# CMPE 273 : Lab 1

Student ID : 012445563

**DROPBOX ( SERVER 2 and CLIENT 2 )**

***Goals of the system:***

The main goal of this system is to develop a dropbox like application which provides users functionalities like file upload, download file, share files via email id, star mark important files with the of React JS and SQL server.

***Purpose of developing the system:***

The purpose behind making dropbox is to provide user an online storage system to maintain his important files on the cloud.

***System Design:***

* The design technique used in this system is MERN Stack but MySQL as databse instead of traditional MongoDB for MERN stack. Also, this system follows client-server architecture where all the functionalities are performed on server using Node JS (N in MERN Stack).
* Here, using Node JS, asynchronous calls are made to database using callback functions which reduces the wait time of the system by performing other actions while waiting for database IOs.
* MySQL’s in-buit functionality connection pooling functionality is used to provide 20 concurrent database connections from connection pool. To connect with database tables, mysql dependency of Node JS is used. Client-Sessions on the server side help to maintain one session for user.
* To connect Node and React, CORs functionality is used so that clients and servers which are running on different ports are connected and parameters can be asked from React side to node side.
* The system is design using react JS as front end and MySQL as database. The great benefit of using react is that we need the components are automatically rendered whenever a small change occur in the state but it does not re-render all the components. Only those components which are changes are re-rendered e.g. in our system, list of files is displayed via state so whenever any file is uploaded, only this part is re-rendered and rest of the page remains as is. Additional functionalities such of react such as material-ui provides a good UI pattern for react components.
* Also, bcrypt encryption algorithm is used for storing passwords in hashed form in the database which provides additional security mechanism and prevents the system from attacks.
* Hence, this design is most suitable for real time fast application such as Dropbox.

***Performance of the system:***

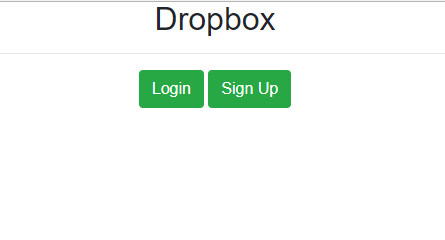
The developed system has following benefits :

1. Connection Pooling : With the help of connection pooling , concurrent database requests are handles in much efficient way than without connection pooling. This was tested on JMeter. And we can see that connection pooling provides much faster results and hence decreases the response time.
2. The system serves all the request very quickly with all the files ready.
3. The system provides all the proper data as and when required.
4. As React JS is used, the system refreshes only updated part of the system hence making the system respond to changes much faster.
5. Using the bcryot algorithm, password is saved and hence security measures are taken into consideration.
6. To reuse the code, API calls are written in modular way such that the same calls are made for similar functionalities and it enhances the performance.

The combination of React JS as front end and Node JS as backend increases the performance by using unique functionality of both of the JS.

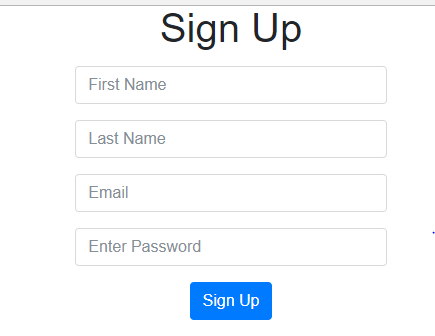
Application Navigation in the built system:

1. Initial page:



1. Sign Up:

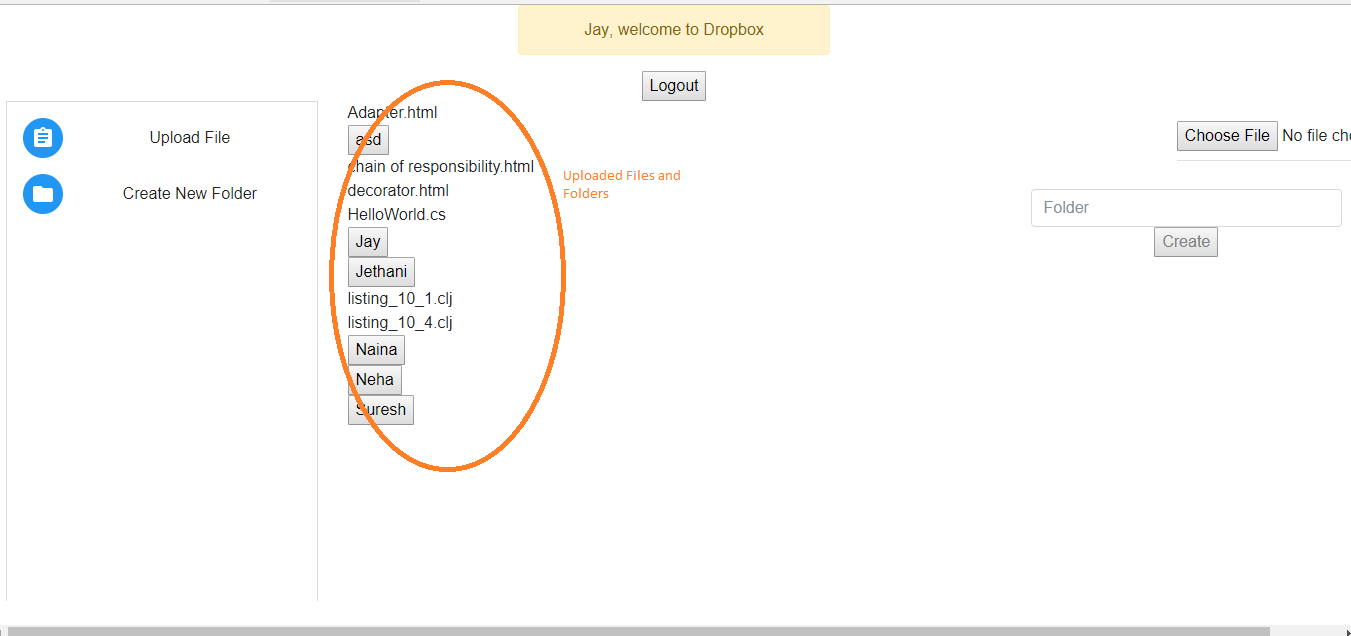
A user can sign up to dropbox with a unique email id by entering first name, last name, email and password. Also, these fields have validations on each field being non-empty, emaild should of proper email format and password must contain one capital, one small letter, one special character and one number. These validations are performed on client side and returned as JSON response.



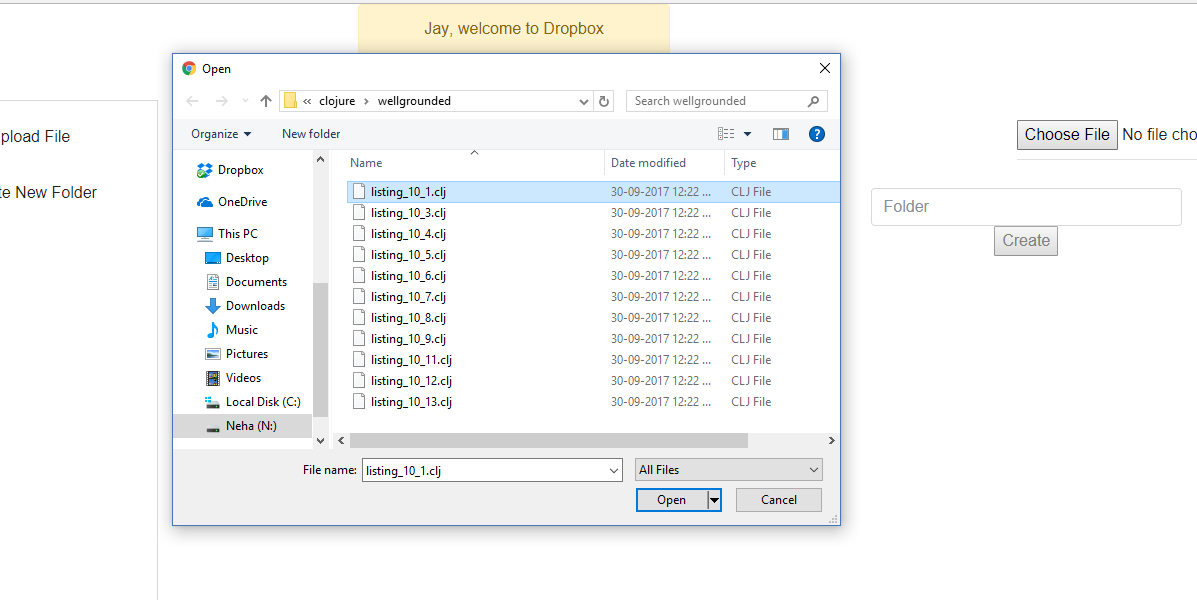
1. Login:

If user already exists then he can sign in using email id and password.

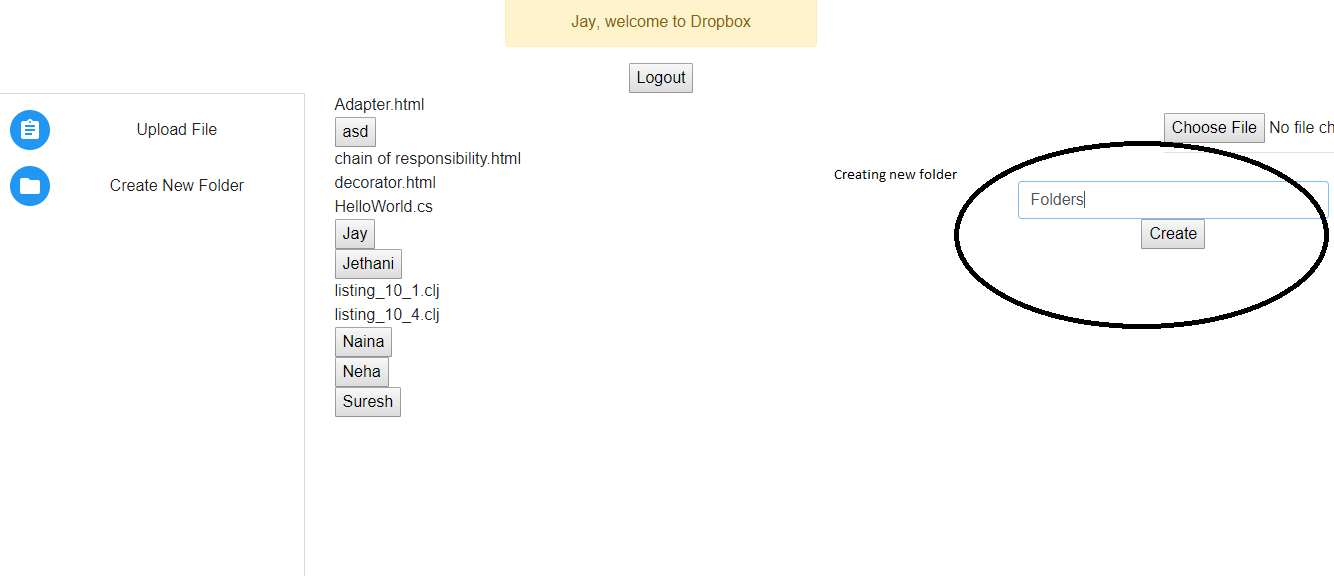
1. On the home page, user has option to upload file, create new directory and a list of all his files and folders which he has uploaded. User can create directory inside directory also and can upload the file anywhere in the hierarchy.



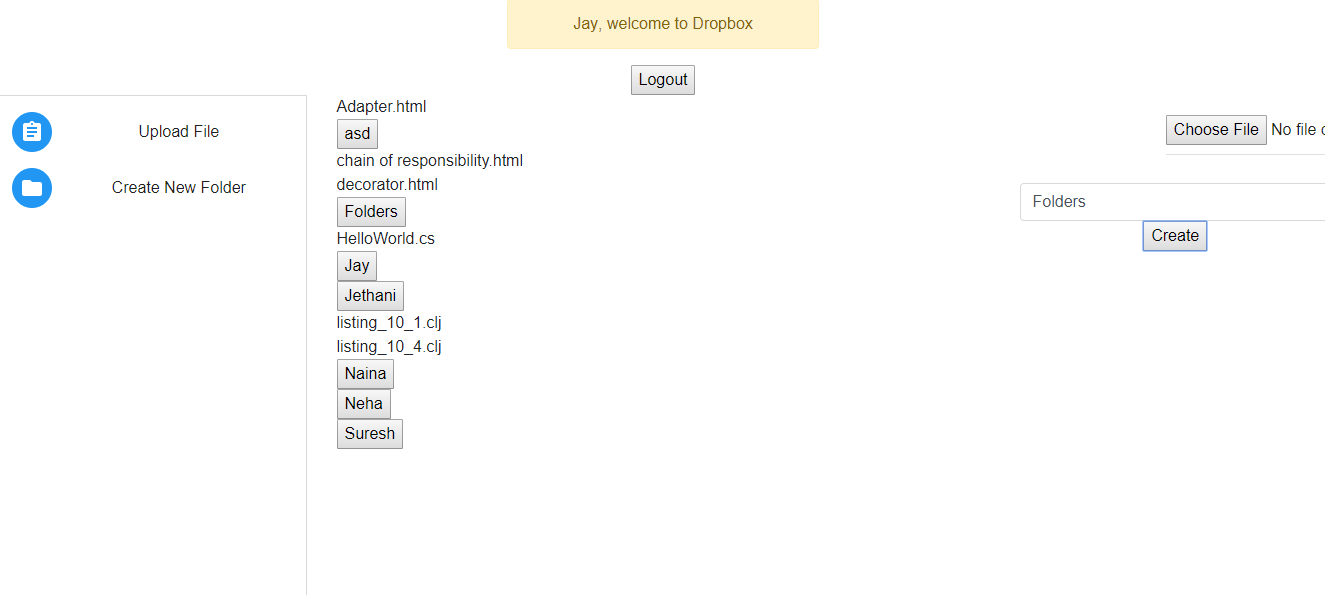
File Upload :



Create Folder:



Created folder ‘Folders’ is added to the list



Inside folder ‘Neha’



1. By clicking on the filename, a user can download the uploaded file.x

If a user exists with the same email id, it won’t be allowed to sign up with the same email id again.

***QUESTIONS:***

***Explain the encryption algorithm used in your application. Mention different encryption algorithms available and the reason for your selection of the algorithm used.***

The word encryption means hidden. Encryption is used to store sensitive information such as password, SSN to store in a form which is not directly readable. This encrypted form is made via applying encryption technique on the original input.

Most used algorithms in today’s industry are:

1. Triple DES
2. RSA
3. Blowfish
4. TwoFish
5. AES
6. SHA-512
7. MD5
8. Salt and Hash

In my project, I have used bcrypt algorithm.

What is bcrypt ?

Bcrypt uses hashing tecgnique with multiple iterations of hashing. Bcrypt gives same hash everytime an input is given to it. This algorithm is based on blowfish symmetric block cipher cryptographic algorithm.

Bcrypt can stretch key as long as we want and hence it can make the process slower. So brute force attackes can be avoided using bcrypt.

***Compare the results of graphs with and without connection pooling of database. Explain the result in detail and describe the connection pooling algorithm used in your code.***

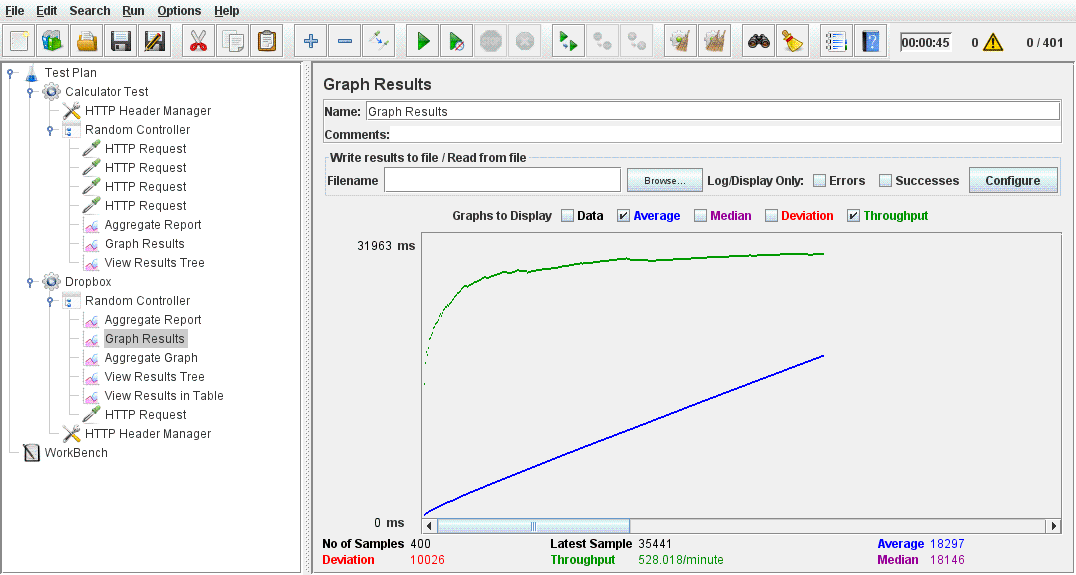
In software engineering, a cache of database connections called ‘connection pool’ is maintained so that the connections can be reused when future requests to the database are required. Here, instead of creating and destroying requests, we use existing requests whenever required. It saves the cost of opening the request everytime. If connections exhaust in the pool, new connections are added to the pool.

Comparision :

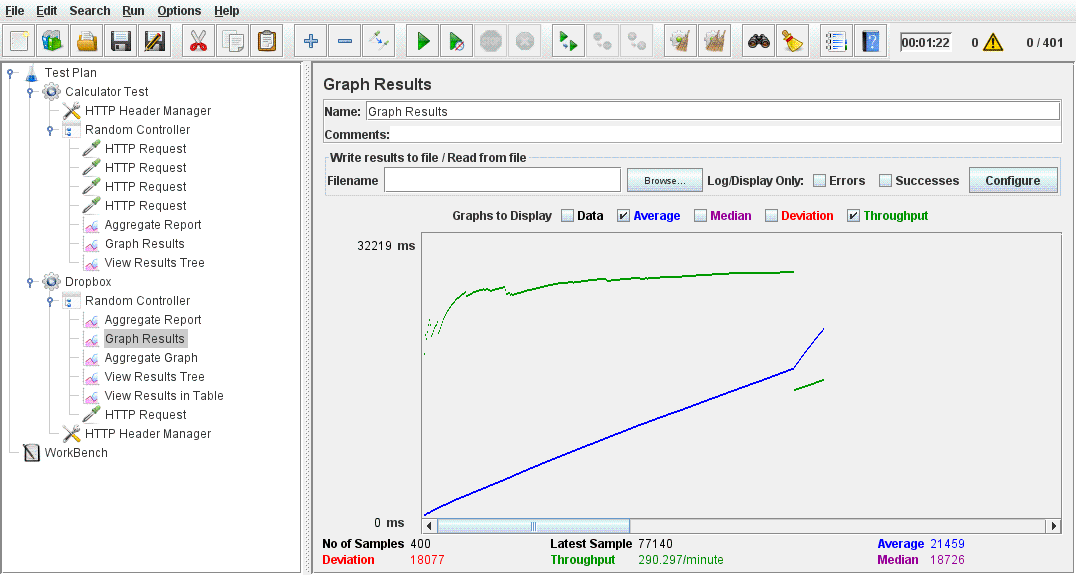
Average time taken to respond to concuurent requests is much faster when connection pooling is used for database connection. E.g. Througput for 400 users for simultaneous login request is 528 requests per minute whereas without connection pooling, this figure is much slower which is 290 requests per minute. Also, average time taken is smaller with connection pooling.

Following is the graph for 400 users simultaneously requesting login :

With connection pooling:

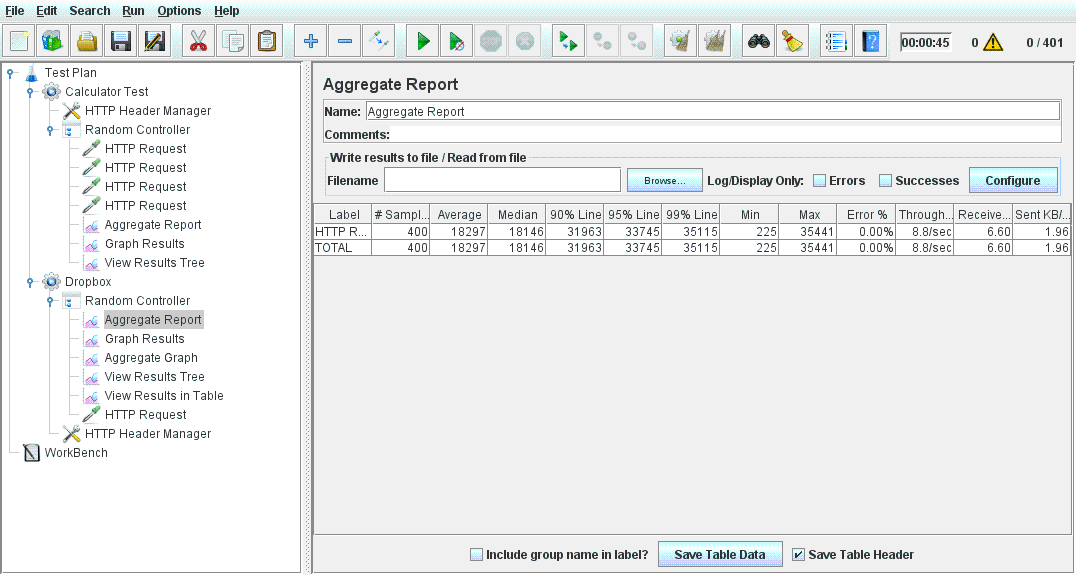


Without connection pooling :

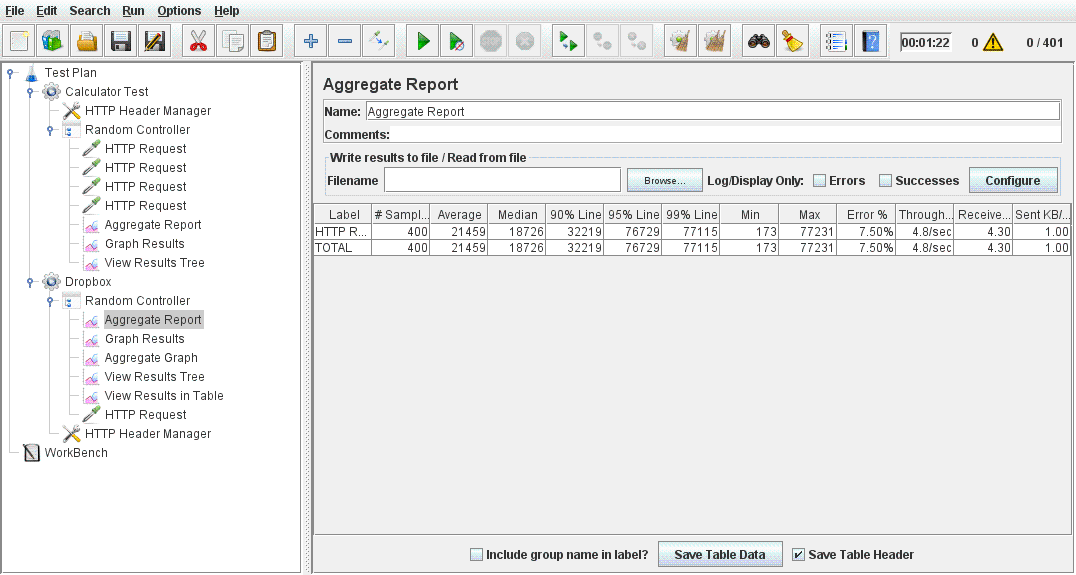


Aggregate Report :

( With connection pooling ):



Without connection pooling :



In my application, I have used default pool functionality provided by MySQL. In this connection pool creation, 20 connections are added to connection pool using createPool method of MySQL.

***What is SQL caching? What all types of SQL caching is available and which suits your code the most. You don’t need to implement the caching, write pseudo code or explain in detail.***

SQL caching is a technique where cache is maintained in front of a database to improve data access read/write performance. SQL caching reduces the number of queries hitting the database and we can directly use caching database memory. By SQL caching, we can distribute repitive query workload to a separate caching server.

The suitable caching technique for my code is Memcached.

MemCached stores data In a big hash tabke in terms of key/value pair. In the application, queries are used to fetch data when users logs in to authenticate the user. We can cache the sql query and use it whenever user logs in.

Explanation :

First, we need to create memcache and add it to server.

Query : “select \* from users where email\_id = ? “

Here, key can be SQL query and email id and corresponding result as value. If the key value pair is not present in database whenever a request is made , we get the result data from database server and store it to cache.

If the data in cache is not used since very long time, we can destroy that object and add more frequently used resultset.

Similarly, we can use other queries to store caching server such as displaying recent user activity.

***Is your session strategy horizontally scalable? If YES, explain your session handling strategy. If NO, then explain how can you achieve it.***

Yes, the provided system is horizontally scalable. The session I have used in the system is default client sessions. With the help of the sessions, data is available across all the servers if we use multiple servers.

Whenever user logs in , its session is created and it is maintained the server so user can activate same page via multiple tabs.

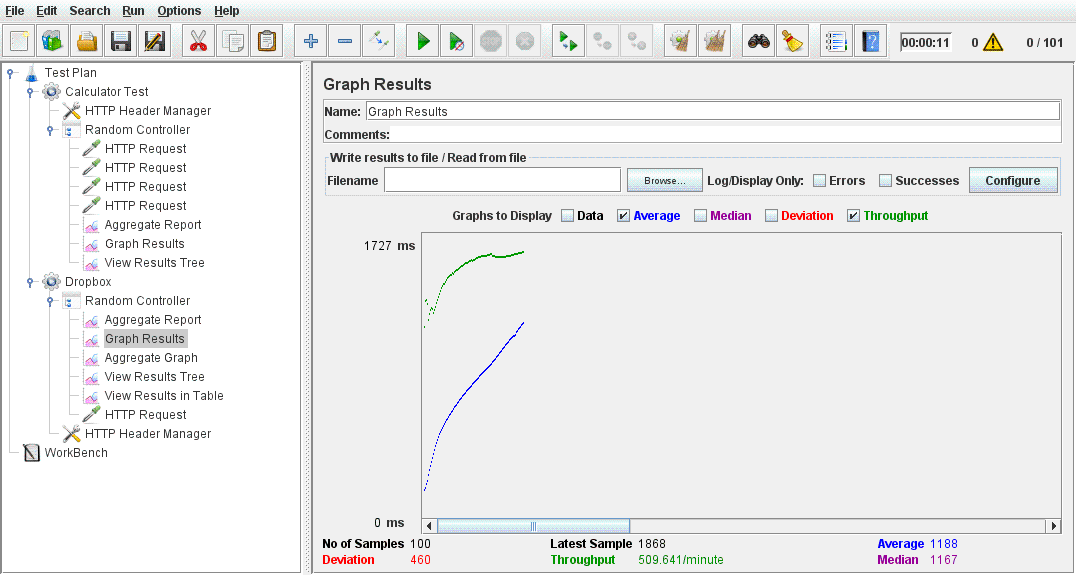
Here, server operations are called via multiple API calls. Each API call maintains the session for the user and hence horizontal scalable.

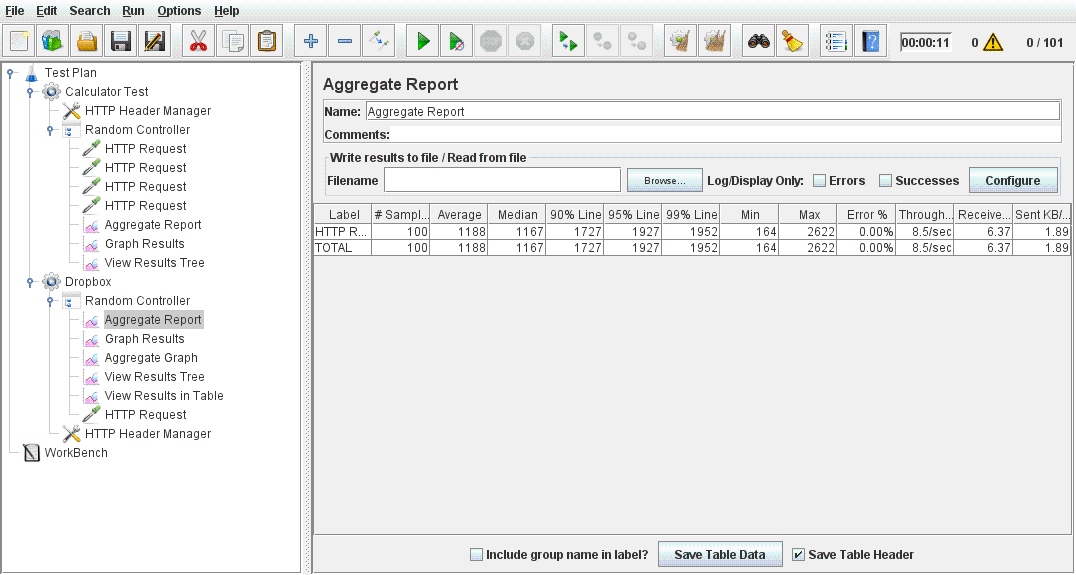
**Tests on JMeter for dropbox:**

**Testing for Login Method**

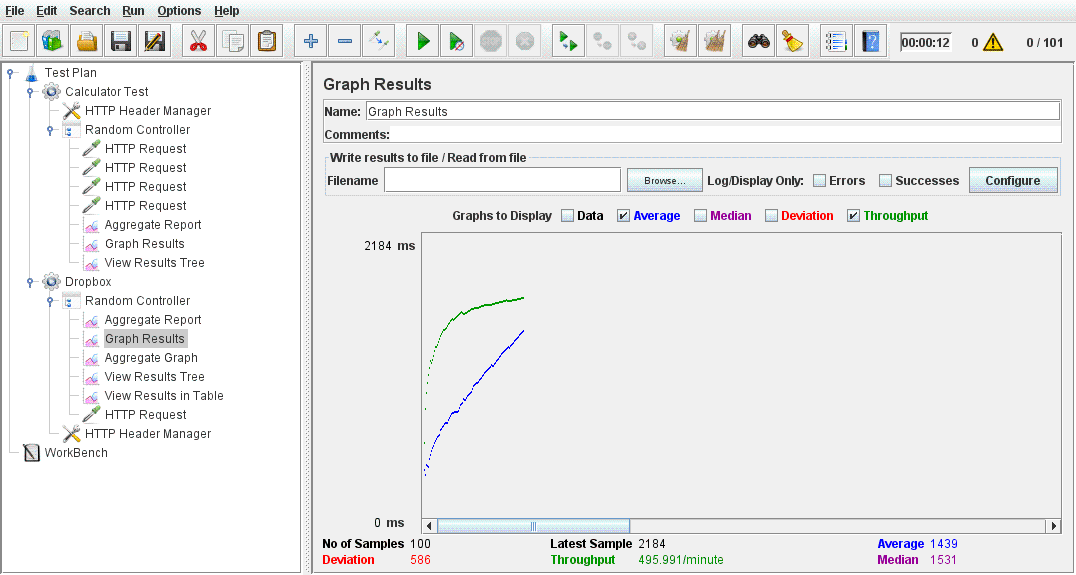
1. **For 100 Users**

**With connection pooling**

****

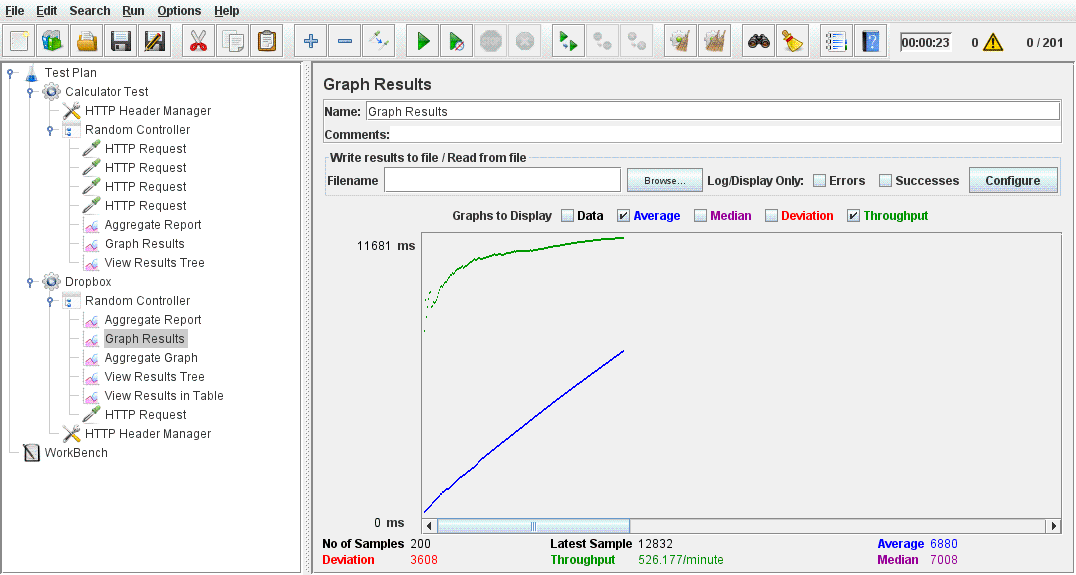
****

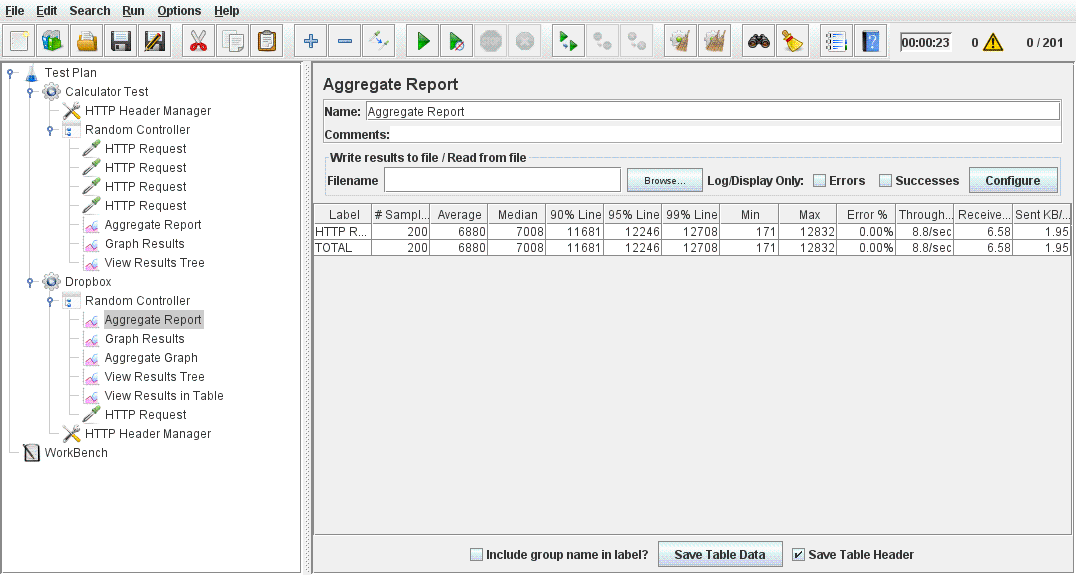
**Without connection pooling**

****

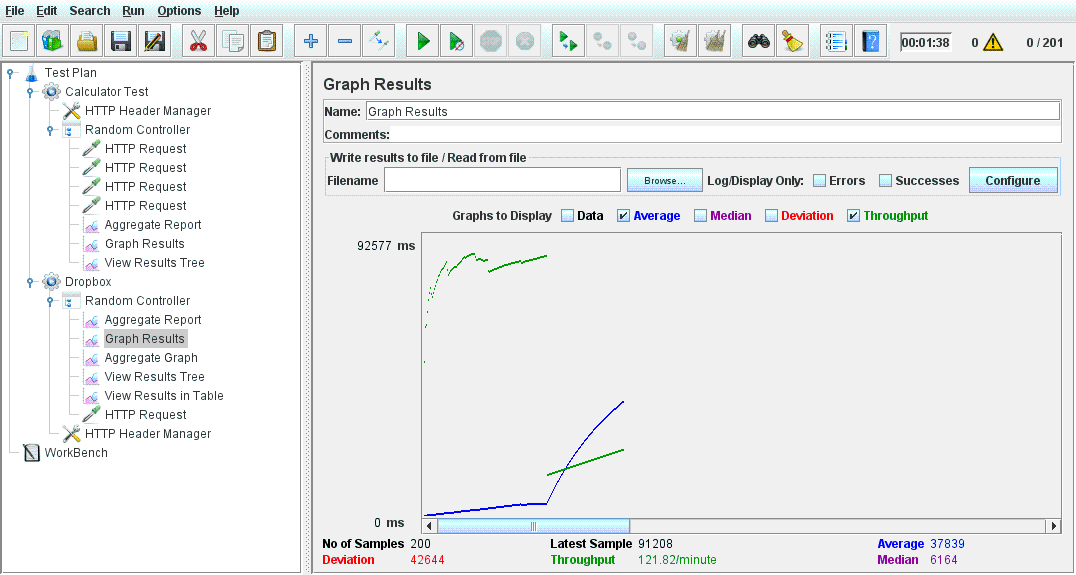
1. **For 200 Users**

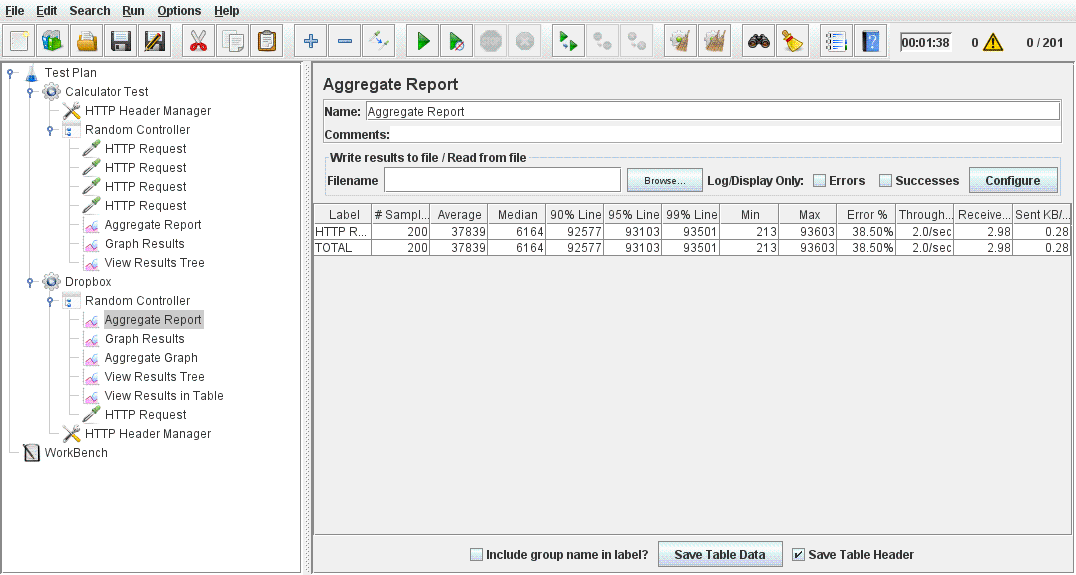
**With connection pooling**

****

****

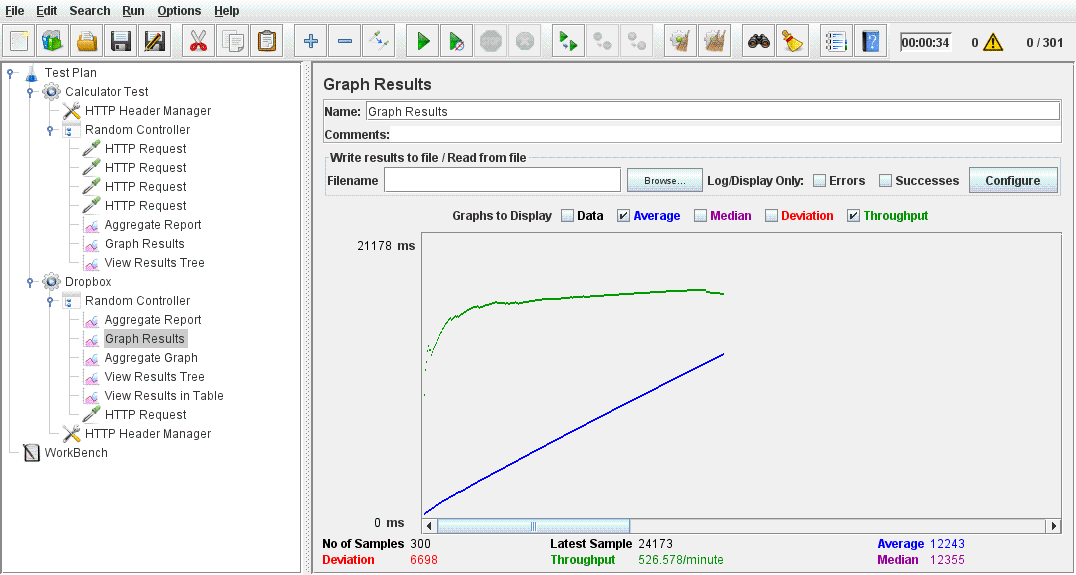
**Without connection pooling**

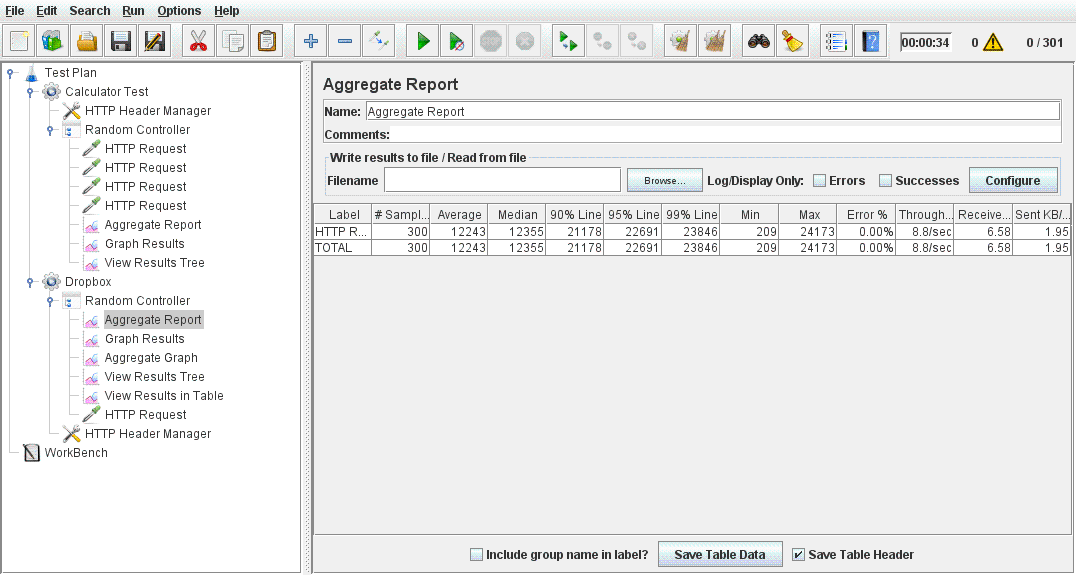
****

****

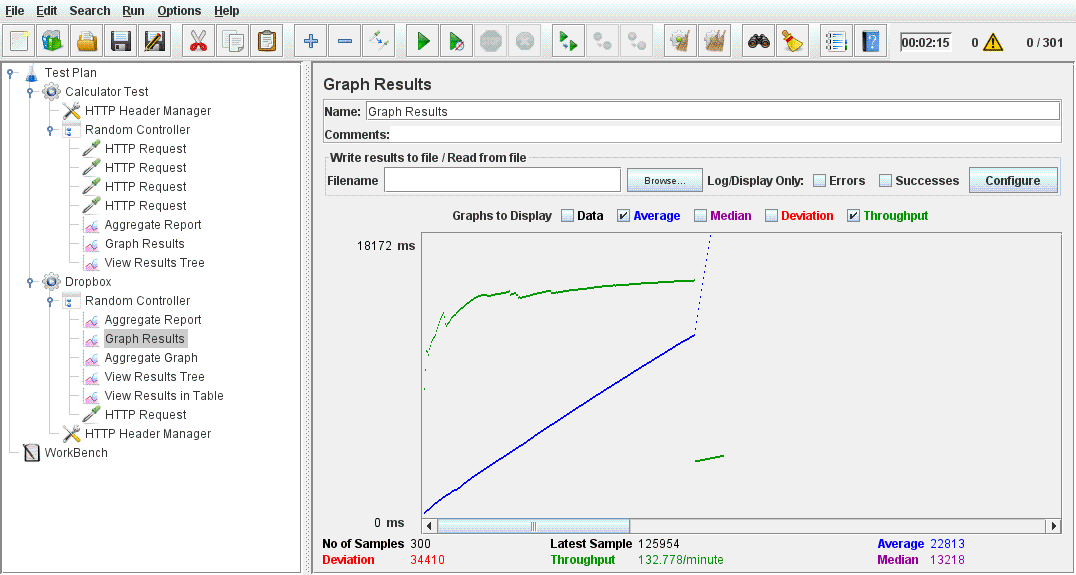
1. **For 300 Users**

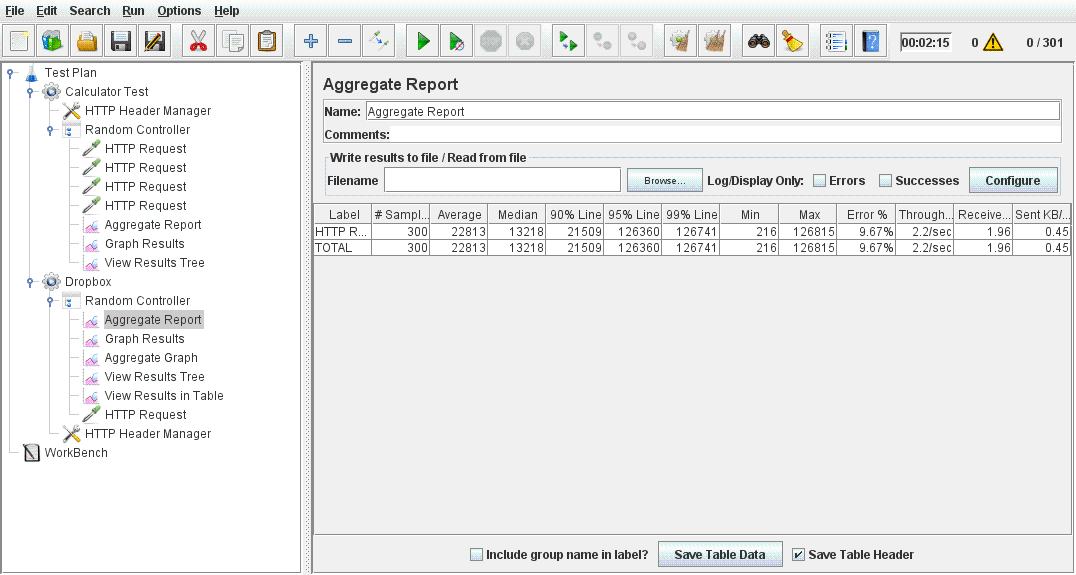
**With connection pooling**

****

****

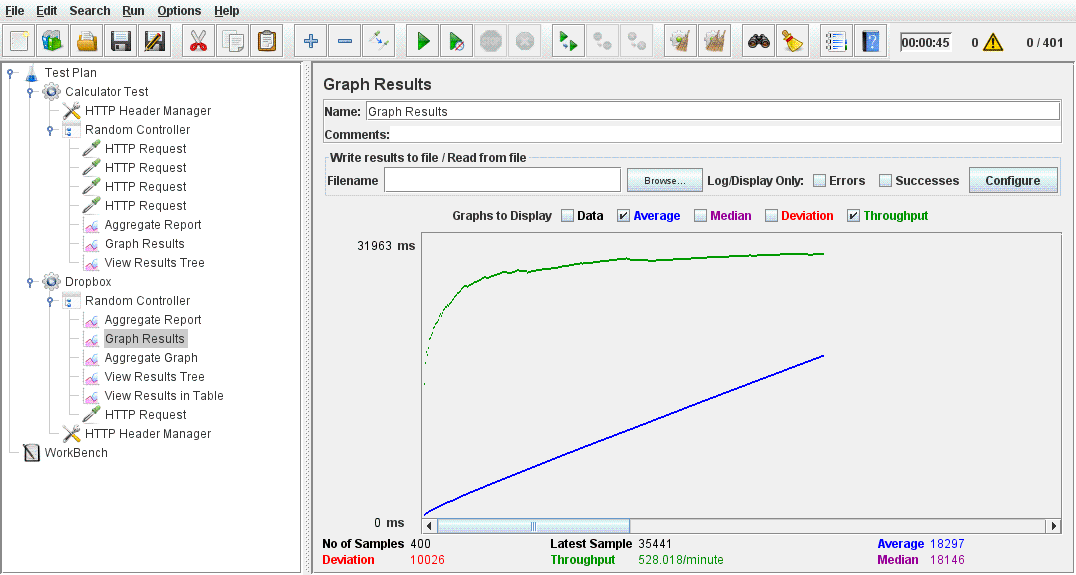
**Without connection pooling**

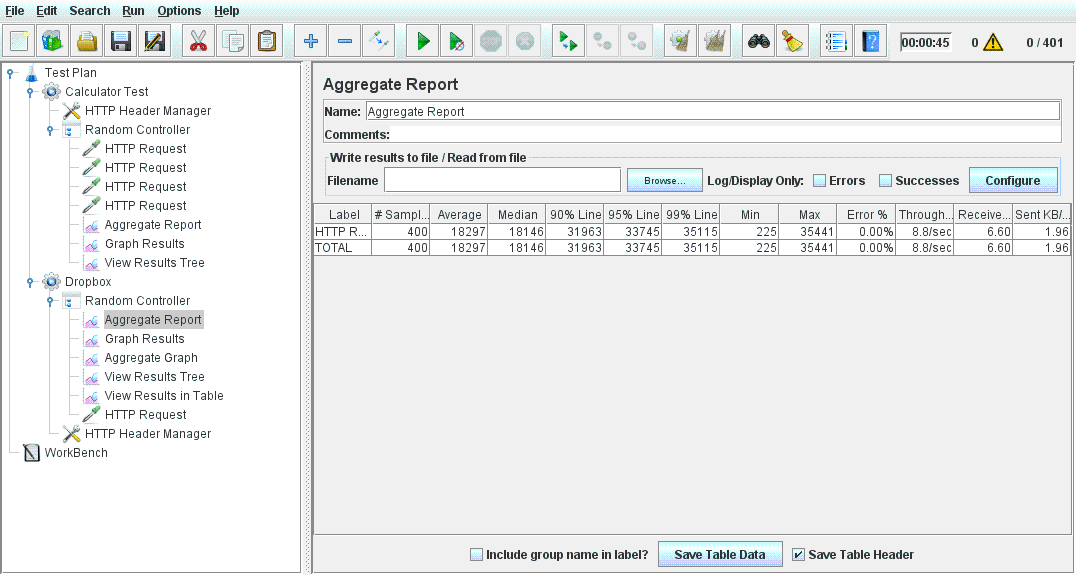
****

****

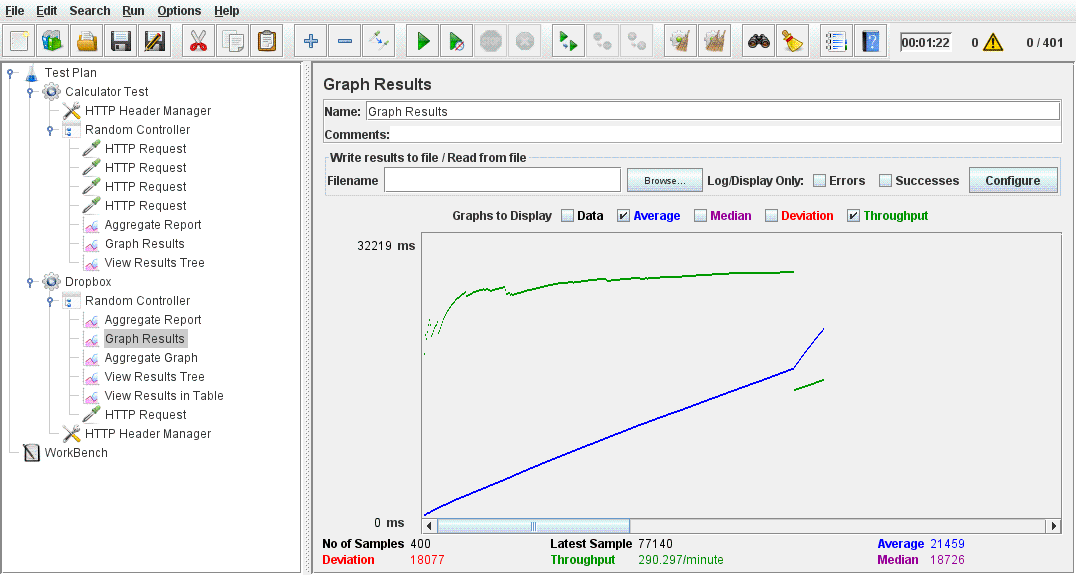
1. **For 400 Users**

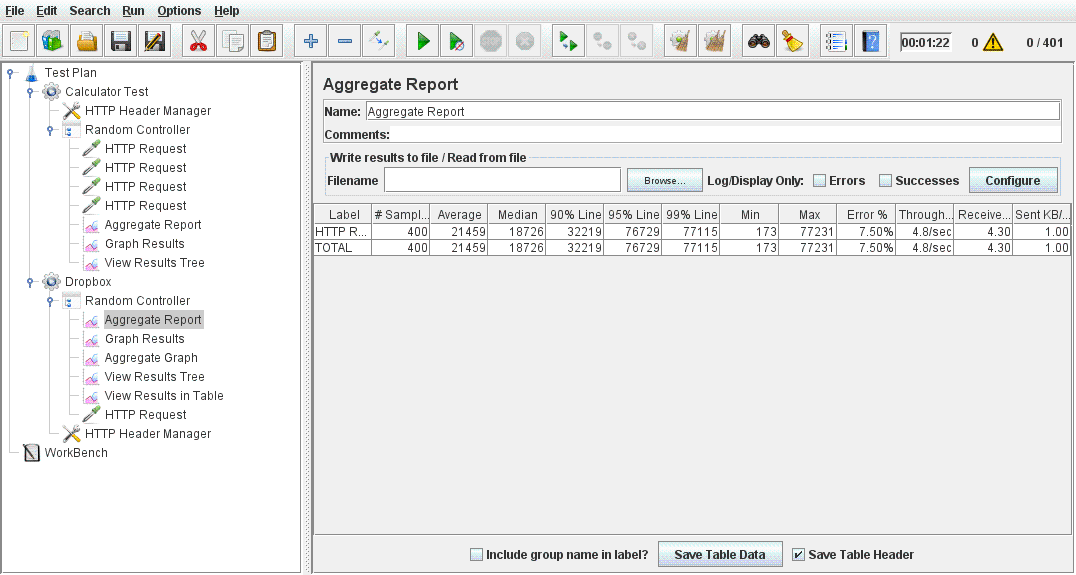
**With connection pooling**

****

****

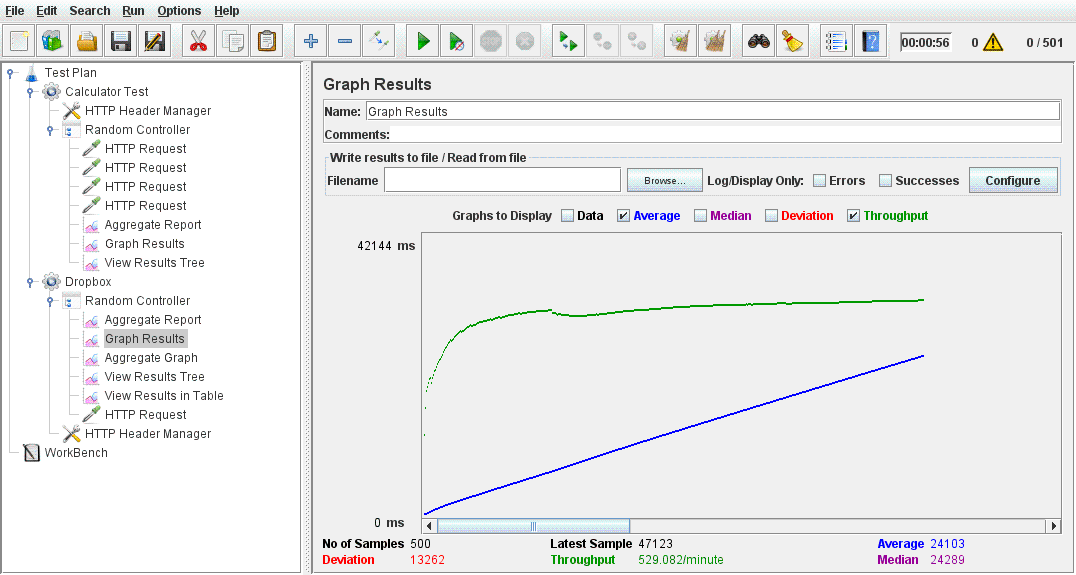
**Without connection pooling**

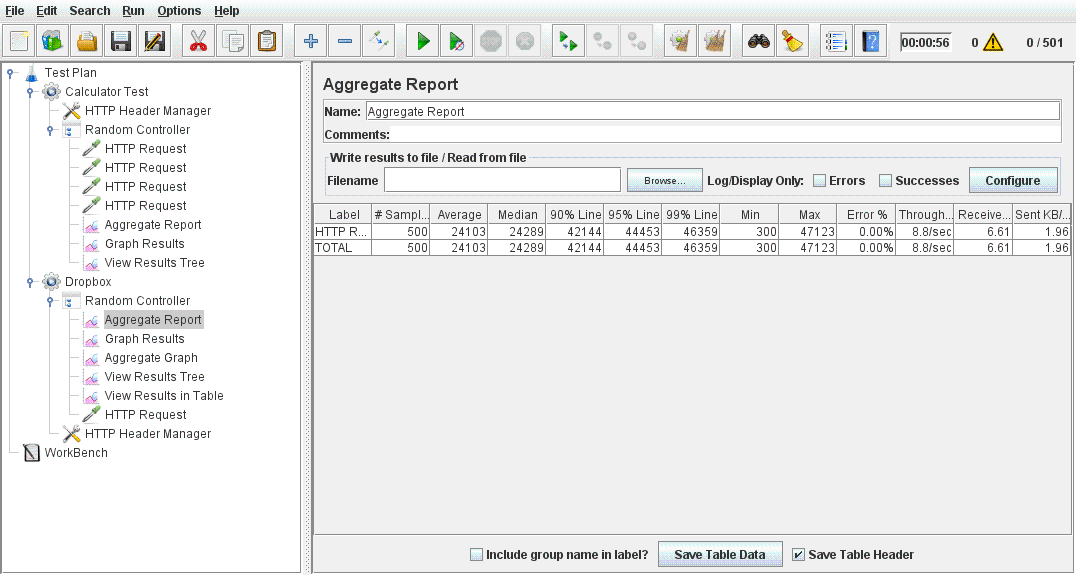
****

****

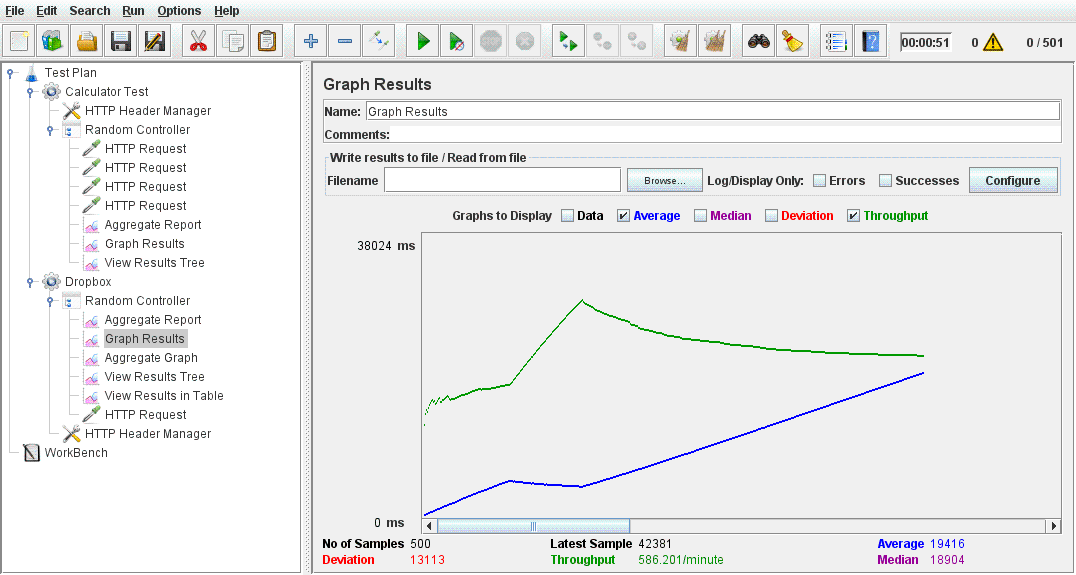
1. **For 500 Users**

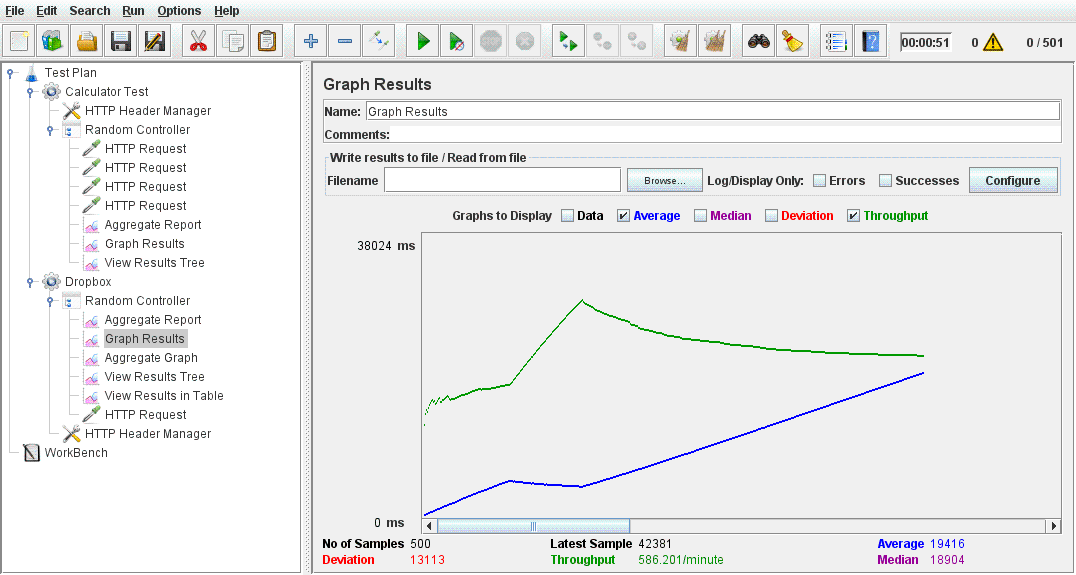
**With connection pooling**

****

****

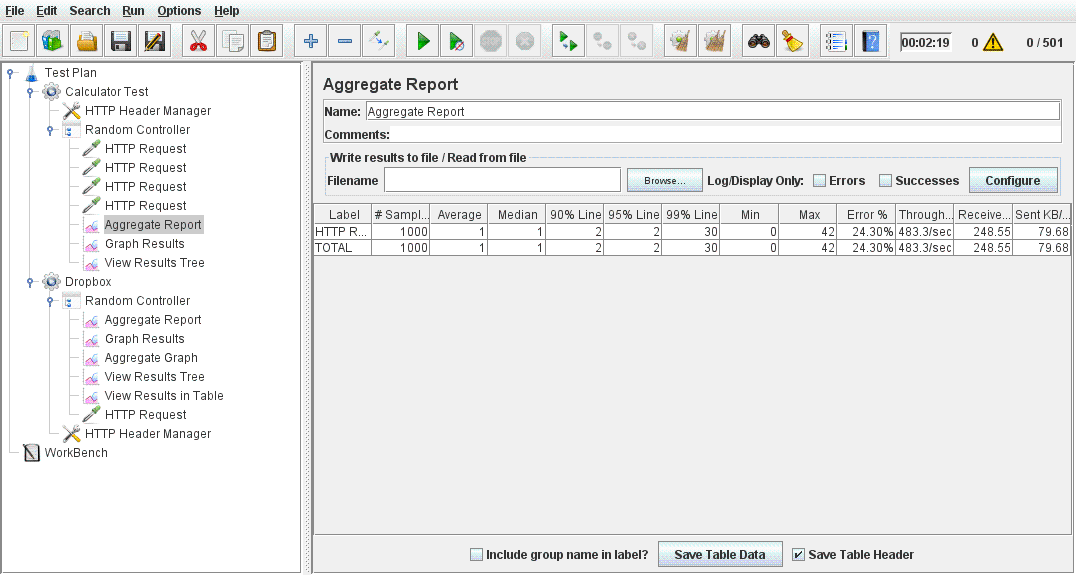
**Without connection pooling**

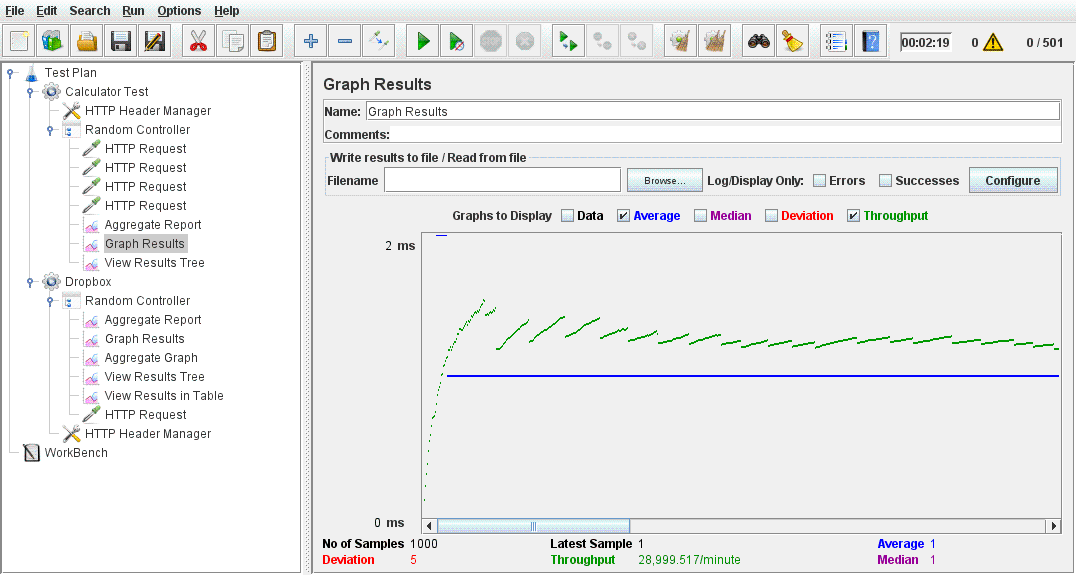
****

****

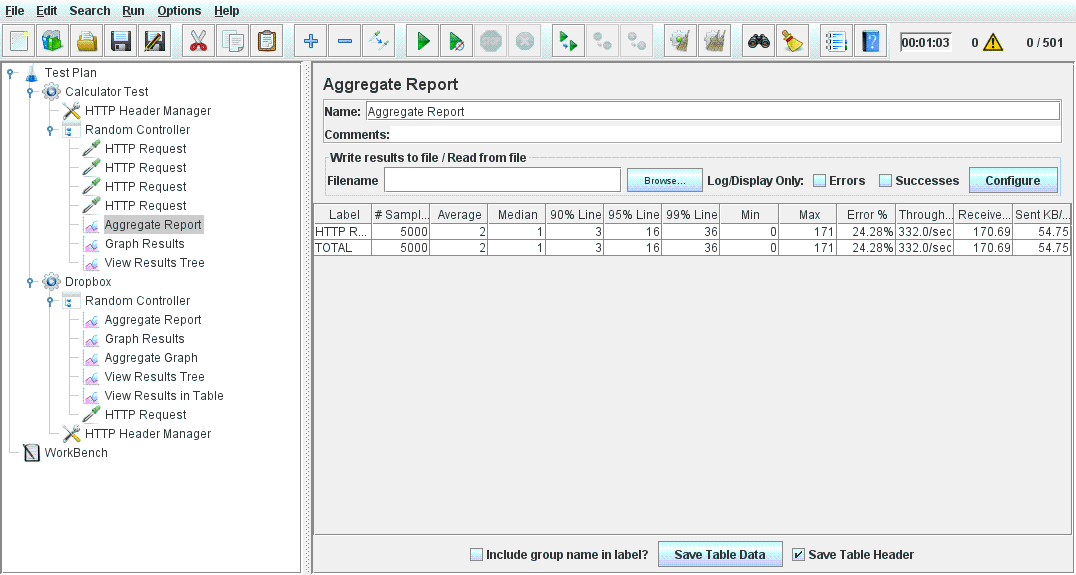
**Testing on JMeter for Calculator :**

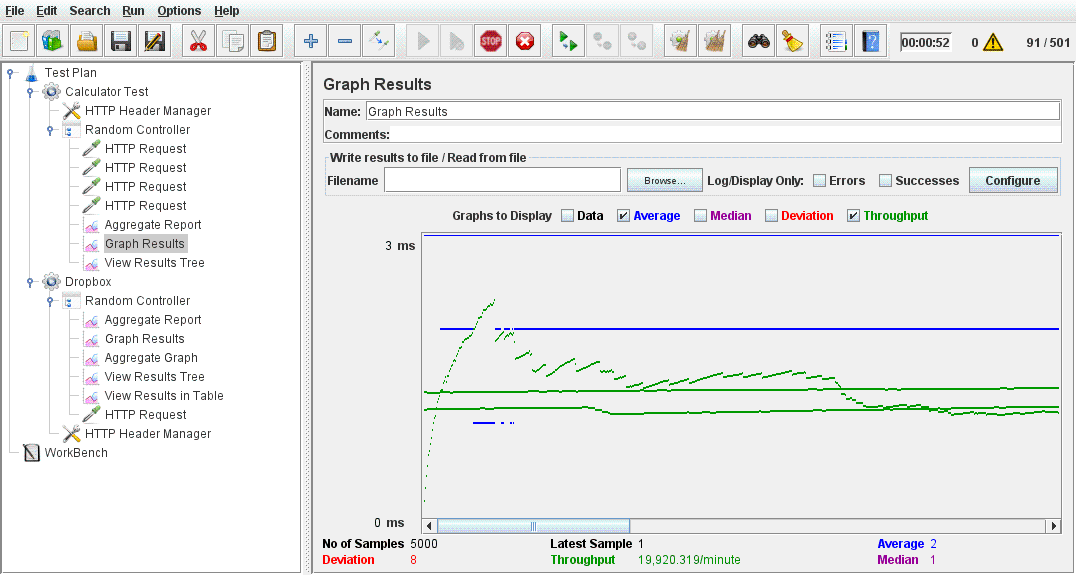
1. Invoke 1,000 calculator calls on randomly selected tasks and print average time to perform each operation.





1. Invoke 5,000 calculator calls on randomly selected tasks and print average time to perform each operation.





1. Invoke 100 concurrent users with 1000 calls each to calculator on randomly selected tasks and print average time to perform each operation.

