that the maximum number of concurrent sessions my Web application can support under 5 seconds is 6.**

7. Using ideas of distributed systems, offer solution to scale up your Web application to support increase traffic demand while still maintaining responsiveness of key URL paths of the application.

## Answer:

The main idea from distributed systems that would be beneficial in scaling up my Web application to support increase demand in traffic while maintaining responsiveness of key URL paths would be to implement multiple server that would handle different types of requests, such as reading a file. This will reduce the amount of computation that the single server has to do to accomplish its task, in the case of reading a file there can be multiple servers that can be used to distribute reading blocks of the filling and sending it to the web application as messages that way the main operation of reading the file is distributed and can be accomplished quicker. The same can be done for reading large images or performing any large computations. Once other idea that could help is to localize the data to a single content holder, such as a database which would allow the data to be accessed in an efficient manner and would involve less computation for Web application. As long as multiple servers are layout to retrieve data from the database and present it to its own web server. This would involve multiple servers, which take the max number of requests that they can handle and then switch to another server. In this case when a client performs a request it will find the server is best suitable to perform the operation using average computation time analysis.