CMPE Lab 2 - Report

Submitted by Keerthi Akella(013858819)

# Introduction:

Grubhub is a food ordering and food delivering online application which connects users to nearby restaurants. Through this app buyers can order their favorite food from the nearby restaurants, can check order status and upcoming orders. The restaurant owners can display their menu items and take orders from buyers and can update the order status. The technologies used to build this web application are react, node js, MongoDB and kafka.

The main technical goals of this project are Implementing RESTful webservices, reusing react components and testing the application using various tools like J meter and Mocha.

# System Design:

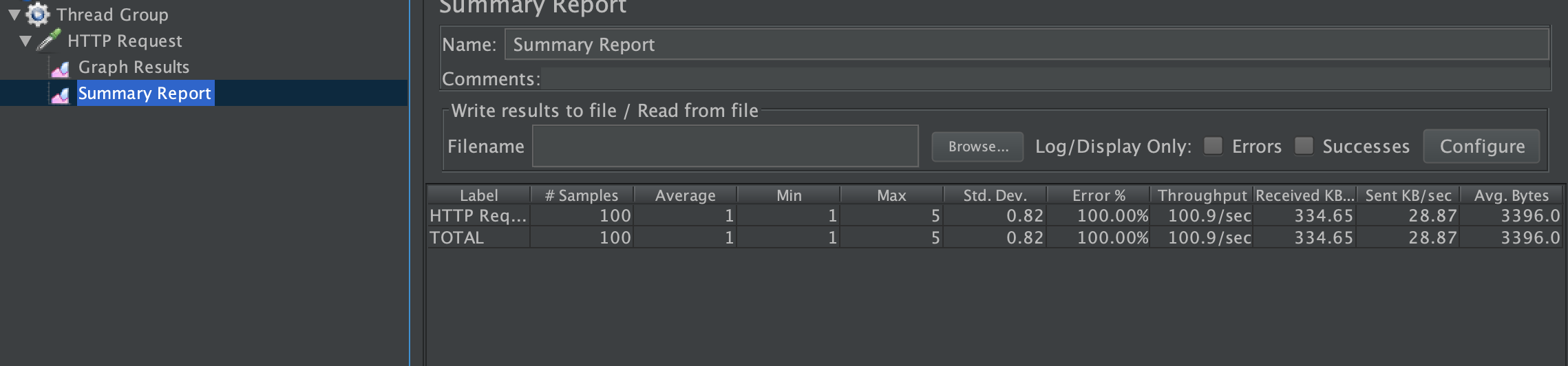


# Performance

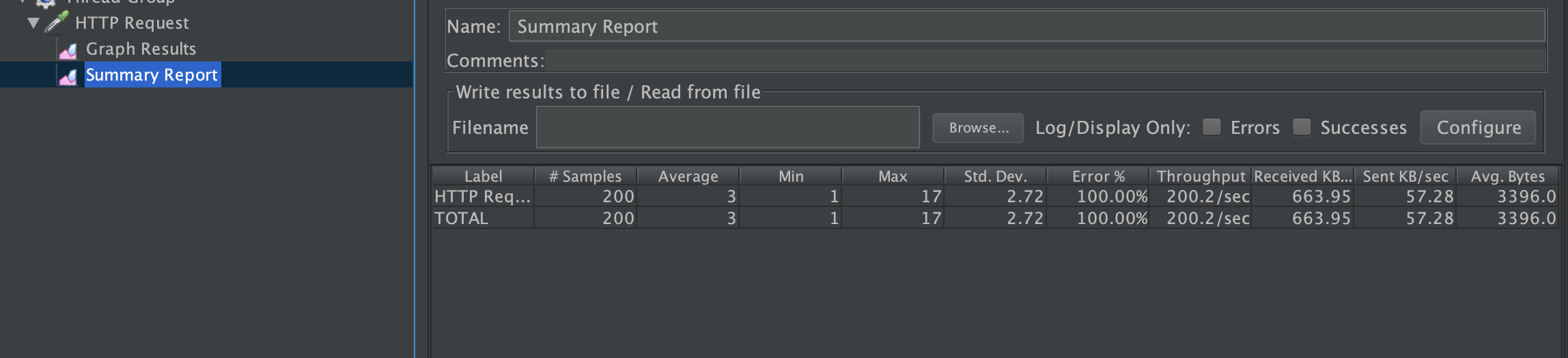
## J meter:

## With pooling

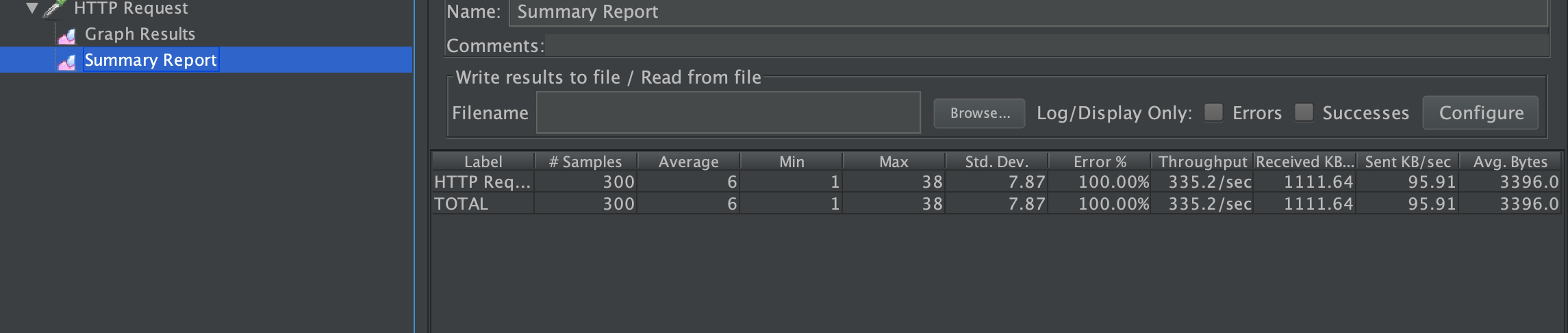
### 100 Users



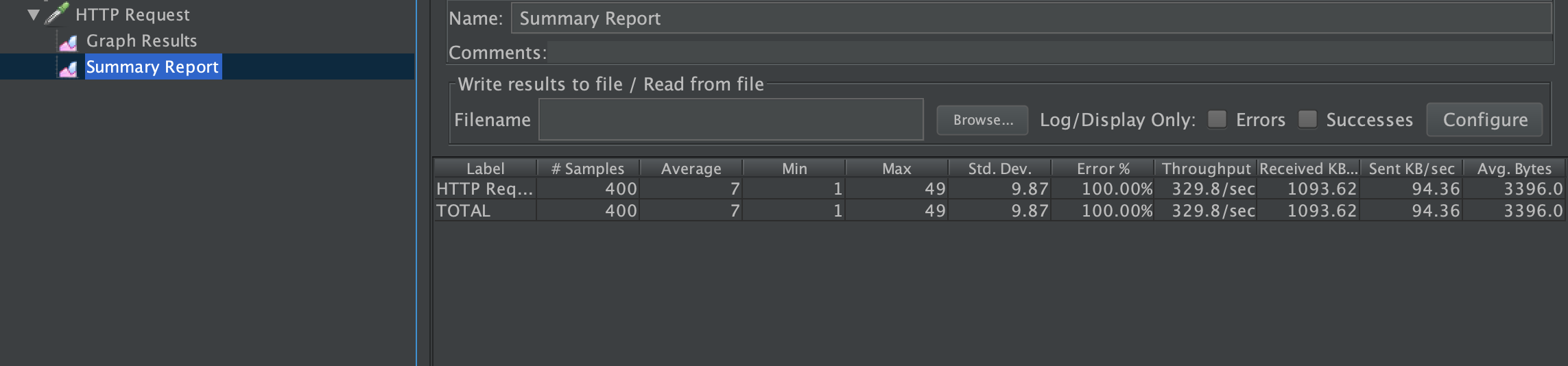
### 200 Users



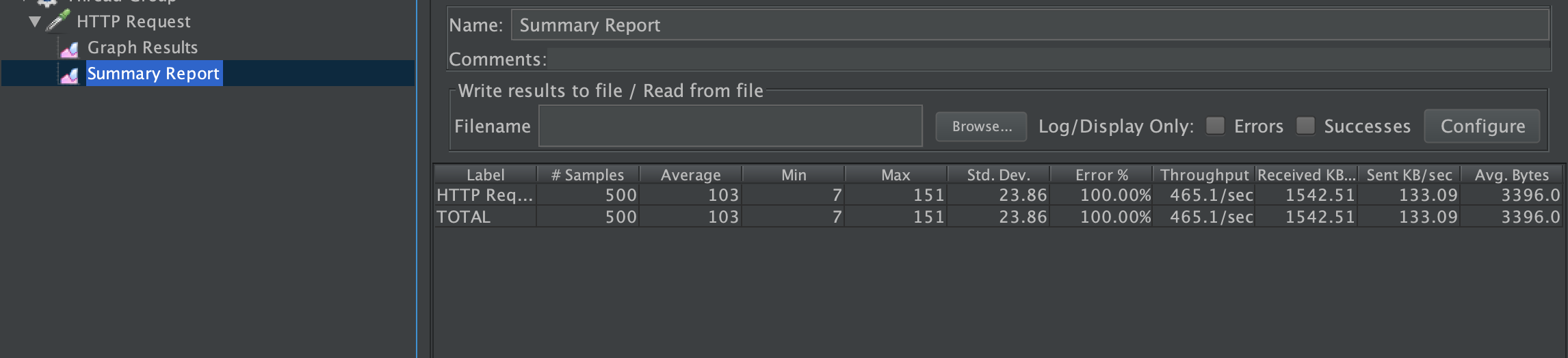
### 300 Users



### 400 Users

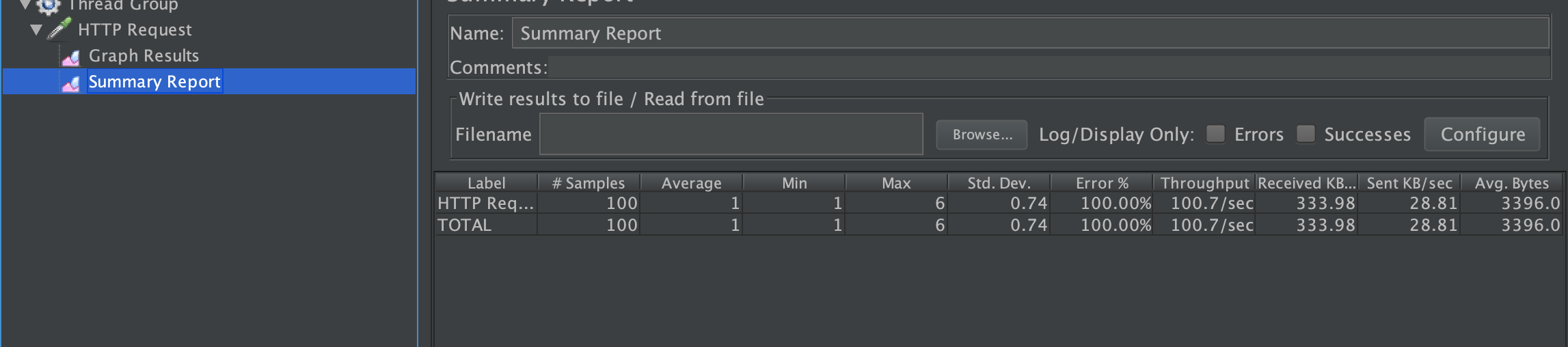


### 500 Users

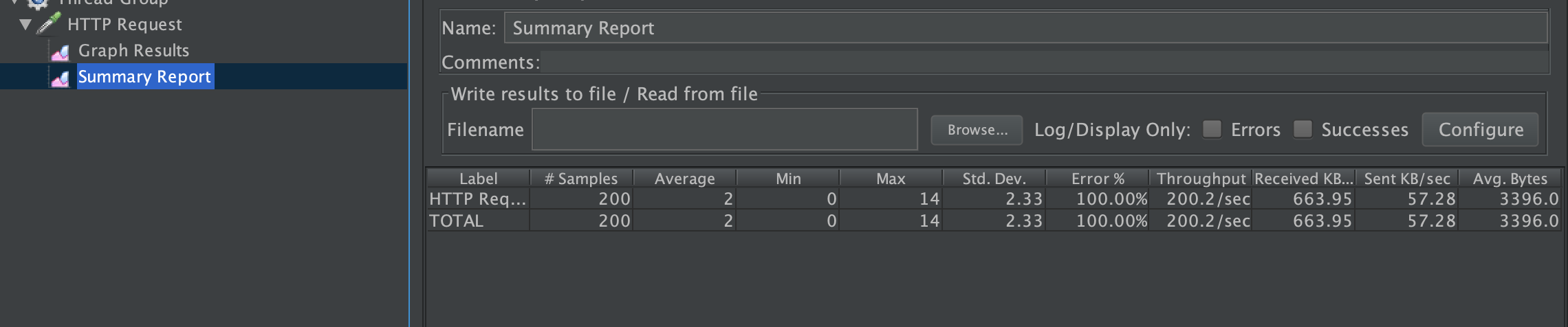


## Without Pooling

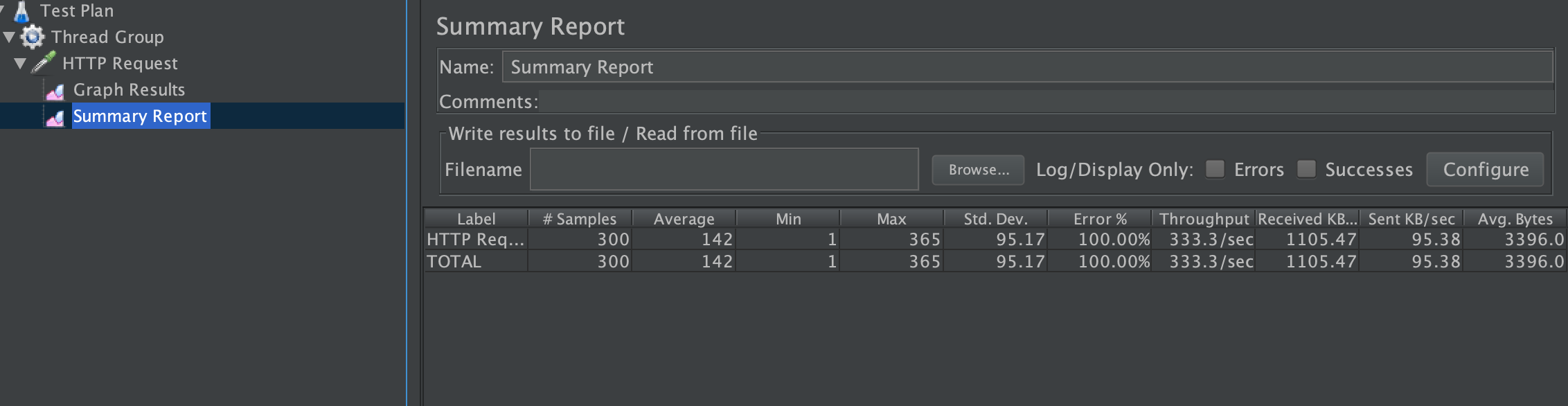
### 100 Users



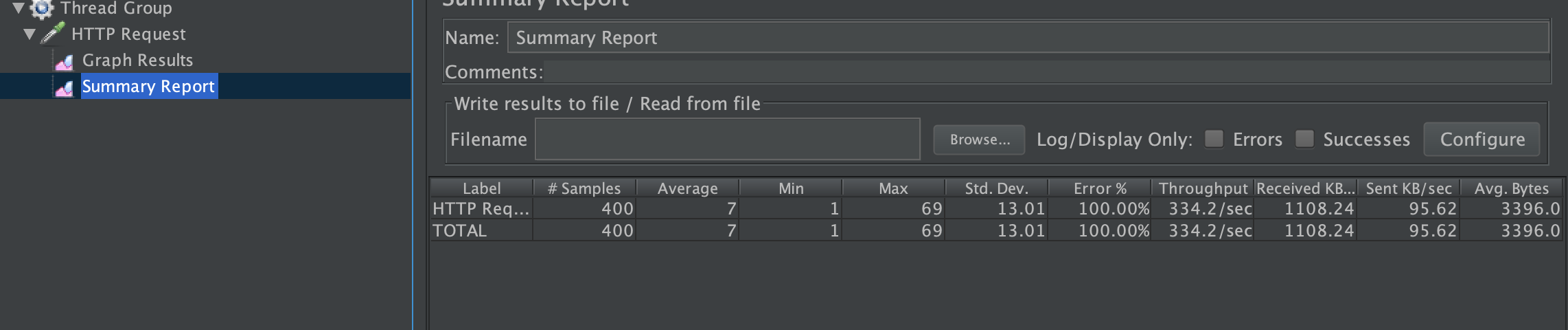
### 200 Users



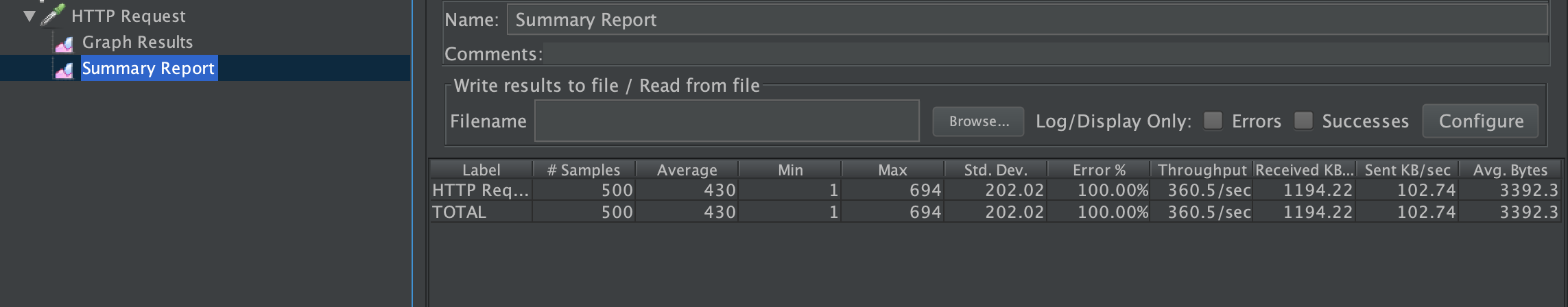
### 300 Users



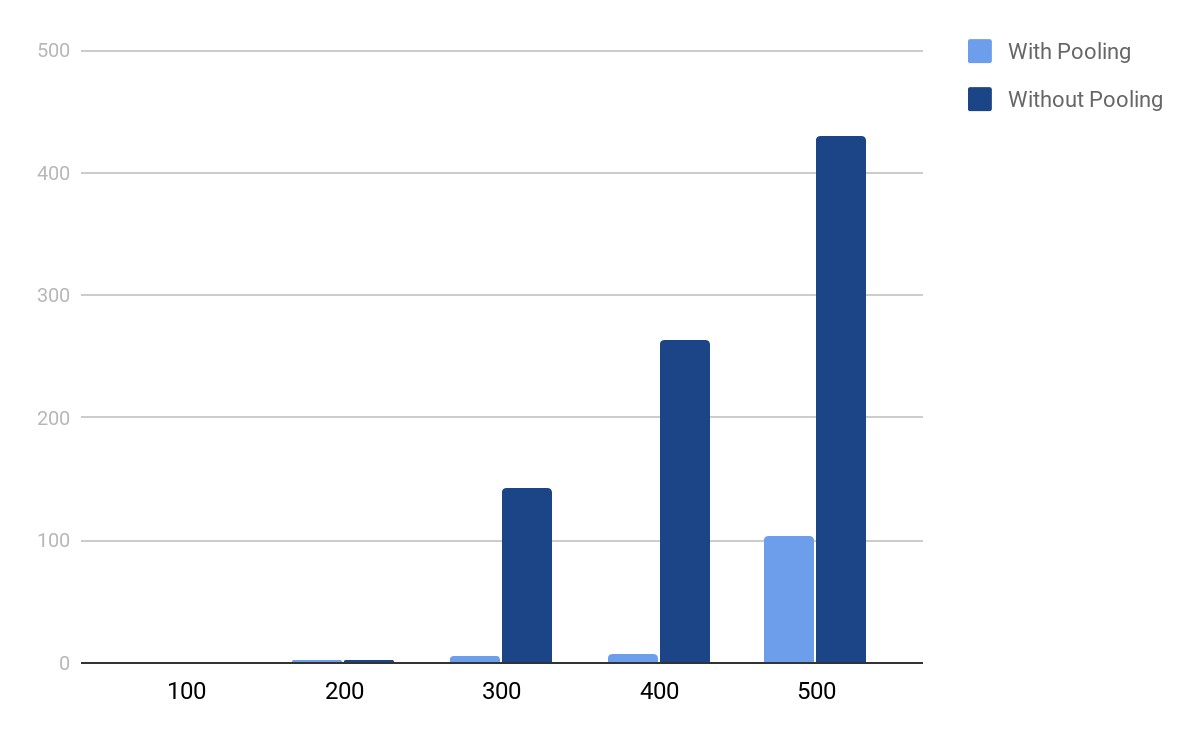
### 400 Users



### 500 Users



## Analysis Graph:

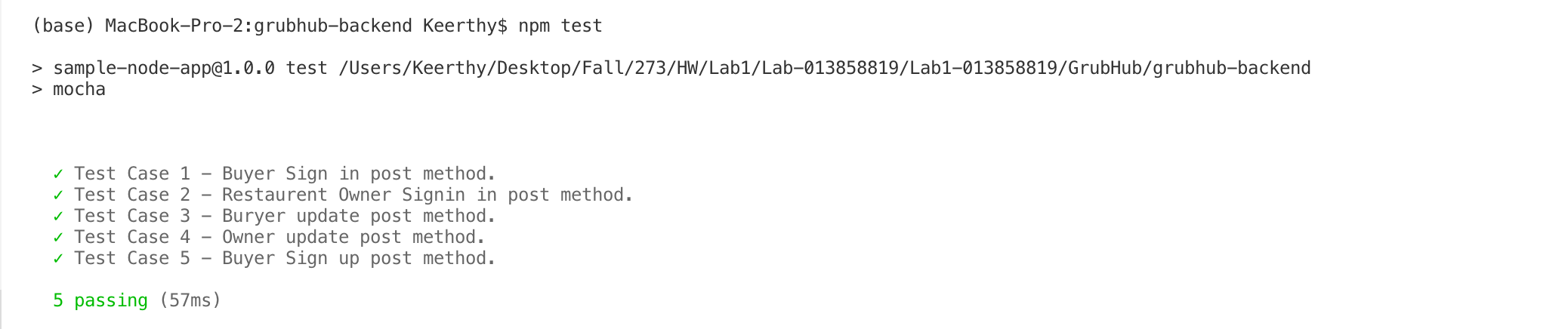


From the above graphical representation we can see a drastic difference in average times between using a connection pool and with not using connection pool.

Case1 (without connection pool): Here, there is a lot of load on backend. The average time taken is very high compared to the case ‘with connection pool’

Case2 (with connection pool): When we use a connection pool and create one upfront we have a reduced connection creation time, controlled resource usage and simplified programming model.

## Mocha:



# Questions

### 

In the lab1, I used hashing process to maintain password security. I validated user by checking his details using cookies and returned a parameter ‘validUser’ with true/false.

In lab2, I am using passport authentication from which i can able to store user’s information in payload and tokens are signed. By this it gives access to only authenticated users

### 

We all know kafka can handle 3,20,000 messages per second. But compared to the Jmeter tests from lab1 and lab2. The average time is higher using kafka than without using kafka.

The reasons could be, Network latency: In lab1 the architecture is simpler there is frontend, backend and database. But in lab2, there is frontend, backend and kafka backend and to communicate to the database it is through kafka.

As we said above to communicate database it is only through kafka, so there must be time interval between request and response

### 

MongoDB is schema less and structure of database is clear. It uses internal memory for storing the working set that enables the faster access of data. So, to implement MongoBD for rich queries and large data that results in good performance. We can access the data with single command.

MySQL is a relational database, it has joints in case if the data is distributed in multiple tables and it has a clear and organised schema. To store data that has to access from multiple tables and to have a clear picture of database we can use MySQL.