Student Name: Siqi Li
GitHub Repo Under NEU Organization:
https://github.ccs.neu.edu/lisiqi/CS6650-DSBS-Repo

Explanation About Client Structure and How it works:

➔ *Database Design:*

- One table names "Puchases" to store all purchase request as each row input
- Table contains:
  - UUID as primary key, and avoid data duplication
  - StoreID as Integer
  - CustomerID as Integer
  - Data as String -> with the given date send by client2 purchase URL
  - CreateTime as Timestamp, takes the server system time while insert to DB
  - PurchaseBody as Json that contains all purchase items information

➔ *Total Three Packages @ "SuperMarketServer":*

MVC pattern used

- model: (general utils for Server)
  - PurchaseRecord:
    - the data access object for each purchase request received
    - Contains attributes match the database purchases table for each columns
- dal: (the DB connection)
  - DBCPDataSource:
    - Use HikariCP as the Database Connection Pool configuration
    - set a max pool size of 15
    - set max possible lifetime for connection in pool for 20 mins
  - PurchaseDao:
    - Setup and close Database Connection Pool (DBCP) connections
    - Insert data to purchase table with executing SQL queries
    - Enforce successful insertion, when insertion fail, keep inserting until successful, avoid failed insertion that give a failed http response back to client and cause a resend of duplicate purchase call
- tools: (the purchase servlet)
  - PurchaseServlet: the server that handles purchase requests
    - have a DBCPDatasource pool attribute for DB connections
    - valid http request if contains required URL path as header information
    - valid http request body is not empty
    - Get information from the URL path and request body to insert to DB, and get DBCP (Database Connection Pool) connections for each post request and close DBCP connection after done insertion
    - write captured information to response body, and send back

➔ *Total two Packages @ "SuperMarketClient":*

- model: (general utils for both client2)
  - Record: the record object used to store all latency results for Client2
  - LogStderr: the error logger for write out system error to err.txt file
  - HttpMethod: Enum for POST and GET
  - CmdParser: set up required and optional command line input options to run client2
- part2: (package for client 2 to generate requested outputs)
  - Clinet2:
    - Main for execute all threads by calling SingleThread in package part2
    - Check if command line arguments are valid
    - Generate output results for part 2, print out to console
  - SingleThread:
    - the single runnable object represents as one thread to make purchase requests with remote server
    - when send purchase request failed, retry maximum 3 times to make purchase request send successfully
    - Use AtomicInteger to count the total successful/unsuccessful requests
    - Use CountDownLatch to initiate phase 2 and phase 3 threads
    - designed for Client 2 to capture start and end time of each request, in order to get latency time
    - use LogStderr to log errors into txt file
  - RecordWriter:
    - Runnable object used to generate Records object and write to csv file
    - Using BlockingQueue, and work as a Consumer, when consider each SingleThread are all producers of the queue
  - ReportGenerator:
    - Generate output results for part 2, print out to console
    - Read in all Records stored in CSV and get all Latency in to a list and sort list
    - Analysis all latency list, and get mean, median, max and p99 response time

Below are the output results for Client2 and analysis Charts:

➔ Results based on run Server on EC2, and run client on EC2 and database as RDS

➔ *With 32 Threads Single Server:*

```
[ec2-user@ip-172-31-26-136 ~]$ java -jar SuperMarketClient.jar -ip 3.90.186.171 -s 32
Generate Output Result For Part2.Client2 Based On Following Parameters:
IP Address: 3.90.186.171, Max Store: 32, numOfCustomersPerStore: 1000, maxItemId: 100000
numPurchasesPerHour: 300, numItemsPerPurchase: 5, @ Date: 2021-01-01
================================================================================
1. Total number of successful requests sent: 86400
2. Total number of unsuccessful requests: 0
3. Mean response time for POSTs:  10.664 milliseconds
4. Median response time for POSTs:  9.000 milliseconds
5. The total run time (wall time) for all phases to complete: 42.479 second
6. Throughput: 2033.946 requests/second
7. P99 (99th percentile) response time for POSTs: 30 milliseconds
8. Max response time for POSTs: 547 milliseconds
```

➔ *With 32 Threads Load Balanced Server:*

```
[ec2-user@ip-172-31-26-20 ~]$ java -jar SuperMarketClient.jar -ip DSBSLoadBalancer-485728369.us-east-1.elb.amazonaws.com -s 32
Generate Output Result For Part2.Client2 Based On Following Parameters:
IP Address: DSBSLoadBalancer-485728369.us-east-1.elb.amazonaws.com, Max Store: 32, numOfCustomersPerStore: 1000, maxItemId: 100000
numPurchasesPerHour: 300, numItemsPerPurchase: 5, @ Date: 2021-01-01
================================================================================
1. Total number of successful requests sent: 86400
2. Total number of unsuccessful requests: 0
3. Mean response time for POSTs:  7.481 milliseconds
4. Median response time for POSTs:  7.000 milliseconds
5. The total run time (wall time) for all phases to complete: 30.897 second
6. Throughput: 2796.388 requests/second
7. P99 (99th percentile) response time for POSTs: 19 milliseconds
8. Max response time for POSTs: 269 milliseconds
```

➔ *With 64 Threads Single Server:*

```
[ec2-user@ip-172-31-26-136 ~]$ java -jar SuperMarketClient.jar -ip 3.90.186.171 -s 64
Generate Output Result For Part2.Client2 Based On Following Parameters:
IP Address: 3.90.186.171, Max Store: 64, numOfCustomersPerStore: 1000, maxItemId: 100000
numPurchasesPerHour: 300, numItemsPerPurchase: 5, @ Date: 2021-01-01
================================================================================
1. Total number of successful requests sent: 172800
2. Total number of unsuccessful requests: 0
3. Mean response time for POSTs:  20.369 milliseconds
4. Median response time for POSTs:  19.000 milliseconds
5. The total run time (wall time) for all phases to complete: 74.129 second
6. Throughput: 2331.072 requests/second
7. P99 (99th percentile) response time for POSTs: 76 milliseconds
8. Max response time for POSTs: 564 milliseconds
```

➔ *With 64 Threads Load Balanced Server:*

```
[ec2-user@ip-172-31-26-20 ~]$ java -jar SuperMarketClient.jar -ip DSBSLoadBalancer-485728369.us-east-1.elb.amazonaws.com -s 64
Generate Output Result For Part2.Client2 Based On Following Parameters:
IP Address: DSBSLoadBalancer-485728369.us-east-1.elb.amazonaws.com, Max Store: 64, numOfCustomersPerStore: 1000, maxItemId: 100000
numPurchasesPerHour: 300, numItemsPerPurchase: 5, @ Date: 2021-01-01
================================================================================
1. Total number of successful requests sent: 172800
2. Total number of unsuccessful requests: 0
3. Mean response time for POSTs:  8.231 milliseconds
4. Median response time for POSTs:  7.000 milliseconds
5. The total run time (wall time) for all phases to complete: 33.979 second
6. Throughput: 5085.494 requests/second
7. P99 (99th percentile) response time for POSTs: 23 milliseconds
8. Max response time for POSTs: 317 milliseconds
```

➔ *With 128 Threads Single Server:*

```
[ec2-user@ip-172-31-26-136 ~]$ java -jar SuperMarketClient.jar -ip 3.90.186.171 -s 128
Generate Output Result For Part2.Client2 Based On Following Parameters:
IP Address: 3.90.186.171, Max Store: 128, numOfCustomersPerStore: 1000, maxItemId: 100000
numPurchasesPerHour: 300, numItemsPerPurchase: 5, @ Date: 2021-01-01
================================================================================
1. Total number of successful requests sent: 345600
2. Total number of unsuccessful requests: 0
3. Mean response time for POSTs:  40.256 milliseconds
4. Median response time for POSTs:  40.000 milliseconds
5. The total run time (wall time) for all phases to complete: 144.448 second
6. Throughput: 2392.556 requests/second
7. P99 (99th percentile) response time for POSTs: 125 milliseconds
8. Max response time for POSTs: 665 milliseconds
```

➔ *With 128 Threads Load Balanced Server:*

```
[ec2-user@ip-172-31-26-20 ~]$ java -jar SuperMarketClient.jar -ip DSBSLoadBalancer-485728369.us-east-1.elb.amazonaws.com -s 128
Generate Output Result For Part2.Client2 Based On Following Parameters:
IP Address: DSBSLoadBalancer-485728369.us-east-1.elb.amazonaws.com, Max Store: 128, numOfCustomersPerStore: 1000, maxItemId: 100000
numPurchasesPerHour: 300, numItemsPerPurchase: 5, @ Date: 2021-01-01
================================================================================
1. Total number of successful requests sent: 345600
2. Total number of unsuccessful requests: 0
3. Mean response time for POSTs:  13.268 milliseconds
4. Median response time for POSTs:  10.000 milliseconds
5. The total run time (wall time) for all phases to complete: 51.319 second
6. Throughput: 6734.348 requests/second
7. P99 (99th percentile) response time for POSTs: 48 milliseconds
8. Max response time for POSTs: 287 milliseconds
```

➔ *With 256 Threads Single Server:*

```
[ec2-user@ip-172-31-26-136 ~]$ java -jar SuperMarketClient.jar -ip 3.90.186.171 -s 256
Generate Output Result For Part2.Client2 Based On Following Parameters:
IP Address: 3.90.186.171, Max Store: 256, numOfCustomersPerStore: 1000, maxItemId: 100000
numPurchasesPerHour: 300, numItemsPerPurchase: 5, @ Date: 2021-01-01
================================================================================
1. Total number of successful requests sent: 691200
2. Total number of unsuccessful requests: 0
3. Mean response time for POSTs:  93.342 milliseconds
4. Median response time for POSTs:  93.000 milliseconds
5. The total run time (wall time) for all phases to complete: 327.218 second
6. Throughput: 2112.353 requests/second
7. P99 (99th percentile) response time for POSTs: 286 milliseconds
8. Max response time for POSTs: 1456 milliseconds
```

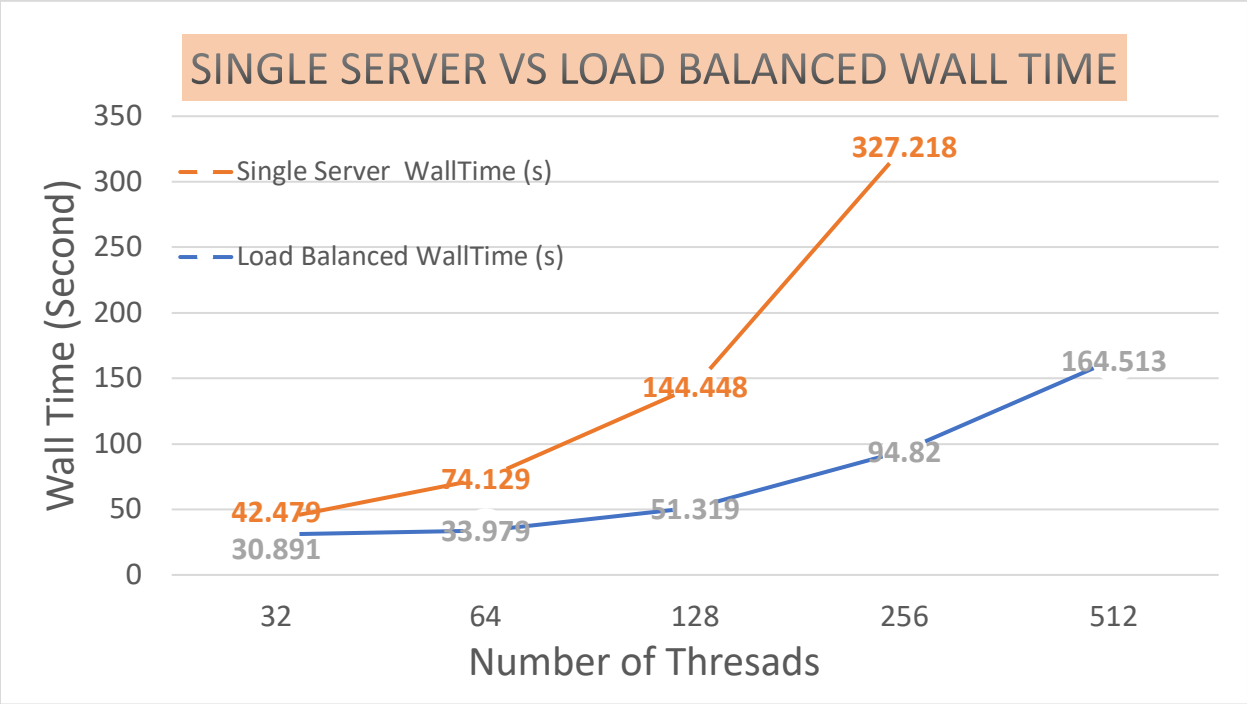
➔ *With 256 Threads Load Balanced Server:*

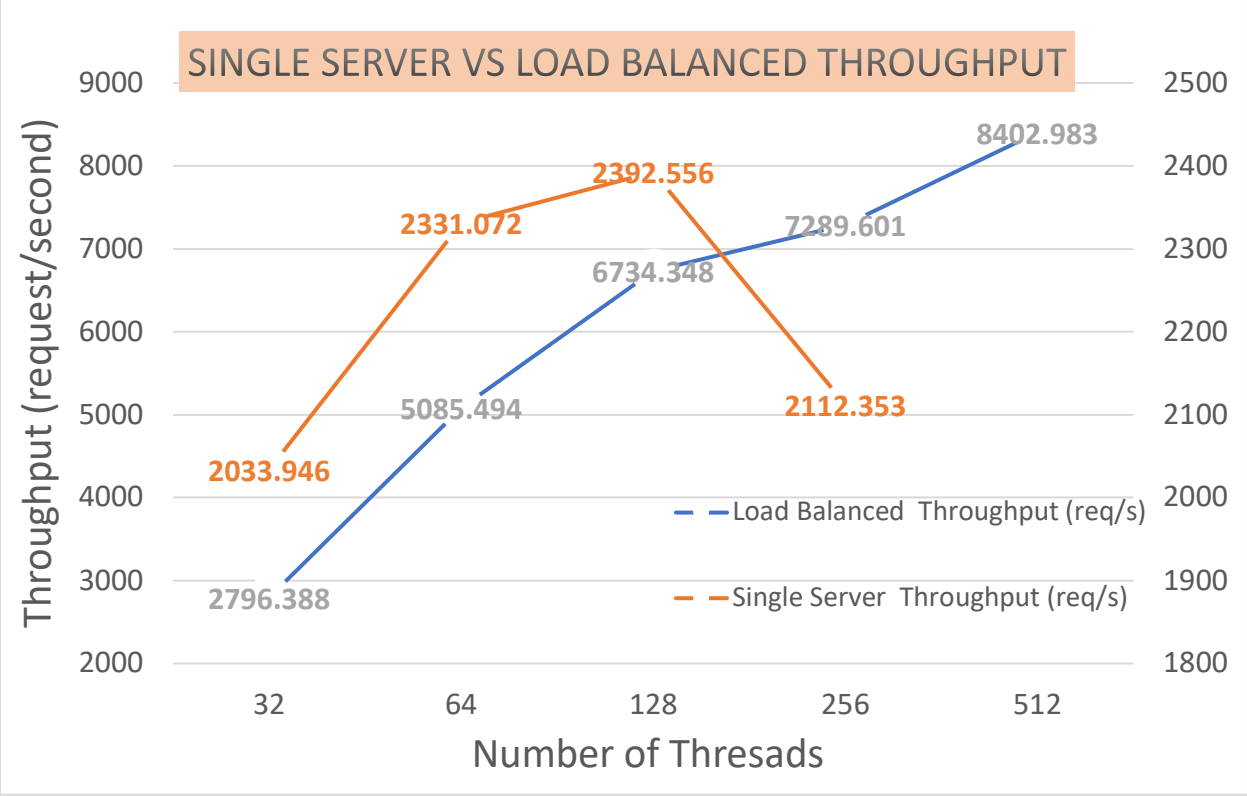
```
[ec2-user@ip-172-31-26-20 ~]$ java -jar SuperMarketClient.jar -ip DSBSLoadBalancer-485728369.us-east-1.elb.amazonaws.com -s 256
Generate Output Result For Part2.Client2 Based On Following Parameters:
IP Address: DSBSLoadBalancer-485728369.us-east-1.elb.amazonaws.com, Max Store: 256, numOfCustomersPerStore: 1000, maxItemId: 100000
numPurchasesPerHour: 300, numItemsPerPurchase: 5, @ Date: 2021-01-01
================================================================================
1. Total number of successful requests sent: 691200
2. Total number of unsuccessful requests: 0
3. Mean response time for POSTs:  26.217 milliseconds
4. Median response time for POSTs:  15.000 milliseconds
5. The total run time (wall time) for all phases to complete: 94.82 second
6. Throughput: 7289.601 requests/second
7. P99 (99th percentile) response time for POSTs: 142 milliseconds
8. Max response time for POSTs: 367 milliseconds
```

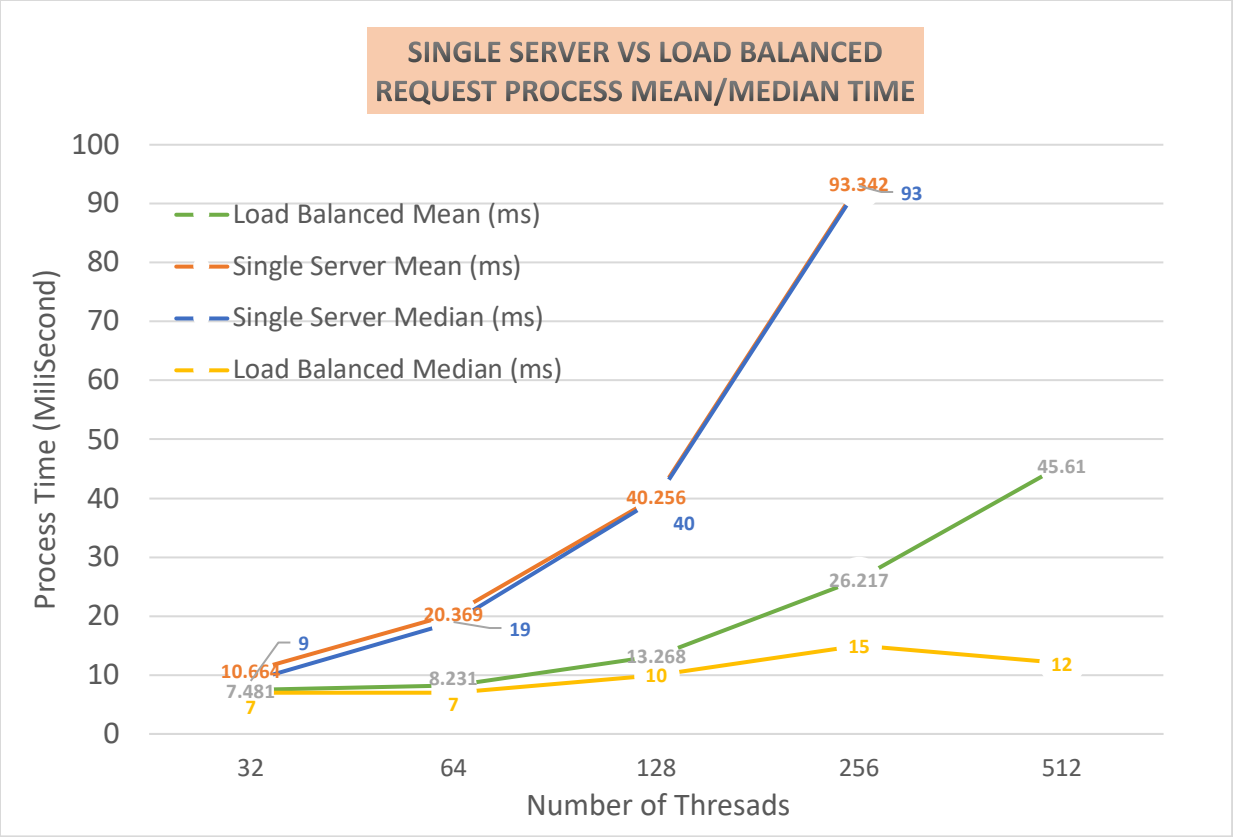
➔ *With 512 Threads Load Balanced Server:*

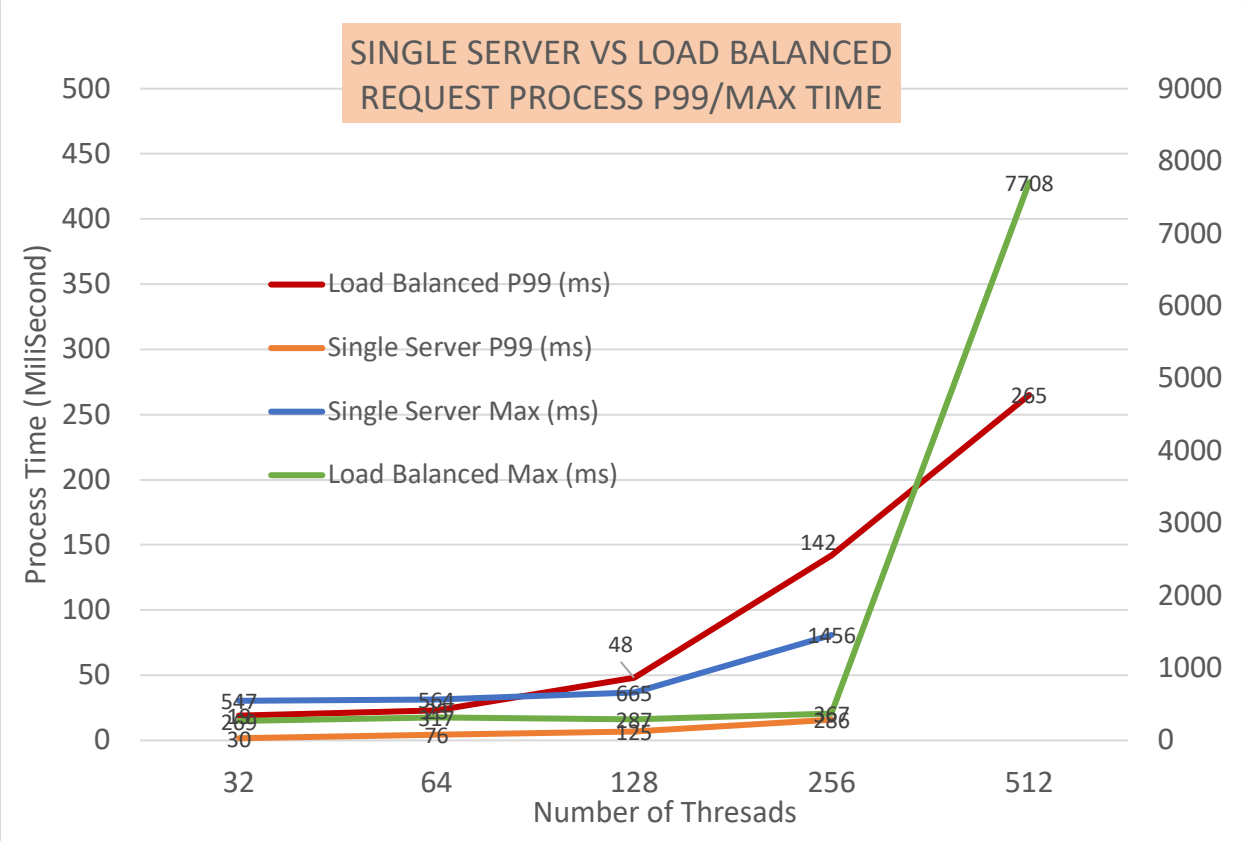
```
Generate Output Result For Part2.Client2 Based On Following Parameters:
IP Address: DSBSLoadBalancer-485728369.us-east-1.elb.amazonaws.com, Max Store: 512, numOfCustomersPerStore: 1000, maxItemId: 100000
numPurchasesPerHour: 300, numItemsPerPurchase: 5, @ Date: 2021-01-01
================================================================================
1. Total number of successful requests sent: 1382400
2. Total number of unsuccessful requests: 2
3. Mean response time for POSTs:  45.610 milliseconds
4. Median response time for POSTs:  12.000 milliseconds
5. The total run time (wall time) for all phases to complete: 164.513 second
6. Throughput: 8402.983 requests/second
7. P99 (99th percentile) response time for POSTs: 265 milliseconds
8. Max response time for POSTs: 7708 milliseconds
```

➔ *Analysis Charts:*






SINGLE SERVER VS LOAD BALANCED REQUEST PROCESS MEAN/MEDIAN TIME
Process Time (MiliSecond)
Number of Thresads
Load Balanced Mean (ms)
Single Server Mean (ms)
Single Server Median (ms)
Load Balanced Median (ms)
100
90
80
70
60
50
40
30
20
10
0
32
64
128
256
512
93.342
93
40.256
40
20.369
19
10.664
9
7.481
7
8.231
7
13.268
10
26.217
15
45.61
12


SINGLE SERVER VS LOAD BALANCED REQUEST PROCESS P99/MAX TIME
Process Time (MiliSecond)
Number of Thresads
Load Balanced P99 (ms)
Single Server P99 (ms)
Single Server Max (ms)
Load Balanced Max (ms)
500
450
400
350
300
250
200
150
100
50
0
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
32
64
128
256
512
7708
265
142
48
1456
665
564
547
287
286
125
76
30