# Performance and Load Test Report

Prepared by: Manoj Kumar Gudala

Reviewed by:

Approved by:

Table of Contents

[Performance and Load Test Report 1](#_Toc65013777)

[1. Summary: - 3](#_Toc65013778)

[2. Introduction 3](#_Toc65013779)

[1. Scope 3](#_Toc65013780)

[2. Glossary 3](#_Toc65013781)

[3. Scenario Specification 4](#_Toc65013782)

[4. Load Test Hardware and OS 5](#_Toc65013783)

[1. Minimum hardware requirements for Linux or Windows: 5](#_Toc65013784)

[2. Minimum OS and software requirements for Linux: 5](#_Toc65013785)

[3. Minimum OS and software requirements for Windows: 5](#_Toc65013786)

[5. Load Test Software 5](#_Toc65013787)

[6. Test Data and Requirements 7](#_Toc65013788)

[7. Test Case Specification 8](#_Toc65013789)

[3. Results 9](#_Toc65013790)

[1. Constant Load performance test 9](#_Toc65013791)

[1. Stepping threads 9](#_Toc65013792)

[2. Summary Report 10](#_Toc65013793)

[3. Error Report 10](#_Toc65013794)

[4. Throughput versus Hits 11](#_Toc65013795)

[5. Transaction time in seconds 11](#_Toc65013796)

[2. Stress Testing 12](#_Toc65013797)

[1. Overview Report 12](#_Toc65013798)

[2. Error Report 12](#_Toc65013799)

[3. Active Thread Over Time 13](#_Toc65013800)

[4. Call Back Responses Throughput 13](#_Toc65013801)

[5. Transaction time in seconds 13](#_Toc65013802)

[4. Conclusions 14](#_Toc65013803)

[5. References 14](#_Toc65013804)

# Summary: -

We have tested the load test for 5000 different merchant’s data across the different Payee. We have tested with 10 to 100 transactions per second we didn’t see any failures and it is able to handle around 30K requests and thereafter we have increased 500 transactions per second steady load we have seen failures of 60 out of 10K

# Introduction

## Scope

This document is a load test report (LTR). The following pages provide the settings, results, and conclusions derived from the load tests, as specified. Successful websites and web applications demonstrate great performance, having the ability to scale up and accurately execute fast while concurrently serving all types of users. Scalability is a web application’s ability to adapt to growing demands without performance degradation. Load testing is the process of putting demands on a system and measuring its responses under stressful conditions in order to identify the maximum operating capacity and pin-point any virtual bottlenecks. Monitoring is the process of verifying that end-users can interact with an existing website or web application as required. While load testing is normally performed before a website or web application is launched, monitoring is continuously required 24x7 after the release. Rather than merely sampling infrastructure parameter values, there are programmed virtual users that accurately test the system under load, and later reused for precise monitoring and alerting on performance related events.

## Glossary

|  |  |
| --- | --- |
| Concurrent Users | The number of users that simultaneously operate the same system at the same time. |
| CPU Utilization | A computer's usage of processing resources, or the amount of work handled by a CPU |
| Disk Utilization | Disk utilization is the amount of space being used on a disk or how busy the  disk is with respect to I/O operations |
| Expected response time | The allowed interval from when a user initiates a request to the instant at which the first part of the response is received at by the application. |
| Memory Utilization | The amount of memory that a software application uses or references while running. |
| Network Utilization | The percentage of a network's bandwidth that is currently being consumed by network traffic. |
| Scalability Testing | The testing of a software application for measuring its capability to scale up or scale out in terms of any of its non-functional capability. |
| Throughput | The rate at which data can be transferred. Throughput is usually measured  in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot. |
| Transactions per second | The number of actions performed per second. For example, the number of database transactions performed per second. |
| Virtual Users | Programmed simulation of user actions used for testing a system in a realistic |

.

## Scenario Specification

Load testing is a part of a more general process known as performance testing. Examples of load testing operations may include: downloading a series of large files from the Internet, running multiple applications on a computer or server simultaneously, assigning many jobs to a printer in a queue, subjecting a server to a large amount of e-mail traffic, writing and reading data to and from a hard disk continuously, etc.

Load testing can be conducted in two ways. Longevity testing, also called endurance testing, evaluates a system's ability to handle a constant, moderate workload for a long time. Volume testing, on the other hand, subjects a system to a heavy workload for a limited time. Either approach makes it possible to pinpoint bottlenecks, bugs and component limitations. For example, a computer may have a fast processor but a limited amount of RAM (random-access memory). Load testing can provide the user with a general idea of how many applications or processes can be run simultaneously while maintaining the rated level of performance.

Load testing differs from stress testing, which evaluates the extent to which a system keeps working when subjected to extreme workloads or when some of its hardware or software has been compromised. The primary goal of load testing is to define the maximum amount of work a system can handle without significant performance degradation.

Performance and load testing is normally performed in order to validate multi-client-server complex system. This load test scenario is based on transactions, we need to test static QR code

* There is already a static QR code generate for each cashier
* when a customer makes a transaction by scanning that in any payment app, we receive a call from ICICI with a response on above URL with given json.
* merchantTranId is the key we use for identifying the transaction. this is what we pass along when we create a QR code (must be unique and must contain TESTSTATIC as prefix)
* on SUCCESS callback we send a notification to merchant using FCM token or Pushy token (whatever is available depending upon device)

ICICI Call back request json -

{

"merchantId" : "XXXXX",  
"subMerchantId" : "XXXX",  
"terminalId" : "XXXX",  
"BankRRN" : "105012733421",  
"merchantTranId" : "TESTSTATIC771964",  
"PayerName" : "null",  
"PayerMobile" : "0000000000",  
"PayerVA" : "abc@abc",  
"PayerAmount" : "1.00",  
"TxnStatus" : "SUCCESS",  
"TxnInitDate" : "20210219122133",  
"TxnCompletionDate" : "20210219201030"  
}

## Load Test Hardware and OS

Load test hardware requirements depend on mainly on the footprint of the virtual client that creates load on the EXAMPLE server side. In the EXAMPLE Web APP system, the client is a Web browser that buffers thousands of EXAMPLES. That creates rather a fat client with huge memory requirements.

### Minimum hardware requirements for Linux or Windows:

CPU Intel® Core™ i5

RAM 32 GB HD or SSD 500 GB

Ethernet 1 GB/s

Internet bandwidth 30M UP, 30M DOWN

### Minimum OS and software requirements for Linux:

Linux Mint 18.3 64-bit

Java virtual machine 64-bit for LINUX VNC

### Minimum OS and software requirements for Windows:

MS-Windows Server 2012 R2 64-bit

Java virtual machine 64-bit for Window

RDP

## Load Test Software

The selected load generation environment is Apache JMeter™ version 4.0, with a standard and extra set of JMeter Plug-ins version x.x.x. The Apache JMeter™ application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions.

Apache JMeter™ may be used to test performance both on static and dynamic resources (files, Servlets, Perl scripts, Java Objects, Data Bases and Queries, FTP Servers and more). It can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types. Moreover, it can be used it to make a graphical analysis of performance or to test your server/script/object behavior under heavy concurrent load.

Apache JMeter™ supports load and performance test many different applications/server/protocol types:

Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, …)

SOAP / REST Webservices

FTP

Database via JDBC LDAP

Message-oriented middleware (MOM) via JMS Mail - SMTP(S), POP3(S) and IMAP(S)

Native commands or shell scripts

TCP

Java Objects

**Additional features:**

Full featured Test IDE that allows fast Test Plan recording (from Browsers or native applications), building and debugging.

Command-line mode (Non GUI / headless mode) to load test from any Java compatible OS

(Linux, Windows, Mac OSX, …)

A complete and ready to present dynamic HTML report

Easy correlation through ability to extract data from most popular response formats, HTML, JSON , XML or any textual format

Complete portability and 100% Java purity.

Full multi-threading framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups.

Caching and offline analysis/replaying of test results. Highly Extensible core:

Pluggable Samplers allow unlimited testing capabilities.

Scriptable Samplers (JSR223-compatible languages like Groovy and BeanShell) Several load statistics may be chosen with pluggable timers.

Data analysis and visualization plugins allow great extensibility as well as personalization. Functions can be used to provide dynamic input to a test or provide data manipulation.

Easy Continuous Integration through 3rd party Open-Source libraries for Maven, Gradle and

Jenkins

**NOTE - JMeter is not a browser:**

JMeter is not a browser, it works at protocol level. As far as web-services and remote services are concerned, JMeter looks like a browser (or rather, multiple browsers); however, JMeter does not perform all the actions supported by browsers. In particular, JMeter does not execute the Javascript found in HTML pages. Nor does it render the HTML pages as a browser does (it's possible to view the response as HTML etc., but the timings are not included in any samples, and only one sample in one

thread is ever displayed at a time).

## Test Data and Requirements

|  |  |  |
| --- | --- | --- |
| **Param** | **Value** | **Comment** |
| Count of Merchants | 5000 |  |
| Sub merchants | 10000 |  |
| terminal\_id | 5000 |  |
| random\_trans\_id | 999999999 |  |
| random\_amount | 99999999 |  |
| random\_status | SUCCESS | Hard-coded |

## Test Case Specification

Following is the Test case Spec Flow

Diagram

Description automatically generated

* End user initiates the Jmeter request/ICICI call back
* Jmeter client stores the request received time against the transaction\_id
* Jmeter send the request to the ICICI Call back API
* Jmeter receives the request from the ICICI call back service
* In the app System it processes the request and ask the push notification service to the Merchant for that transaction
* Merchant/Test Client receives the notification and logs the Request Received time against the same transaction\_id received in the Step 2
* Test Client replies back with 200 OK

Here we are calculating the total transaction time as

Total transaction time = Time logged by the Merchant/Test Client in Step 6 - Time logged by the Jmeter in the Step 2

# Results

## Constant Load performance test

### Stepping threads

Graphical user interface, application, table, Excel

Description automatically generated

### Summary Report

Graphical user interface

Description automatically generated with low confidence

### Error Report

Graphical user interface, application, table

Description automatically generated

### Throughput versus Hits

Chart, line chart

Description automatically generated

### Transaction time in seconds

Chart, pie chart

Description automatically generated

## Stress Testing

This test ran with Following Params

Number of Virtual User – 500

Iterate the Virtual users -- 20

### Overview Report

Graphical user interface, application

Description automatically generated

### Error Report

Graphical user interface, application

Description automatically generated

### Active Thread Over Time

Chart

Description automatically generated

### Call Back Responses Throughput

Chart, line chart

Description automatically generated

### Transaction time in seconds

# Conclusions

Under the load generated by 100 virtual users, ranging from 10 to 100 Virtual Users, the following

happened

* The test duration range was approximately from 10 minutes to 20 minutes.
* We have observed the total transaction\_time varied from 1 second to 9 seconds
* We didn’t see any failure response from the ICICI Call Back Service

When the load increased to 500 virtual users, ranging from 10 to 500 Virtual Users, the following

happened

* The test duration range was approximately from 3 minutes to 5 minutes.
* We have observed the total transaction\_time varied from 1 second to 30+ seconds
* We have seen any failure response from the ICICI Call Back Service, the requests are getting timed out
* There were around 60 failures out of 10K requests

# References

• Software Performance and Scalability: A Quantitative Approach, Henry H. Liu, Wiley 1st edition,

May 26, 2009.

• The Software Verification and Validation: An Engineering and Scientific Approach, Marcus S.

Fisher, Springer 2007 edition, December 20, 2006.

• Performance Testing With JMeter 2.9, Bayo Erinle, Packt Publishing, July 26, 2013.