

Part 1: Ebay Marketplace Application

Introduction

Goal : The Goal of the system is to provide a online web marketplace to its user for selling & buying products. It also allows its user to sell their products either via direct selling or via the bidding system. The users can sign up in the application & then login with their credentials to use the system. The system also maintains the user's transactions which includes the user's item bought or sold by the user, in addition to this system also keeps the users last logged in time & it's displayed on the user's profile. System also has the cart functionality in which users can add the items they want to buy and proceed to checkout for the payment processing.

System Design

Server Side :

- Each functionality is exposing its apis via the routes.
- For persistence we have used MongoDBI database. The database named is “ebayj”
- IN mongoDb we have collections as per the business entities like Users, Advertisements, Bids and Transactions.
- For better optimization of load, we have implemented “Connection Pooling” to share the pre-created connections. These pre created connections are kept in a stack and are given to the in coming requests which require the mongodb connection for querying the database. There is a queue implemented in the “**myMongo.js**” file, in which the requests that require the connection are queued and a connection is allocated to them as & when available. If all connections are busy then the request is pushed back into the queue, where it waits until a connection is available to serve the request.
- Once the user is logged in, its session is maintained on the server. We have used Passport npm module for this functionality.
- The application also takes care of password security & stores the user's password in an encrypted format, converting the plaintext password into a HMAC code using SH1 hash function with a static secret key. Then this encrypted format is stored in the database.
- User can add items to its cart and remove same. It can also reduce or increase the quantity of products it want to buy. This cart is persisted and is maintained until user checks out the cart or removes item from it.
- When user makes a purchase the system does validation on the credit card that is entered, also while signing up, logging .
- After the purchase is made successfully the bought item is either reduced or removed from the list of available items to be bought and the bought item is moved to the user's transactions for him to view later.

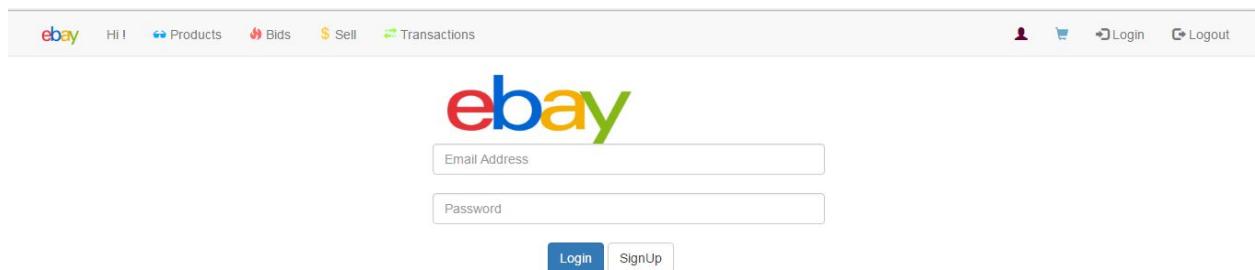
- The bidding system maintains the maximum bid placed against the item and also the bidder for that bid. After the bid ends that is 4 days, the highest bidder for the item wins the bid. This won bids & his purchases can be viewed under transactions.

Client Side :

- We have extensively used angular modules to keep our application a single page app. There is only one index,ejs file that is downloaded to the client when he visits the application url. Each module is created independently & the dependency among the modules is injected as and when required.
- There are modules for each part of the application like header, signin, signup, products list, bids, transactions , payment etc.
- For navigating between these modules we have used ng-route which renders the component as per the url entered in the browser, however there is a check applied, if the user is not logged in he is redirected towards the login page, if already logged in, the user is greeted with the page.
- For the look & feel to be similar to “ebay” we have extensively used bootstrap css classes in our application.
- We have also used Html5 controls such as number , date and email in our application.

Results: Screen captures

1:Login



1.1 Invalid Credentials




Incorrect username or password

l@ebay.com

Login Sign Up

2 Registration

 Sign Up

First name

Last name

password

reenter password

email

mm/dd/yyyy

Ebay handle

Cell phone

Location

Register Cancel

2.2 Registration Form Empty Submit

 Sign Up

* First Name

* Last Name

* Password

* Passwords dont match

* Email

* Birthday

* Ebay Handle




* Cell No


* Location

Register

Cancel

2.3 Registration Form: half filled submit

 Hi ! [Products](#) [Bids](#) [Sell](#) [Transactions](#)   [Login](#) [Logout](#)

 **Sign Up**

* Last Name

* Password

* Passwords dont match

* Email

2.4 Successful Registration :

localhost:3000 says:

Registered use your credentials

☐ Prevent this page from creating additional dialogs.

OK

Sign Up

•

•

l@ebay.com

11/29/1995

l@ebay

9876543210

Loyal City Down Town

Register Cancel

3: HomePage

Hi w! Products Bids Sell Transactions Login Logout

motorola x2

Description : Very heavy duty phone

Seller : w

Price : 2344

Quantity : 1

Add to Cart

Multiport Charger

Description : Very userfule multil port chger 4 ports

Seller : w

Price :

Quantity : 2

Add to Cart

Light Room Lamp

Description : Phillips Light room Lamp

Seller : philips Inc.

Price : 50


Quantity : 2

Add to Cart

3.0 Selling an Item

 Bids







 Sell

 Transactions

Product Details

☐ I want to sell via Bid

3.1 Selling an Item, Unfilled Form Submit

 Hi w! Products Bids Sell Transactions

Product Details

* Item Name

* Description

* Seller

* Price

* Quantity

Item Name

Description

Seller

☐ I want to sell via Bid

Item Price

Quantity

Submit

Cancel

3.2 Successful Submission of Advertisement

ebayHi w!ProductsBids\$ SellTransactions

Product Details

Cookies

Chocolate Cookie

ParleG

☐ I want to sell via Bid

10

2

SubmitCancel

localhost:3000 says:
Submitted ur add
☐ Prevent this page from crea

3.3: Bids

ebayHi !ProductsBids\$ SellTransactionsLoginLogout

motorola x2

Description - asdasd
Seller - qa@email.com
Quantity - 1
Bid Ends - 2016-10-14
Max Bid - 126\$
Bidder - q

Your Bid \$

Bid

Woodland Shoes

Description - pair of wood land shoes
Seller - qa@email.com
Quantity - 1
Bid Ends - 2016-10-15
Max Bid - 2\$
Bidder - q






Your Bid \$

Bid

Mattress

Description - Spring Maxwell mattress
Seller - Harshit
Quantity - 1

3.4 Selling an item via Bid :

 Hi w!  Products  Bids  Sell  Transactions

Product Details

☒ I want to sell via Bid

3.5 Before Placing a Bid on an existing item

Park Avenue

Description - deo by PArk Avenue
Seller - Park Avenue Inc.
Quantity - 5
Bid Ends ⌚ - 2016-11-10
Max Bid - 0\$
Bidder -

3.6 After placing a BID (update bid max amount and Bidder name)

Park Avenue

Description - deo by PArk Avenue
Seller - Park Avenue Inc.
Quantity - 5
Bid Ends 🕒 - 2016-11-10
Max Bid - **3\$**
Bidder - w

Bid

4: Transactions

ebay

Hi w!

Products

Bids

Sell

Transactions

👤

🛒

Login

Logout

Purchases

📦

bubble gum

Description : mint flaovor bubble gum used
Seller : w@email.com
Price : 2
Quantity : 3

Icard

Description : 12
Seller : wq
Price : 12
Quantity : 1

bubble gum

Description : mint flaovor bubble gum used
Seller : w@email.com
Price : 2
Quantity : 3

Bids Won

★

Description : MEga ULTRA MEGA SUPER MEGA PRIZE
Seller : qa
Price :
Quantity : 100

5: Profile:

Profile

Last Login  : 2016-11-06 04:21:05

Loyal

User

l@ebay.com

1995-11-29

l@ebay

Cell phone

Loyal City Down Town

Done

6 : Cart

Cart Summary

Amount : 900

[Proceed to Checkout](#)

Mousepad

LG mouse Pad

450

1 Quantity :

[Remove](#)

Table + chair

wooden table chair
qa@email.com

2 Quantity :

[Remove](#)

Mousepad

LG mouse Pad

450

1 Quantity :

[Remove](#)

7.1: Payment :

Payment

Amount : 900

Shubham Malviya

9876543210123456

Pay

Cancel

7.2 Invalid Card Details at payment

Payment

Incorrect Card Details

Amount : 125

Name on Card

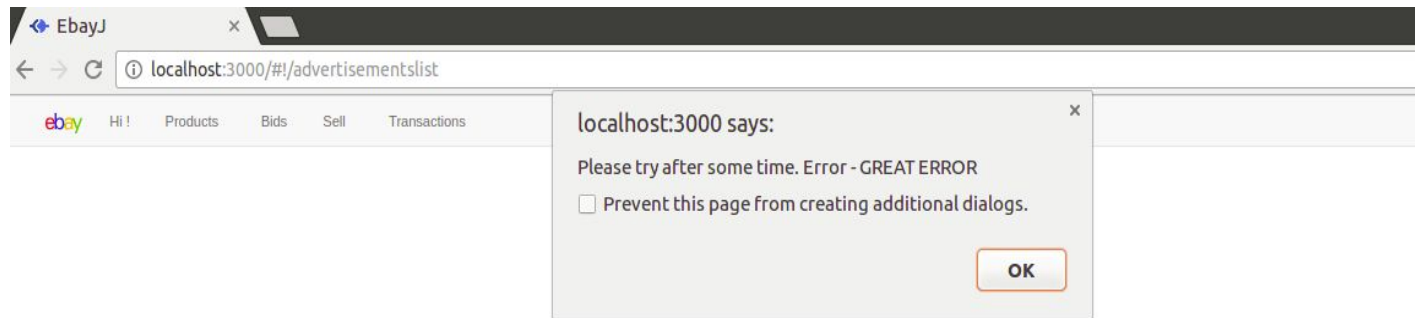
Card Number

Pay

Cancel

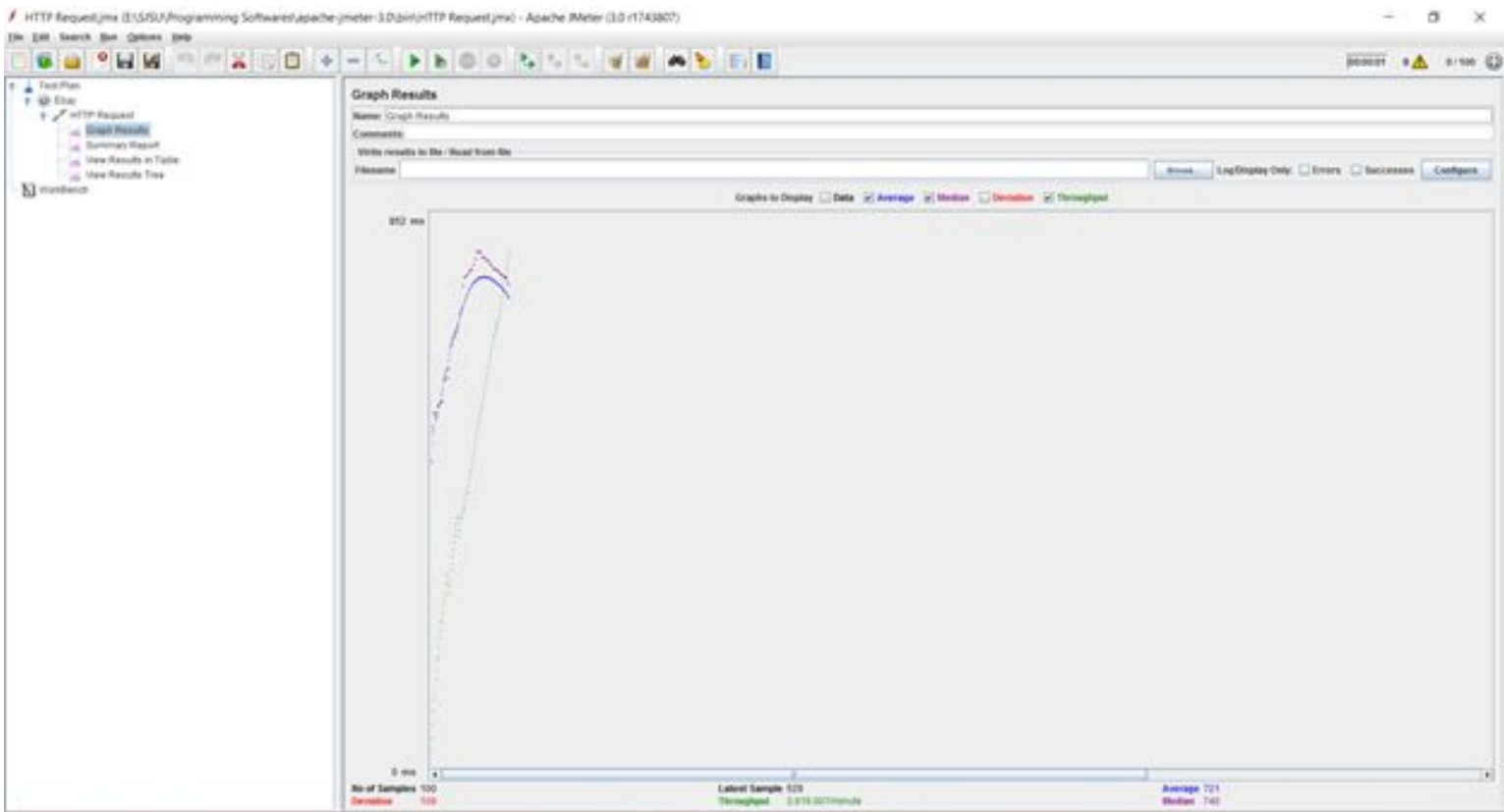
8 : Any Unexpected Exception at Back End

Handled as alert on the client side.

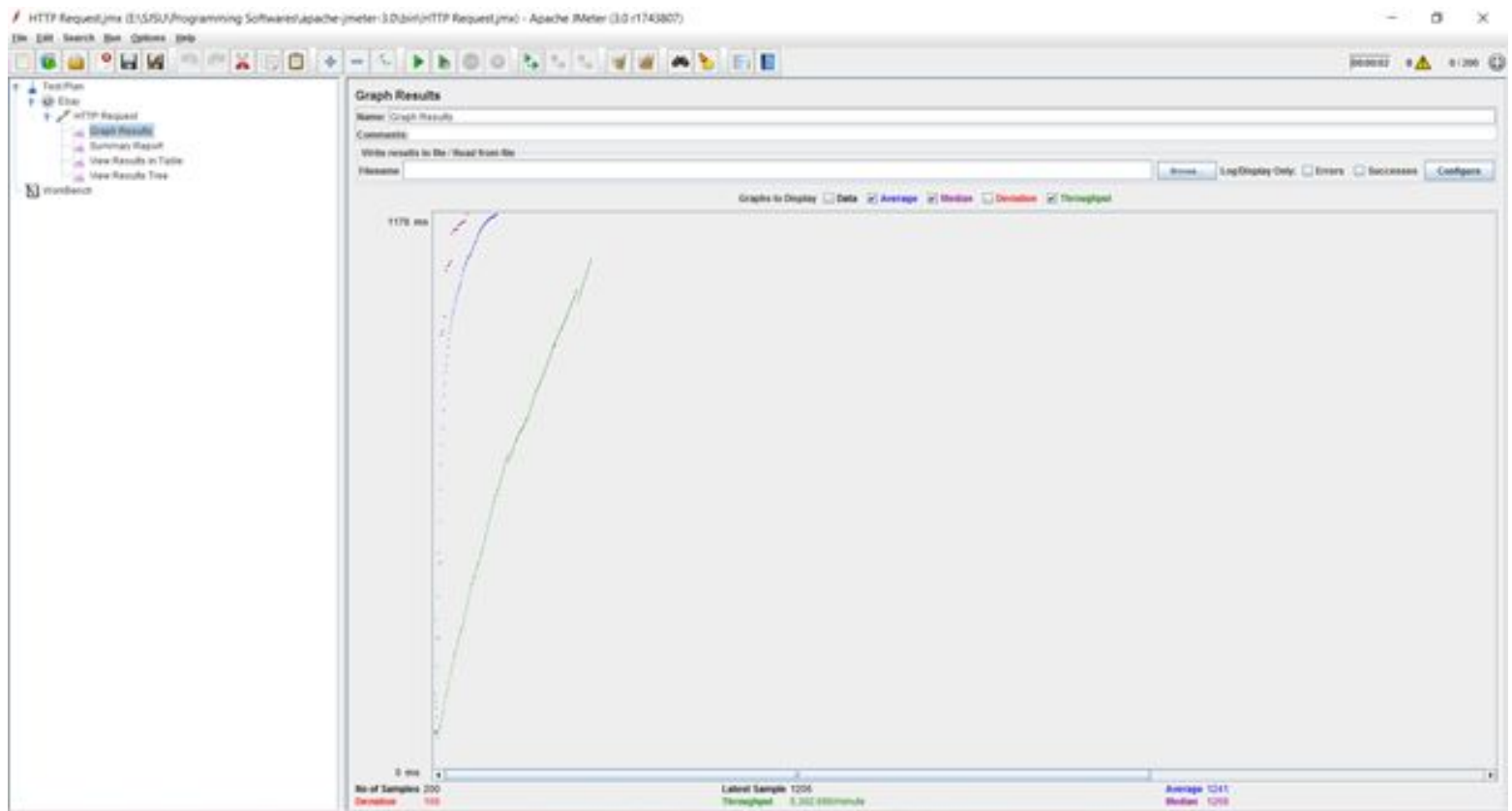


Performance Graphs

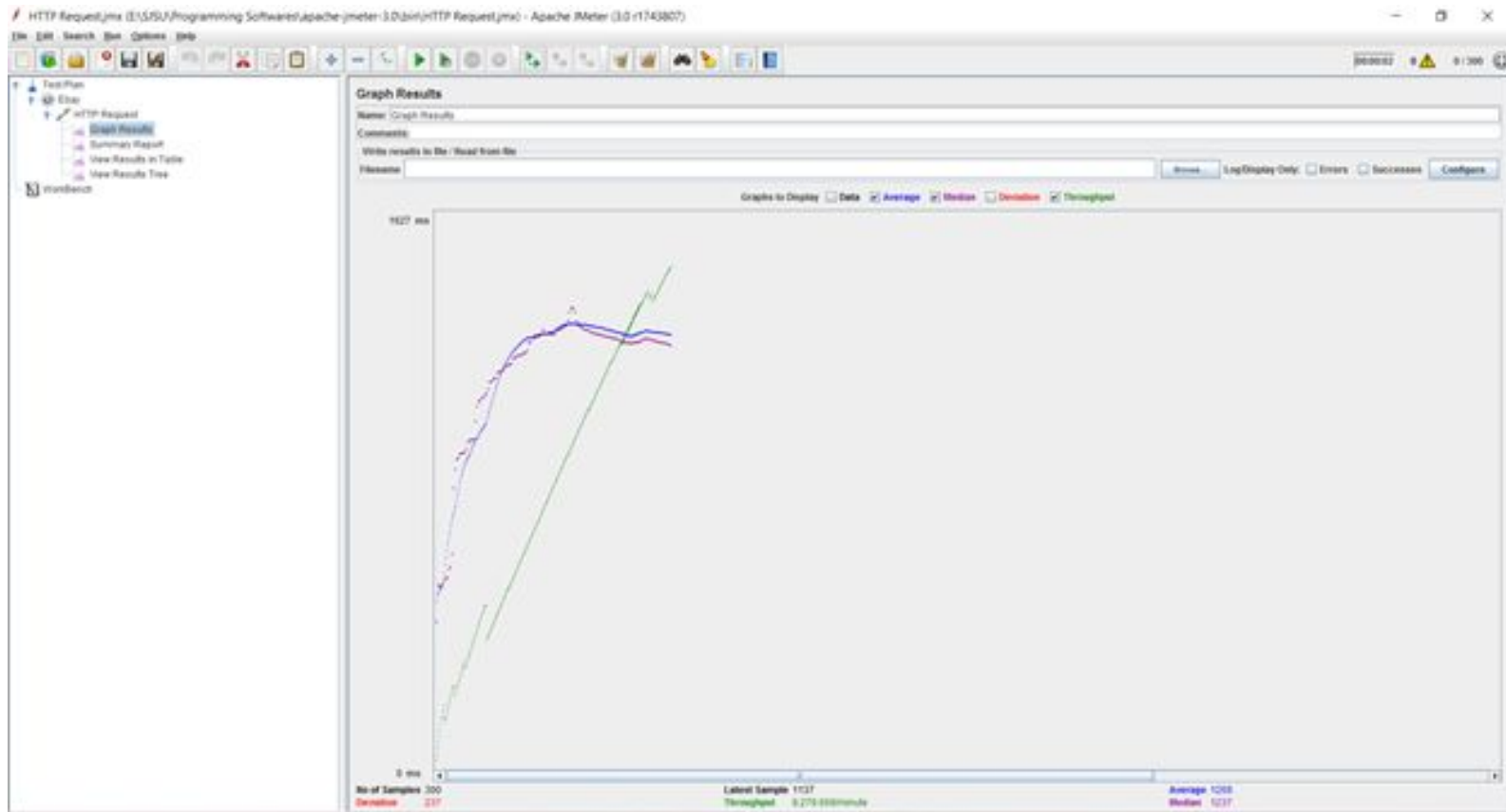
100 Calls Without Connection Pooling



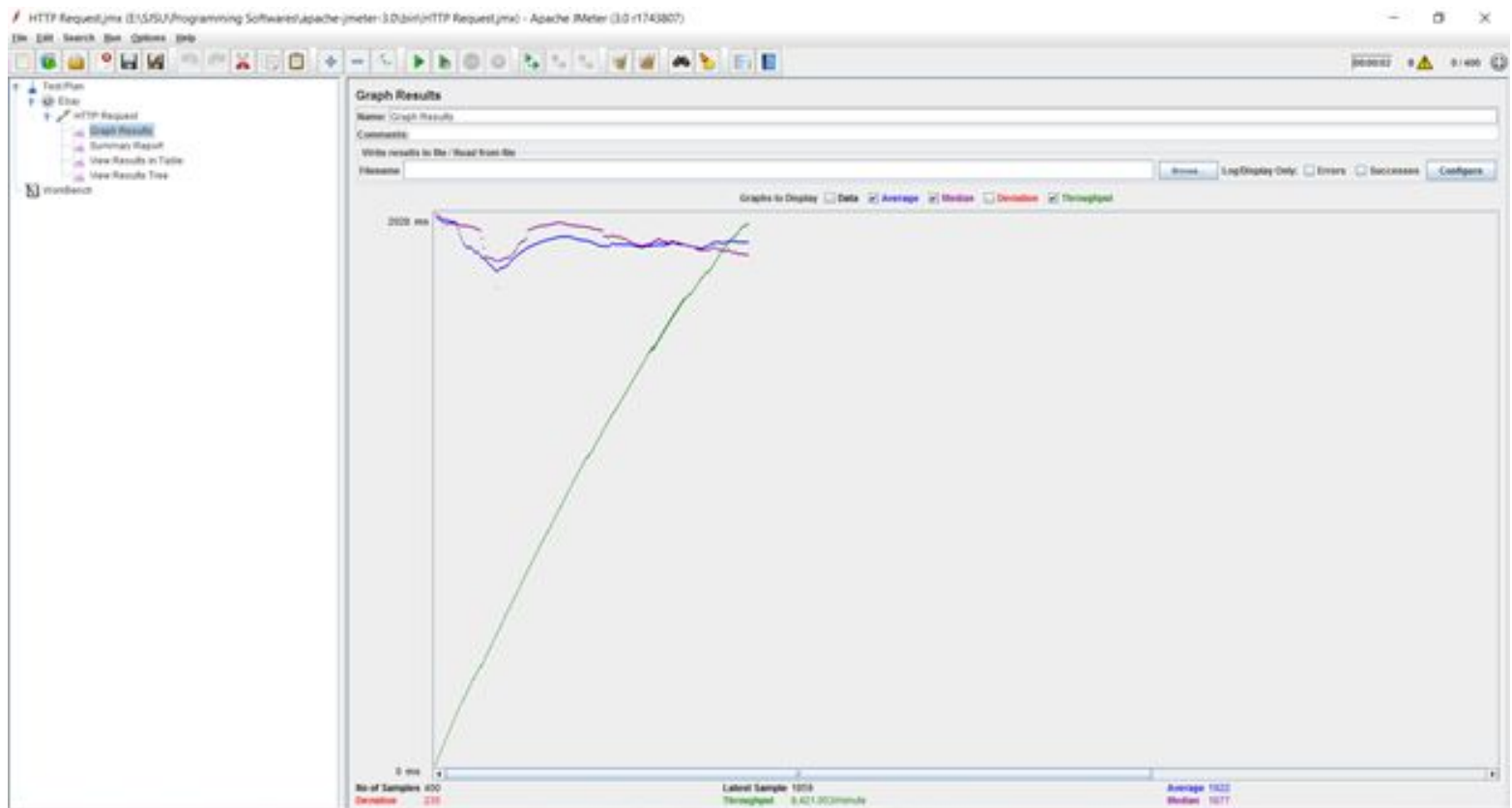
2) 200 Without connection Pooling



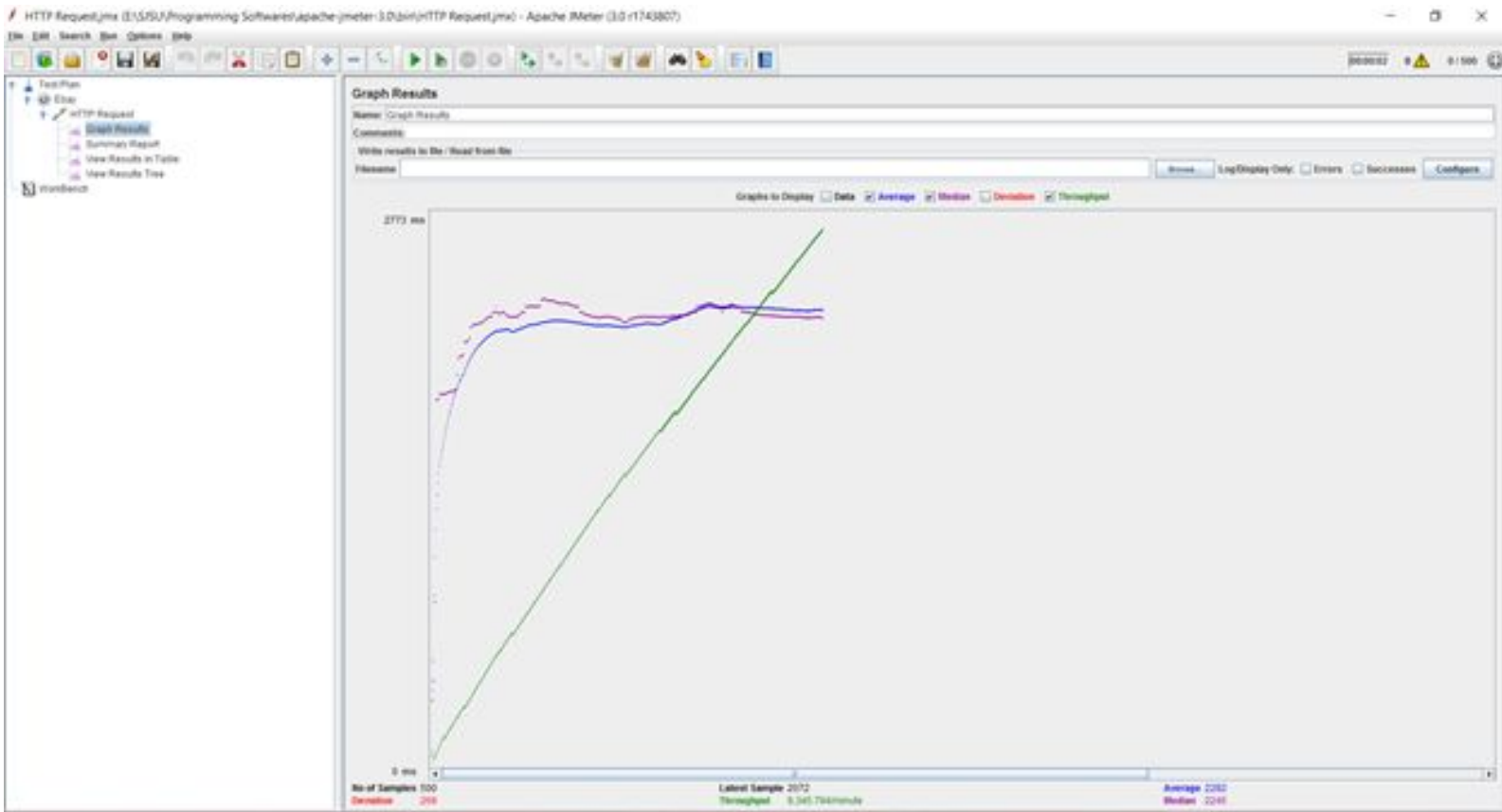
3) 300 requests Without Connection Pooling



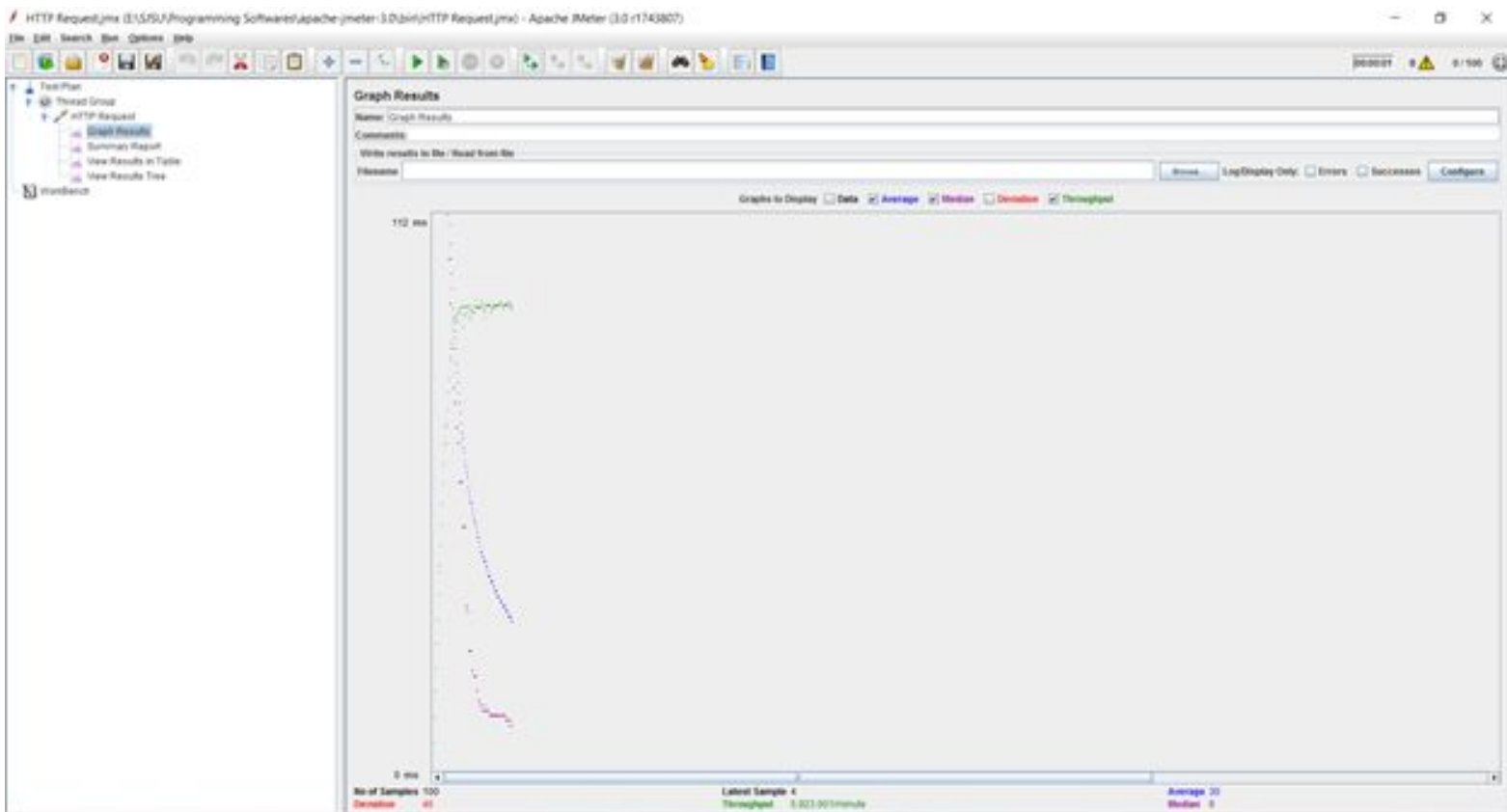
4) 400 requests without connection pooling



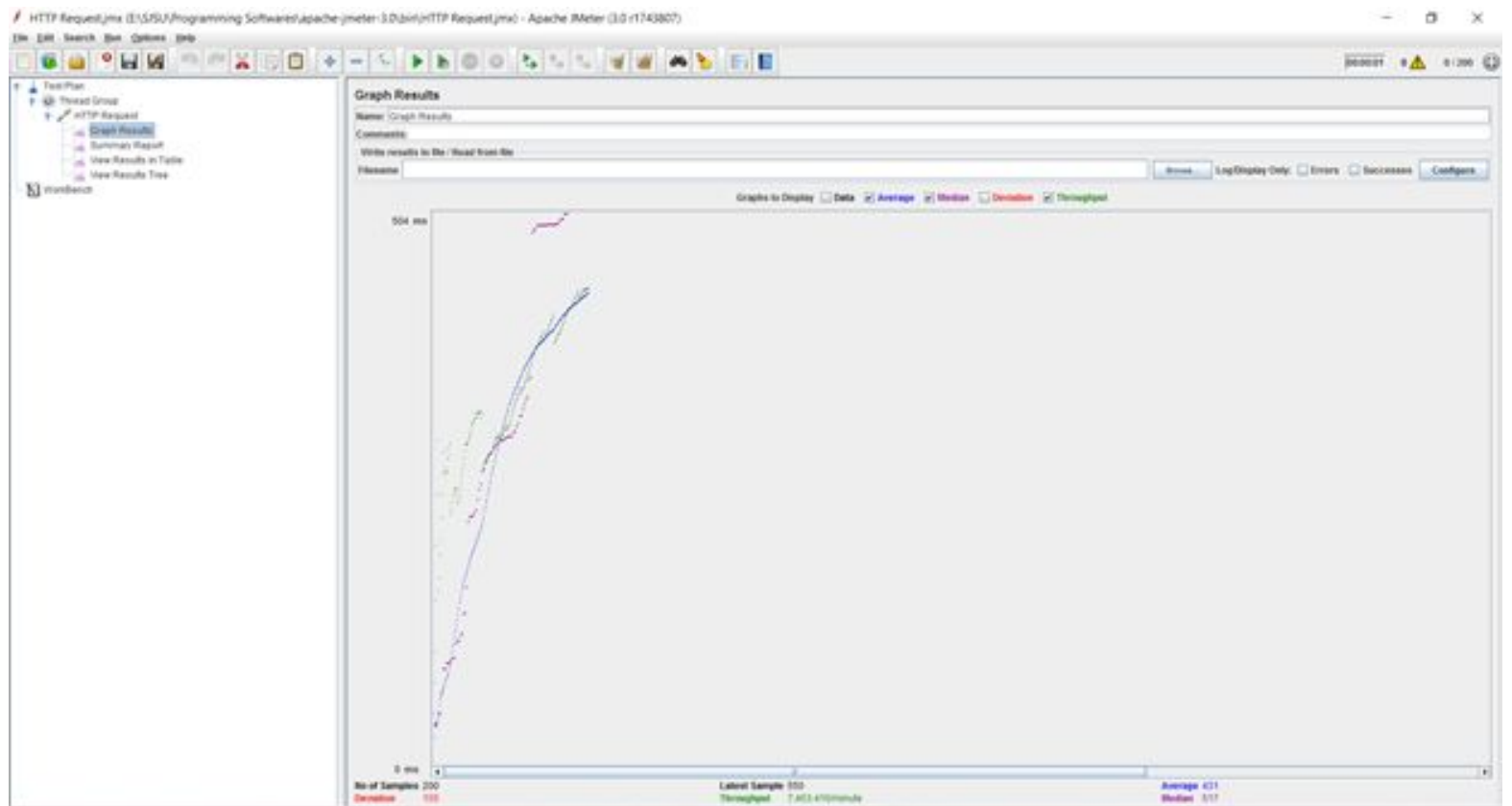
5) 500 requests without connection pooling



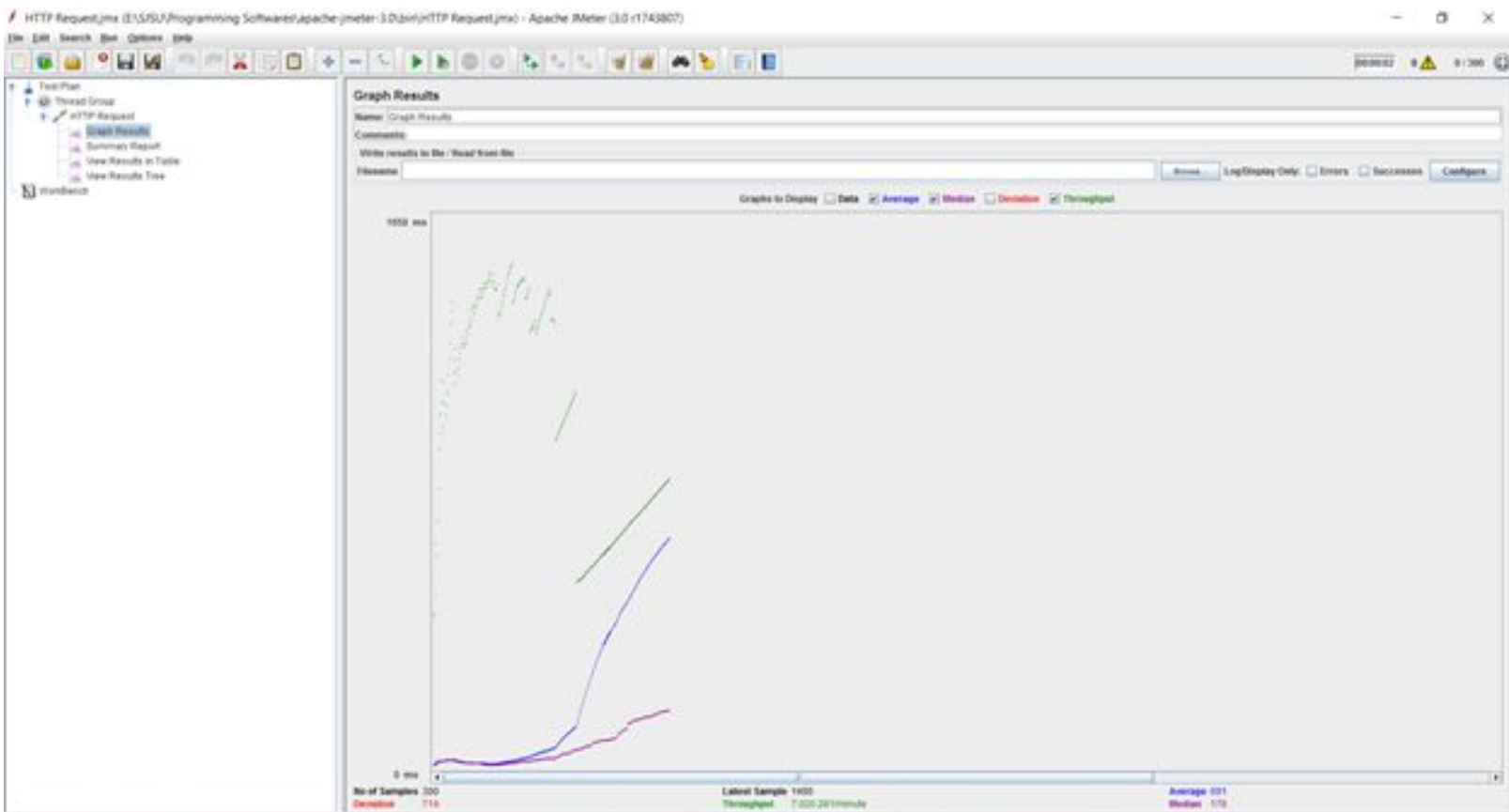
100) With connection Pooling



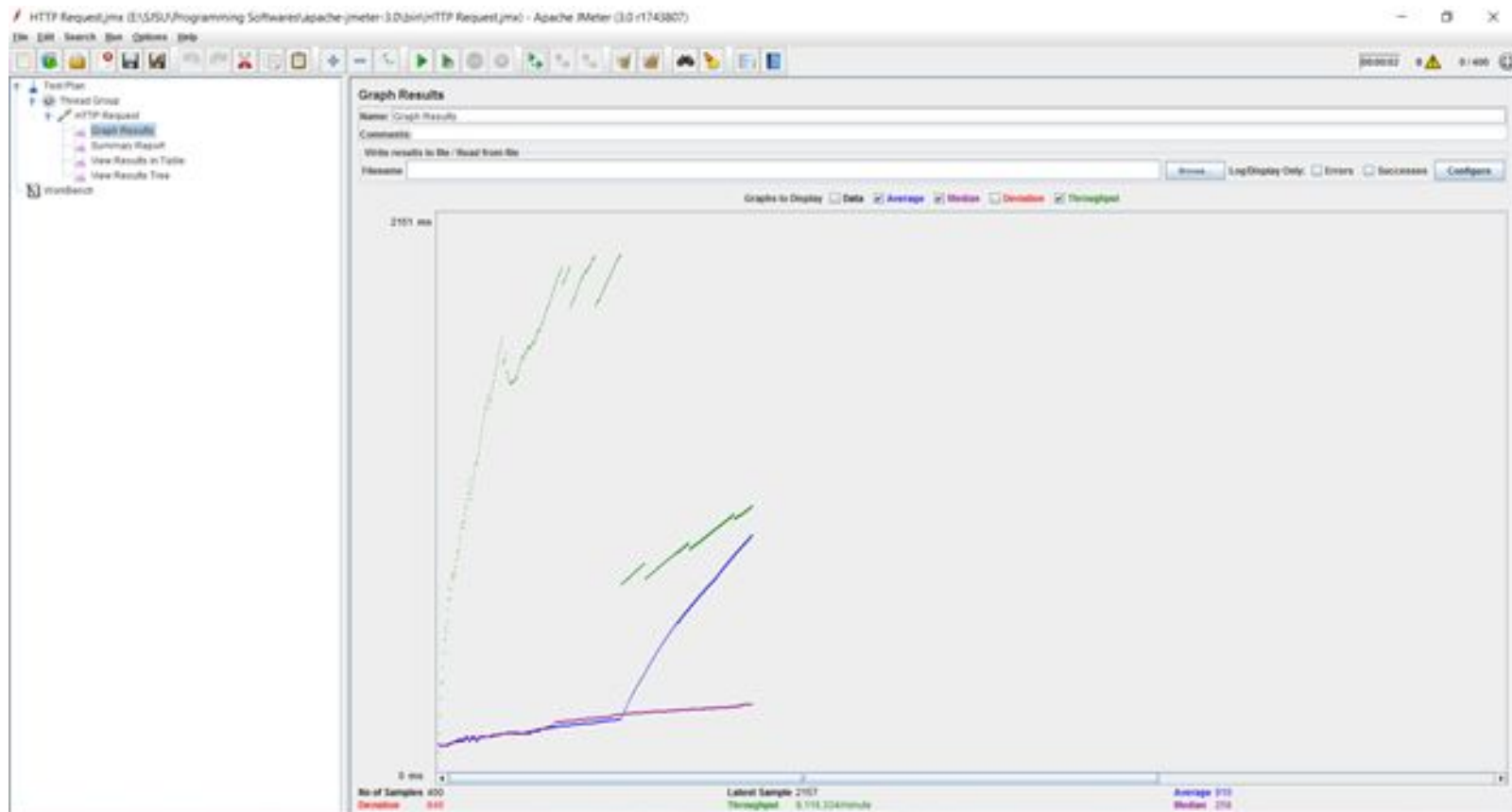
200) With connection Pooling



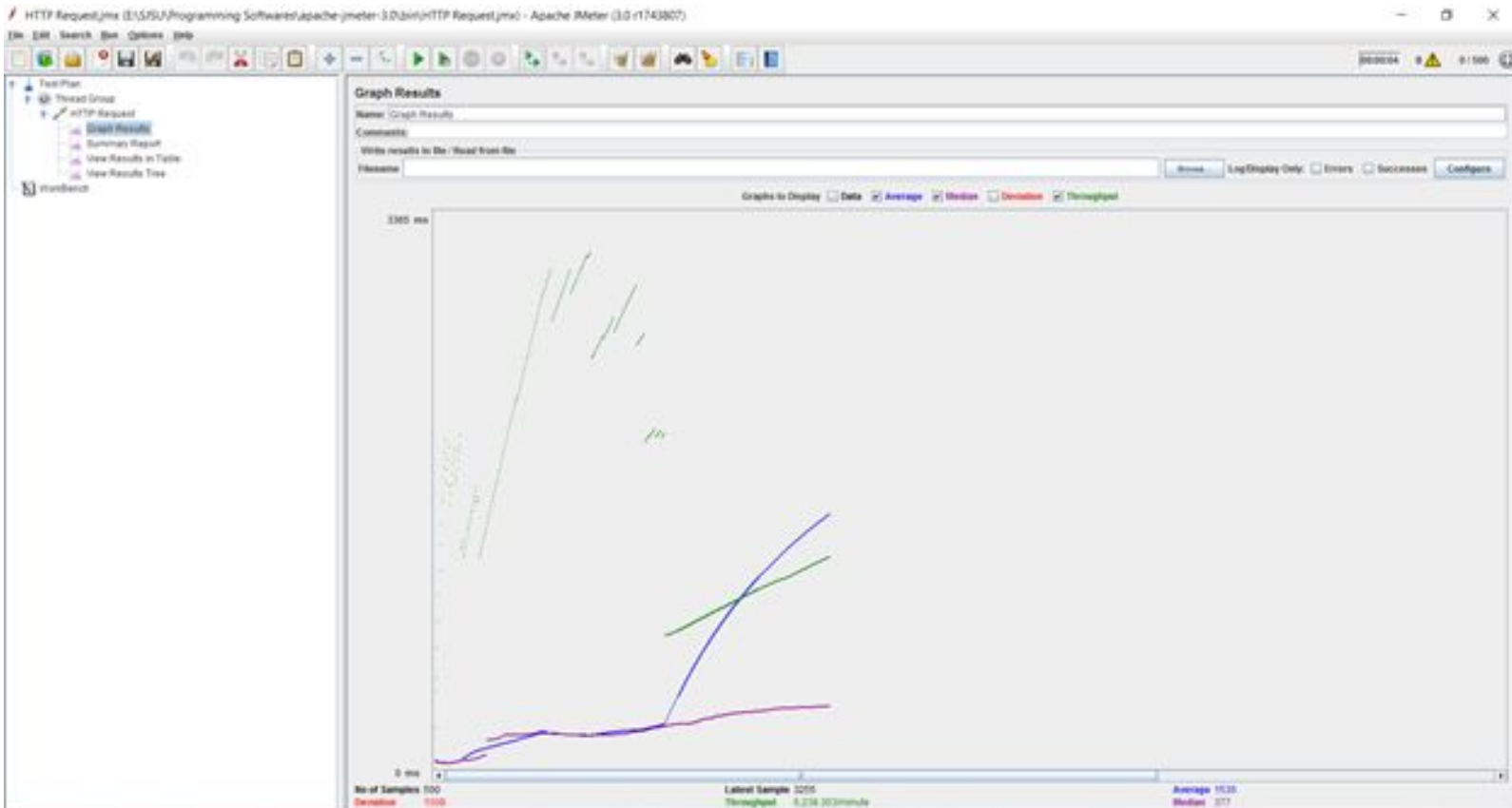
300) With connection Pooling



400) With connection Pooling

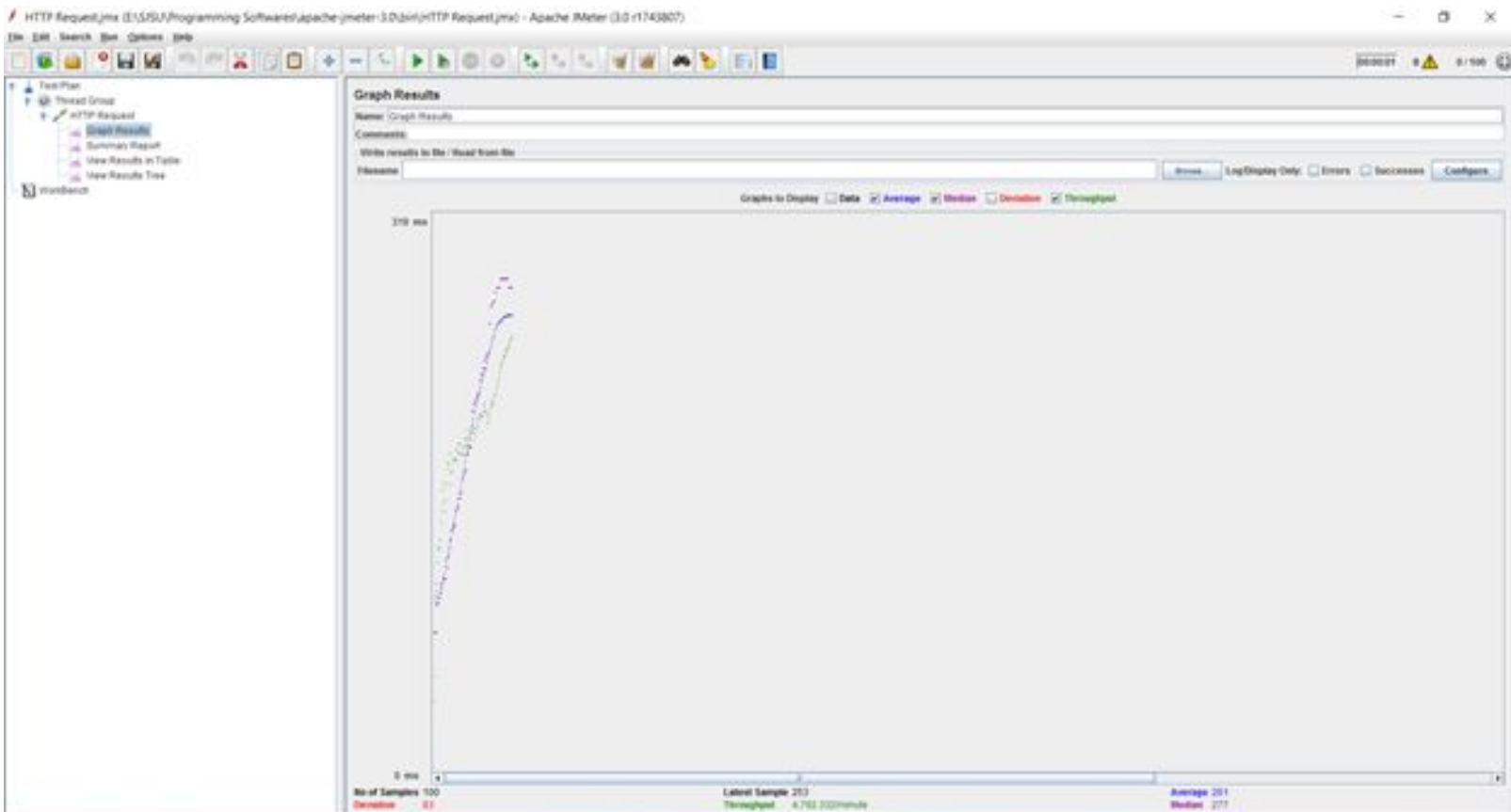


500) With connection Pooling

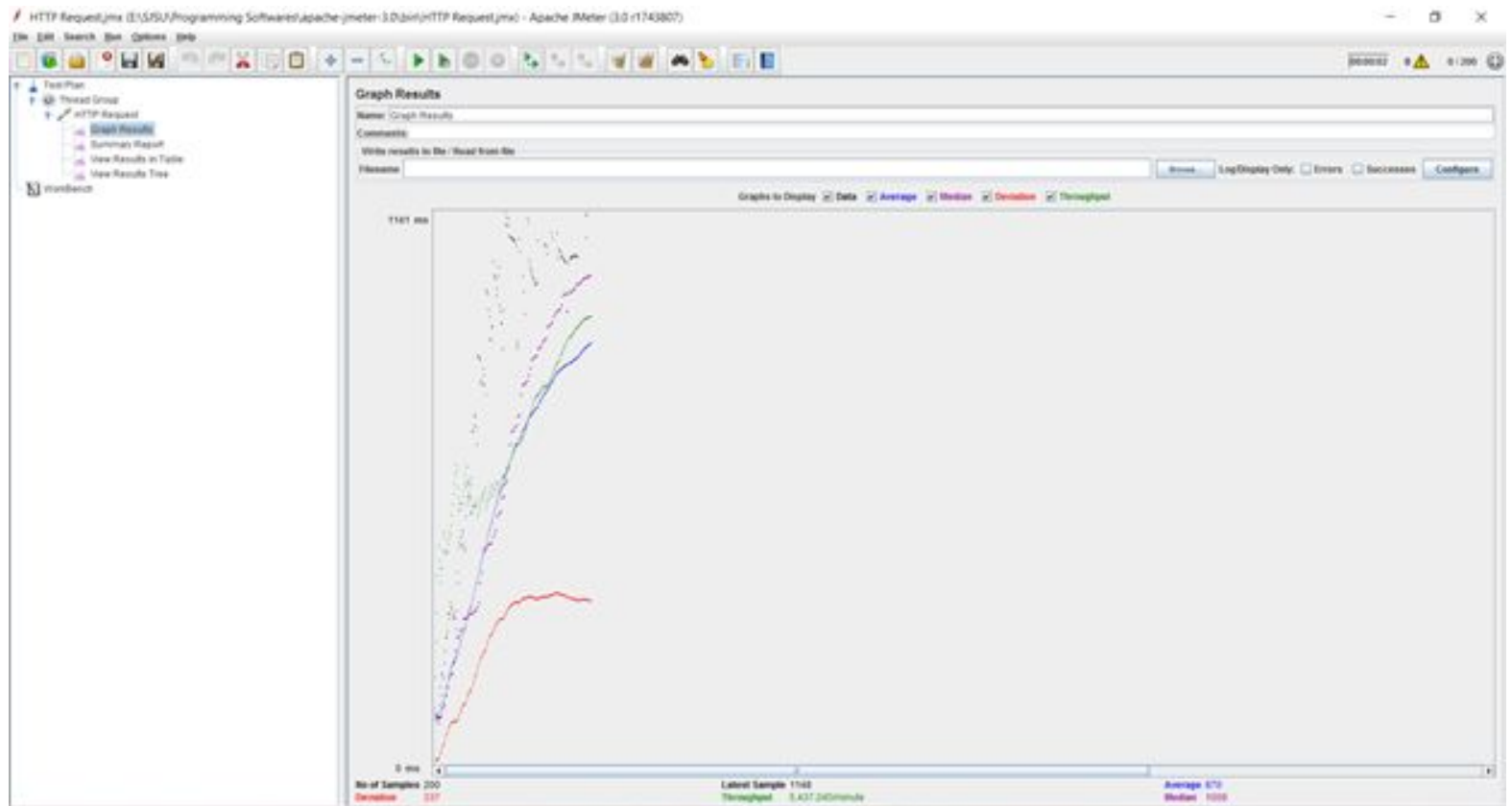


Performance Graph With and Without RabbitMQ

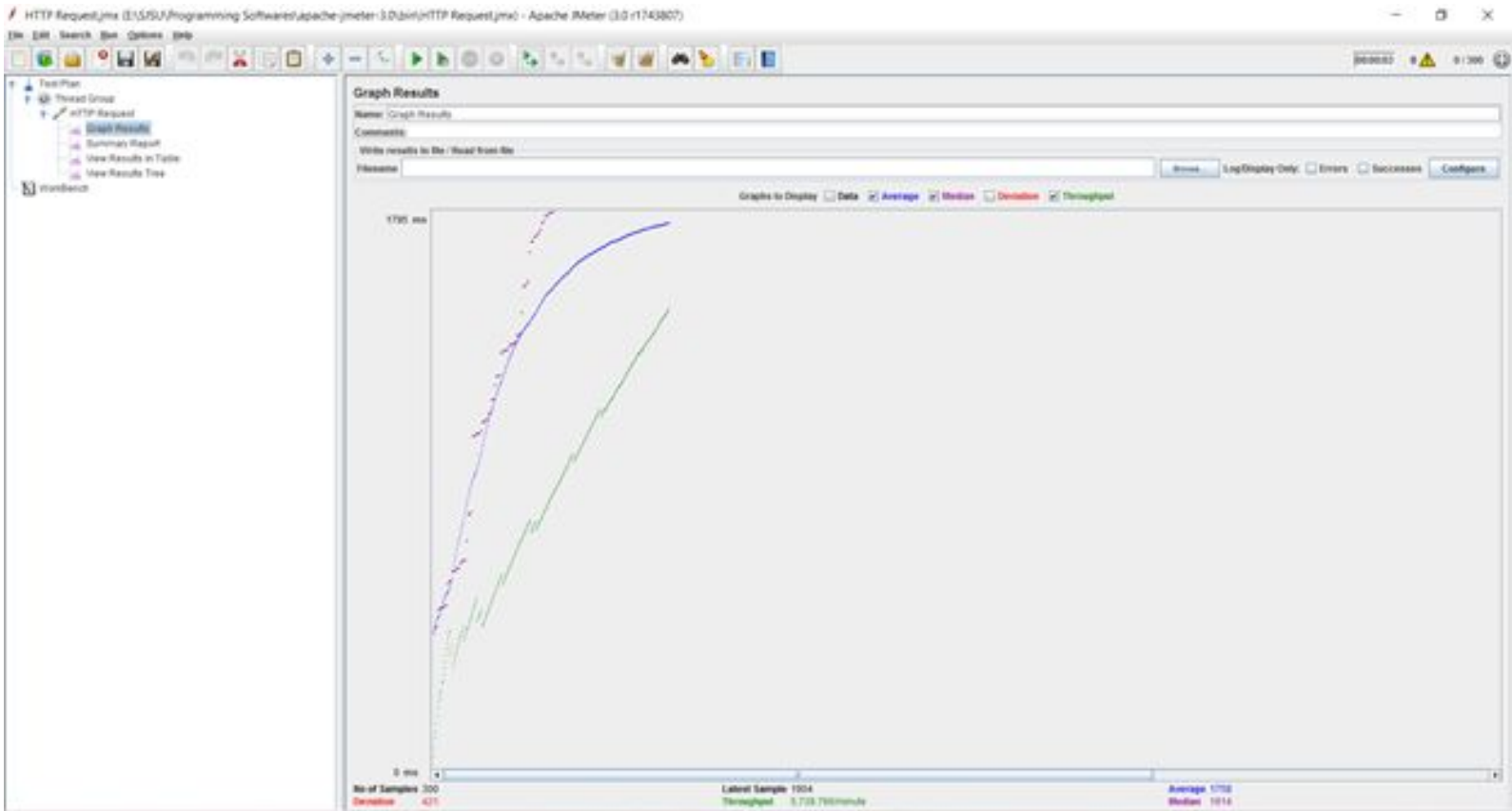
1) 100 Calls Without RabbitMQ



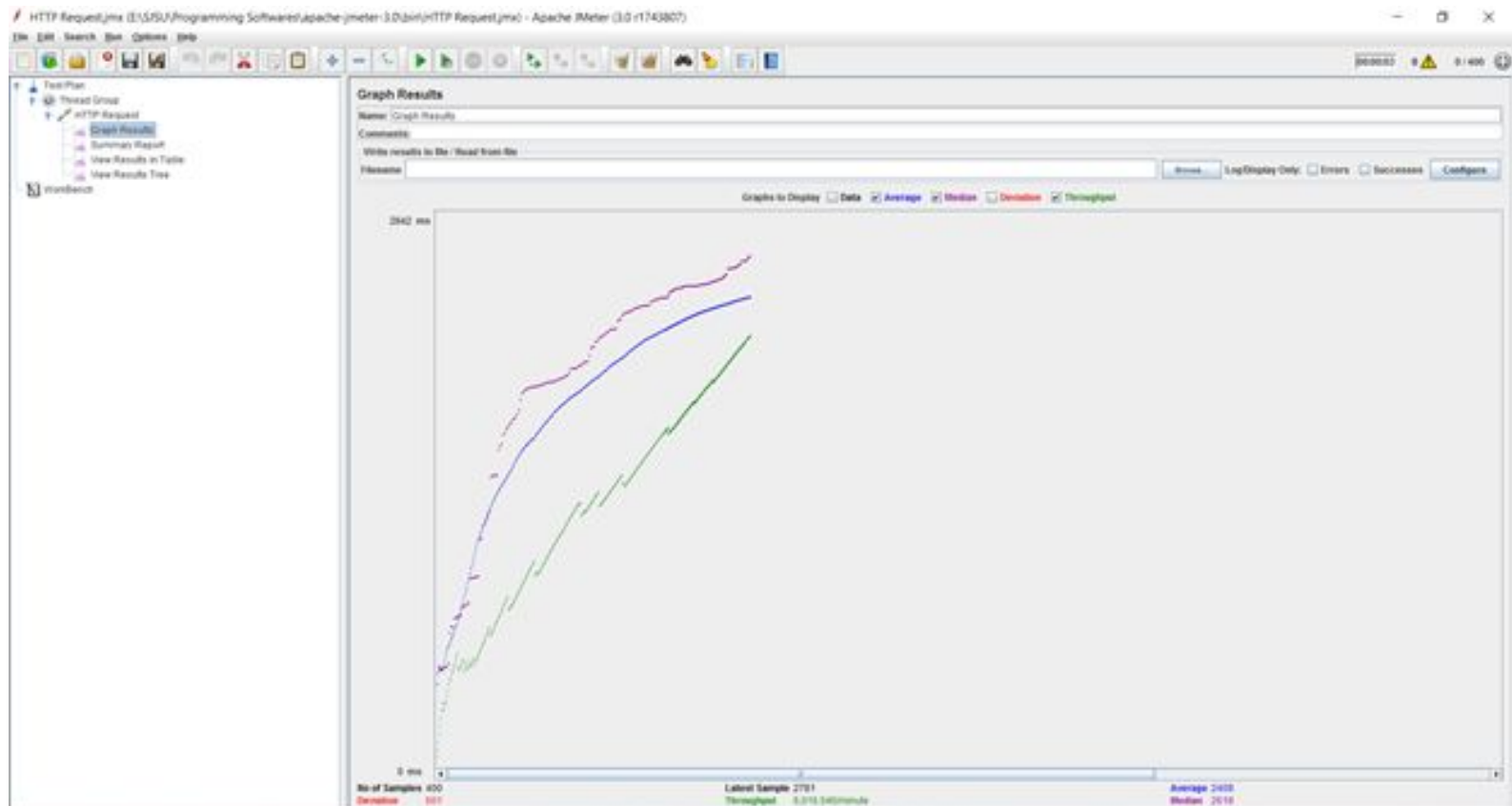
2) 200 Calls Without RabbitMQ



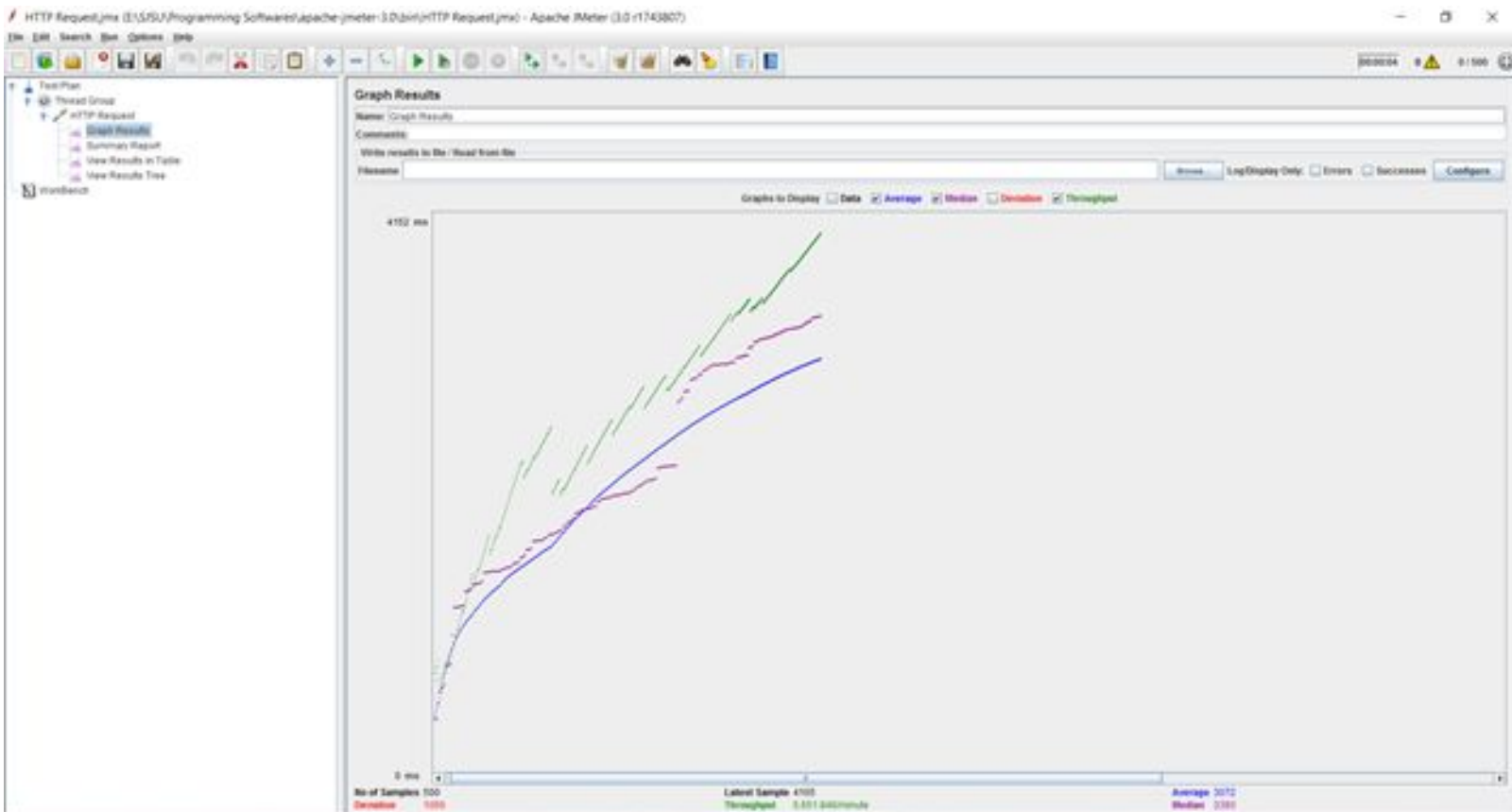
3) 300 calls Without RabbitMQ



4) 400 Calls Without RabbitMQ

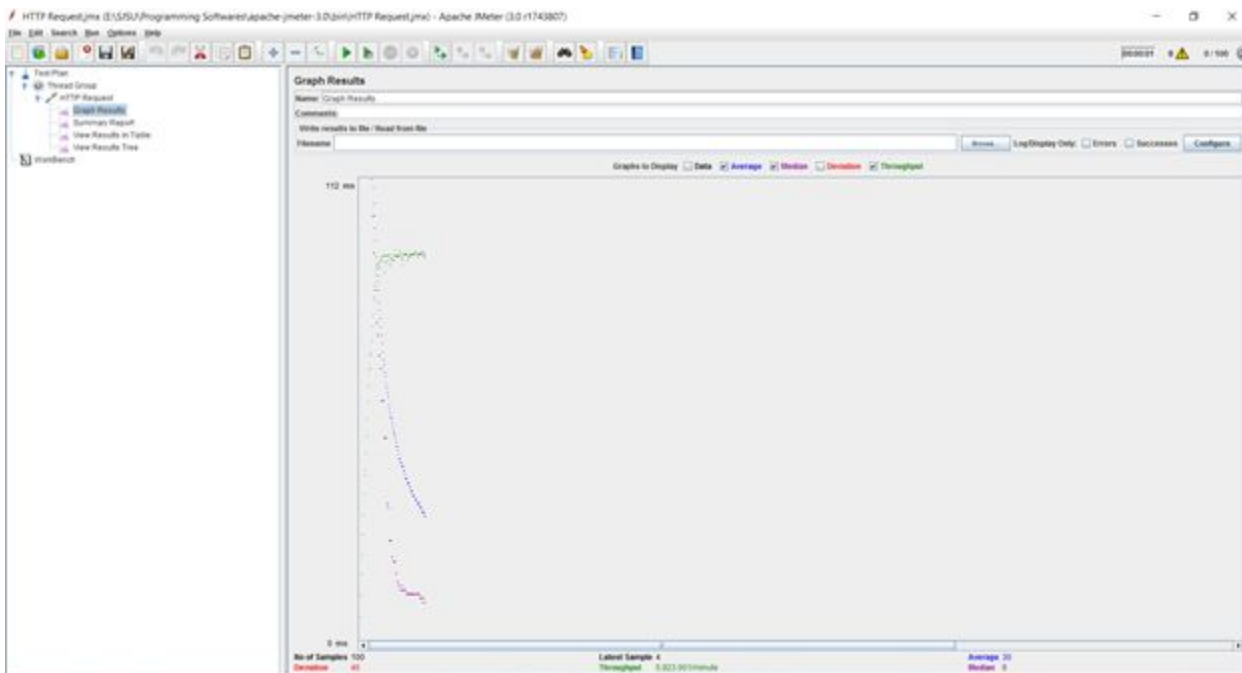


5) 500 Calls Without RabbitMQ

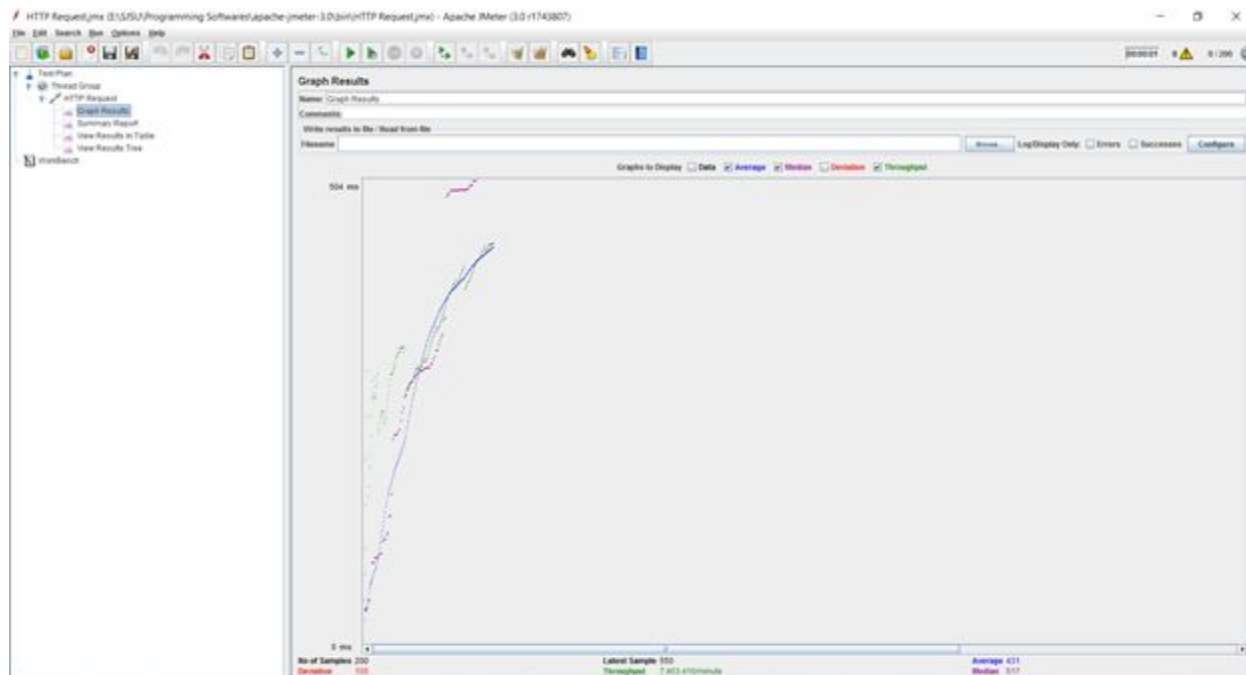


With RabbitMQ

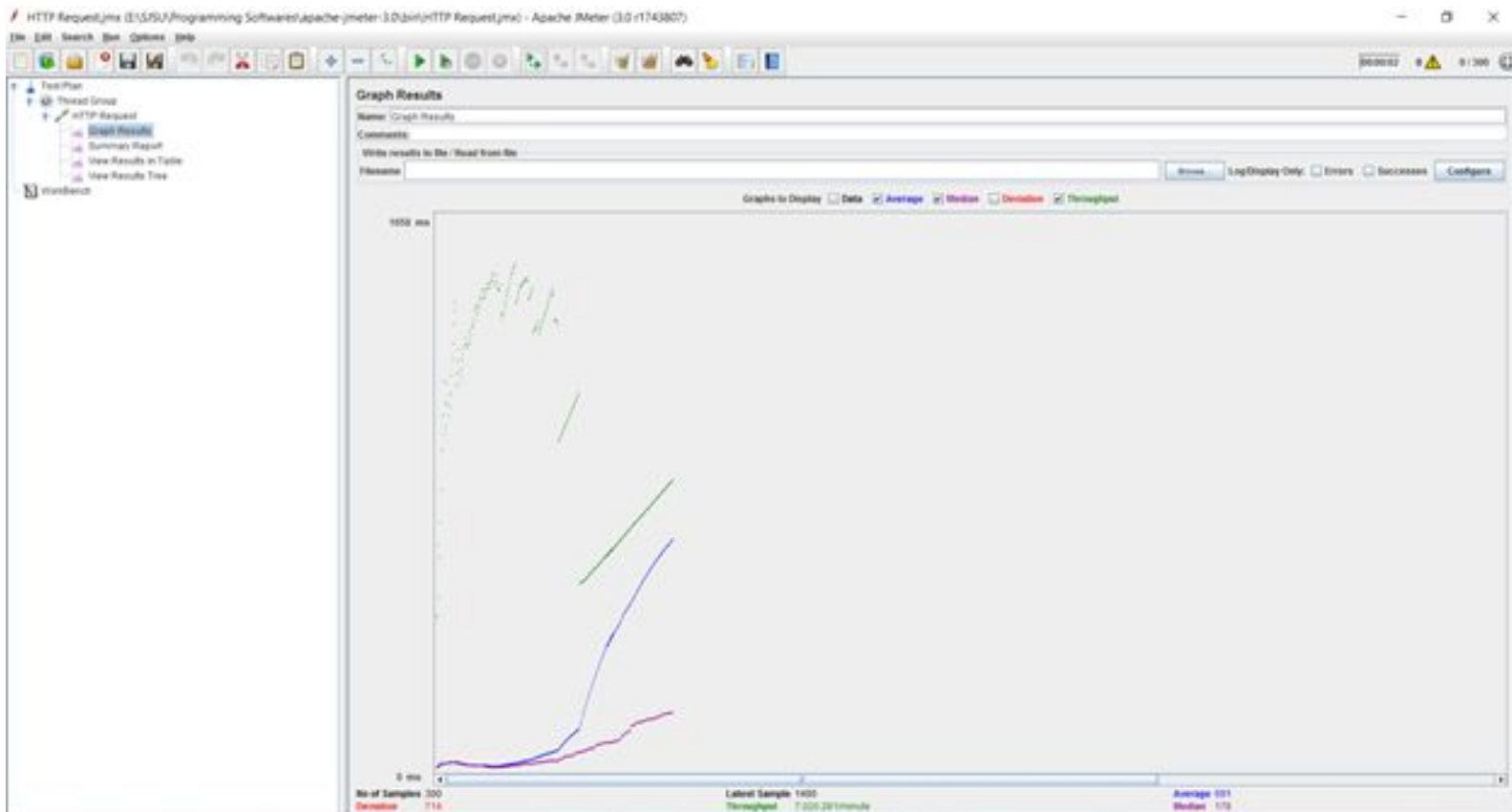
100) With RabbitMQ



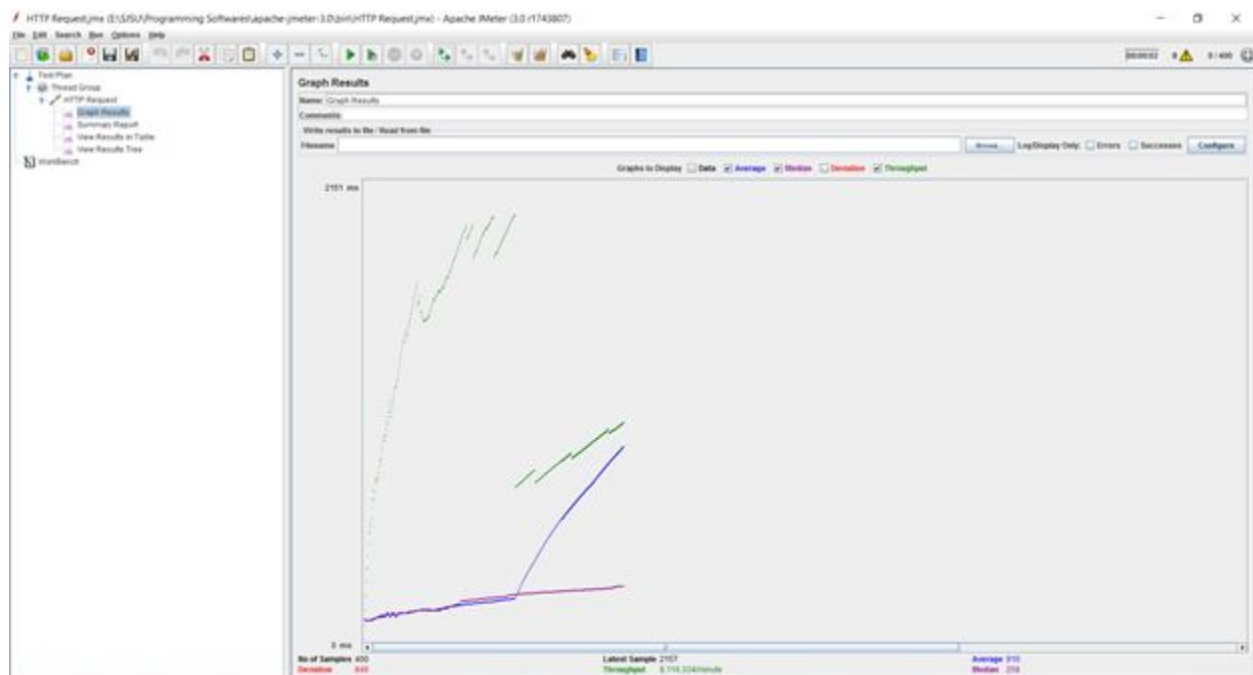
200) With RabbitMQ



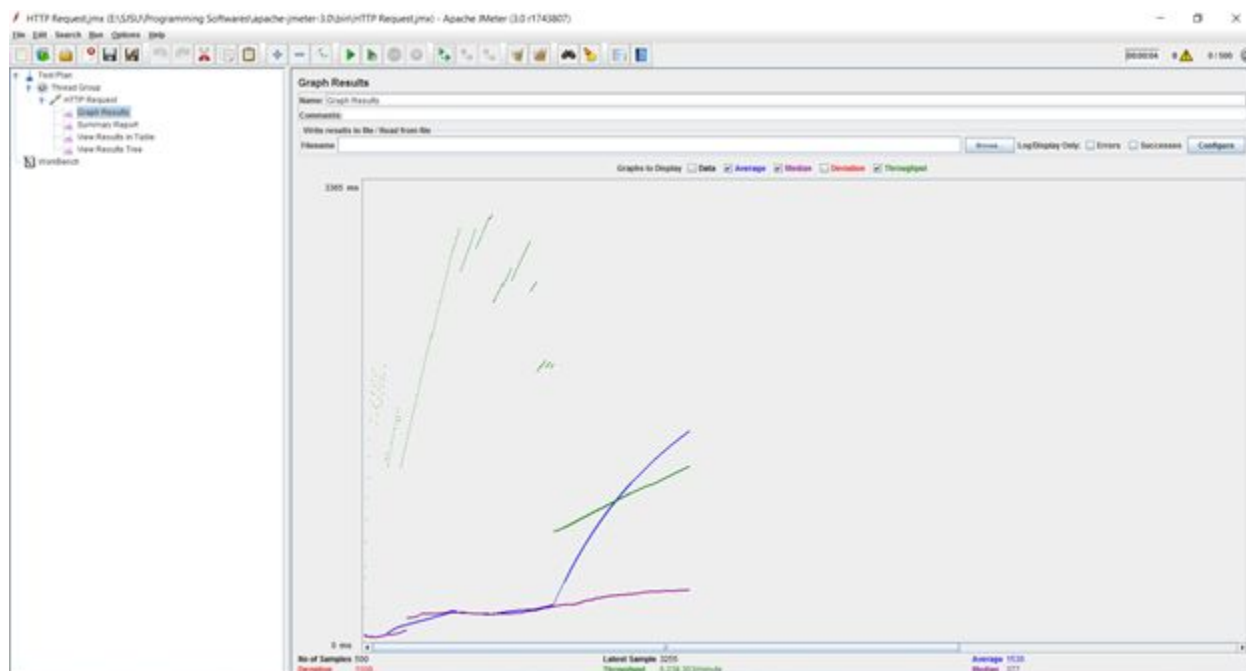
3) 300 calls With RabbitMQ



4) 400 calls With RabbitMQ

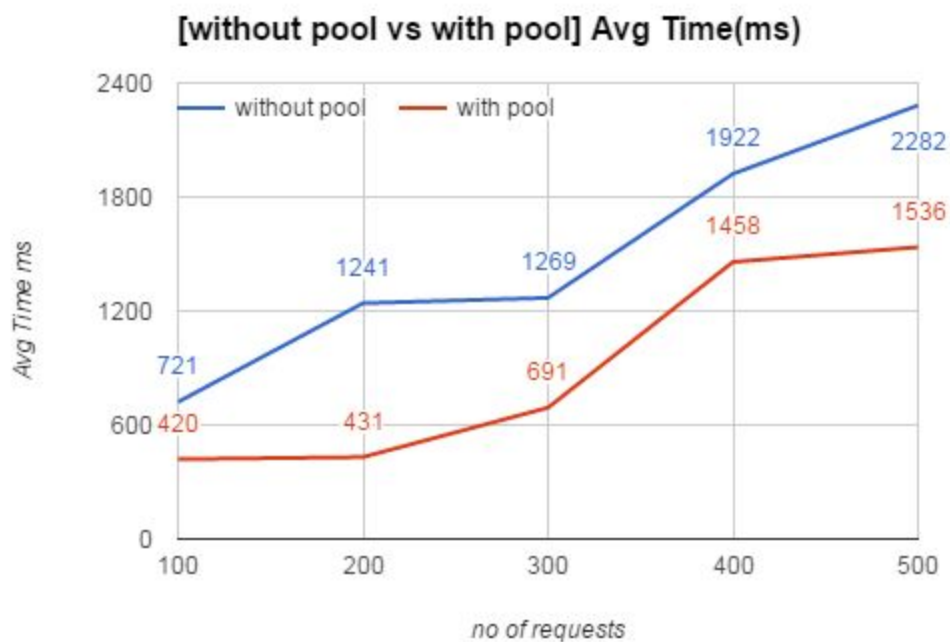


5) 500 calls With RabbitMQ



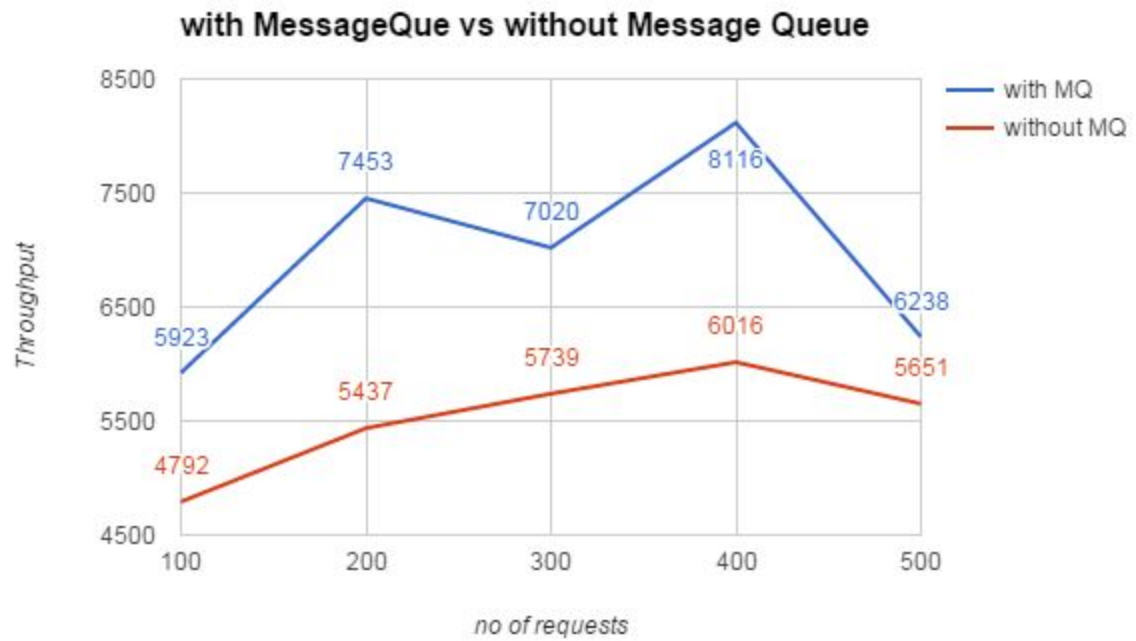
Comparison Of Without Connection Pooling vs With Connection Pooling

Connection Pooling	Avg time ms	Avg Time ms
no of requests	without pool	with pool
100	721	420
200	1241	431
300	1269	691
400	1922	1458
500	2282	1536



Comparison Of With RabbitMQ vs Without RabbitMQ

	Throughput	Throughput
no of requests	with MQ	without MQ
100	5923	4792
200	7453	5437
300	7020	5739
400	8116	6016
500	6238	5651



Part 2 : Answers to Questions

Architecture of the RabbitMQ interaction:

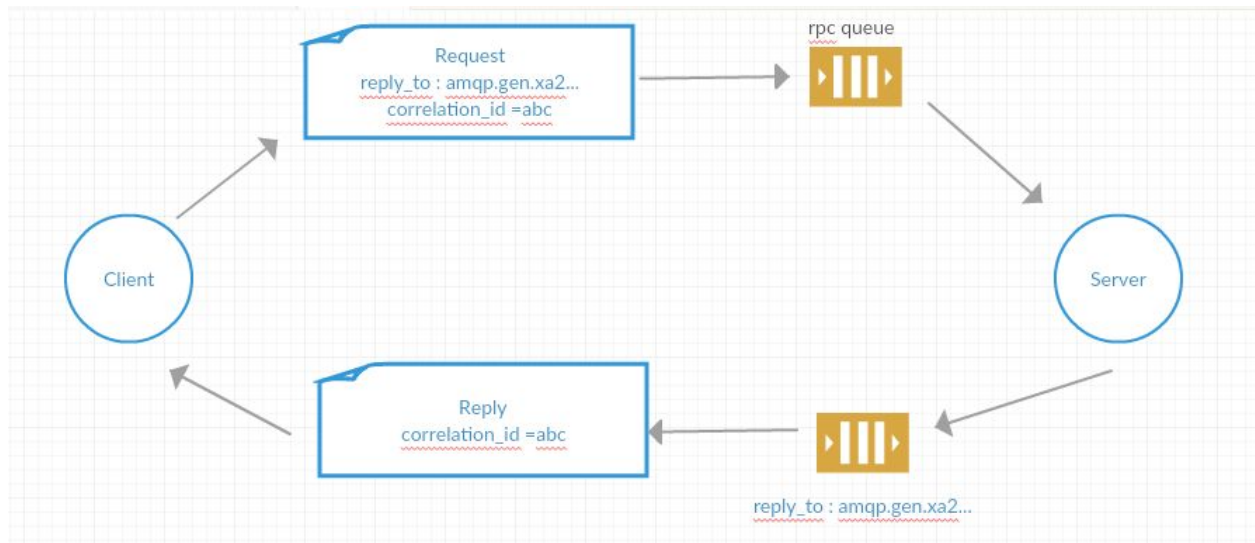
Below are the points explaining the role and sequence of events, of client and server in the AMQP setting.

1. The client (rpc client) sends commands to the underlying amqp protocol to instantiate a message queue with a unique queue name after this step the rabbit server subscribes has to subscribe to this queue, with this subscription set up the rabbitmq server can easily listen up to the requests that are published into the queue. But there is one part still pending , which is for the client to subscribe to the response queue. This response queue is utilized by the rabbit Server to publish the processed output, which the client has requested. That's how we end up having two queues one being the request queue and another being the response queue

2. After the first step we have underlying architecture ready for us to send and receive messages. But we have still not identified who is going to be the producer for the request queue. This is where our web application comes into play. The web application also known as "producer" sends a message to RabbitMQ, using the queue that we have defined in the previous step, and includes the data from the request, like name and email into the message payload. This payload will contain the data required for the request processing, in addition to the payload the client also pushes a callback function into the response queue which will be called when the server has finished processing the request. But In Case the request took too long for receiving the response we have to add a timeout function, which limits the upper bound that a request can stay in the queue without receiving a response. If the timeout has expired the request will get removed from the message queue and the final call back will be called with a timeout error. There is one more important information that is inserted with the payload, to uniquely identify each request a correlation_id is generated which uniquely identifies each request.

3. With our housekeeping ready on both the ends it's time to publish the message into the queue. After the Producer (client) has published a message in tot the queue, an exchange receives the messages and routes them to correct message queues with the help of given queue name. This exchange takes place within the RabbitMQ and the message is published into the queue.

4. Now the request has arrived at the consumer end of the requests queue, which is the rabbitMq server. From here server read the message payload and calls a service which will perform the processing and generate a result. The service actually holds the business logic and also the reference for database, it is the only layer which will hold the reference to the database as it is the only place where the business logic resides. After the service has executed it gives back the response to the rabbit server, which then calls the response callback from the response queue with passing the result as parameter. It fetches the callback from response queue based on the correlation id of the request that it is processing right now.



1. Explain what performance change RabbitMQ provides? Elaborate on the results of throughput with and without using RabbitMQ. If you find any increase/decrease in the throughput, explain the reason for the same.

Ans: As we can observe from the recorded performance of our application with using RabbitMQ, that we have achieved greater throughput for our application. As compared to our previous architecture where we had no concept of producer and consumer, there was only one server which had to manage every request from start to end. In our new architecture we can have multiple producers and consumers to handle operations. To give an example, we can have one producer dedicated for lets say login request. Our login route client can be the producer and we can have multiple consumer on the rabbitmq server side, which are subscribed to the login queue. Now when a bunch of login requests arrive we can have our consumers serving all these login request in parallel. This ability to have many consumers work along each side, gives us a tremendous performance boost. And it is also visible in the performance comparison graphs. Similar thing can be done on the rabbitmq server side as well, rather than having only single consumers on the response queue, we can have multiple consumers subscribed to the response queue. In this way any available consumer can take the response and forward the response. With this configuration we have gained a greater throughput in our application.

2. Compare passport authentication process with the authentication process used in Lab1.

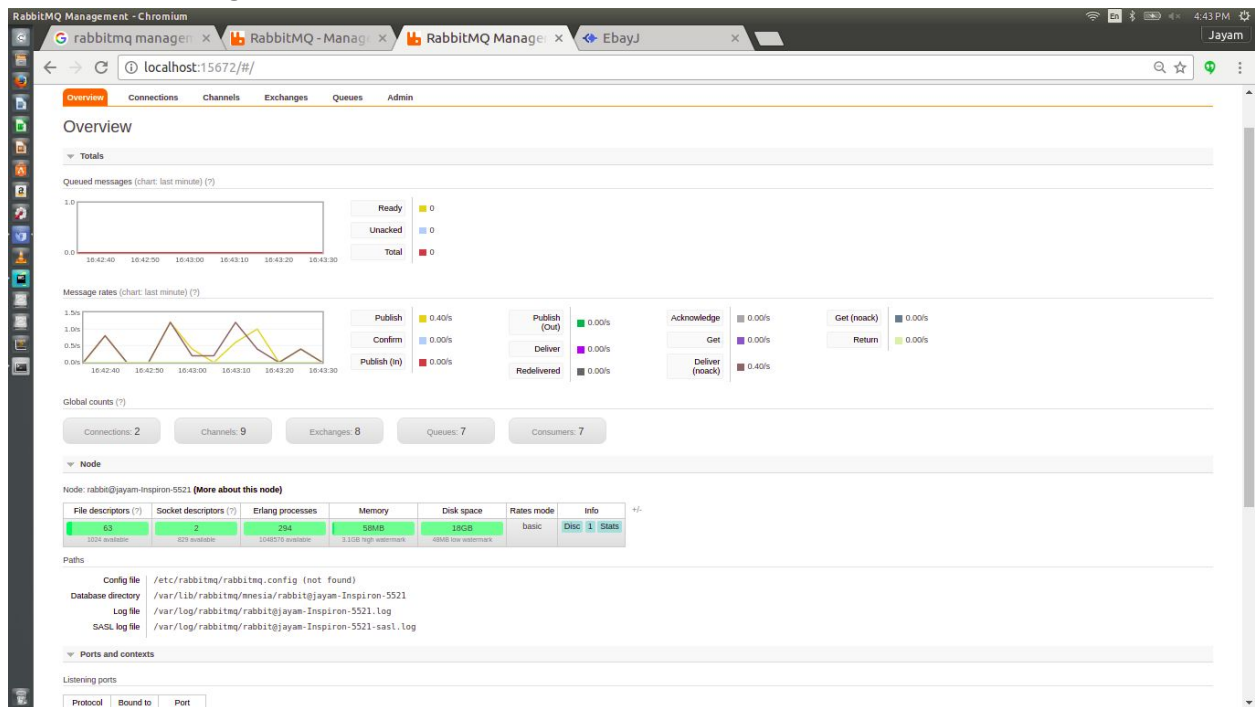
Ans : One advantage we observed of using passport for authentication was it provided a neat way to plugin in various types of authentication strategies in our code. The concept of strategies gives us a separation of concern in design. This concept of using multiple strategies for authentication is a definitely a big win over our traditional authentication method. Another advantage that we observed which is in couple with the nosql MongoDB that we have used. On successful authentication, as passport is coupled with “mongo-store” it will automatically create a new collection named “Sessions” with in the current database. And will populate with the current user object associated with the request. We can define what objects we want to serialize upon the successful authentication of the request. Once the user object is serialized into Session collection, we can utilize these objects for checking if the request is already authenticated or not. Upon the request from the same user the user object is automatically deserialized for the corresponding request. As compared to the previous method where we have to keep the user's details in the client-session object. Passport authentication act as a interceptor in between and proceeds the request attaching the authentication result to the next function, which is an ideal modelling of the authentication layer within the business. Also we can add more strategies as config to our passport, this will help us in enabling our application to utilize oauth and use social media passport strategies to provide an ability for user to login with their social media accounts.

3. If given an option to implement MySQL and MongoDB both in your application, specify which data of the applications will you store in MongoDB and MySQL respectively.

Ans : In my opinion the major advantage of NoSql Db over Relational Db is the improved read and write performance. As unstructured data is allowed we can have diverse objects within the same collection, this enables us to store the information related to the entity to be stored just alongside with the object. Also Nosql is very ideal to store the data that is not going to get update once written, some examples of data that falls under such category are the History of transactions that the user has had in past, purchases, bids won, sold items, Reviews of the products or items etc. Also we can horizontally scale the mongodb with sharding and distribute the database to multiple nodes. This feature is not possible with Mysql, because it can only scale up vertically, means we can only increase the strength of a single instance of the database machine. Apart from this Mysql relational database is ideal for transaction processing, in our project it would be a good fit for the bidding system, where we have to keep updating the bids depending upon the the bid amount or if the bid duration is over. Also Mysql allows us to link information across tables through the use of Foreign keys, another ideal place to benefit from this ability would be the shopping cart functionality, where we have to maintain if the product is added in the cart and if so then we keep a reference of the original product in the cart,

so that we don't necessarily commit the product to the customer before him buying the product. Also to maintain the last logging activity of the users Mysql is an ideal choice.

RabbitMQ Management Console:



Mocha Test Cases run

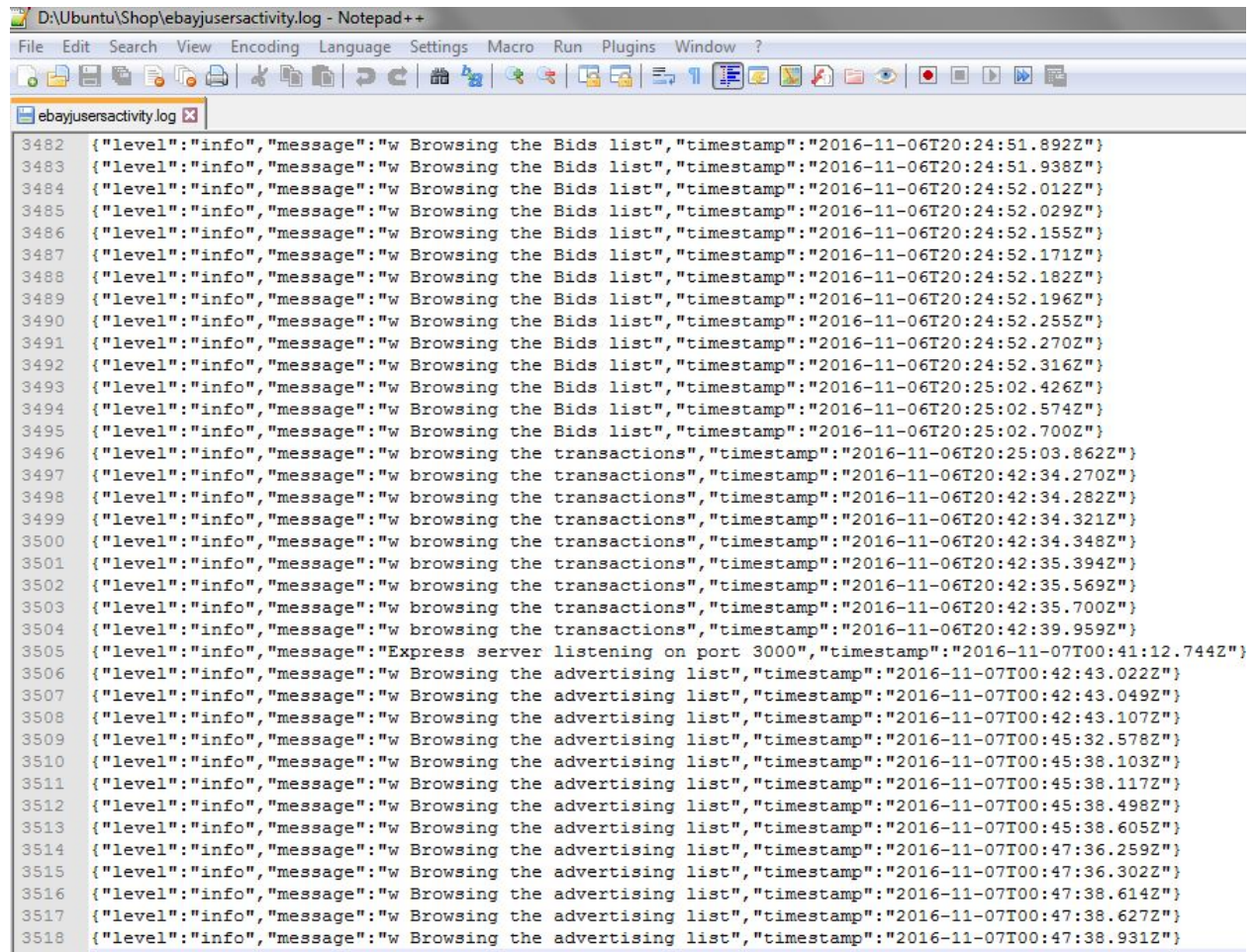
```
jayam@jayam-Inspiron-5521: ~/WebstormProjects/Shop/node_modules
  at /home/jayam/WebstormProjects/Shop/node_modules/express/node_modules/connect/node_modules/send/lib/send.js:320:26
  at FSReqWrap.oncomplete (fs.js:82:15)
  ✓ should return empty if the user is not logged in

  Login
  username w
  password w
  done function verified(err, user, info) {
    if (err) { return self.error(err); }
    if (!user) { return self.fail(info); }
    self.success(user, info);
  }
  Pushing Entry in queue - [object Object]
  As response que already present, moving ahead to publish in msgque - function () {
    console.log('publishing entry ' + correlationId+ ' -- '+Date.now());
    self.connection.publish(queue_name, msg_payload, {
      correlationId: correlationId,
      contentType: CONTENT_TYPE,
      contentEncoding: CONTENT_ENCODING,
      replyTo: self.response_queue
    });
  }
  publishing entry d66a2a67b6dbcf27c4e33cde86662af8 -- 1478462280465
  subscribing at respose queue - amq.gen-iQvAWPzYQpUeo2oq4X6U9Q -- 1478462280468
  received message d66a2a67b6dbcf27c4e33cde86662af8 -- 1478462280468
  call back of actual request
  Client received results
  [ { _id: '5817c9f1bb3230130529cc72',
    firstname: 'w',
    lastname: 'w',
    password: '3ffac31a441f95de7ffeacd498f1e5461ece31da',
    email: 'w',
    birthday: '2016-12-31',
    location: 'w',
    ebayhandle: 'w',
    lastLogin: '2016-11-06T06:01:23.948Z' } ]
  null
  UOB { username: 'w',
    password: '3ffac31a441f95de7ffeacd498f1e5461ece31da' }
  Here ...
  err ... null
  info ... undefined
  user ... { username: 'w',
    password: '3ffac31a441f95de7ffeacd498f1e5461ece31da' }
  abt to enter login { username: 'w',
    password: '3ffac31a441f95de7ffeacd498f1e5461ece31da' }
  abt to enter login
  ZZZZZZlogin
  {"username":"w","password":"3ffac31a441f95de7ffeacd498f1e5461ece31da"}
  ZZZZZZabout to wxit
  Http Status Code should Be 200 - 200
  ✓ should login (52ms)

  5 passing (180ms)

jayam@jayam-Inspiron-5521:~/WebstormProjects/Shop/node_modules$
```

Snippet of Ebayuseractivity.log



```
3482 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:24:51.892Z"}
3483 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:24:51.938Z"}
3484 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:24:52.012Z"}
3485 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:24:52.029Z"}
3486 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:24:52.155Z"}
3487 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:24:52.171Z"}
3488 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:24:52.182Z"}
3489 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:24:52.196Z"}
3490 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:24:52.255Z"}
3491 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:24:52.270Z"}
3492 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:24:52.316Z"}
3493 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:25:02.426Z"}
3494 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:25:02.574Z"}
3495 {"level":"info","message":"w Browsing the Bids list","timestamp":"2016-11-06T20:25:02.700Z"}
3496 {"level":"info","message":"w browsing the transactions","timestamp":"2016-11-06T20:25:03.862Z"}
3497 {"level":"info","message":"w browsing the transactions","timestamp":"2016-11-06T20:42:34.270Z"}
3498 {"level":"info","message":"w browsing the transactions","timestamp":"2016-11-06T20:42:34.282Z"}
3499 {"level":"info","message":"w browsing the transactions","timestamp":"2016-11-06T20:42:34.321Z"}
3500 {"level":"info","message":"w browsing the transactions","timestamp":"2016-11-06T20:42:34.348Z"}
3501 {"level":"info","message":"w browsing the transactions","timestamp":"2016-11-06T20:42:35.394Z"}
3502 {"level":"info","message":"w browsing the transactions","timestamp":"2016-11-06T20:42:35.569Z"}
3503 {"level":"info","message":"w browsing the transactions","timestamp":"2016-11-06T20:42:35.700Z"}
3504 {"level":"info","message":"w browsing the transactions","timestamp":"2016-11-06T20:42:39.959Z"}
3505 {"level":"info","message":"Express server listening on port 3000","timestamp":"2016-11-07T00:41:12.744Z"}
3506 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:42:43.022Z"}
3507 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:42:43.049Z"}
3508 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:42:43.107Z"}
3509 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:45:32.578Z"}
3510 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:45:38.103Z"}
3511 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:45:38.117Z"}
3512 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:45:38.498Z"}
3513 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:45:38.605Z"}
3514 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:47:36.259Z"}
3515 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:47:36.302Z"}
3516 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:47:38.614Z"}
3517 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:47:38.627Z"}
3518 {"level":"info","message":"w Browsing the advertising list","timestamp":"2016-11-07T00:47:38.931Z"}
```