**Project Report**

**Performance Analysis and Modeling of Software Systems, Fall Semester 2019**

PAULA ANDREA CASTAÑEDA HOYOS,  [PA.CASTANEDA@alumnos.upm.es](mailto:%20PA.CASTANEDA@alumnos.upm.es)

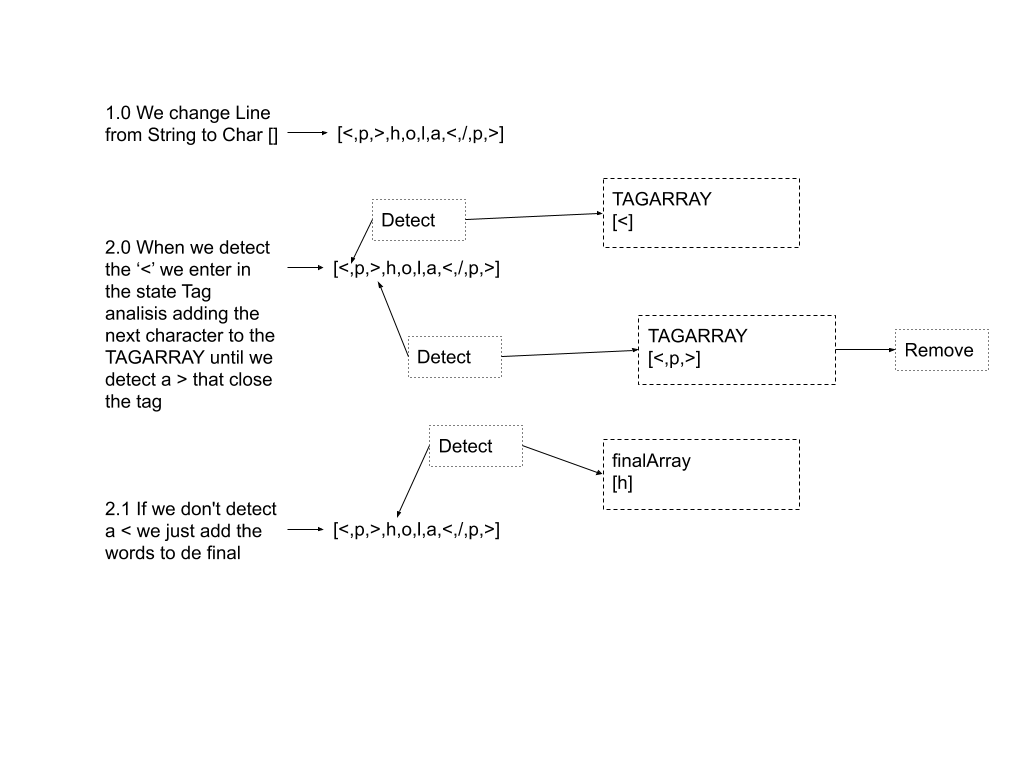
LUIS DOMÍNGUEZ LÓPEZ, [luis.dominguez.lopez@alumnos.upm.es](mailto:luis.dominguez.lopez@alumnos.upm.es)

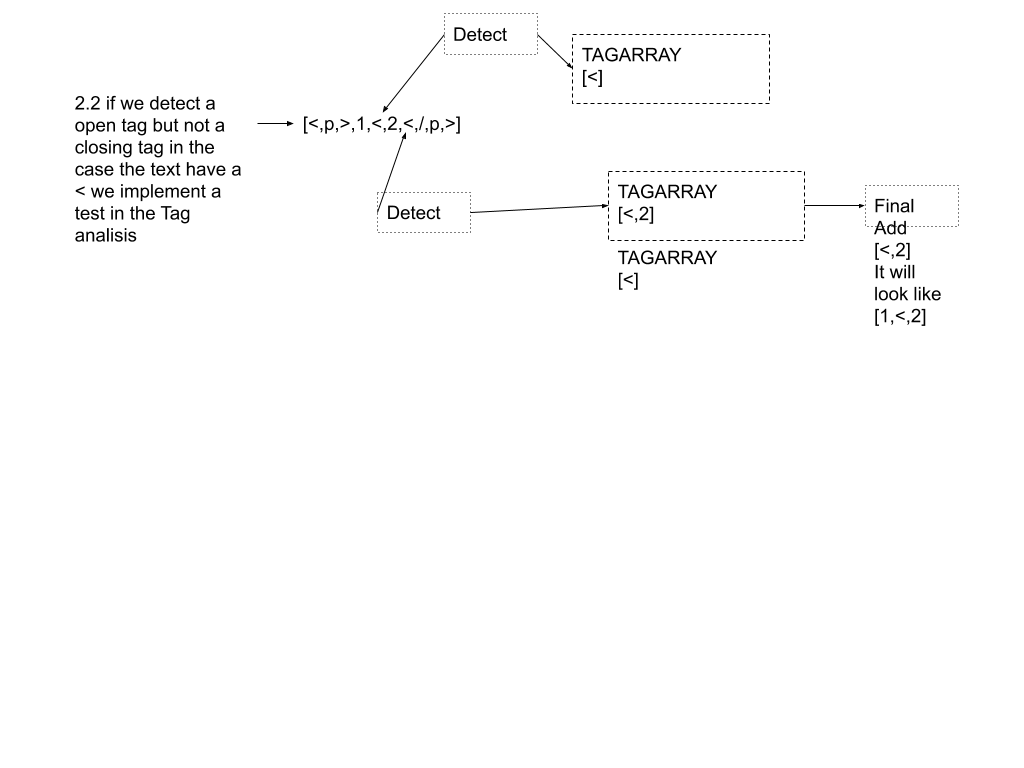
# Implementation overview

# We extended the system with the features that were requested (1. Cleaning of document from special words, 2. Add fine-grained statistics gathering inside the server. 3. Extend the server into a multi-threaded implementation). We implemented those features in a progress way that means that we didn't implement all the features from the beginning because one of our purpose was evaluate the system with each improvement. The first phase one we just calculate the times that was requesting in each method of the system, in the second phase we implemented the cleaning method of the HTML files and in the last one, we implemented the multithreading to handle the calls on the server.

## Cleaning

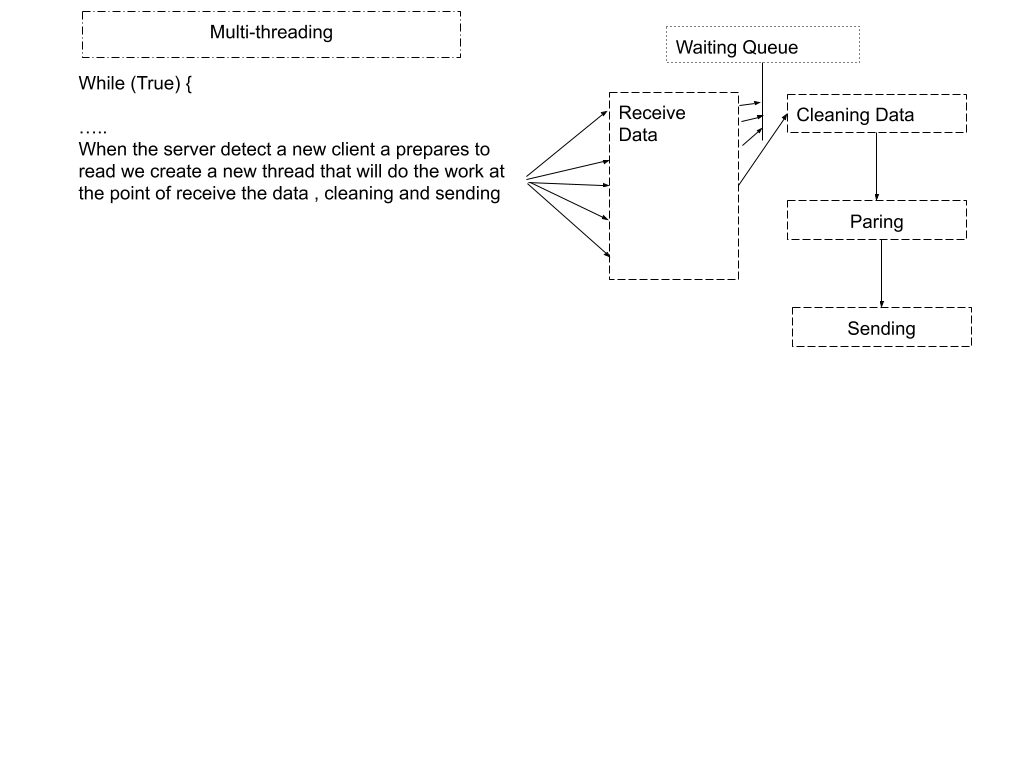
## We have implement the next algorithm





We have make a class call CleaningAlg.java where you can try the algorithm and see the test that we have done to the system.

## 1.2 multi-threading



We understand that the idea of the threading was that when a new client arrive we have to create a new thread that works in the cleaning and count process, so we create the thread at the point that the server detects a new client, and we use synchronized in the cleaning methods to assure that we don’t have problems with memory and concurrency.

Technically speaking what we do is that in the while(true) of the server, where the server waits to the clients, we have implement a thread that becomes alive when the text is ready so the while true don’t stop listening at that moment.

## Add fine-grained statistics gathering inside the server

* Time spent until the entire document has been received: We are going to use the equation of Mean service rate μ = 1 / E[s] that means the Time to process a job (“useful work”, no queueing), we have to remember that the system is going to verify in the server class if the buffer associated with the client has a full document in it. We start counting since the moment the server starts receiving data from client class we calculated the difference between the time when we start receiving the data file and the last time we received the last data with the funtion System.nanoTime();
* Time spent performing the word count: In the doWordCount we calculated two different times. First we calculate the difference when the server starts doing the WordCount till it finished cleaning the file that means wait for the response of the method Cleaning that we incorporate to the software and then we calculated the difference between it starts this process and finished counting all the file.
* Time spent serializing the results:We used the same logic to find the time spent in just one method or process of the server. We calculated the difference between the beginning time when the server starts to clean the document and when it finished it.

# 2. Baselines

## 2.1 Test Environment and Experimental Design

We created the two machines in Amazon but in the set up we could only select the type t2.micro for free, we couldn’t make machines with type t2.xlarge and connect to each other.

## 2.2 Throughput and Response Time

This is an open system and it was measure by the client and we focused on the number of jobs and number of files the client is going to use to make the full document that the server is going to receive. The follow images represent the behave of the throughput in their different states according with the among of calls that the server is receiving by the client.

**FIRST STATE WITH OUT MULTI THREATING AND CLEANING METHOD**

**SECOND STATE WITH CLEANING METHOD BUT WITH OUT MULTI THREATING**

**LAST STATE WITH CLEANING METHOD AND MULTI THREATING**

## 2.3 Discussion

We can see that at the beginning the systems was going to take the tendency to be overload, the calls to the server are less because are few task that the client is doing to process the files but in the second case when we put on the method of cleaning we started to control more unexpected situations in the HTML file that’s make the client spend more time to make his own file and more calls on the system are done so in the second case the calls are increasing because the system is taking more time in cleaning all the files but is making sure that the final document that the server is going to receive will be clean and without mistakes. The final states with the multi-threading is taking a normal tendency, its increasing till it has to handle 3 files, at the four the throughput is decreasing so we can say that at this point the server spent more time working in process the full document at this point the behave of thr throughput is decreasing and is easy to understand that more jobs or files to process the throughput will be less. We can predict that if the number of files increase the system will be on a point that may collapse by overload

**TIME CLEANING AND BEHAVE OF THE MULTITHREADING**

**SECOND STATE WITH CLEANING METHOD BUT WITH OUT MULTI-THREATING**

**LAST STATE WITH CLEANING METHOD AND MULTI-THREATING}**

In the case of multi-threads, the system has better results but with more number of documents we see more fluctuation in the processing job as a result of accepting each job when it arrives to the server. In the result we also appreciate that depending of the number of threads and the threads of the CPU the time can be reduce or can increment if the program is not taking in account concurrency and the number of threads, so if we are overheat the time can increase. We also see that depending of the concurrency of the threads the time can increment or decrease, but in general the system obtains better results than in the other systems.

In the second State without multi-thread we appreciate that the respond time is more stable and the time don’t really fluctuate. We think that it depends of the size of the documents and the number of tags in the text.

# 3. Effect of Document Size

Here we focused on how long the server is working with the document on the method in which the server is cleaning, counting and generating the final file where is going to show the number of words per document. In order to evaluate this we decided to load different number of files.

## 3.1 Time Spent in Server

**FIRST STATE WITH OUT MULTI THREATING AND CLEANING METHOD**

**SECOND STATE WITH CLEANING METHOD BUT WITH OUT MULTI THREATING**

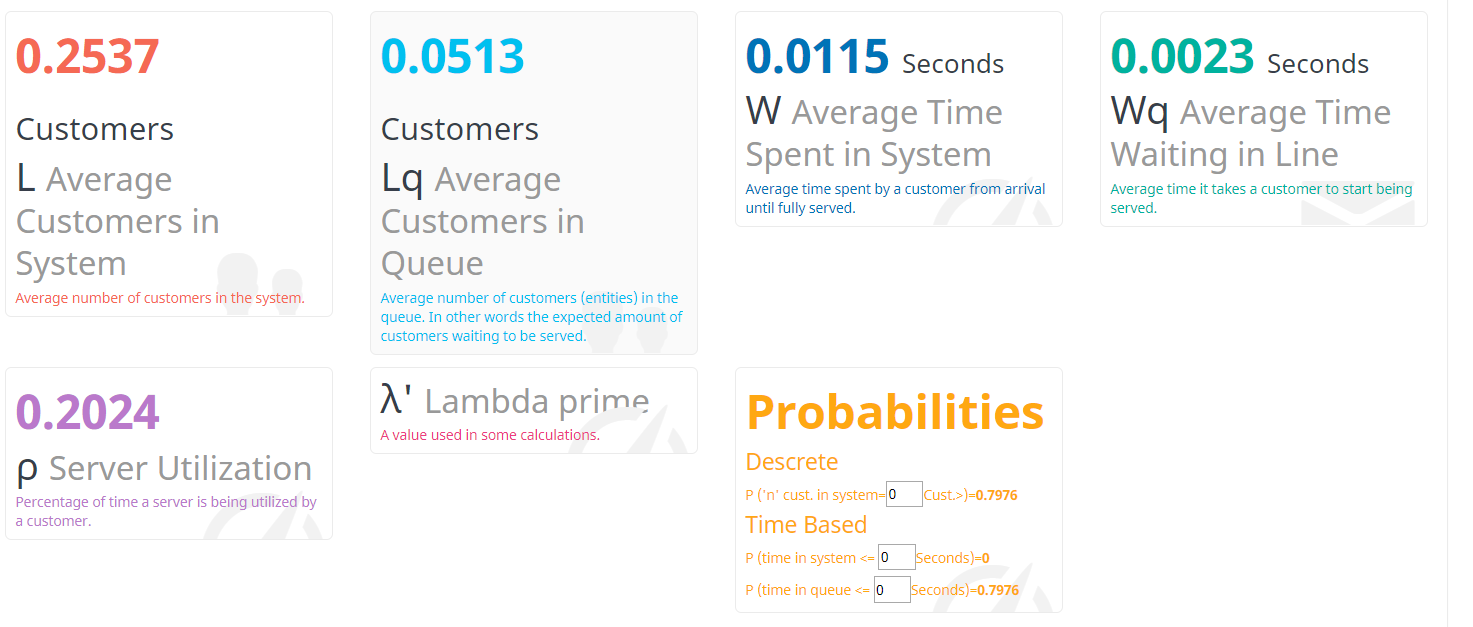
**LAST STATE WITH CLEANING METHOD AND MULTI THREATING**

## 3.2 Discussion

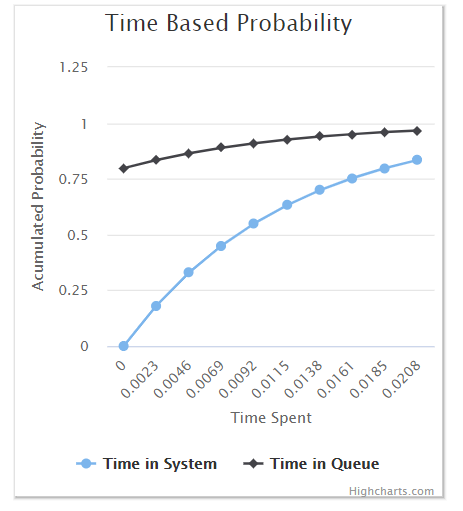
## The graphics shows the way how long it takes the server to count the full document that has received from the client and what we can see is that the tendency in the first and second phase when the threads has not been implements is that the time is increasing according the number or documents are being loan in the system, when we implement the multithreading we can see that at some points the time is decreasing because the idea of incorporate threads is in some way reduce the time the server takes to count all the files.

# 4. Modeling

## 4.1 M/M/m

With all the implementations, with the cleaning method and the multi threatening implemented the predict behave of the system(response time and throughput), we made this evaluation with 1 server and a size of 10 files per client to process.

* Average Customers in Queue: 0.0513
* Average time spent in system: 0.0115
* Average time waiting in line: 0.0023
* Server Utilization: 0.2024



4.2 Discussion

We can see that after the implant of the multithreading the system is tending to improve his response time and still being stable, the average time waiting in line is low and is tending to stabilize. If we look the Server Utilization we see that is 0,2024 , 1>0,2024 so the system is stable. We also appreciate in the time-based probability that we stay more time in the queue than in the system. So, we can think that we have to many clients in the queue and we can try to have more clients in the system so we reduce the time in queue.