INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

PERFORMANCE ANALYSIS OF ROCKSDB AND ITS INTEGRATION IN OPENEDX

PRINCIPAL INVESTIGATOR

Prof. D B Phatak

Project-in Charge: Mr. Nagesh Karmali and Miss Firuza Aibara

Project Mentor

Nithin S

Project Members

Abhishek Chattopadhyay

Ekta Agrawal

Simran Goyal

LAST UPDATE: July 4, 2017

Summer Internship 2017

Project Approval Certificate

Department of Computer Science and Engineering

Indian Institute of Technology Bombay

The project entitled "**Performance Analysis of RocksDB and it's Integration in OpenEdx**" submitted by Abhishek Chattopadhyay, Ekta Agrawal and Simran Goyal is approved for Summer Internship 2017 programme from 8th May 2017 to 5th july 2017, at Department of Computer Science and Engineering, IIT Bombay.

_____	_____
Prof. Deepak B. Phatak,	**Mr. Nagesh Karmali,**
Dept. of CSE, IITB	**Dept. of CSE, IITB**
Principal Investigator	**Project Incharge**

Place: Indian Institute of Technology, Bombay

Date: July 05, 2017

Declaration

I declare that this written submission represents my ideas in my own language and in case when others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Abhishek Chattopadhyay

IIT (ISM) Dhanbad

_____

Ekta Agrawal

NIT Raipur

_____

Simran Goyal

NIT Raipur

_____

Acknowledgement

We, the summer interns of the team Performance Analysis of RocksDB and its Integration into OpenEdX, want to acknowledge our deep gratitude to all those who have helped us put our ideas to perfection in our assigned tasks to a level well above simplicity and into something concrete and unique. We, wholeheartedly thank Prof.D.B. Phatak for having faith in us, selecting us to be a part of his valuable project and for constantly motivating us to do better. We are very thankful to our project incharge Mr.Nagesh Karmali and Miss Firuza Aibara and our mentor Nithin S for their valuable suggestions. They were and are always there to show us the right track when needed help. With the help of their brilliant guidance and encouragement, we all were able to complete our tasks properly and were up to the mark in all tasks assigned. During the process, we got a chance to see the stronger side of our technical and non-technical aspects and also strengthen our concepts. We would also like to thank Mr. Abhilash from the ASL lab and Mr. Muralee and Miss. Silpa from the T10Kt Lab for their constant support on the technical issues that we faced. Last but not the least, we wholeheartedly thank all our other colleagues working in different projects under Prof. D.B Phatak for helping us evolve better with their critical advice.

Abstract

The issue of storage space is very paramount for any Organisation, which needs to store a huge amount of User-specific data. Open-EdX is a MOOC (Massive Online Open Course) Platform that provides online courses and is used by multiple organisations worldwide. RocksDB is an embedded, high-performance, key-value storage engine developed at Facebook Inc.

This project focuses on Performance Testing and Analysis of RocksDB v/s InnoDB (the default transaction engine used by MySQL) and subsequently trying to integrate RocksDB with OpenEdX.

All tests have been performed with the help of JMeter, an open-source Performance Measurement software, with the results displayed in Graphical and Tabulated Forms.

List of Figures

List of Tables

Table of Contents

1. Introduction

MySQL/MariaDB

MySQL is an open-source relational database management system (RDBMS) software which provides an implementation of a SQL database very well suited for small to medium web pages. The database is free and open source with a commercial license available.

MySQL is currently owned by Oracle who bought it off Sun Microsystems.

Common applications with MySQL support include PHP and Java based web applications that require backend storage viz. Drupal, Joomla, etc. A lot of applications that use MySQL are geared towards the LAMP stack (Linux, Apache, MySQL, and PHP).

MySQL has one major advantage since it is free; it is usually available on shared hosting packages and can be easily set up in a Linux, UNIX or Windows environment. If a web application requires multiple instances of a database or load balancing or sharding, it is easy to set up multiple instances of the database, requiring only the hardware costs, as opposed to commercial databases that would require a single license for each instance.

After the acquisition of Sun Microsystems by Oracle, some people were concerned about the future of the free version of MySQL. As a result, the original author of MySQL started a project that uses a fork of MySQL to build a new database service named MariaDB (https://mariadb.org), which is intended to be compatible as a replacement for MySQL and also implement new storage engines in the future.

Apache JMeter

The Apache JMeter application is an open source software,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.

Stefano Mazzocchi of the Apache Software Foundation was the original developer of JMeter. He wrote it primarily to test the performance of Apache JServ (now called as Apache Tomcat project). Apache later redesigned JMeter to enhance the GUI and to add functional testing capabilities.

Features of Jmeter are:

- Since it is an open source software, it is freely available.
- It has simple and intuitive GUI.
- It can conduct load and performance test for many server types like Web - HTTP, HTTPS, SOAP, Database via JDBC, LDAP, JMS, Mail – POP3,TCP,Java Objects.
- It is a platform-independent tool. On Linux/Unix, JMeter can be invoked by clicking on JMeter shell script. On Windows, it can be invoked by starting the jmeter.bat file.
- It has full Swing and lightweight component support (precompiled JAR uses packages javax.swing.*).
- JMeter store its test plans in XML format. This means you can generate a test plan using a text editor.
- Its full multithreading framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups.
- Various kinds of Samplers are available for different types of testing.Timers in the test scripts can help in the simulation of different types of load.
- Plug-ins can be added to further extend the tool.

- It can also be used to perform automated and functional testing of the applications.

Open EdX

EdX is a MOOC(Massive Open Online Course) platform started by MIT and Harvard in collaboration for revolutionizing online education. EdX is among the most popular MOOCs in the world because of the advance features it provides for making online education of student fruitful and more effective worldwide. It also provides the author an effective way to design a course.

OpenEdX is the open source platform based on the EdX platform and is created on the Django framework (written in python).

The Open edX platform is a free--and open source--course management system (CMS) that was originally developed by edX. The Open edX platform is used all over the world to host Massive Open Online Courses (MOOCs) as well as smaller classes and training modules.

1.1. Purpose

The project is divided into two major portions:

A. The benchmarking of two MySQL Engines namely, RocksDB and InnoDB and compare their relative performances.

B. Trying to integrate RocksDB into OpenEdX and if integration is successful, then benchmark it

The main purpose in making this document is to describe the newly introduced features in RocksDB. The working and tests performed with this new engine have been described to make any further enhancements in future.

1.2. Scope

Minimizing space amplification is important to efficient hardware use because storage space is the bottleneck in production environments. SSDs process far fewer reads/s and writes/s during peak times under InnoDB than what the hardware is capable of.

RocksDB provides an efficient solution to this issue, under production conditions. As a result it may be possible to use the same engine enabled database system for the OpenEdX framework, thus enabling more efficient usage of disk space and faster execution time.

1.3. Abbreviations and Definitions

1	XML	eXtensible Markup Language
2	JSP	Java Server Pages
3	J2EE	Java 2 Enterprise Edition
4	SQL	Structured Query Language
5	XLS	Excel File
6	GUI	Graphical User Interface
7	ID	Identification number
8	HTTP	Hypertext Transfer Protocol
9	JRE	Java runtime environment
10	HTML	Hyper Text Markup Language
11	FTP	File Transfer Protocol
12	SMTP	Simple Mail Transfer Protocol
13	SUT	System Under Test
14	AUT	Application Under Test
15	TPC	Transaction Processing Performance Council
16	LSM	Log-Structured Merge Tree
17	WAL	Write Ahead Log
18	SST	Sorted String Table
19	SSD	Solid State Drive

Table 1. Abbreviations and Definitions

1.4. Motivation

The factors that motivated us to look into RocksDB as an alternative storage engine are:

- Minimizing space amplification is important to efficient hardware use because storage space is the bottleneck in production environments.
- SSDs process far fewer reads/s and writes/s during peak times under InnoDB than what the hardware is capable of since the query rate per node is relatively low.
- RocksDB provides an efficient solution to this issue, under production conditions since:

2. Objective

The main objective of the project is to conduct a feasibility test of using RocksDB as a storage engine in MySQL/MariaDB, instead of the usual InnoDB engine.

Currently, MySQL uses InnoDB as the default storage engine to perform all the transactions going through. This has some significant drawbacks since:

- The B+ tree data structure used gets fragmented easily, resulting in additional disk space usage.
- Also compression in InnoDB has a space overhead.
- InnoDB adopts "update in place" architecture. Even though there is an update in just 1 record, an entire page has to be written back to storage.

On the other hand,

- RocksDB (LSM) database uses "Append Only" model in which edits are written to WAL.
- The edits are then periodically merged with SST files .
- RocksDB uses LSM and is optimised for reducing space usage.

3. Design Considerations

The following softwares have been used to develop, modify and test MariaDB:

- MariaDB
- RocksDB plugin for MariaDB
- Mozilla Firefox
- Apache JMeter 3.2
- Plugin Manager for Apache JMeter
- Spawner Data Generator

All these software are Open Source and freely available.

The Operating System that we used was Ubuntu 16.04 LTE. The various dependencies for the build and execution of the source are as follows:

- Apache JMeter 3.2 requires JDK8 or above to run.
- MariaDB version should be 10.2 or higher, to include support for RocksDB plugin.

3.1. General Constraints

MariaDB and JMeter have some limitations which include the following. Note that these are necessary for successful processing of the software.

- For changes made in the *jmeter* or *jmeter.properties* file to take effect, JMeter needs to be restarted
- Similarly, MariaDB has to be restarted after making changes in the *my.cnf* file, for the changes to take effect.
- Memory may need to be managed after continuous use of JMeter.

3.2. Goals and Guidelines

The main goal behind this project is to check the feasibility of RocksDB as an alternate to the InnoDB Storage Engine.

- Benchmark the two instances of the database, record the changes observed and do a relative study of feasibility under different scenarios.
- To provide the user with knowledge about the functionality to be used under different conditions.

3.3. Technologies Used

- **Java**

It is a platform independent and object oriented language. Java is used in a wide variety of computing platforms from embedded devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end. Apache JMeter uses Java platform for the development process.

  - JDBC: JMeter uses Java Database Connection for interacting with databases at the backend.
  - JMS: Java Message Service is used by JMeter to exchange messages between clients.
  - Java Mail Api: Apache JMeter uses Java-mail api to use mail services through jmeter.
  - AWT/Swings: Java AWT and Swings has been used in JMeter to generate the user interface.
  - Socket Connection: JMeter uses socket connection packages to implement different protocols.

- **Protocols**

Apache JMeter uses large number of protocols for the virtual users using different web services.

- HTTP: Hypertext Transfer Protocol (HTTP) is used by the http samplers and http config element in the test plan.
- HTTPS: Hypertext Transfer Protocol Secure (HTTPS) is used by JMeter for secured HTTP requests.
- SMTP: SMTP samplers in Apache JMeter uses (Simple Mail Transfer Protocol) SMTP protocols for mail services
- FTP: FTP (File Transfer Protocol) is used by jmeter for the ftp requests (samplers and config element)
- TCP: JMeter uses TCP (Transmission Control Protocol) for TCP Samplers which used to send tcp packets from client to server.
- LDAP: JMeter uses to LDAP samplers to send requests to LDAP servers

- **MariaDB**

It is the open-source fork world's most used open source relational database management that runs as a server providing multi-user access to a number of databases.

- **OpenEdX**

The Open edX Platform is a web-based system for creating, delivering, and analyzing online courses at massive scale. It is the software that powers edx.org and many other online education sites.There are a handful of major components in the Open edX project. The primary component is edx-platform, which contains the learning management and course authoring applications. Where possible, components communicate using stable, documented APIs. Almost all of the

server-side code in the Open edX project is written in Python, with Django as the web application framework.

4. Architecture

MariaDB's Storage Engine Architecture

The MySQL pluggable storage engine architecture enables a database user to select a specialized storage engine for a particular application need while being completely shielded from the need to manage any specific application coding requirements. The server architecture isolates the program user and the Database Administrator from all of the low-level implementation details at the storage level, providing a consistent and easy application model and API.

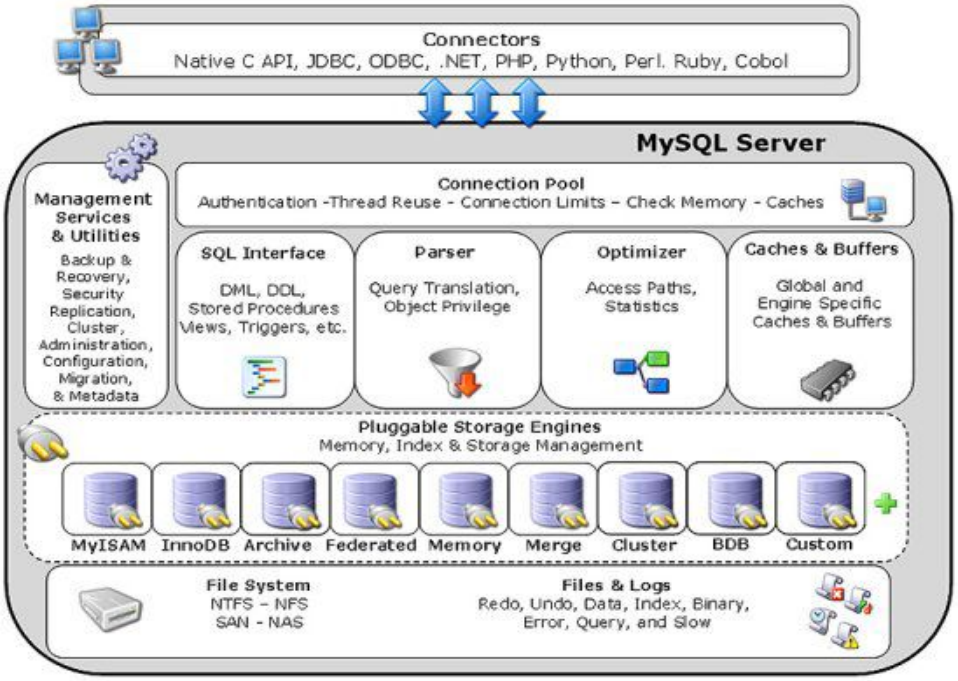


Figure 1. Architecture of MariaDB

The pluggable storage engine architecture provides a standard set of management and support services that are common among all underlying storage engines. The storage engines themselves are the components of the database server that actually perform

actions on the underlying data that is maintained at the physical server level. A major benefit of using specific storage engines is that only the features required for a particular application are delivered, and therefore you have less system overhead in the database, with the end result being more efficient and higher database performance.

RocksDB Engine

RocksDB is a C++ based library that stores keys and values. The keys and values are arbitrary byte streams, wherein the keys are stored in a sorted sequence. New writes occur to new places in the storage where a background compaction process eliminates duplicate entries and processes delete markers.

Data is stored in the form of a LSM tree. There is support for atomically writing a set of keys into the database, and backward and forward iterations over the keys are supported. RocksDB is built using a "pluggable" architecture, which makes it easy to replace parts of it without impacting the overall architecture of the system.

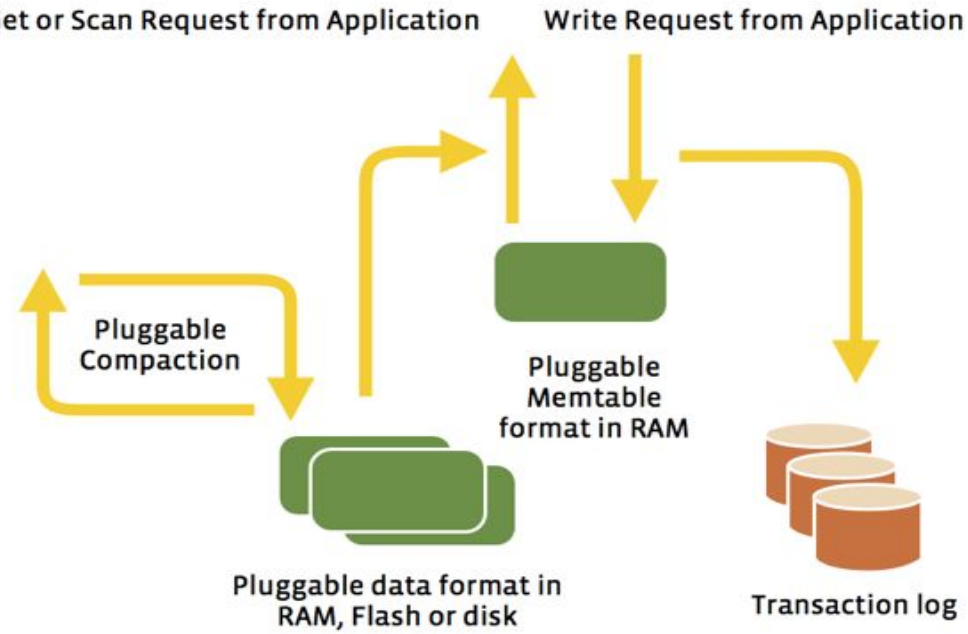


Figure 2. Architecture of RocksDB

JMeter

All load generation tools consist of a “protocol engine” which replicates the traffic i.e. the number of requests that would be generated by the real users driving the interface of the system under test.

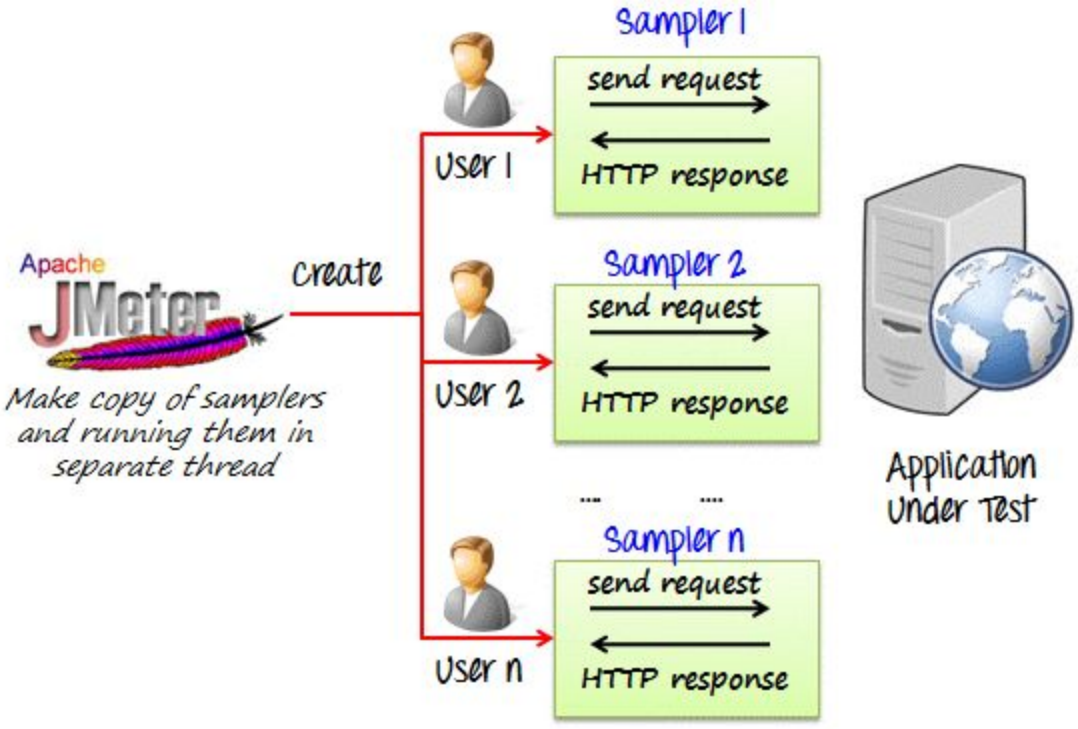


Figure 3. Architecture of Jmeter

The protocol engine is synchronous in nature. In simpler words, this means that for any virtual user, the request are synchronous. Thus after firing the first request, the protocol engine waits for the corresponding response before firing the next request. The response of this request is discarded only after the next request is sent and its response is recorded.

The maximum number of concurrent virtual users which can be simulated (threads) depends on the average memory/CPU specifications for each virtual user These specs,

in turn, depend on the complexity of the script being run and the application's response time.

Open-EdX

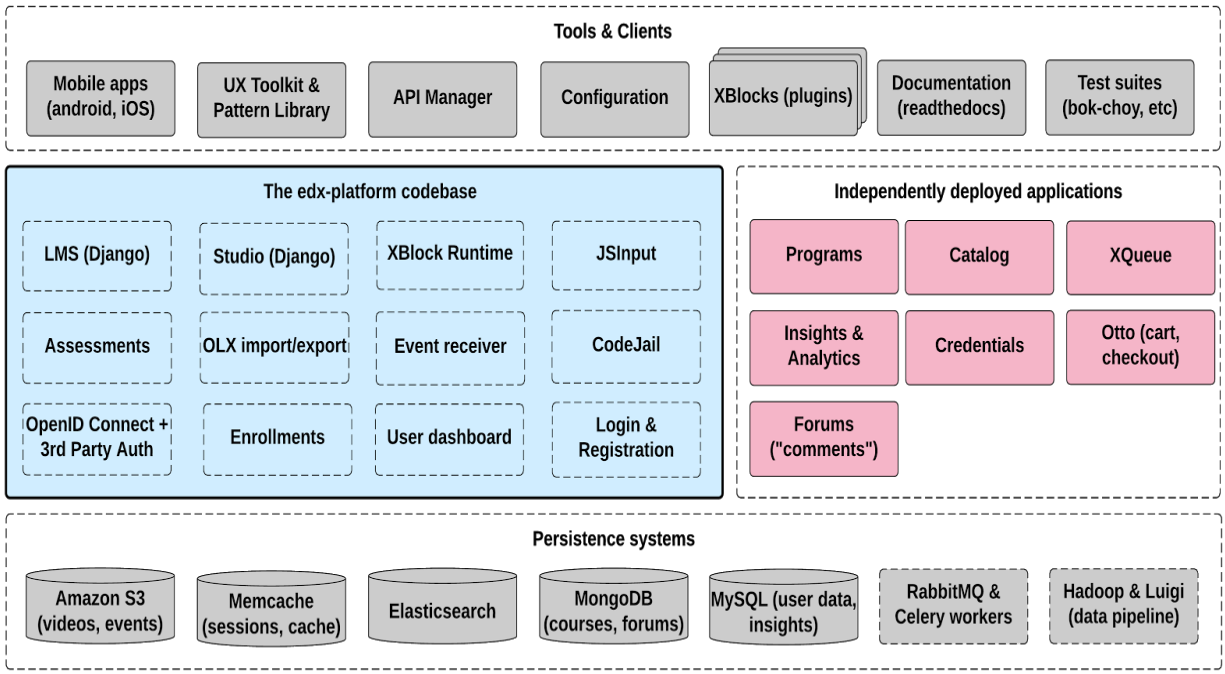


Figure 4. Architecture of OpenEdx

There are a handful of major components in the Open edX project which communicate using stable, documented APIs.The main component of the Open edX architecture is edx-platform, which contains the learning management and course authoring applications (LMS and Studio, respectively).This service is supported by a collection of other autonomous web services called independently deployed applications (IDAs). Individual IDA's are deployed to add functionality to the EdX Platform.

Key Components

- **Learning Management System (LMS)**

  The LMS is the visible front end of the Open edX project. Learners take courses using the LMS. The LMS also provides an instructor dashboard that users who have the Admin or Staff role can access by selecting the Instructor option.

  The LMS uses a number of data stores. Courses are stored in MongoDB. Per-learner data is stored in MySQL. As learners move through courses and interact with them, events are published to the analytics pipeline for collection, analysis, and reporting.

- **Studio**

  Studio is the course authoring environment. Course teams use it to create and update courses. Studio writes its courses to the same Mongo database that the LMS uses.

5. Design and Implementation

1. MariaDB

a. Entity-Relationship Diagram

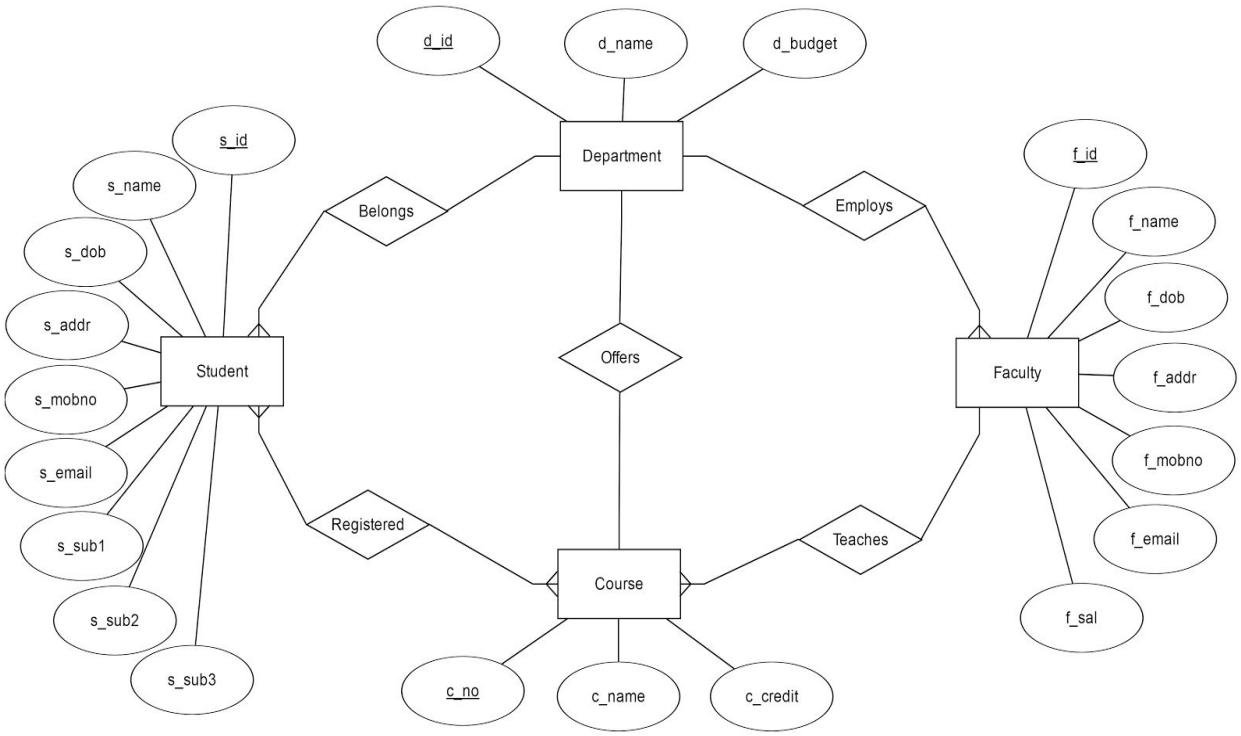


Figure 5. ER Diagram

b. Class Diagram

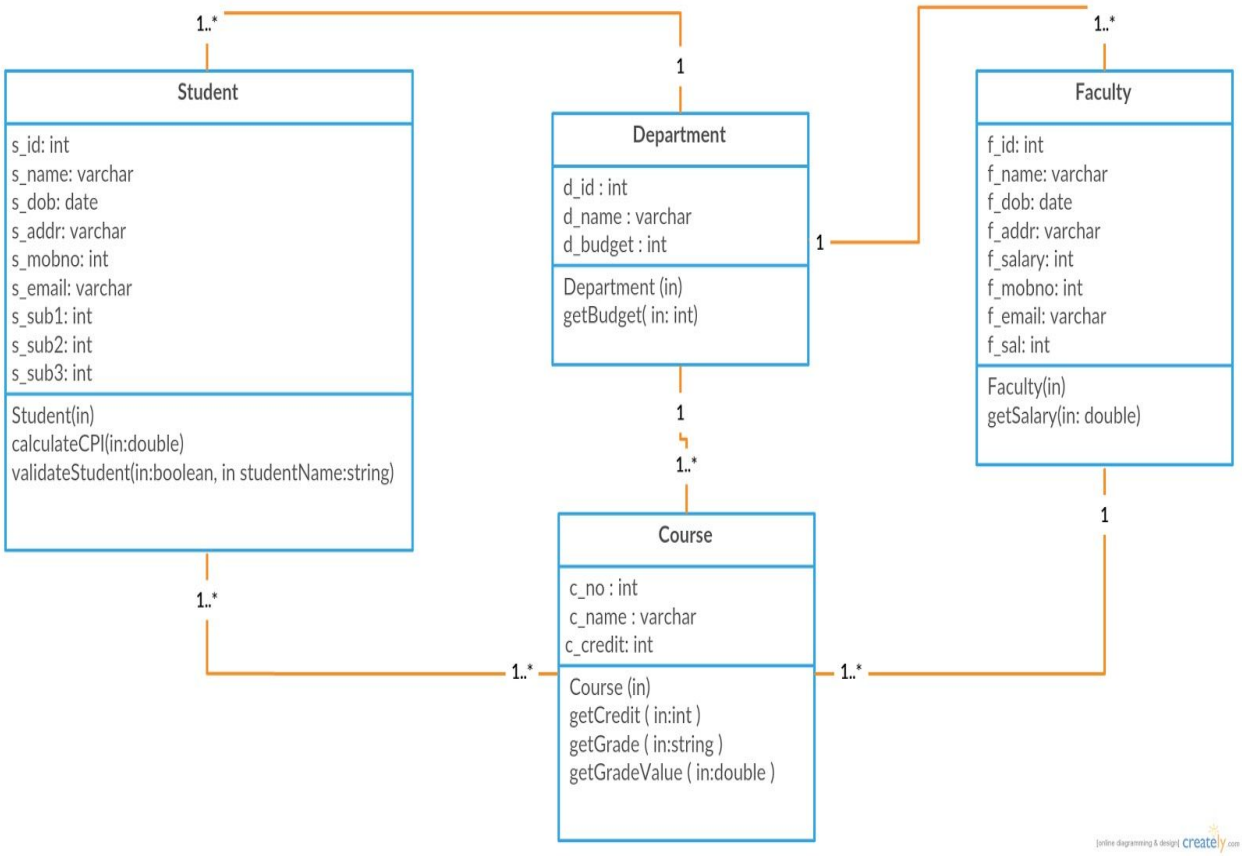


Figure 6. Class Diagram

c. Use Case Diagram

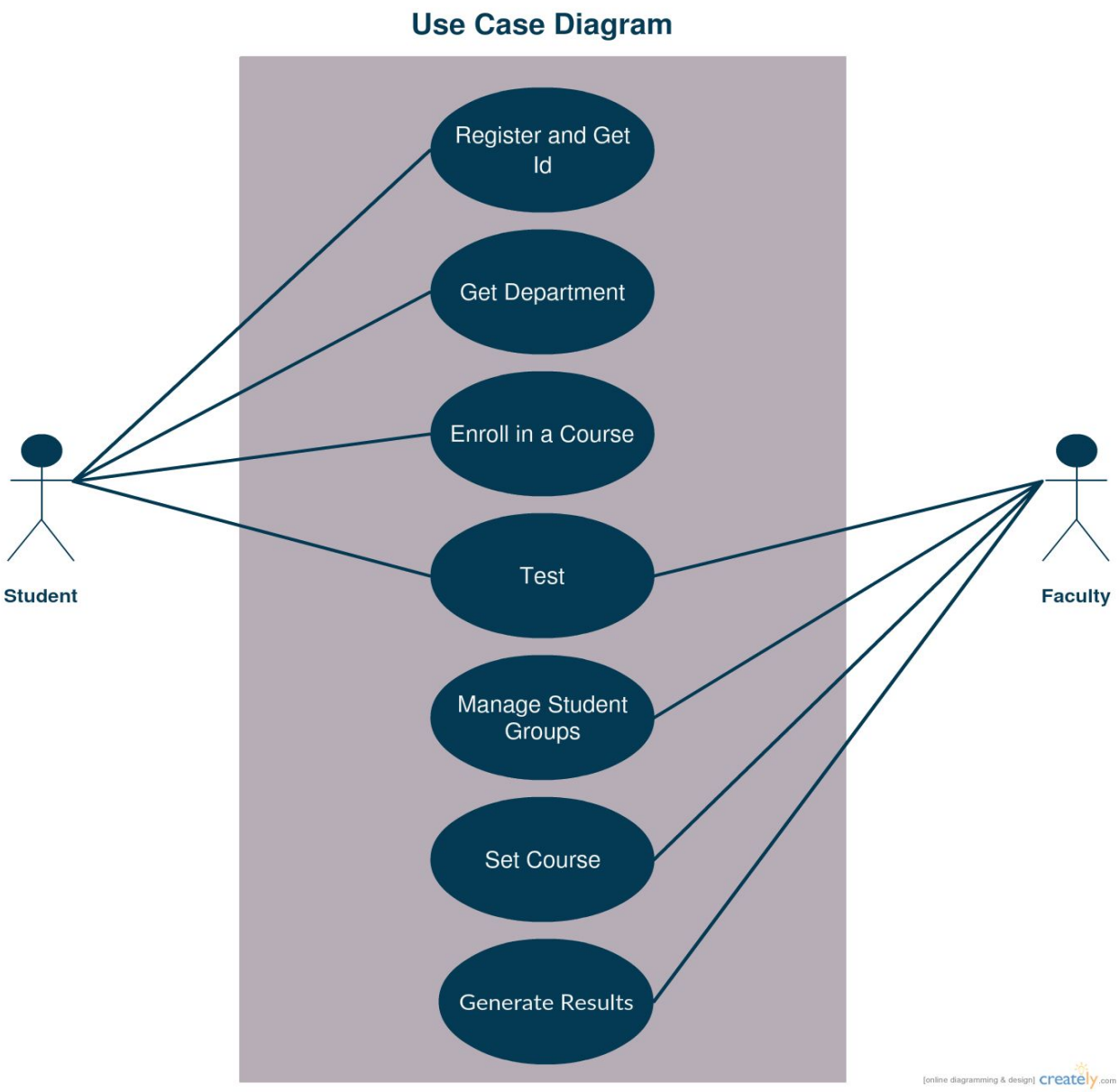


Figure 7. Use Case Diagram

2. JMeter Class Diagram

a. Assertion

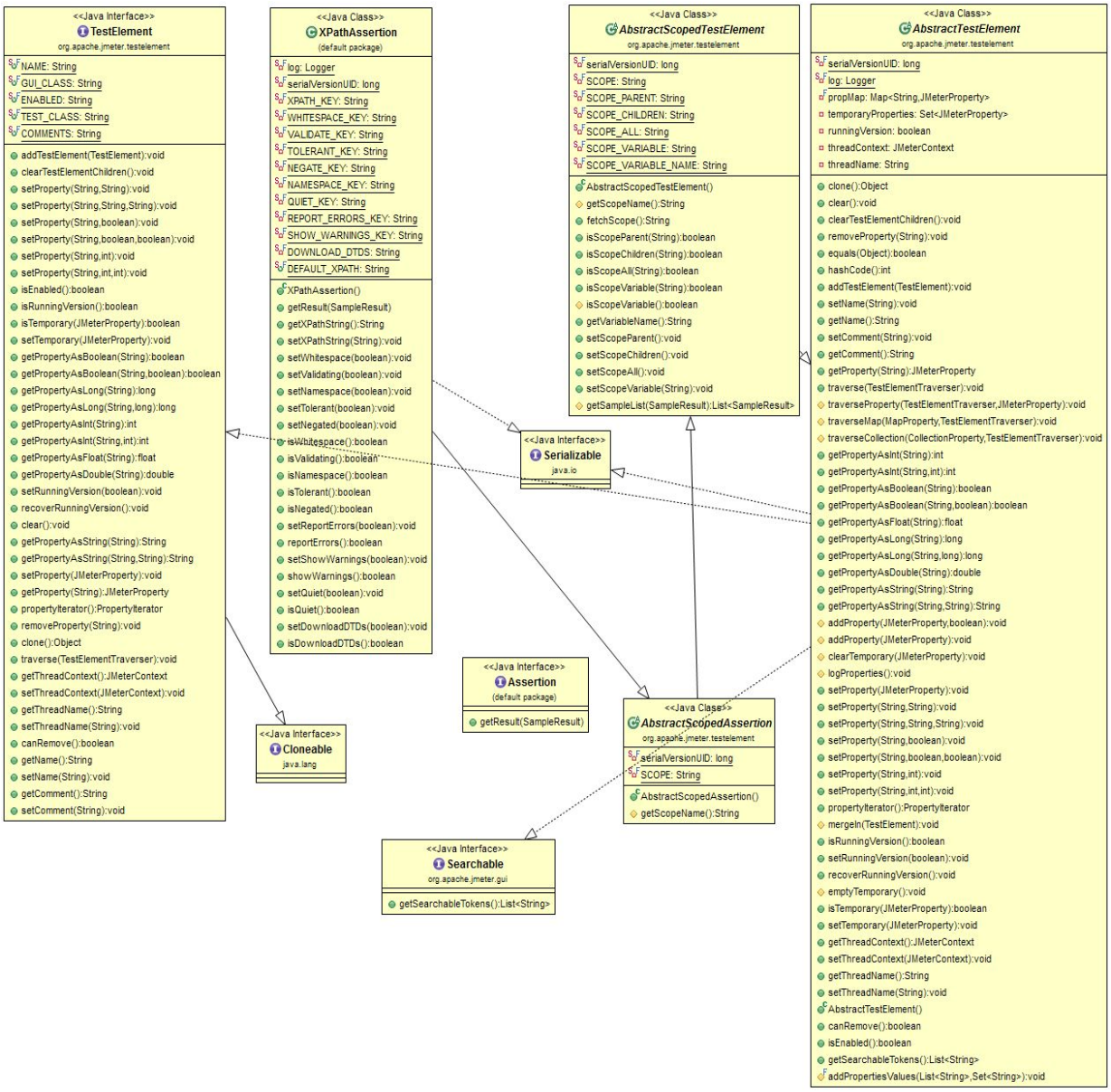
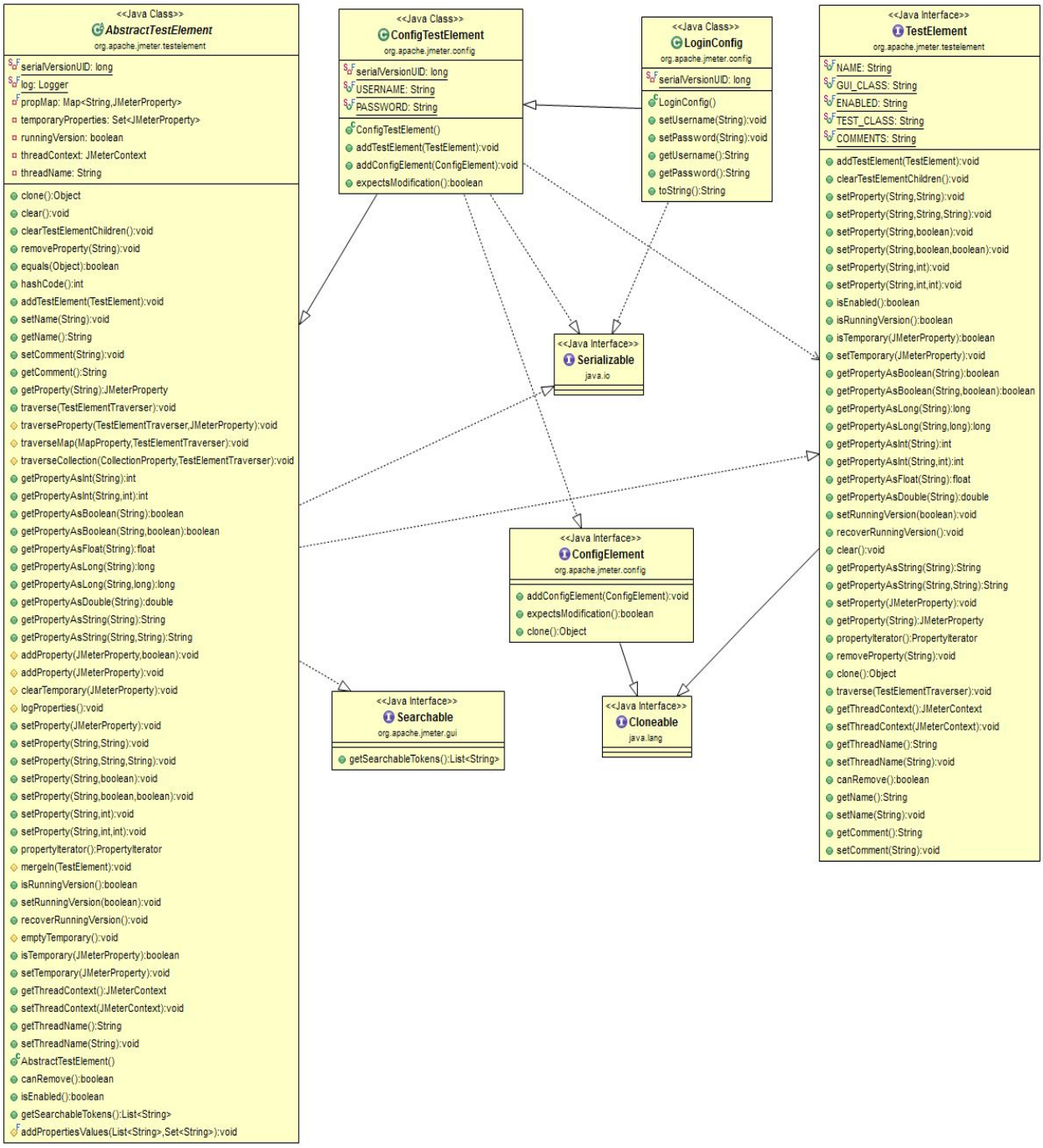


Figure 8. Assertion


<<Java Class>>
AbstractTestElement
org.apache.jmeter.testelement
serialVersionUID: long
log: Logger
propMap: Map<String,JMeterProperty>
temporaryProperties: Set<JMeterProperty>
runningVersion: boolean
threadContext: JMeterContext
threadName: String
clone():Object
clear():void
clearTestElementChildren():void
removeProperty(String):void
equals(Object):boolean
hashCode():int
addTestElement(TestElement):void
setName(String):void
getName():String
setComment(String):void
getComment():String
getProperty(String):JMeterProperty
traverse(TestElementTraverser):void
traverseProperty(TestElementTraverser,JMeterProperty):void
traverseMap(MapProperty,TestElementTraverser):void
traverseCollection(CollectionProperty,TestElementTraverser):void
getPropertyAsInt(String):int
getPropertyAsInt(String,int):int
getPropertyAsBoolean(String):boolean
getPropertyAsBoolean(String,boolean):boolean
getPropertyAsFloat(String):float
getPropertyAsLong(String):long
getPropertyAsLong(String,long):long
getPropertyAsDouble(String):double
getPropertyAsString(String):String
getPropertyAsString(String,String):String
addProperty(JMeterProperty,boolean):void
addProperty(JMeterProperty):void
clearTemporary(JMeterProperty):void
logProperties():void
setProperty(JMeterProperty):void
setProperty(String,String):void
setProperty(String,String,String):void
setProperty(String,boolean):void
setProperty(String,boolean,boolean):void
setProperty(String,int):void
setProperty(String,int,int):void
propertyIterator():PropertyIterator
mergeIn(TestElement):void
isRunningVersion():boolean
setRunningVersion(boolean):void
recoverRunningVersion():void
emptyTemporary():void
isTemporary(JMeterProperty):boolean
setTemporary(JMeterProperty):void
getThreadContext():JMeterContext
setThreadContext(JMeterContext):void
getThreadName():String
setThreadName(String):void
AbstractTestElement()
canRemove():boolean
isEnabled():boolean
getSearchableTokens():List<String>
addPropertiesValues(List<String>,Set<String>):void
<<Java Class>>
ConfigTestElement
org.apache.jmeter.config
serialVersionUID: long
USERNAME: String
PASSWORD: String
ConfigTestElement()
addTestElement(TestElement):void
addConfigElement(ConfigElement):void
expectsModification():boolean
<<Java Class>>
LoginConfig
org.apache.jmeter.config
serialVersionUID: long
LoginConfig()
setUsername(String):void
setPassword(String):void
getUsername():String
getPassword():String
toString():String
<<Java Interface>>
TestElement
org.apache.jmeter.testelement
NAME: String
GUI_CLASS: String
ENABLED: String
TEST_CLASS: String
COMMENTS: String
addTestElement(TestElement):void
clearTestElementChildren():void
setProperty(String,String):void
setProperty(String,String,String):void
setProperty(String,boolean):void
setProperty(String,boolean,boolean):void
setProperty(String,int):void
setProperty(String,int,int):void
isEnabled():boolean
isRunningVersion():boolean
isTemporary(JMeterProperty):boolean
setTemporary(JMeterProperty):void
getPropertyAsBoolean(String):boolean
getPropertyAsBoolean(String,boolean):boolean
getPropertyAsLong(String):long
getPropertyAsLong(String,long):long
getPropertyAsInt(String):int
getPropertyAsInt(String,int):int
getPropertyAsFloat(String):float
getPropertyAsDouble(String):double
setRunningVersion(boolean):void
recoverRunningVersion():void
clear():void
getPropertyAsString(String):String
getPropertyAsString(String,String):String
setProperty(JMeterProperty):void
getProperty(String):JMeterProperty
propertyIterator():PropertyIterator
removeProperty(String):void
clone():Object
traverse(TestElementTraverser):void
getThreadContext():JMeterContext
setThreadContext(JMeterContext):void
getThreadName():String
setThreadName(String):void
canRemove():boolean
getName():String
setName(String):void
getComment():String
setComment(String):void
<<Java Interface>>
Serializable
java.io
<<Java Interface>>
ConfigElement
org.apache.jmeter.config
addConfigElement(ConfigElement):void
expectsModification():boolean
clone():Object
<<Java Interface>>
Searchable
org.apache.jmeter.gui
getSearchableTokens():List<String>
<<Java Interface>>
Cloneable
java.lang

b. Configuration Elements

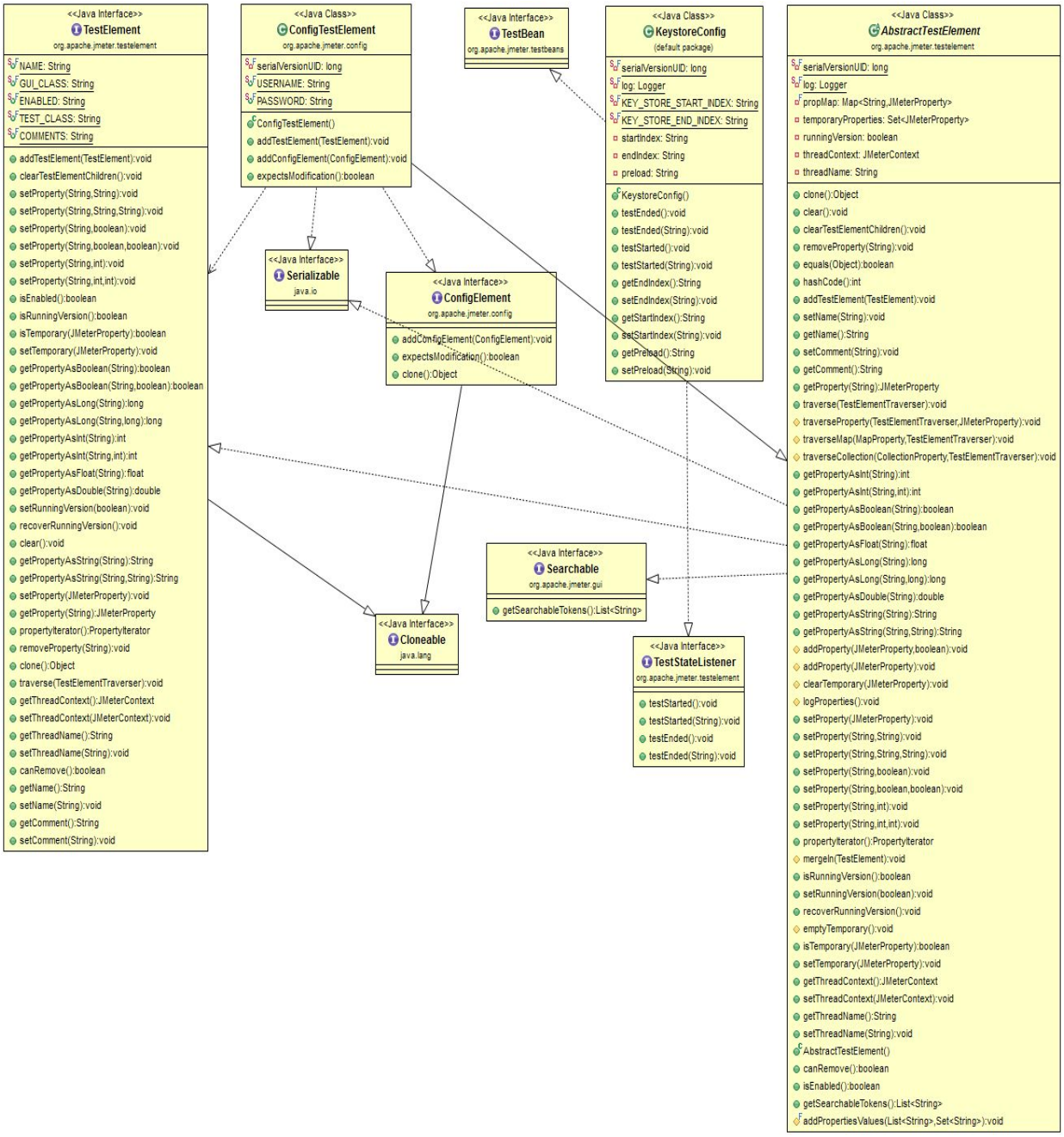
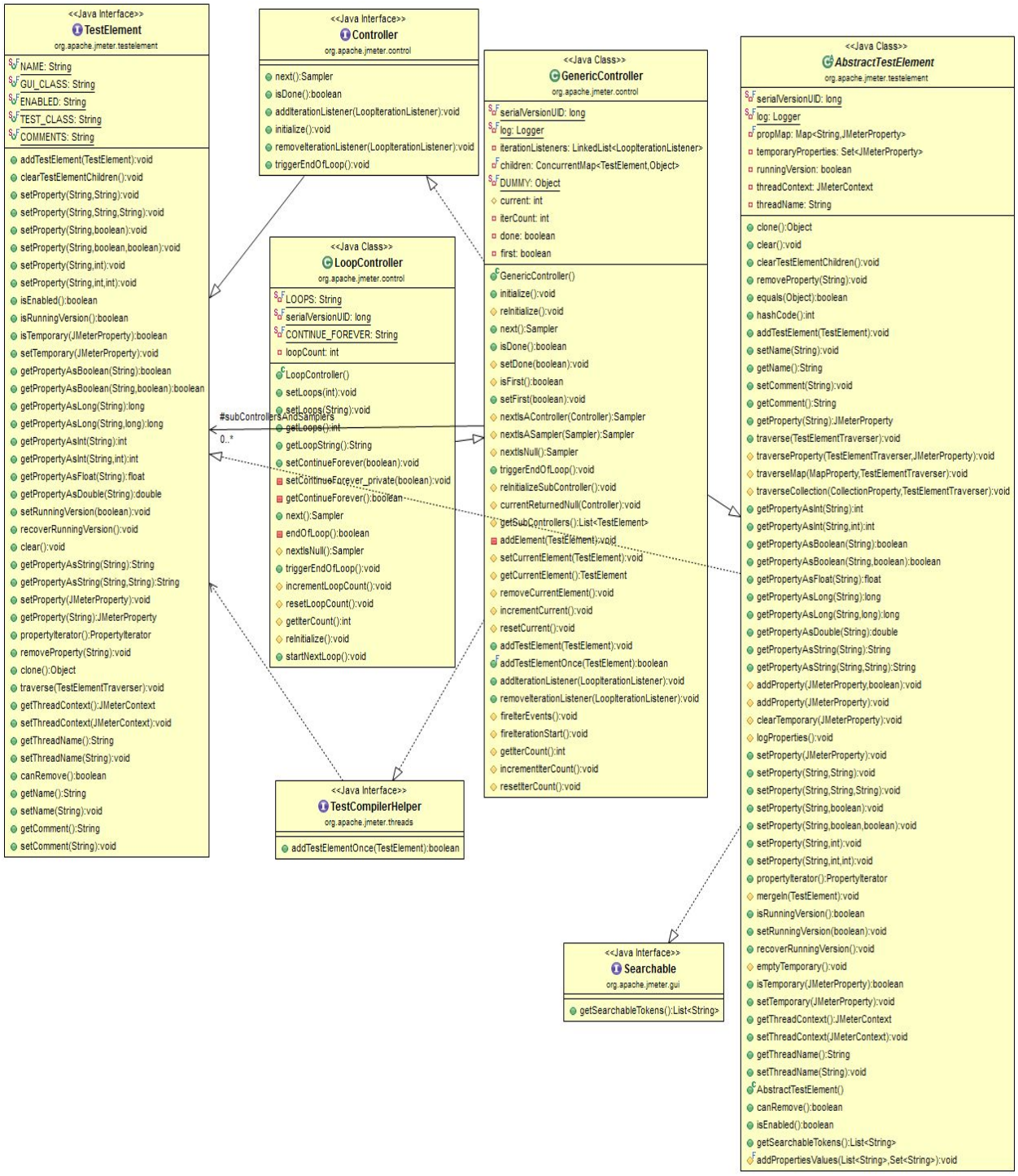


Figure 9. Configuration Elements

c. Controller Element



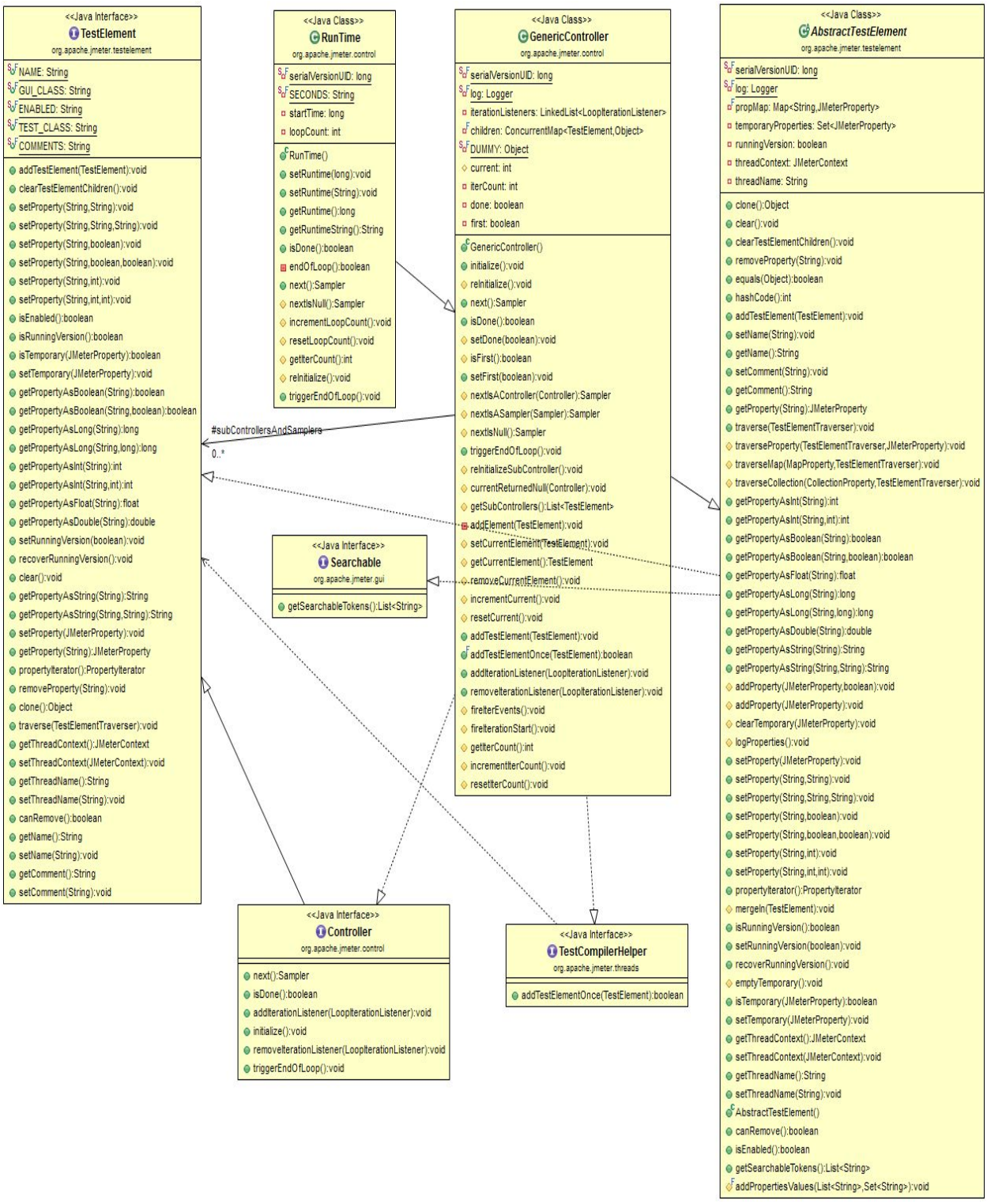


Figure 10. Controller Element

6. Detailed Description

MySQL

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company.

Components of a SQL:

- **MySQL Server:** It is the database server. It processes all queries and manipulates databases and tables.
- **MySQL Clients:** These are programs that communicate with the server.
- **Data Directory:** This is where MySQL stores the databases.
- **Storage Engine:** Manages how the information is organised, stored and accessed. MySQL uses a pluggable storage engine architecture.

MySQL Server

The MySQL Server manages the changes in the data directory. The database and tables are present in the datadirectory. It also manages the connections from clients and their access to the database;.

These main components of the MySQL server are:

- **mysqld:** the database server
- **mysqld_safe:**: a script to start mysqld that adds some safety features. It is especially useful when server starts with errors.
- **mysql.server:** a Unix script that starts mysqld_safe
- **mysqld_multi:** Manages multiple mysqld servers in the same machine

MySQL Clients

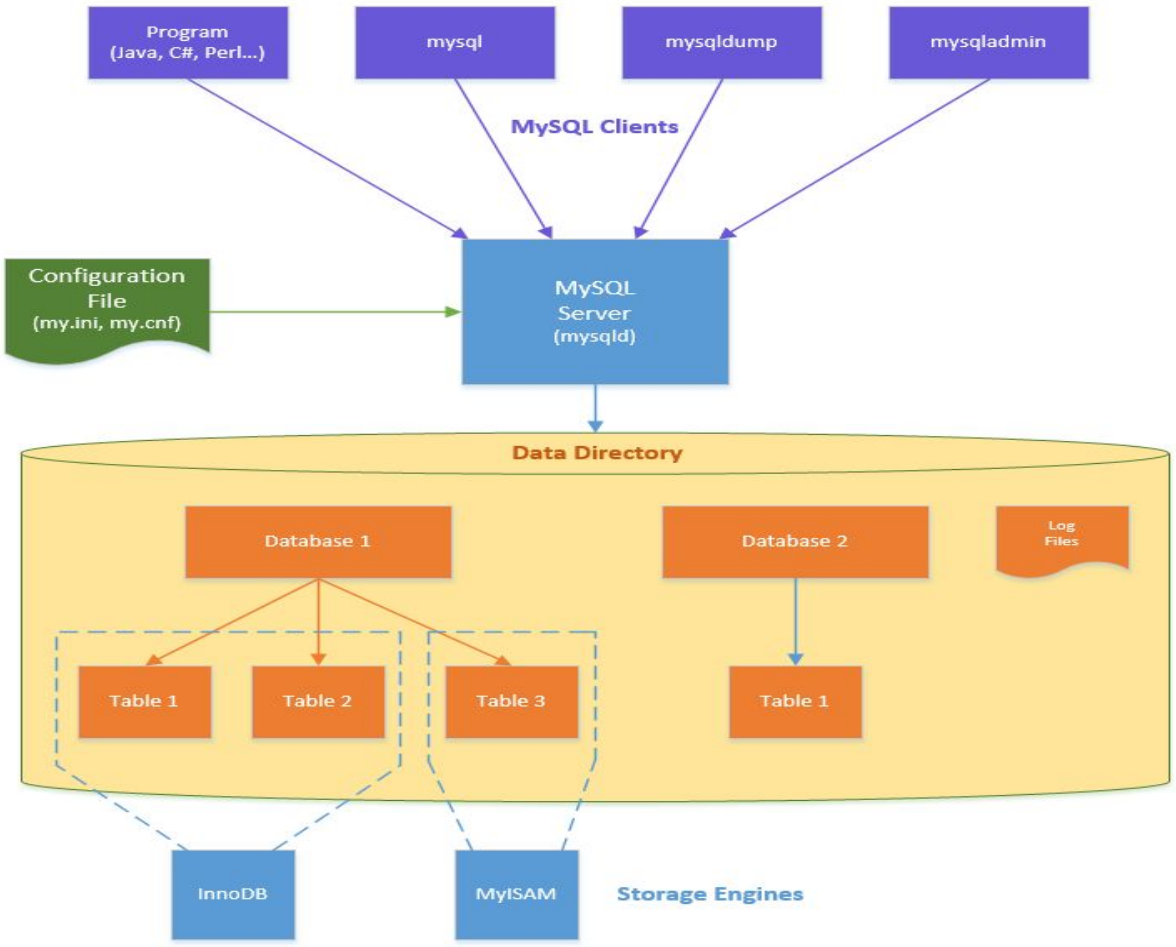


Figure 11. MySQL Clients

Clients connect to the MySQL Server to access the Server's database and table. The main components of a Client are:

- **mysql:** The program where SQL statements are written and executed in the server..
- **mysqladmin:** It is an administrative tool which configures, monitors or shuts down the server.
- **mysqldump:** It is a tool used to backup the database;

- **mysqlcheck and myisamchk:** Tools to check the integrity of tables and help repair them.

Data Directory

The data directory contains the information managed by the SQL server, including tables, triggers, stored procedures, database, status files and log files.

- The default location of the data directory in Ubuntu is :***~/etc/mysql***/
- At server startup, you can change the data directory with the *--datadir* option

  You can also change the data directory in the configuration file (*my.cnf)*
  [mysqld]
  datadir="*Path/to/data/directory*"

- To show the location of the current data directory, you can run this SQL command:
  mysql> *SHOW VARIABLES LIKE 'datadir';*

Storage Engines

A storage engine manages how the table information is organised, stored and accessed. MySql uses a pluggable storage engine architecture which consists of:

- MyISAM
- BLACKHOLE
- CSV
- MEMORY
- ARCHIVE
- MERGE
- InnoDB

MariaDB

MariaDB Server is one of the most popular database servers in the world. It was made by the original developers of MySQL and is open source. It's users include Wikipedia, WordPress.com and Google.

MariaDB turns data into structured information for a wide array of applications, ranging from banking to websites. It is an enhanced, drop-in replacement for MySQL. MariaDB is used because it is fast, scalable and robust, with a rich ecosystem of storage engines, plugins and many other tools make it very versatile for a wide variety of use cases.

MariaDB is developed as open source software and as a relational database it provides an SQL interface for accessing data. The latest versions of MariaDB also include GIS and JSON features.

Reasons to migrate to MariaDB:

- MariaDB development is more open and vibrant i.e all the development decisions can be reviewed and debated on a public mailing list for all the bugs present in the the public bug tracking list. Also, contributing to MariaDB with patches is easy and patch flow is transparent in the fully public and up-to-date code repository.
- Quicker and more transparent security releases.
- More number of cutting edge features.

  For example GIS features were introduced already in the 5.3 series of MariaDB, which makes storing coordinates and querying location data easy.

  Dynamic column support (MariaDB only) is interesting because it allows for NoSQL type functionality, and thus one single database interface can provide both SQL and "not only SQL" for diverse software project needs.

- More storage engines and better performance.

  By default,the storage engine of MySQL/MariaDB is InnoDB.

Storage Engines which are also included in the MariaDB Source and binary packages (other than those present in MySQL) are:

- ColumnStore
- RocksDB
- Aria
- FederatedX
- OQGRAPH
- SphinxSE
- TokuDB (in MariaDB 5.5 and later)
- CONNECT (in MariaDB 10.0 and later)
- SEQUENCE (in MariaDB 10.0 and later)
- Spider and Cassandra (in MariaDB 10.0 and later)

InnoDB vs RocksDB

Advantages of InnoDB:

1. Data integrity and foreign key constraints: Foreign keys are used to establish a relationship between columns in one table and those in another.

Example: Let us assume that we create a student-application in which library books are loaned to members. In such a database, a foreign key constraint would check that a member exists before a book is checked out by them. Similarly, one can not remove a user until their books are returned.

2. Transactions: InnoDB tables support transactions. A transaction allows multiple SQL commands to be treated as a single and reliable unit.

Example: Let us consider a banking application in which a transaction consists of sending money from one account to another. The transaction would only be committed if both accounts were altered successfully. If anything failed, the database would be rolled-back to a previous state.

3. Row-level locking: InnoDB uses row-level locking, rather than using a table-level one. On inserting, updating or deleting a row, only changes to the required row take place.

Disadvantages of InnoDB:

1. **Fragmentation of B +Tree index:**

   B+Tree indexes get fragmented if not inserted sequentially and fragmentation increases 30~50% additional space.

2. **InnoDB compression has a large space overhead:**

   InnoDB compresses per page basis. For example, even though InnoDB compression could compress data from 16KB to 5KB (68.75% reduction), it actually uses 8KB, so compression efficiency deteriorates from 68.75% to 50%.

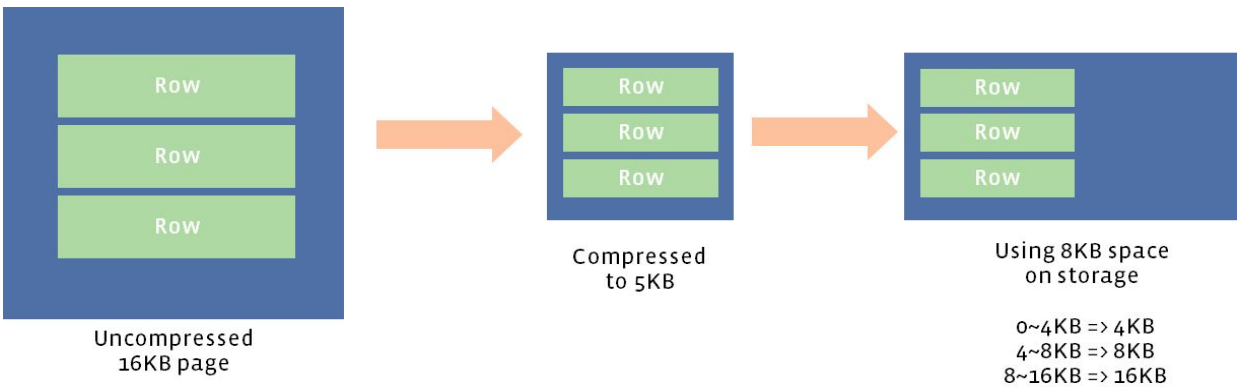


Figure 12. InnoDB Compression

3. **InnoDB adopts "update in place" architecture:**

   Even on updating a single record, an entire page where the row is present, becomes dirty, and the dirty page has to be written back to storage.

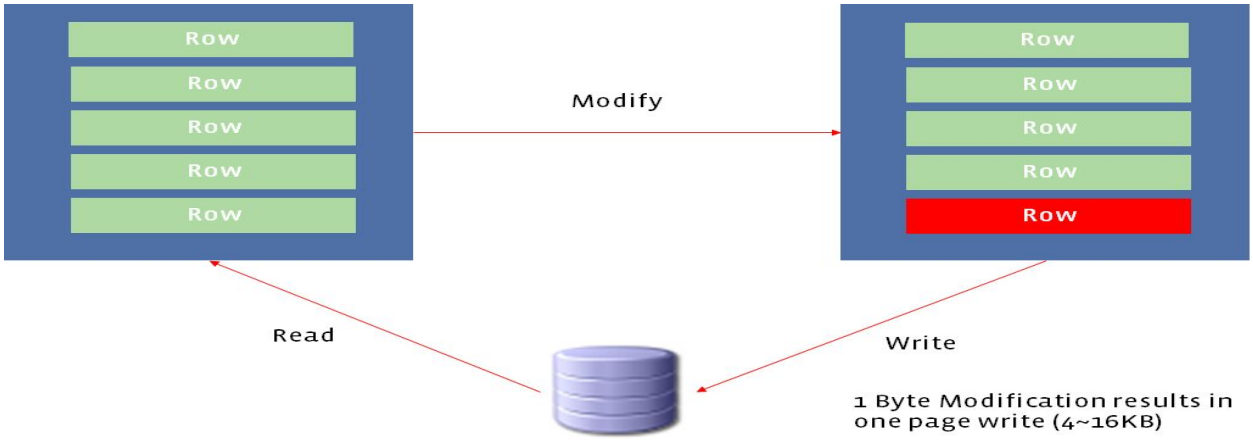


Figure 13. Updation in InnoDB

Where to use InnoDB:

1. InnoDB is the most viable option if one has to create a reliable data-driven web application.

2. InnoDB is certainly the best choice for online shops, financial applications or any project where data integrity is essential. Defining tables is more complex, but the application will be more robust and will most probably need less server-side code.

RocksDB:

1. RocksDB is a C++ library providing an embedded key-value store, where keys and values are arbitrary byte streams. It was developed at Facebook based on LevelDB and provides backwards-compatible support for LevelDB APIs.
2. RocksDB is optimized for Flash with extremely low latencies. RocksDB uses a Log Structured Database Engine for storage,which is written entirely in C++.
3. RocksDB features highly flexible configuration settings that may be tuned to run on a variety of production environments, including pure memory, Flash, hard disks or HDFS. It supports various compression algorithms and good tools for production support and debugging.

Features of RocksDB:

1. Designed for application servers wanting to store up to a few terabytes of data on locally attached Flash drives or in RAM.
2. Optimized for storing small to medium size key-values on fast storage -- flash devices or in-memory.
3. Scales linearly with number of CPUs so that it works well on processors with many cores.

Advantages of RocksDB:

1. Compression Efficiency: One of the biggest reasons RocksDB compresses better than InnoDB is that it does not use fixed page sizes. When a page is compressed to 5 KB, it will use only 5 KB of storage, compared with 8 KB with InnoDB.

2. Append only and compaction: RocksDB uses a smaller number of sequential reads and writes because access patterns for writes are maintained in a persistent key-value store based on LSM. RocksDB database uses "Append Only" model, thus edits are written to WAL, then periodically merged with SST

files. Less storage I/O capacity per update is needed since small number of random reads and writes are used, unlike indexing structures.

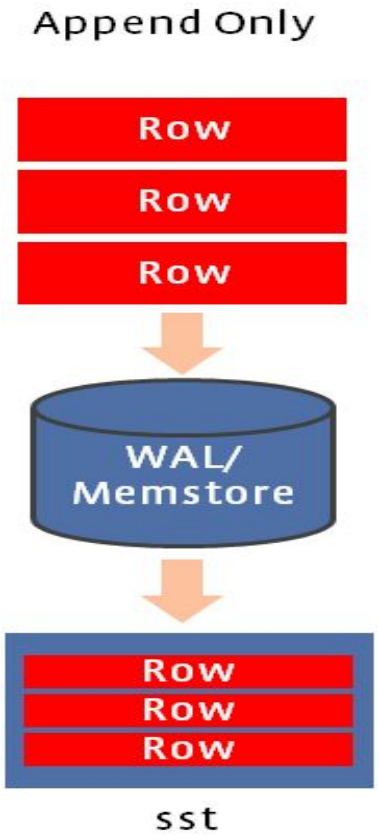


Figure 14. Compaction in RocksDB

3. Prefix key encoding : RocksDB trims identical prefix bytes when possible, which can save a lot of space. When using multi-column indexes, very often first few bytes are the identical to previous index entry. Then RocksDB optimizes space by not storing duplicate prefix byte stream.

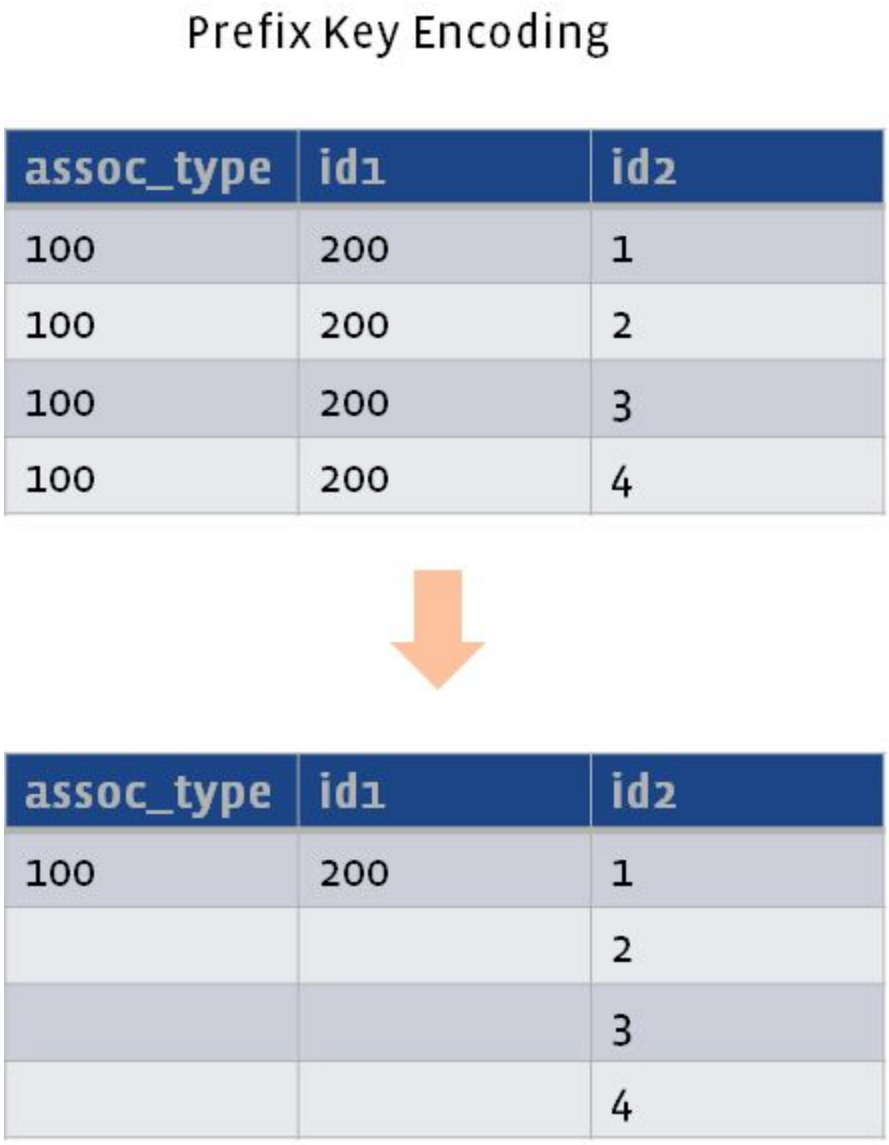


Figure 15. Prefix Key Encoding

4. Zero- filling Metadata: RocksDB has metadata per key/value, whereas InnoDB has metadata per record. InnoDB metadata is 13 bytes per record with 6 bytes for the transaction identifier and 7 bytes for the rollback pointer, and is not compressed. RocksDB metadata consists of a 7 byte sequence identifier and 1 byte operation type identifier, and is compressed. RocksDB also replaces the

sequence identifier with the value zero when it is not needed to resolve multi-version concurrency control (MVCC) visibility.

Zero-Filling metadata

seq id	flag	key	value
1234561	W	k1	v1
1234562	W	k2	v2
1234563	W	k3	v3
1234564	W	k4	v4

seq id	flag	key	value
0	W	k1	v1
0	W	k2	v2
0	W	k3	V3
0	W	k4	V4

Figure 16. Zero-Filling Metadata

My-Rocks limitations:

1. Online DDL is not supported,but fast alteration and dropping of indexes are supported in RocksDB.
2. Exchange Partition is not supported in MyRocks

3. Lack of savepoint support.
4. **Transportable Tablespace, Foreign Key, Spatial Index, and Fulltext Index** are not supported.
5. Statement based binary logging may cause data inconsistency between master and slave because My-Rocks does not support Next-Key Locking.
6. "*_bin" (e.g. latin1_bin) or binary collation should be used on CHAR/VARCHAR indexed columns. By default, My-Rocks prevents creating indexes with non-binary collations (including latin1).
7. Either ORDER BY DESC or ASC is slow. This is because of "Prefix Key Encoding" feature in RocksDB.

JMeter

Apache JMeter consists of large number of components which are used in a variety of test plans. The components have been distributed according to the tasks that they are used for, like samplers are used to specify the tasks that will be done by the threads and timers are used to specify the delay (thinking time) between execution of the samplers. In a similar vein, there are other components like listeners, config elements, pre processors, post-processors, assertions,etc which have several elements under them.

We shall be defining the components that have been used by us:

Thread Groups:

Thread group elements are the beginning points of any test plan. All controllers and samplers are placed under a thread group. Other elements, e.g. Listeners, may be placed directly under the test plan, in which case they will apply to all the thread groups. As the name implies, the thread group element controls the number of threads JMeter will use to execute your test. The controls for a thread group allow you to:

- Set the number of threads
- Set the ramp-up period
- Set the number of times to execute the test

Samplers:

1. **HTTP Request:** This sampler is used to send an HTTP/HTTPS request to a web server. It is also used to control whether or not JMeter parses HTML files for images and other embedded resources and sends HTTP requests to retrieve them.
2. **JDBC Request**: This sampler is used to send JDBC request to a MySQL server.

Logic Controllers:

1. **Simple Controller:** The Simple Logic Controller is used to organize Samplers and other Logic Controllers. The only functionality of his this controller is to provide a storage/ log for the functions being carried out.
2. **Recording Controller:** The Recording Controller acts as a placeholder to indicate where the proxy server should record samples to. While the test is run,it works similar to the Simple Controller. When we record a script using the HTTP(S) Test Script Recorder, all recorded samples will by default be saved under the Recording Controller.

Listeners:

1. **Graph Results:** This listener is used to generate a simple graph which plots sample times.The results are plotted according to a pre defined color schema (present at the bottom of the graph)
    a. The current sample is displayed in black.
    b. The current average of all samples is displayed in blue

c. The current standard deviation is displayed in red
d. The current throughput rate is displayed in green

(All results are displayed w.r.t milliseconds.)

The throughput numbers represent a nearly real-time estimation of the requests per minute that the server handles. This is achieved by adding delays (think times) to the test. It also takes into account JMeter's internal processing time. This can be used to discover the maximum throughput of the server.

2. **View Results Tree:** The View Results Tree displays a tree report of all the test elements and view the result of any sample. It also displays the time needed to reach to this response and some associated response codes.
3. **View Results in Table:** It is similar to the View Results Tree element, with the only difference being that it displays the results in the form of a table.
4. **Aggregate Report:** The aggregate report creates a table row for each different request in the test. For each request, it totals the response information and provides request count, min, max, average, error rate, approximate throughput (request/second) and Kilobytes per second throughput. Once the test is done, the throughput is the actual through for the duration of the entire test.
5. **Aggregate Graph:** The aggregate graph is similar to the aggregate report. The primary difference is that the aggregate graph provides an easy way to generate bar graphs and save the graph as a PNG file.
6. **Simple Data Writer:** The Simple Data Writer listener is used to record the results to a file and not the UI. It provide a simple and effective means of

recording data by eliminating the GUI overhead. When running in non-GUI mode, the -l flag can be used to create a data file.

7. **Summary Report:** The Summary Report is similar to the Aggregate report but takes up less memory space.It creates a table row for each differently named request in your test.

Timers:

1. **Constant Timer:** This timer is used to pause each thread for the same amount of time between requests.
2. **Uniform Random Timer:** This timer pauses each thread request for a random amount of time (X), with each time interval having the same probability of occurring. The total delay is the sum of the random value and the offset value.

Configuration Elements:

1. **CSV DataSet Config:** The CSV DataSet Config is used to read lines from a file, and split them into variables.It may be used for a large amount of data and also in tandem with random variables.
2. **HTTP Cache Manager:** The HTTP Cache Manager adds a caching functionality to HTTP requests within its scope to simulate browser cache feature. Every thread has its own Cache. And by default, the Cache Manager can store a maximum of 5000 items in the cache per thread, using Least Recently Used algorithm.
3. **HTTP Cookie Manager:** The Cookie Manager formulates two tasks:
    a. It records and sends cookies similar to a web browser. If the HTTP Request that we send and the response contain cookies, the the

Manager stores the cooke automatically for future use for the particular web site. Each JMeter thread has its own "cookie storage area".

   b. One can manually add a cookie to the cookie manager. In this case, the cookie is shared by all the threads.

4. **HTTP Request Defaults:** This element is used to set the default values for the HTTP Request controllers. The Server Name/IP and/or Port Number can be added in this for requesting purposes.
5. **JDBC Connection Configuration:** This Config element creates a database connection that is used by the JDBC Request Sampler from the given JDBC Connection settings.
6. **User Defined Variables:** The User Defined Variables element can define an initial set of variables, one that remains constant throughout the execution of the test plan.

Post Processor Elements:

1. **Regular Expression Extractor:** This is used to extract values from a server response using a regular expression. As a post-processor, this executes after each Sample request in its scope, applying the regular expression, extracting the requested values, generate template string and store the result into the given variable name.

Order of Execution of Components:

1. **Configuration elements**
2. **Preprocessors**
3. **Timers**
4. **Sampler**
5. **Post-Processors (unless SampleResult is null)**
6. **Assertions (unless SampleResult is null)**
7. **Listeners (unless SampleResult is null)**

7. Test Demonstrations

1. Checking for the RocksDB plugin:

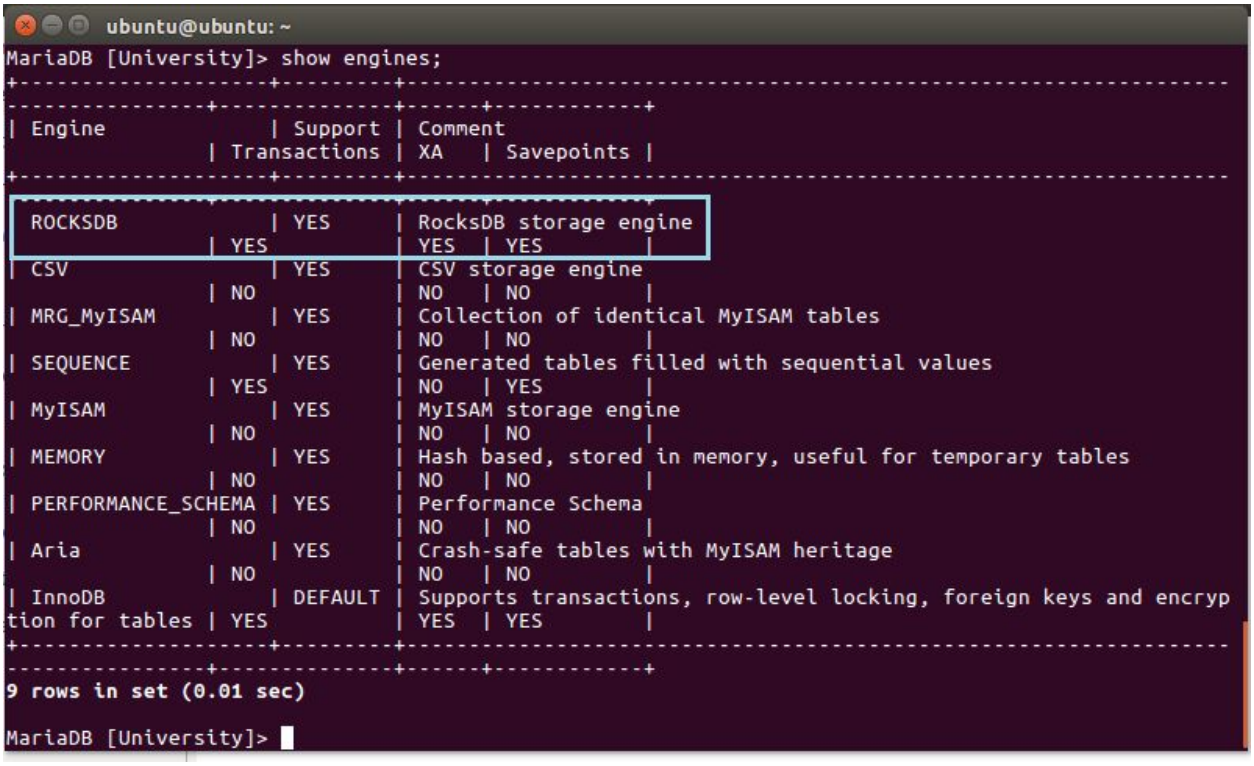


Figure 17. Checking for RocksDB Plugin

To check whether the RocksDB plugin has been installed successfully, enter into the MariaDB command line and type the following

MariaDB [(none)] > *show engines;*

2. Running an Insert Query Test Plan on JMeter

a. Add a Thread Group to the Test Plan

Right Click on Test Plan->Add->Threads(Users)->Thread Group

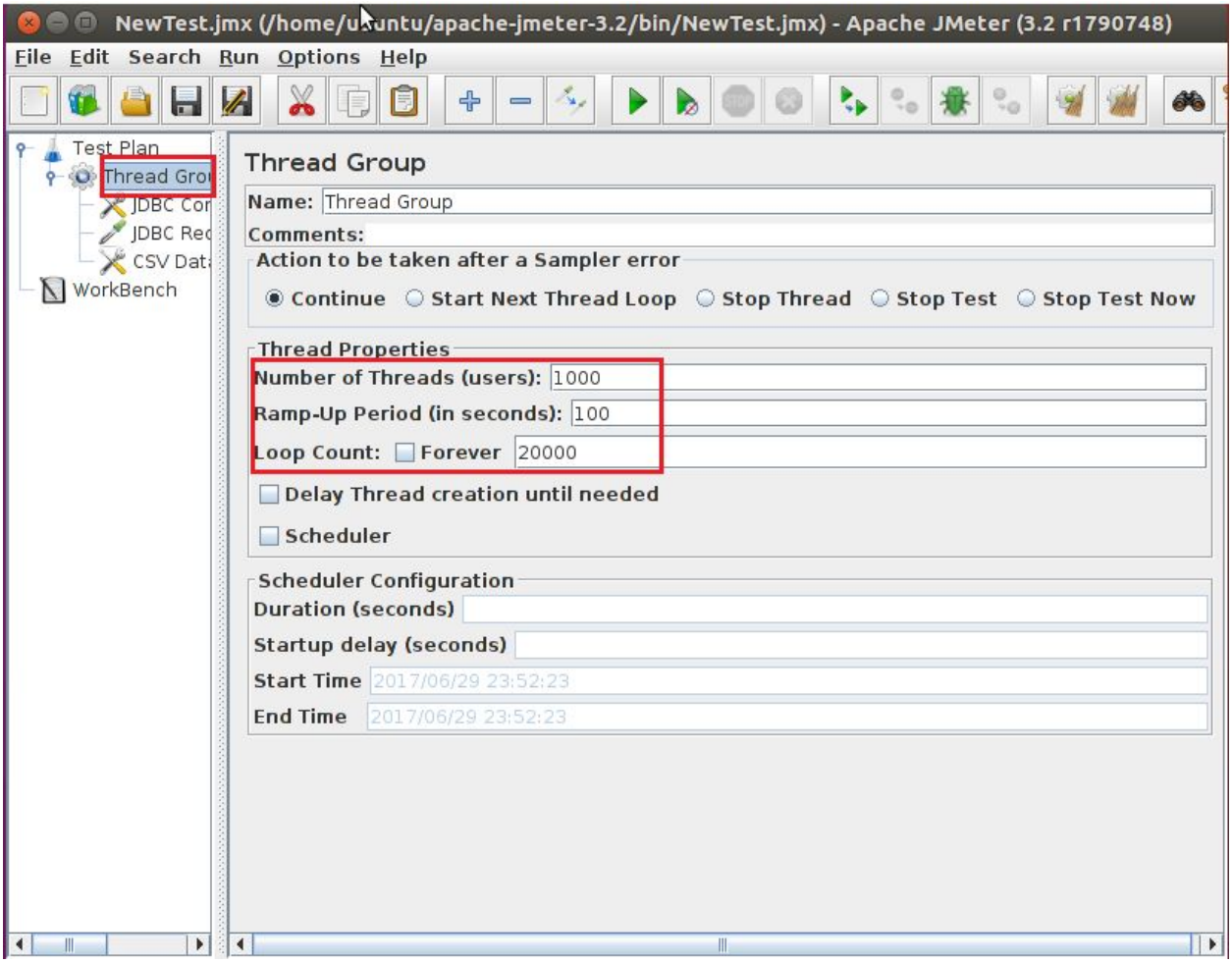


Figure 18. Thread Group

b. Setting Up the JDBC Connection Configuration

Right Click on Thread Group -> Add -> Config Element -> JDBC Connection Configuration

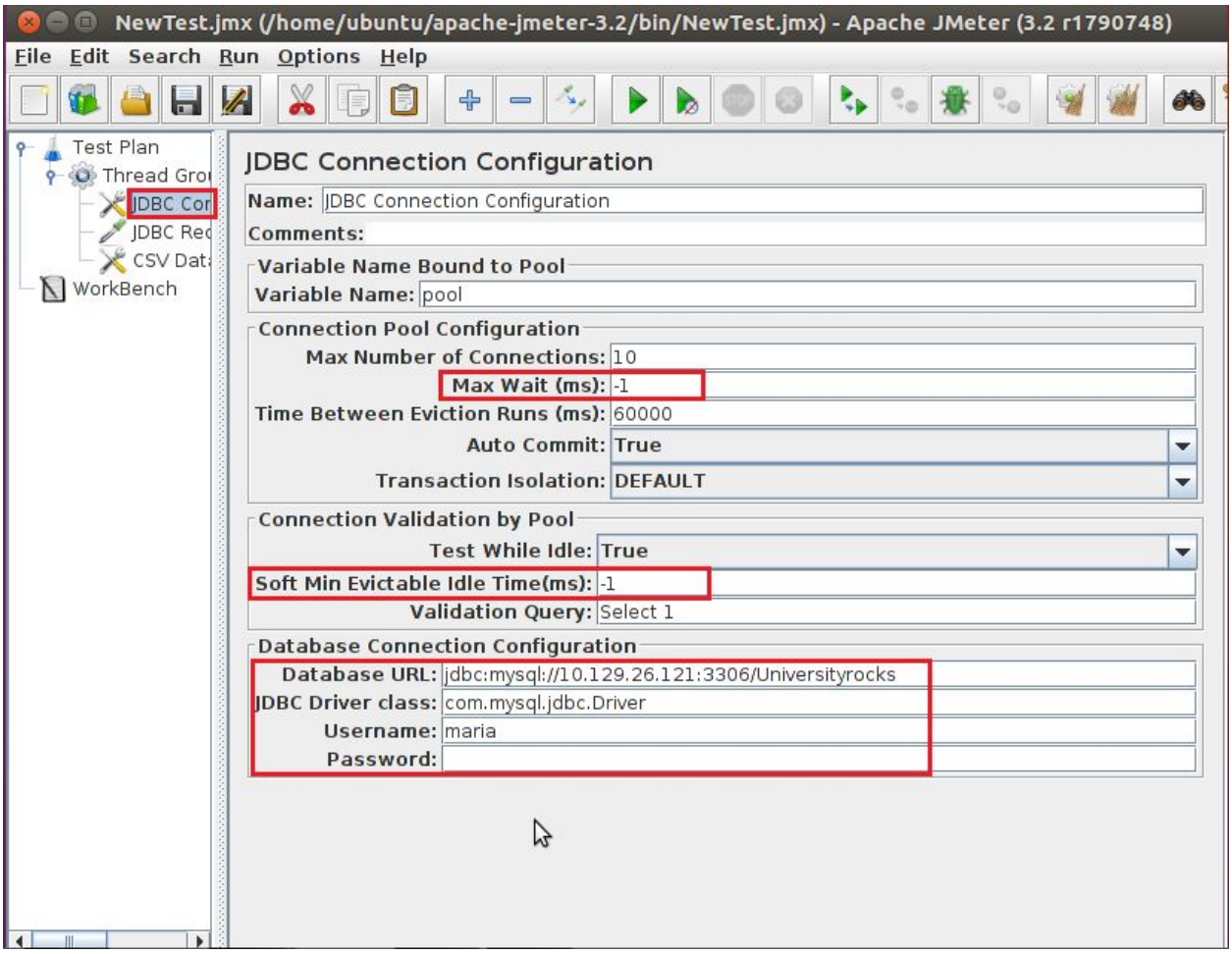


Figure 19. JDBC Connection Configuration

c. Adding JDBC Requests

Right Click on Thread Group -> Add -> Sampler -> JDBC Request

Note that Query Type should be 'Prepared Update Statement'

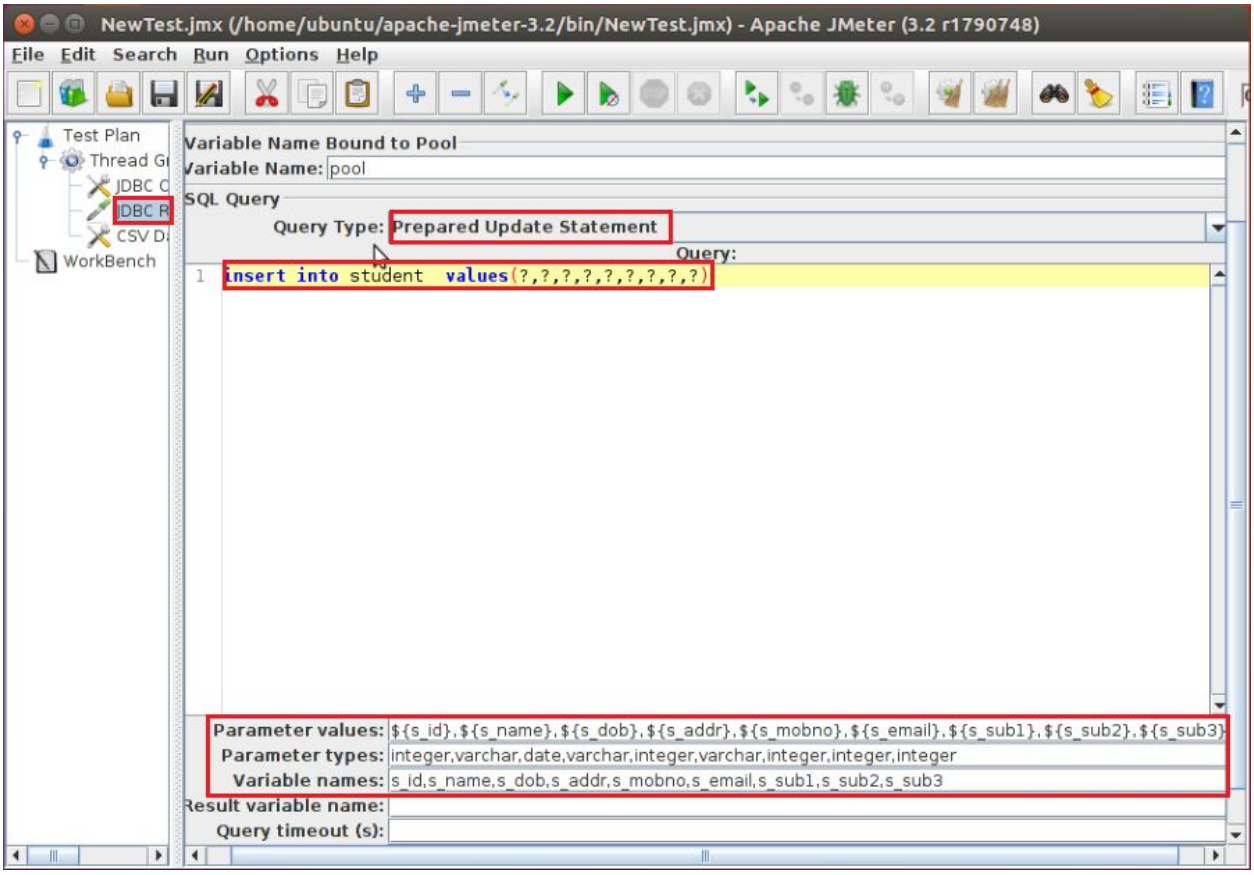


Figure 20. JDBC Requests

d. CSV DataSet Config

Right Click on Thread Group -> Add -> Config Element -> CSV Data Set

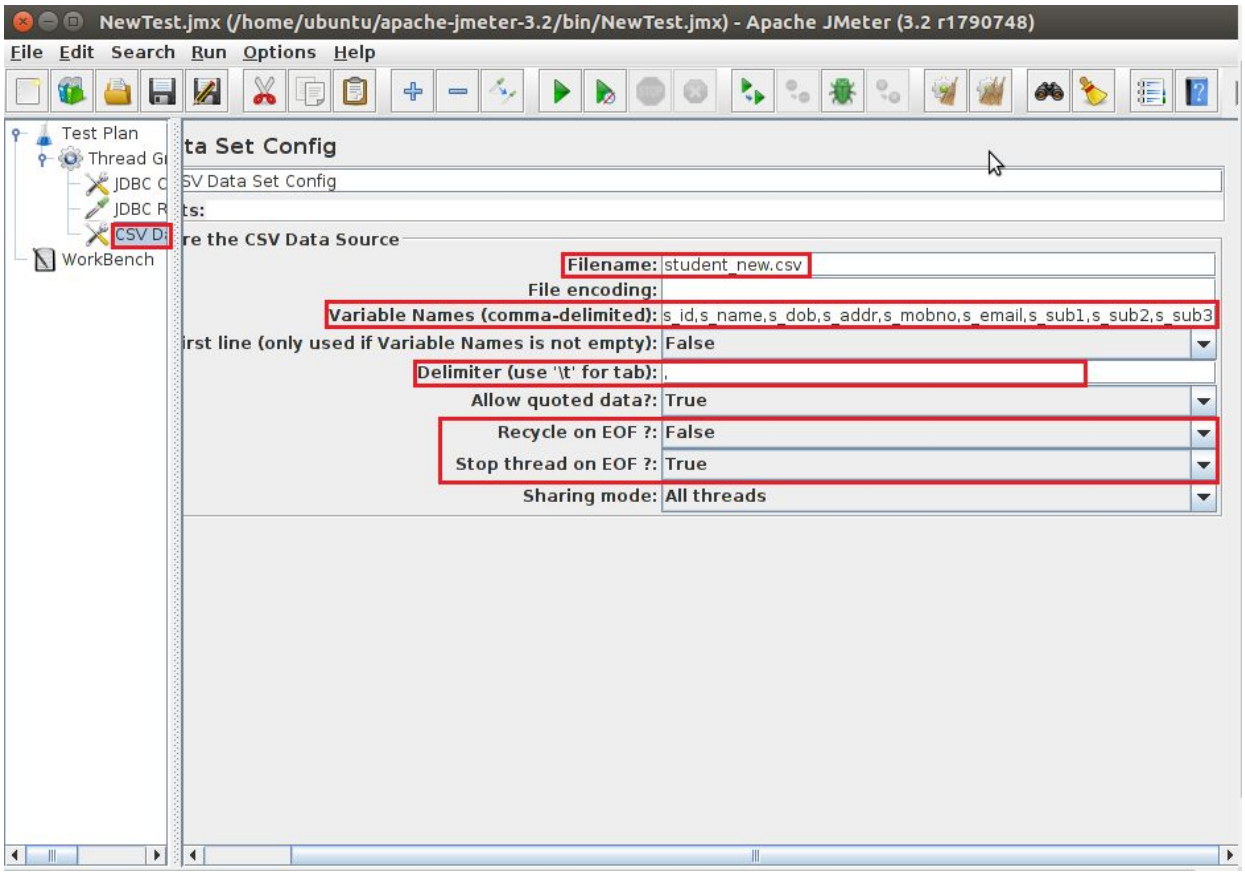


Figure 21. CSV Data Set Config

e. Run Tests in Non-GUI Mode

It is recommended to run the tests in Non-GUI mode without listeners, since using GUI mode and adding listeners tends to take up a lot of unnecessary memory which may result in improper results generation.

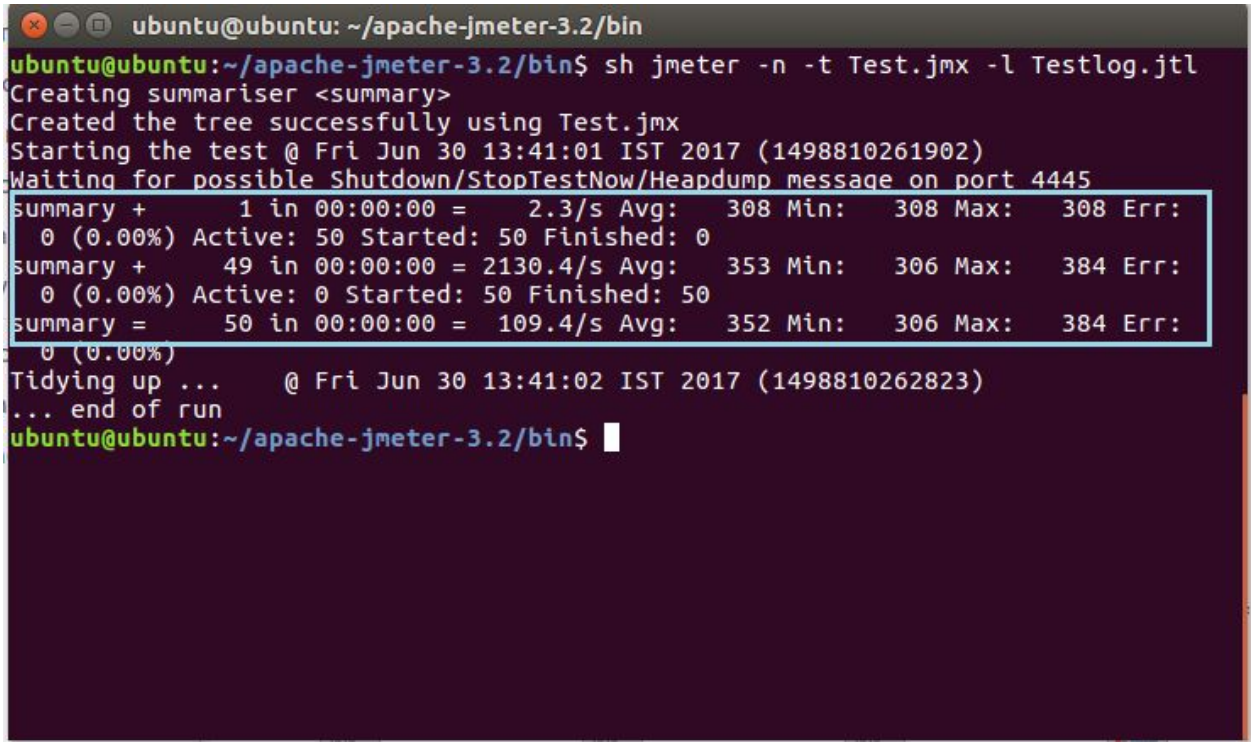


Figure 22. Test in Non-GUI Mode

Why Command Line Mode?

- GUI consumes more resources/money, thus not recommended for heavy testing
- Command line can be integrated with other systems.

To run from command line type:

sh jmeter -n -t [location of your jmeter script file] -l [location of your result file].

- ***-n = execute jmeter in non-gui mode.***

f. View Results Tree

Right Click on Workbench -> Listeners -> View Result Tree or View Result in Table. Add different graph listeners to run the tests.

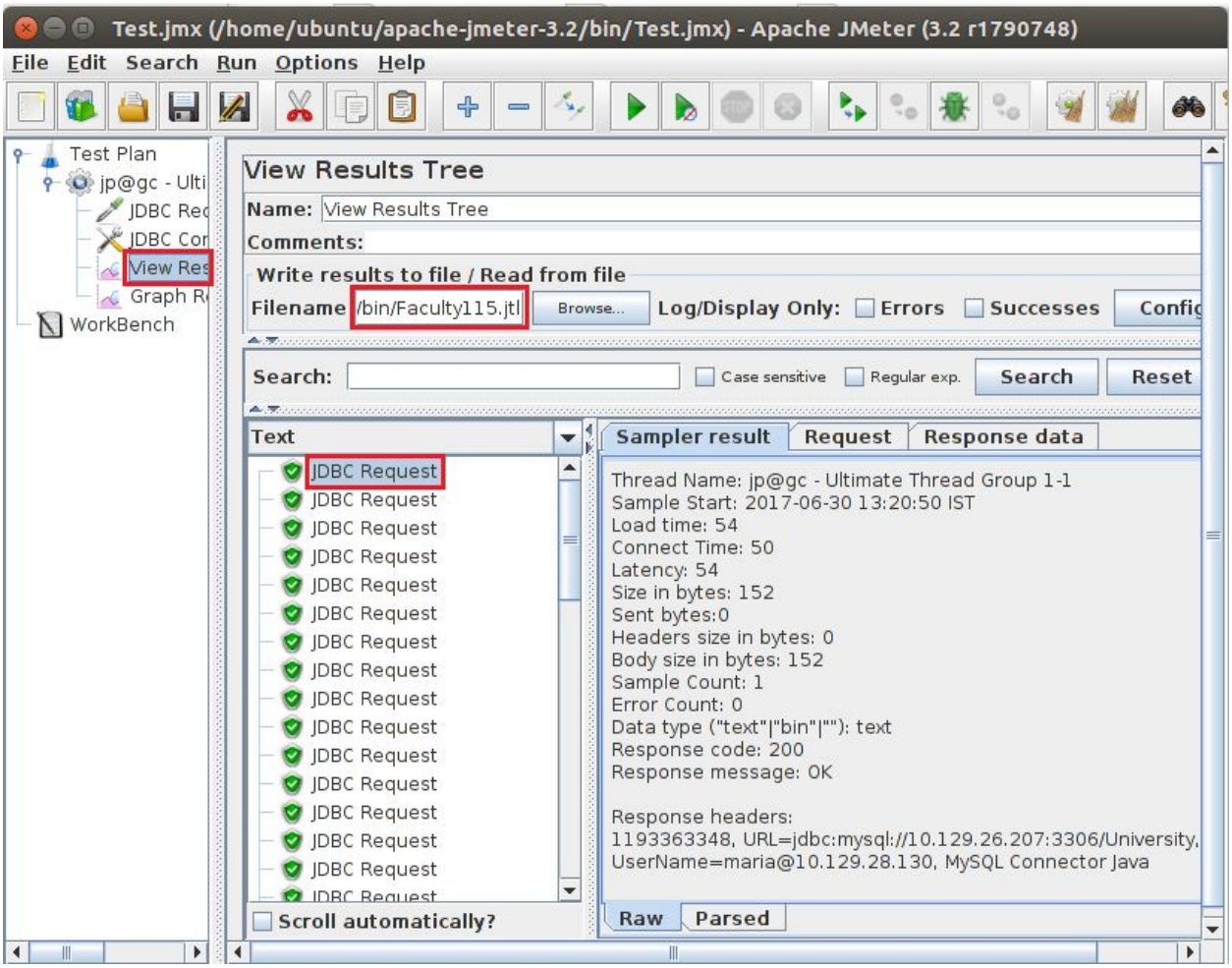


Figure 23. View Results Tree

3. Running Select Query in JMeter

The process of running a select query in JMeter is similar to that of an Insert Query. Only the JDBC Request needs to be changed to Select Statement.

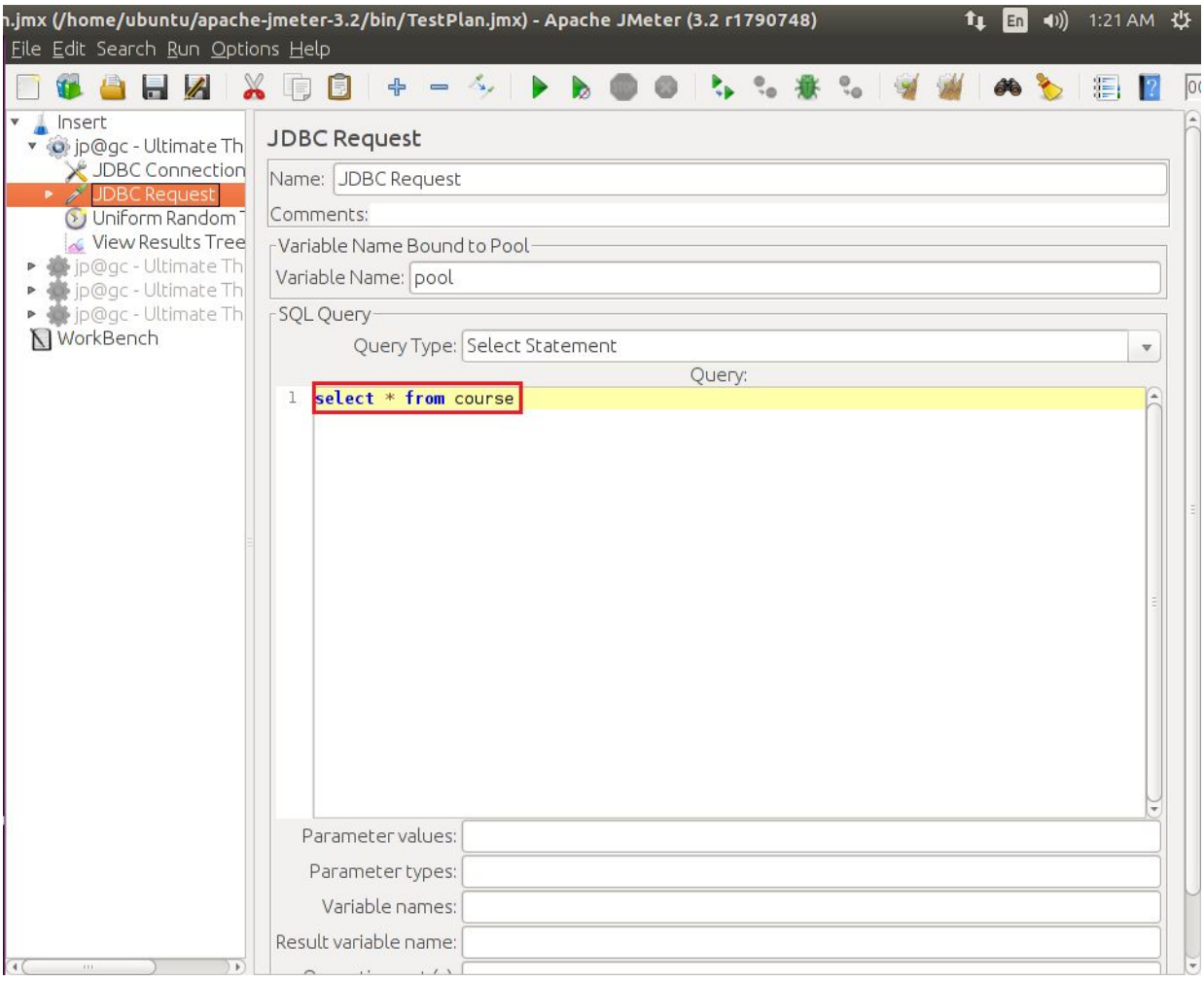


Figure 24. Select Query in Jmeter

4. HTTP(S) Test Script Recording of Websites

a. Add a Thread Group

Add a Thread Group/ Ultimate Thread Group.

b. Adding Cache Manager

RIght Click on Thread Group ->Config Element -> HTTP Cache Manager.

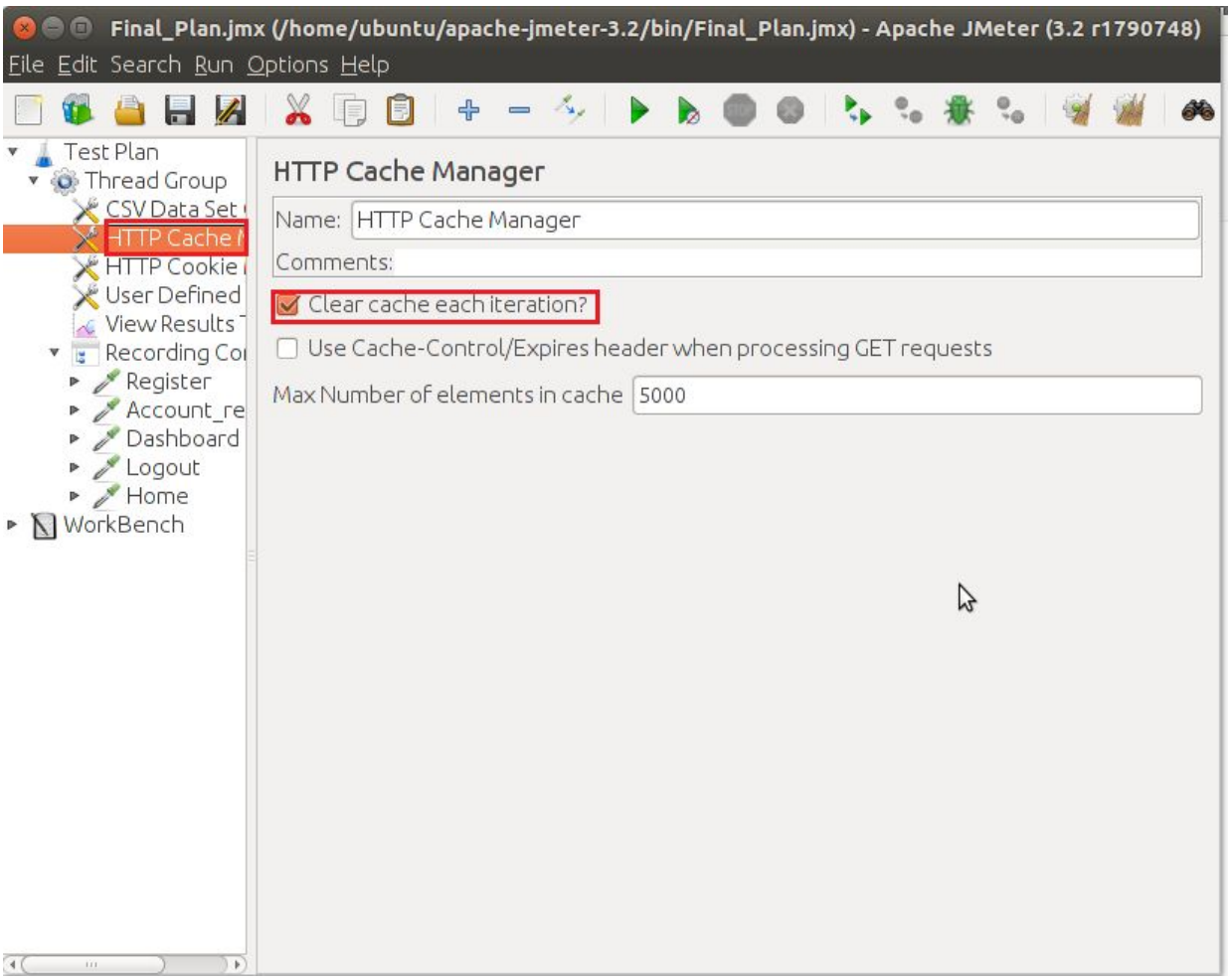
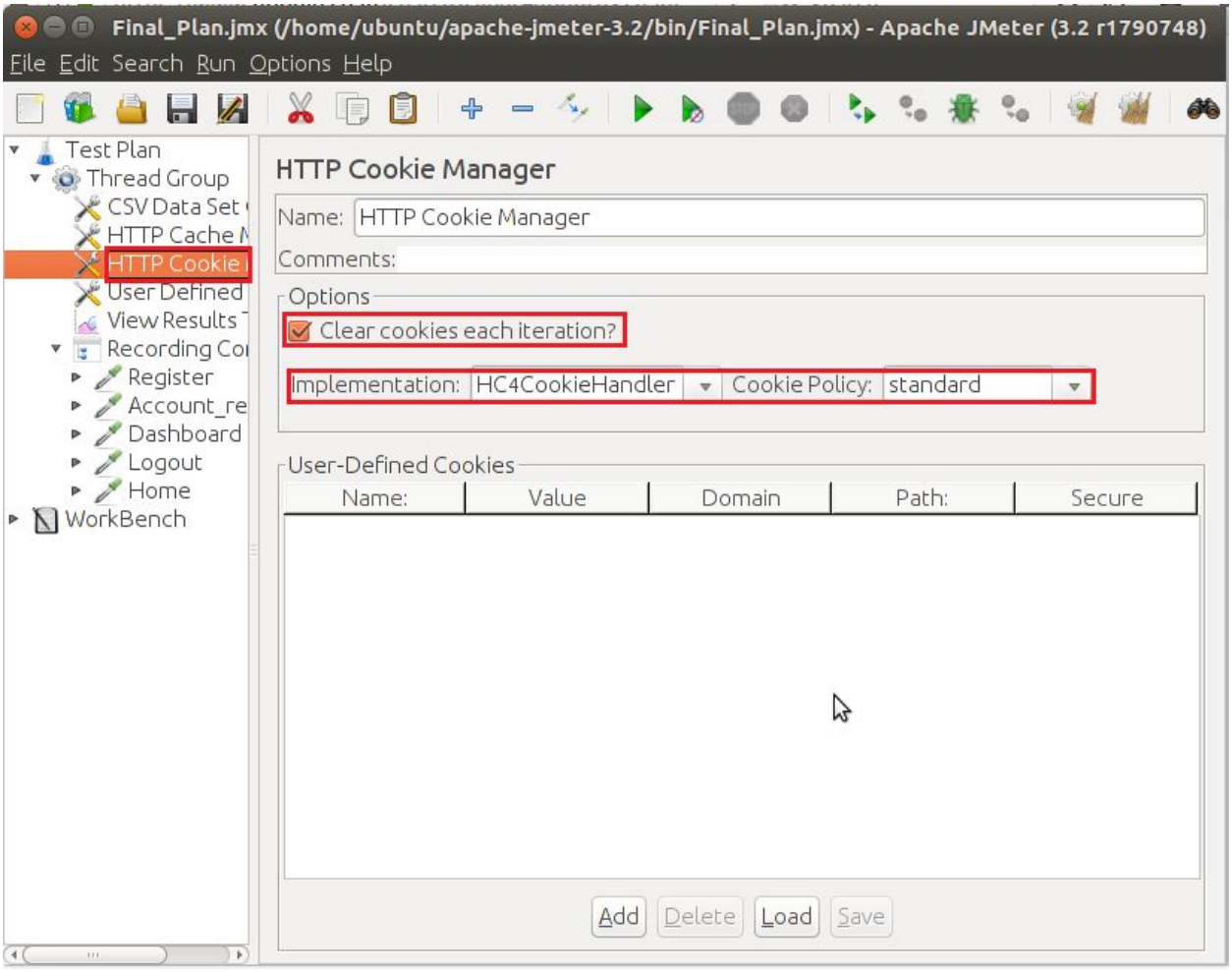


Figure 25. HTTP Cache Manager

c. Add Cookie Manager

RIght Click on Thread Group -> Config Element -> HTTP Cookie



Manager.

Figure 26. HTTP Cookie Manager

d. Add CSV Data Set and a Result Tree

Add a CSV Data Set and Result Tree as explained in the Insert Query Section

e. Add Recording Controller

Right Click on Thread Group -> Logic Controller -> Recording Controller

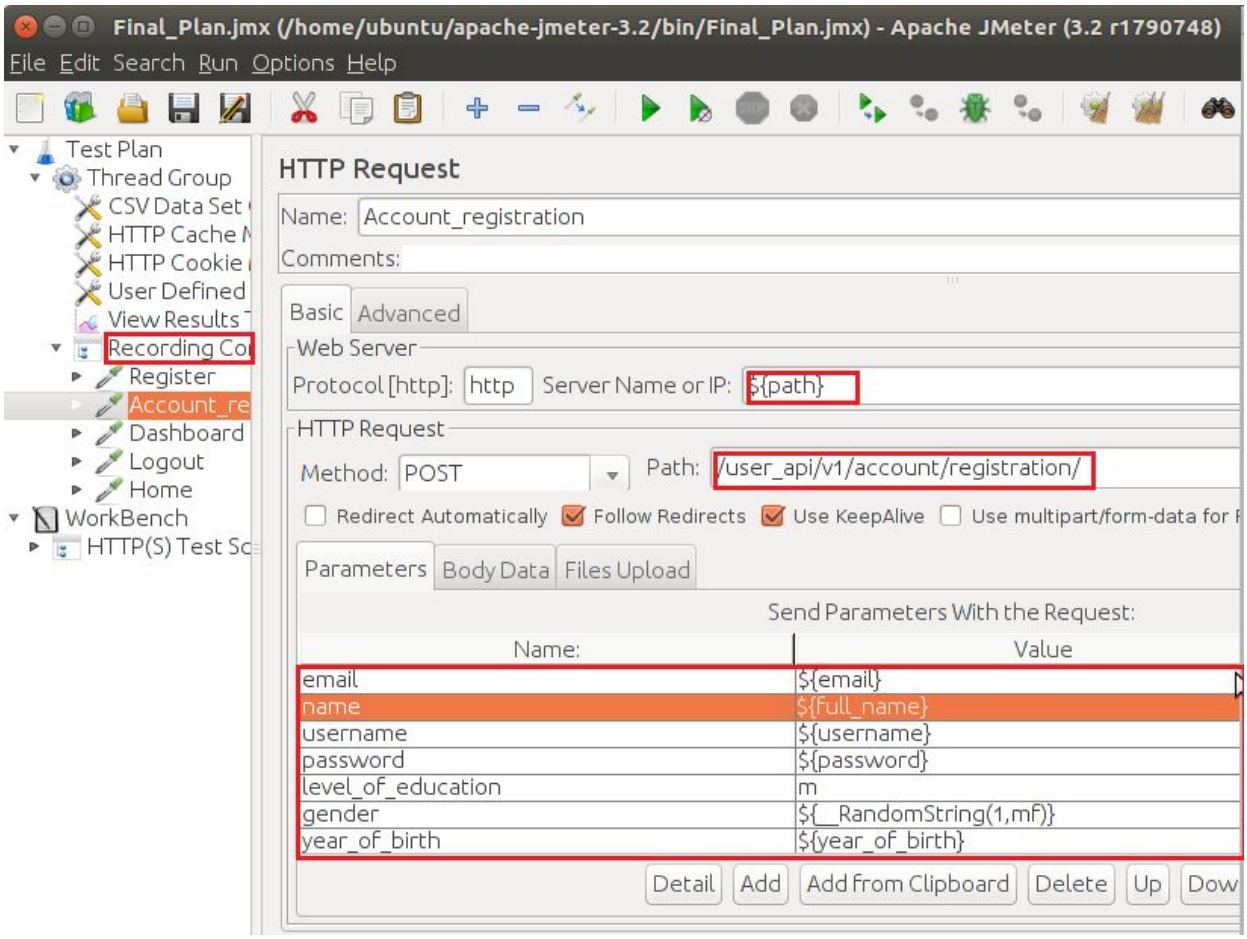


Figure 27. Recording Controller

e. Add HTTP(S) Test Script Recorder

Right Click on Workbench -> Non-Test Elements -> HTTP(S) Test Script Recorder

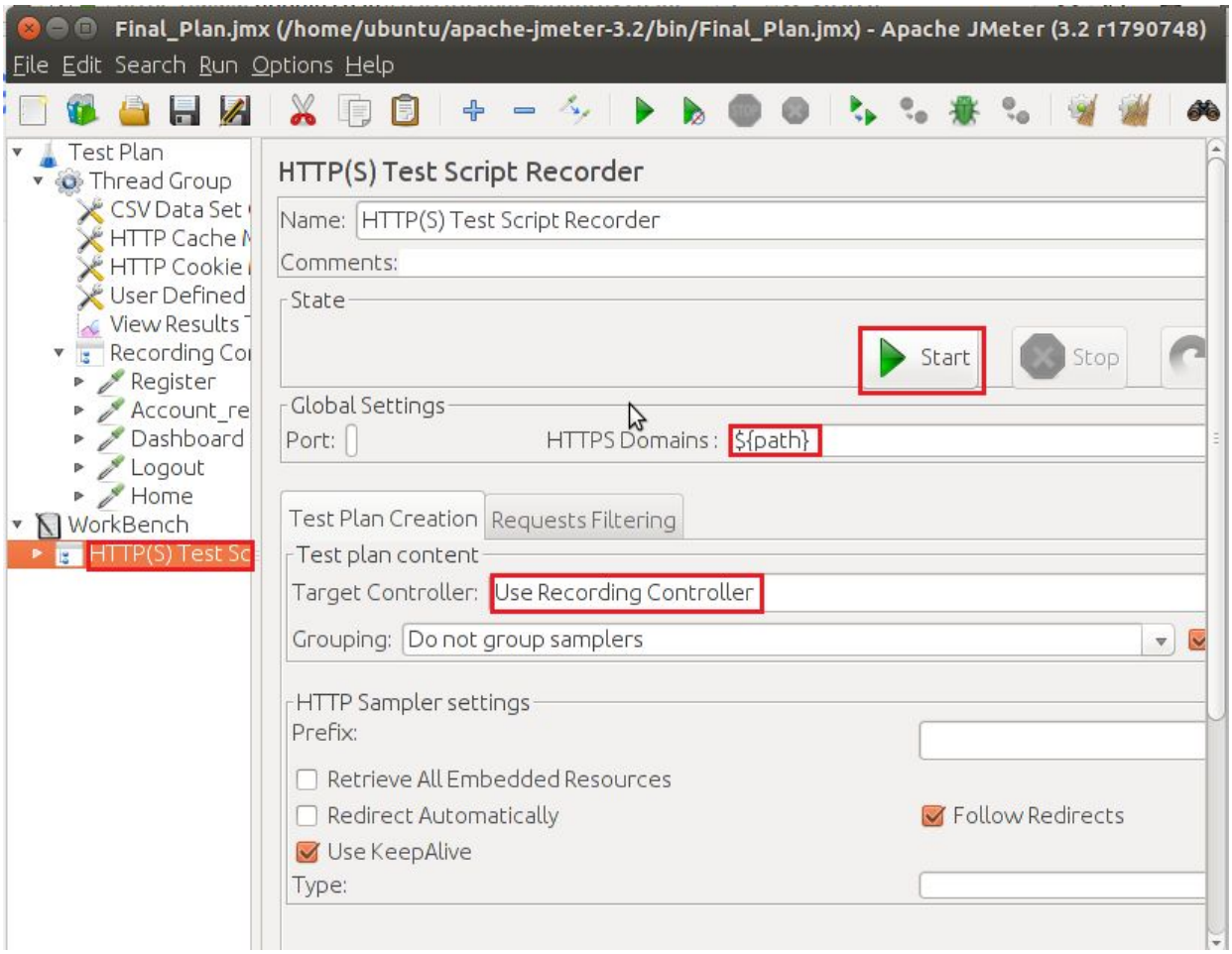


Figure 28. HTTP Test Script Recorder

Running a HTTP(S) Test Script Recorder

1. *Start Firefox and keep JMeter open simultaneously.*
2. *From the toolbar, click “Edit -> Preferences” (or “Tools > Preferences”).*
3. *Select the “Advanced” tab, and “Network” tab.*
4. *Click “Settings” button near the bottom.*
5. *On the new pop-up, check “Manual proxy configuration”. The address and port fields should be enabled now.*
6. *Address – enter “localhost” or the IP address of your system.*
7. *Port – enter “8888”(To find Jmeter Server Proxy Port number: Goto File->Templates).*
8. *Check “Use this proxy server for all protocols”.*
9. *Click “OK” button.*
10. *Clicking on the "Start" button, on the bottom of the Script Recorder window, will start the JMeter proxy server which will be used to intercept and record browser requests. Click on the Start button (of the recorder) now to start recording the script.*
11. *The first time we attempt to run the recorder, it will display an error saying that it can't start because a certificate does not exist. Click OK,then click Start a second time.*
12. *A message that says that a temporary certificate named ApacheJMeterTemporaryRootCA.crt has been created in JMeter bin directory would then be displayed. Click OK and continue.*
13. *Now there should be a little triangle next to the Recording Controller. Click on it to expand and see the requests that it has recorded.*

14. *Refine the list of HTTP requests by simply deleting unwanted entries.*
15. *Once recording is completed, click the "Stop" button at the bottom of the HTTP(S) Test Script Recorder window.*
16. *Firefox will no longer be able to reach any pages (because it is configured to use port 8888 as a proxy)--configure it to use "No proxy" to make it function normally.*

8. Technical Details

Tests were run simultaneously on two VM, (**VM1 and VM2)**, with the following changes made to the configurations, to simulate different production environments.

1. Changes to my.cnf

Path to File: */home/ubuntu/etc/mysql/my.cnf*

The following lines should be added/modified in the [*mysqld*] section:

Property	VM1	VM2
innodb_buffer_pool_size	256	512
innodb_log_file_size	128	256
innodb_buffer_pool_instances	4	8
max_connections	512	1024
innodb_spin_wait_delay	0	0

Table 2. Changes to my.cnf

Uncomment the line, (essential to run RocksDB)

- *binlog_format=ROW*

Add the following line before *skip-external-locking*

- *plugin-load-add=ha_rocksdb.so*

2. **Changes to jmeter.properties file**

   Path: */home/ubuntu/apache-jmeter-3.2/bin*

   The following lines should be added to the jmeter.properties files :

   CookieManager.delete_null_cookies=false

   CookieManager.save.cookies=true

   jmeter.save.saveservice.url=true

   jmeter.save.saveservice.requestHeaders=true

3. **Changes to user.properties file**

   Path: */home/ubuntu/apache-jmeter-3.2/bin*

   The following lines should be added to the user.properties file

   httpclient4.retrycount=1

   hc.parameters.file=hc.parameters

4. **Changes to hc.parameters file**

   Path: /home/ubuntu/apache-jmeter-3.2/bin

   http.connection.stalecheck$Boolean=true

5. Adding the Plugins Manager in Jmeter

Download the *Plugins Manager JAR file (* and put it into JMeter's lib/ext directory. Then start JMeter and go to "Options" menu to access the Plugins Manager.

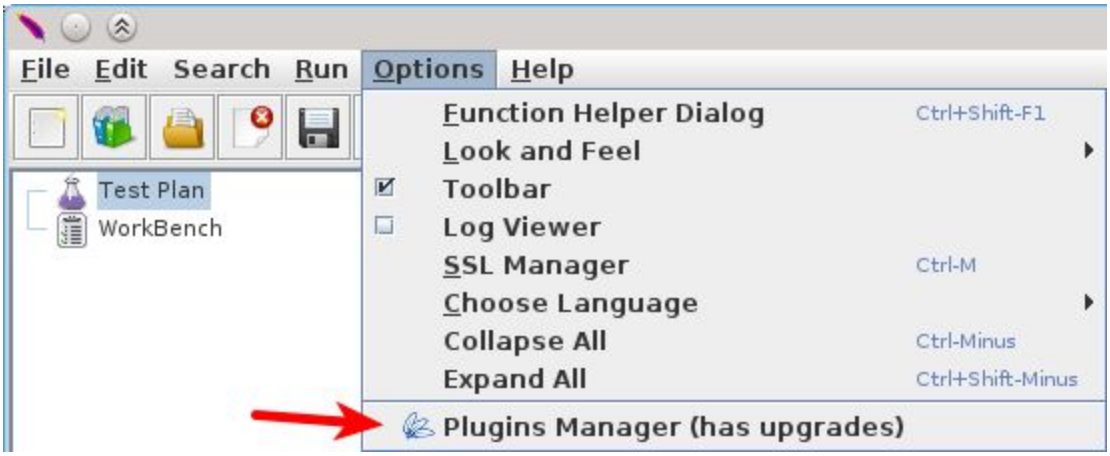


Figure 29. Adding Plugins Manager in Jmeter

The label on the menu item will say "has upgrades" in case any of your installed plugins have upgrades for it.

Clicking on menu item will bring up the plugins manager dialog:

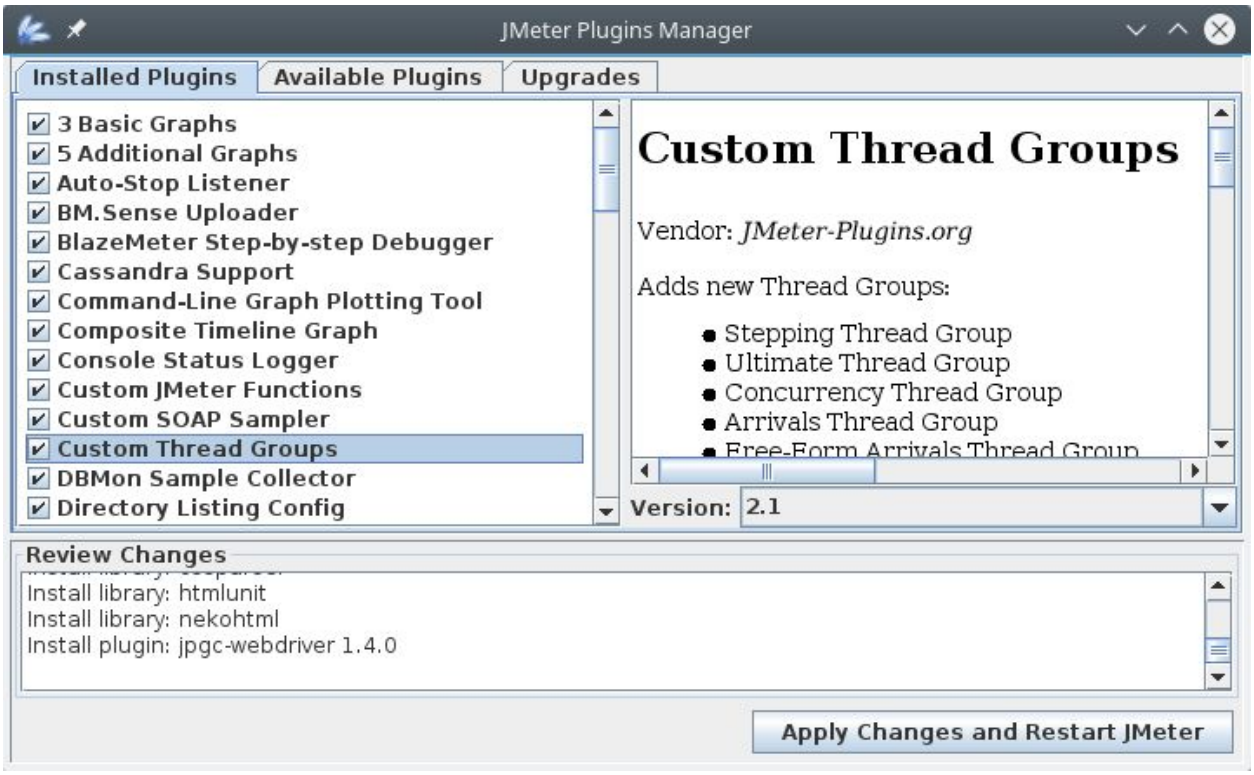


Figure 30. Jmeter Plugins Manager

This dialog has three tabs, a Review Changes area and an Apply button. Removing ticks from plugins on "Installed" tab will uninstall them. Setting ticks on "Available" tab will install the plugin.

Click on Apply Changes and Restart Jmeter.

9. Test Results

Query 1:

INSERT INTO student VALUES
(s_id,s_name,s_dob,s_addr,s_addr,s_email,s_sub1,s_sub2,s_sub3);

- **Number of Threads taken = 1000**
- **Response Latencies V/s Elapsed time :**

1. VM1

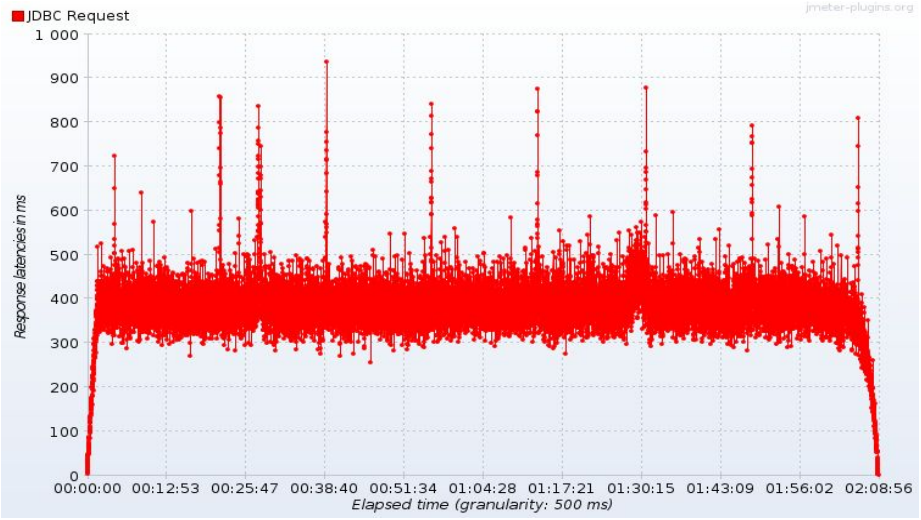


Figure 31. InnoDB(Response Latencies over elapsed time)

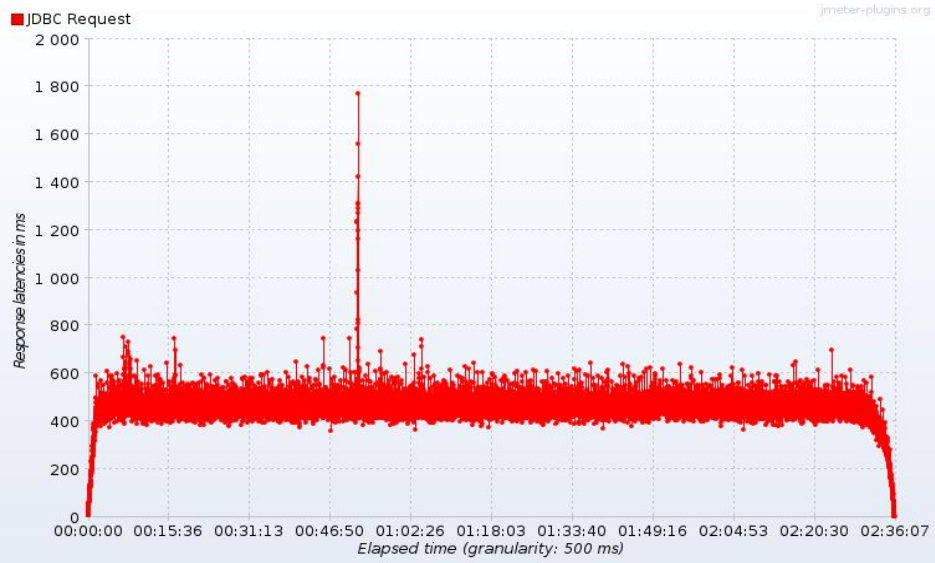


Figure 32. RocksDB (Response Latencies over elapsed time)

2. VM2:

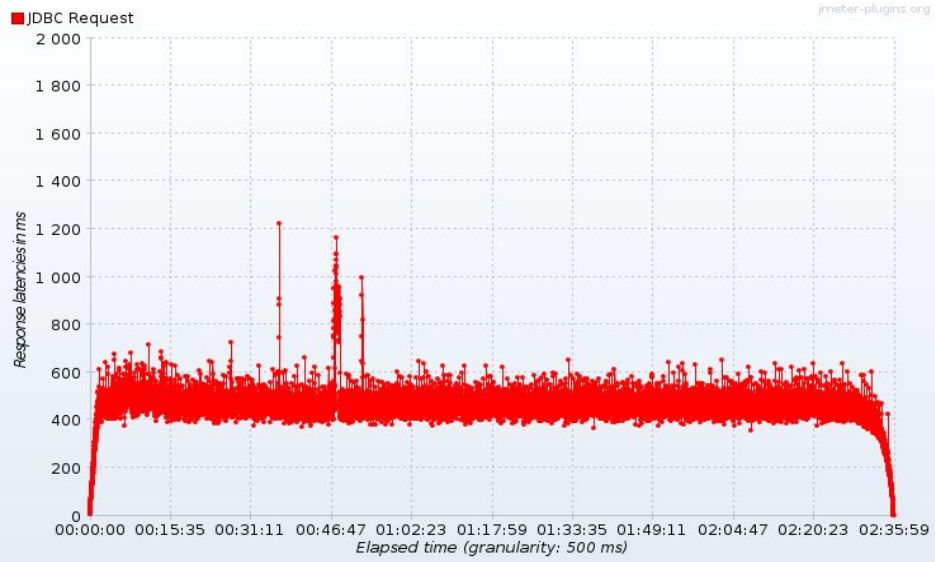


Figure 33. InnoDB (Response Latencies over elapsed time)

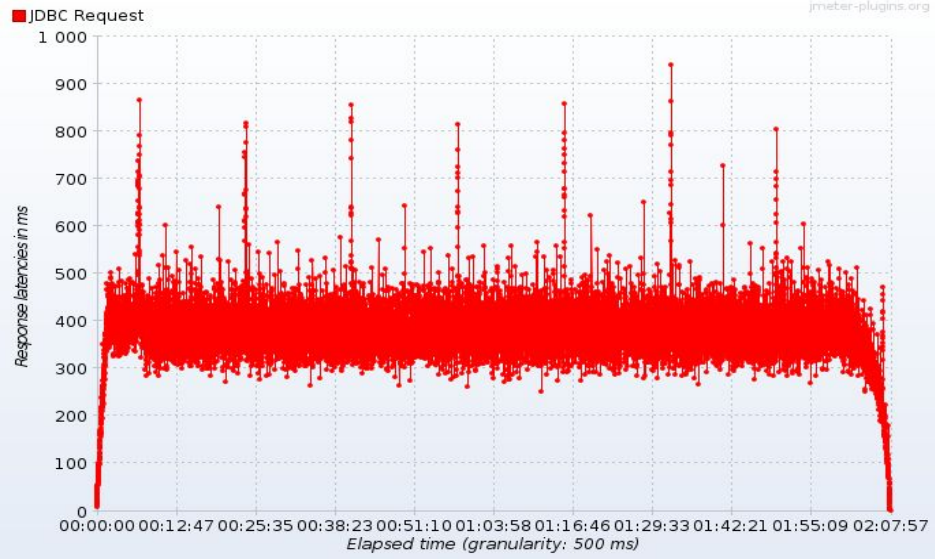


Figure 34. RocksDB(Response Latencies over elapsed time)

- **Number of estimated transactions V/s Number of active threads**

1. VM1:

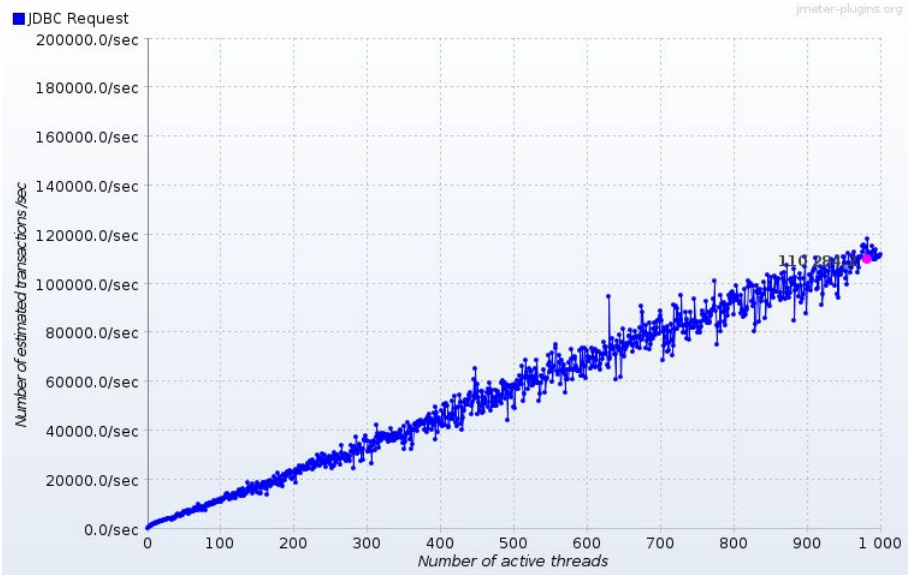


Figure 35. InnoDB(Estimated Transactions V/s Active Threads)

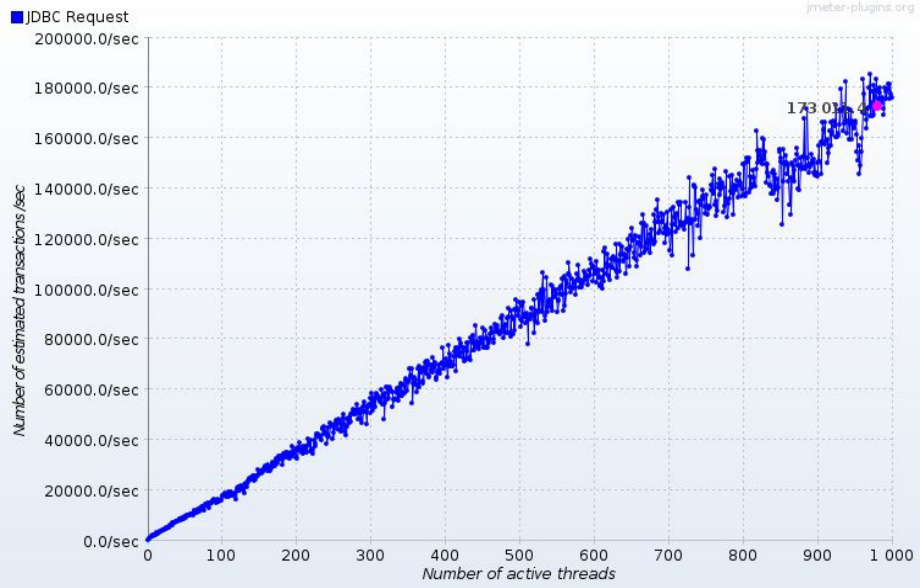


Figure 36. RocksDB(Estimated Transactions V/s Active Threads)

2. VM 2:

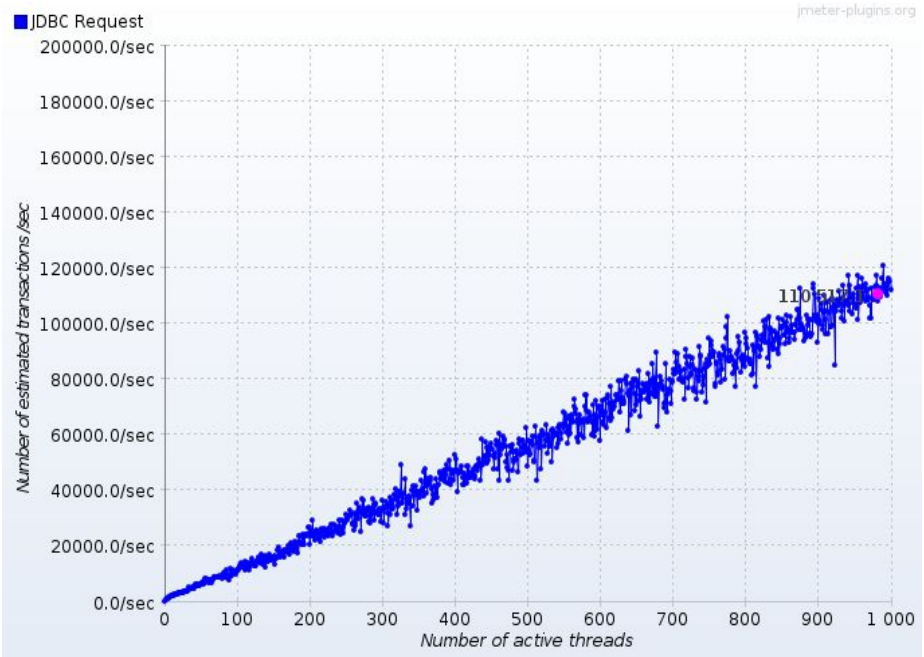


Figure 37. InnoDB(Estimated Transactions V/s Active Threads)

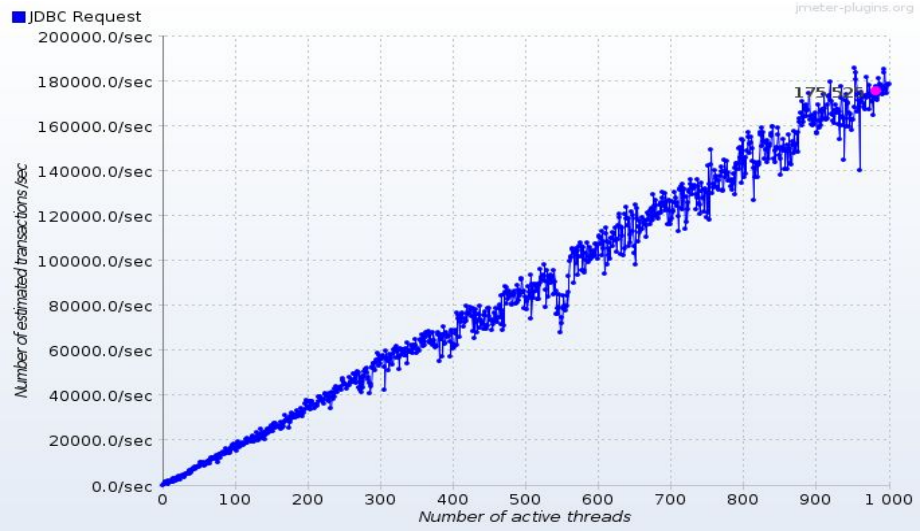


Figure 38. RocksDB(Estimated Transactions V/s Active Threads)

Query 2:

SELECT COUNT(s_id) FROM student GROUP BY s_sub1, s_sub2 ;

- **Number of Active Threads : 9000**
- **Response Latencies V/s elapsed time :**

1. VM1

Figure 39. InnoDB(Response Latencies over elapsed time)

Figure 40. RocksDB(Response Latencies over elapsed time)

2. VM2

Figure 41. InnoDB(Response Latencies over elapsed time)

Figure 42. RocksDB(Response Latencies over elapsed time)

- **Number of estimated transactions V/s Number of active threads :**

1. VM1:

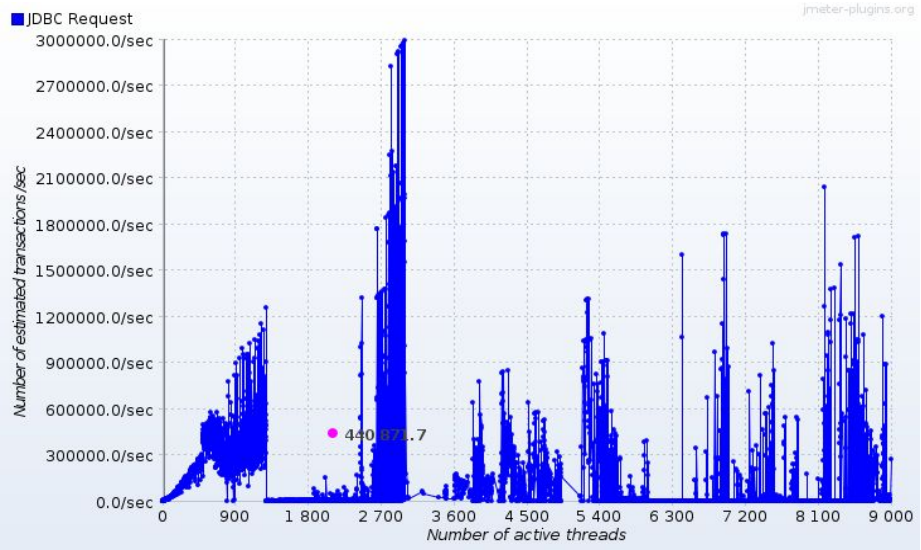


Figure 43. InnoDB(Estimated Transactions V/s Active Threads)

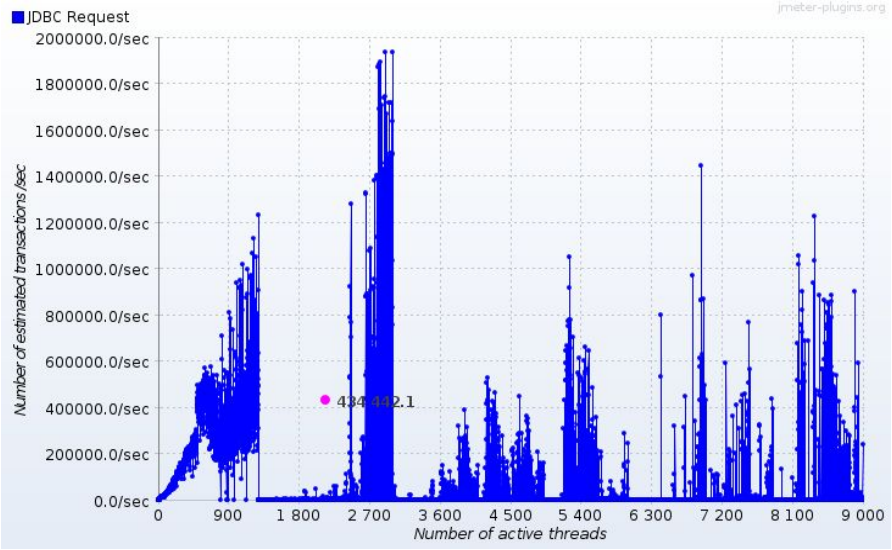


Figure 44. RocksDB(Estimated Transactions V/s Active Threads)

2. VM2:

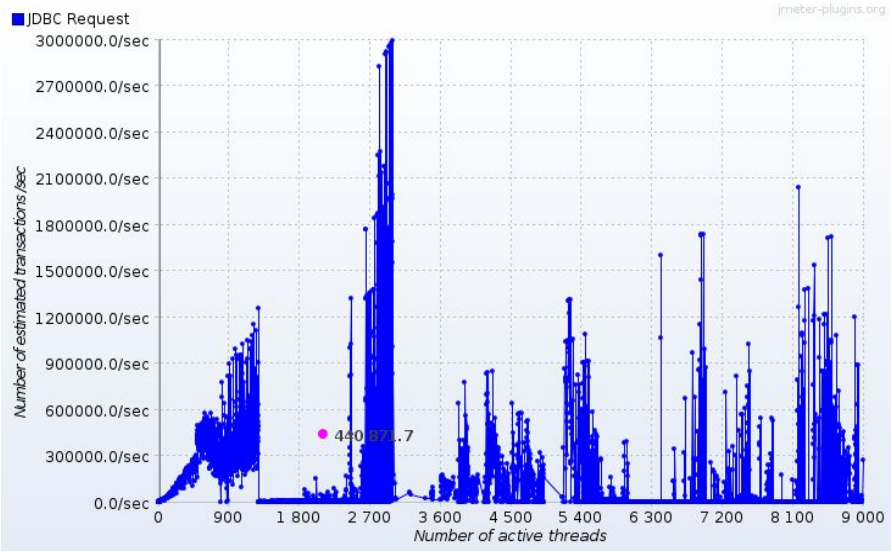


Figure 45. InnoDB(Estimated Transactions V/s Active Threads)

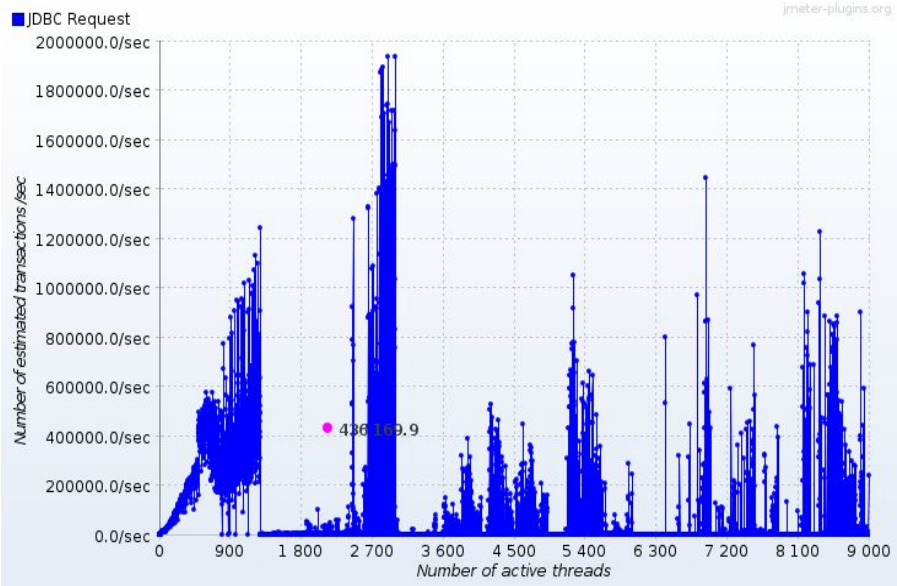


Figure 46. RocksDB(Estimated Transactions V/s Active Threads)

Query 3:

DELETE from STUDENT where s_id =20000000;

- **Number of Threads =500**
- **Response Latencies V/s elapsed time**

1. VM1:

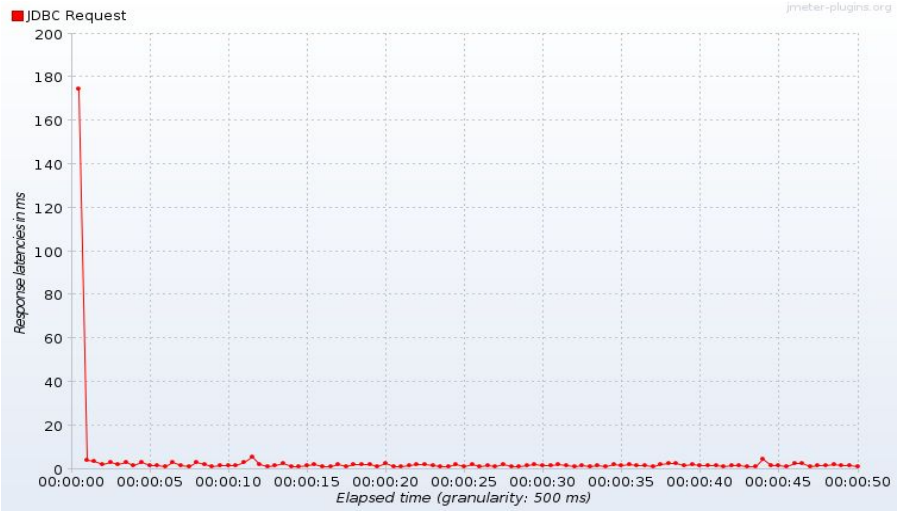


Figure 47. InnoDB(Response Latencies over elapsed time)

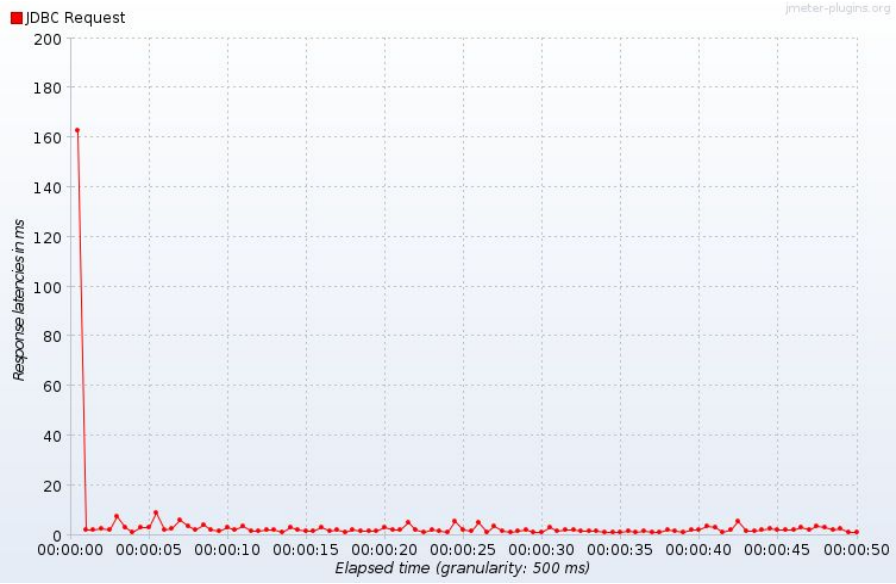


Figure 48. RocksDB(Response Latencies over elapsed time)

2. VM2:

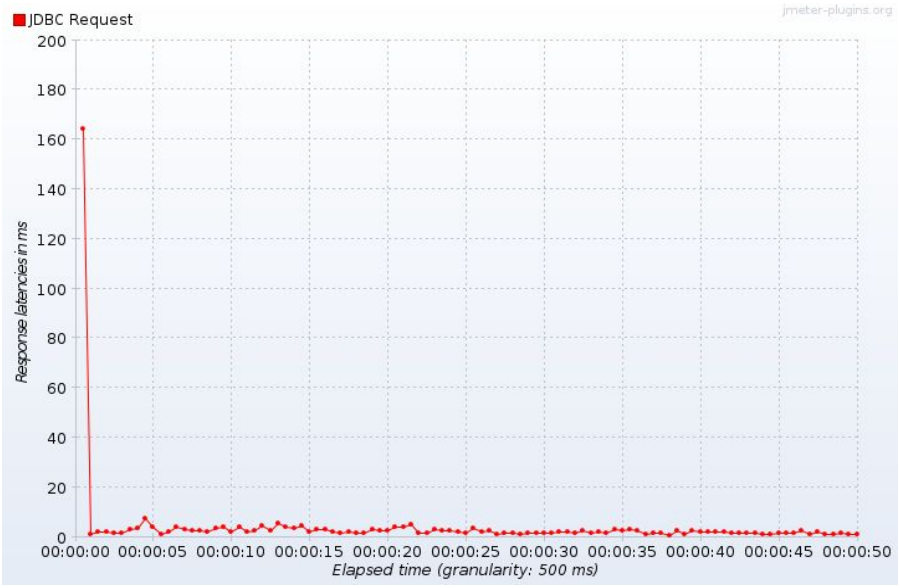


Figure 49. InnoDB(Response Latencies over elapsed time)

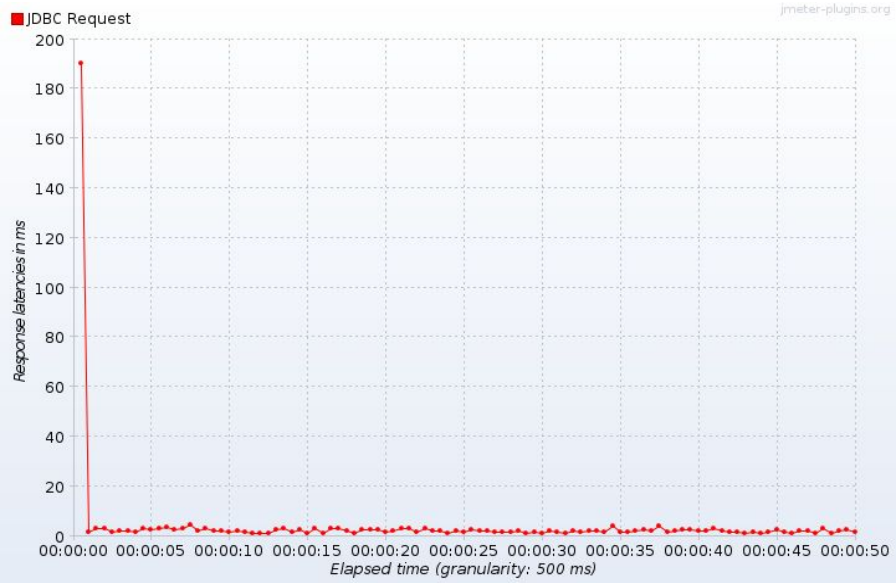


Figure 50. RocksDB(Response Latencies over elapsed time)

- **No of Estimated Transaction / No of Active Thread:**

1.VM 1:

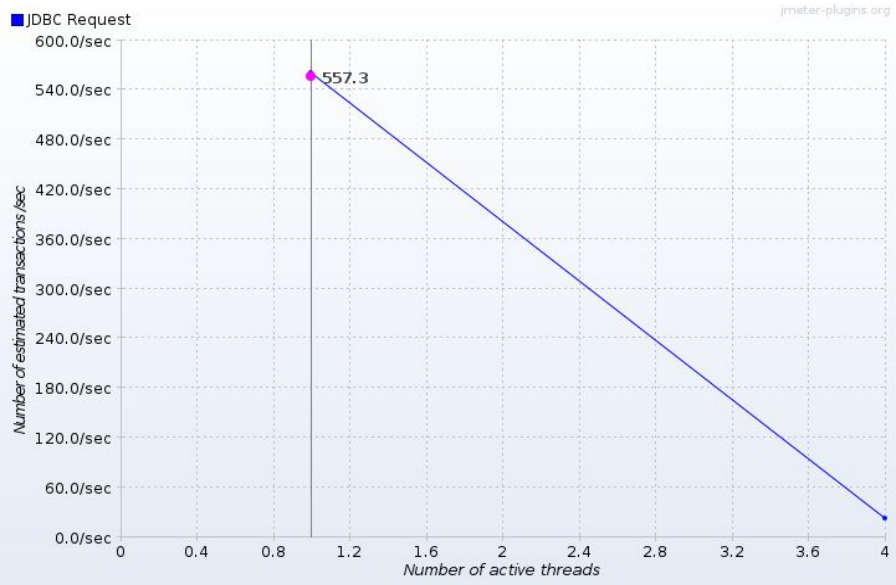


Figure 51. InnoDB(Estimated transactions over active threads)

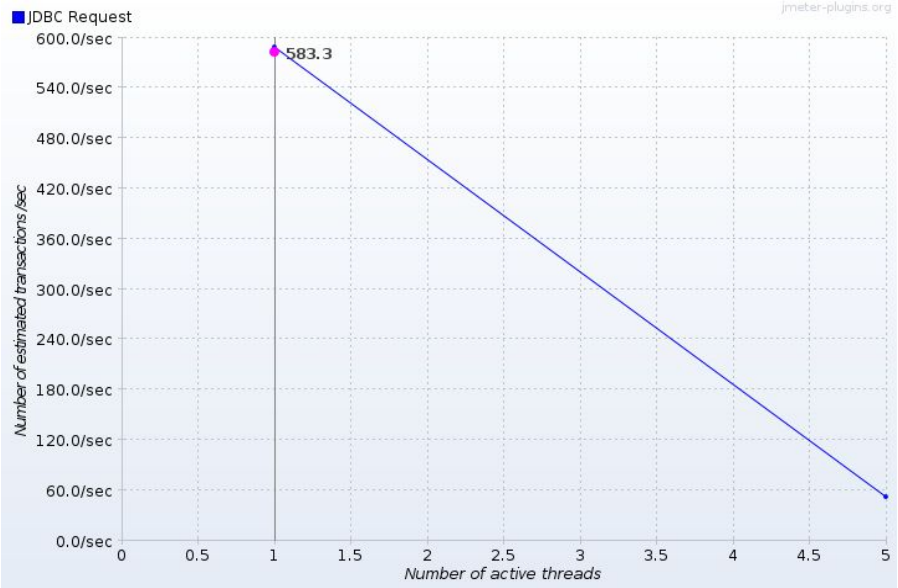


Figure 52. InnoDB(Estimated transactions over active threads)

2.VM2:

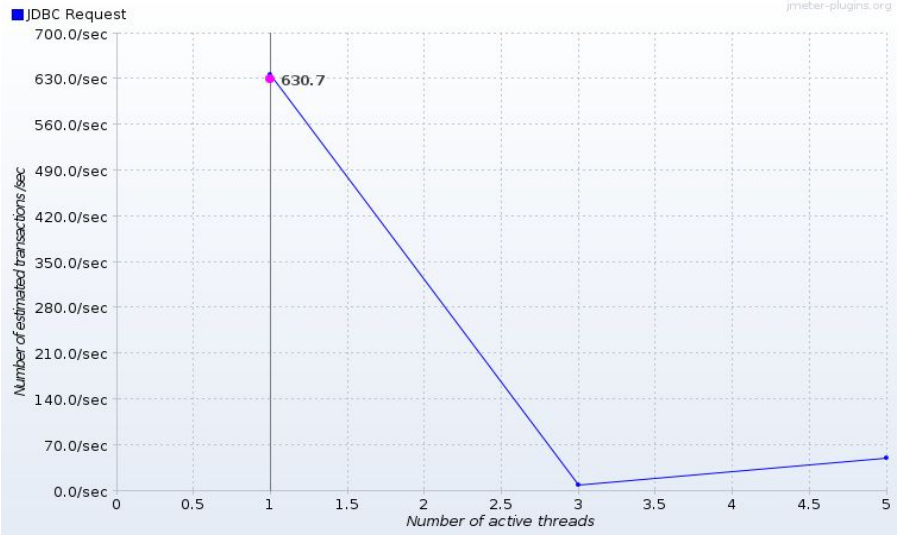


Figure 53. InnoDB(Estimated transactions over active threads)

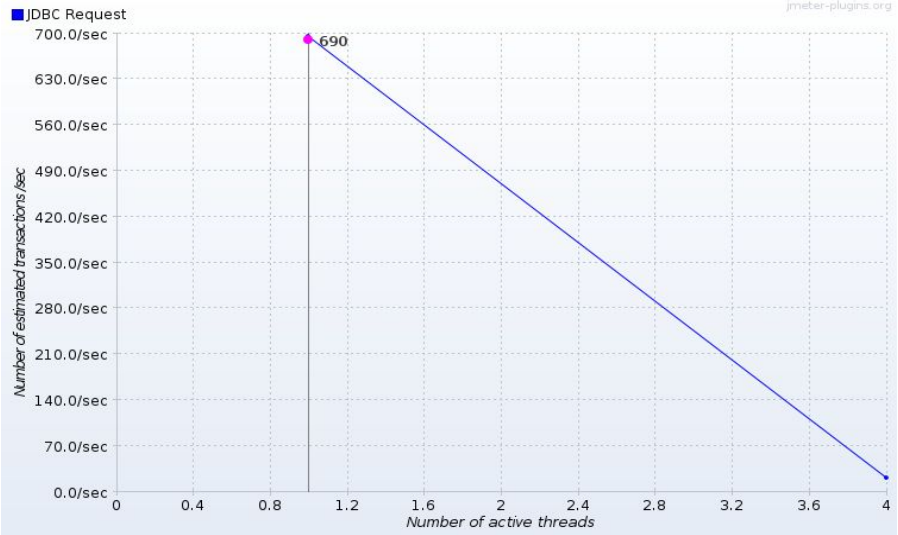


Figure 54. RocksDB(Estimated transactions over active threads)

Query 4:

UPDATE SET s_name =’EMMA' WHERE s_id =24;

- **Number of Threads = 500**
- **Number of estimated transactions V/s Number of active threads :**
    1. VM1

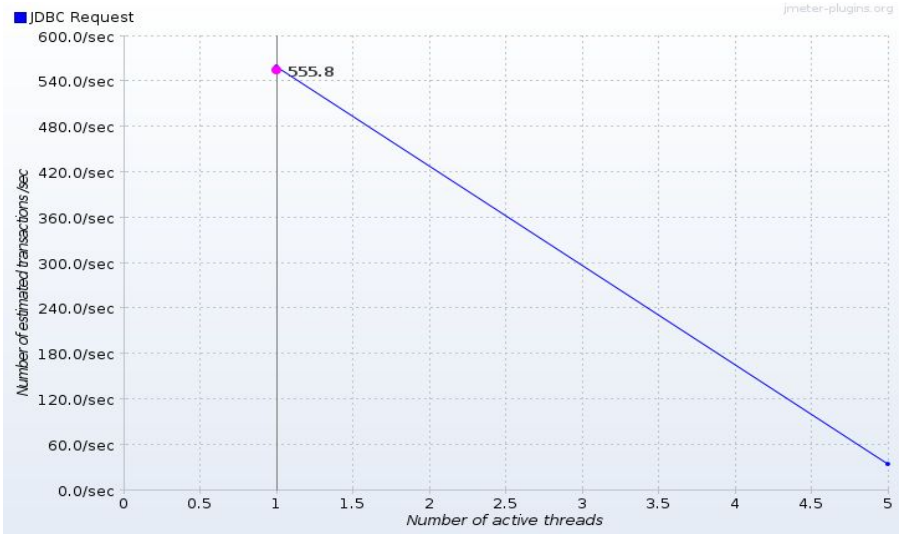


Figure 55. InnoDB (Estimated transactions over active threads)

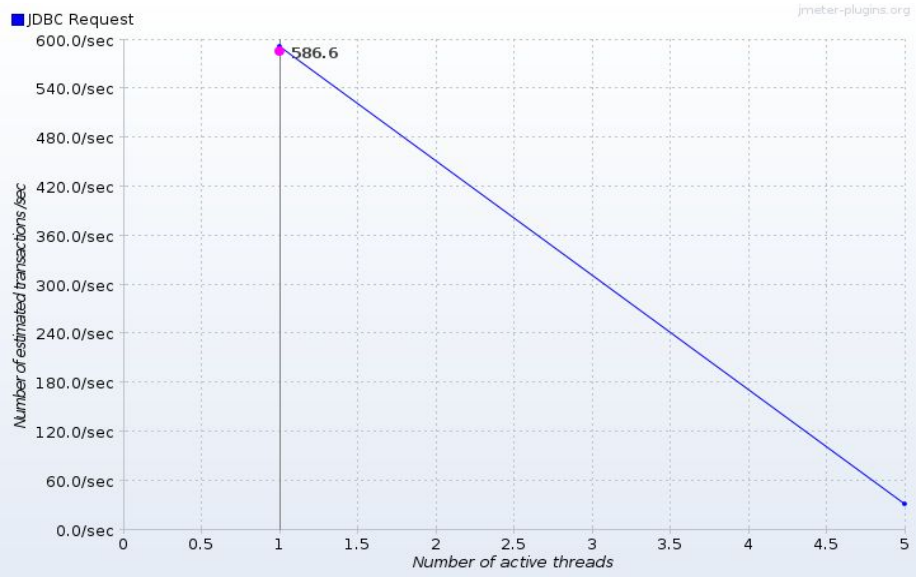


Figure 56. RocksDB(Estimated transactions over active threads)

2. VM2

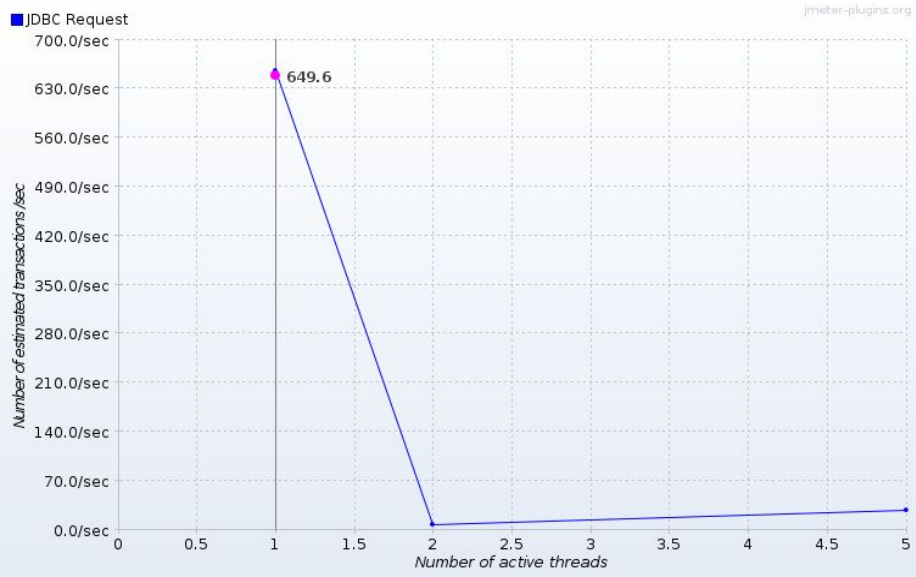


Figure 57. InnoDB(Estimated transactions over active threads)

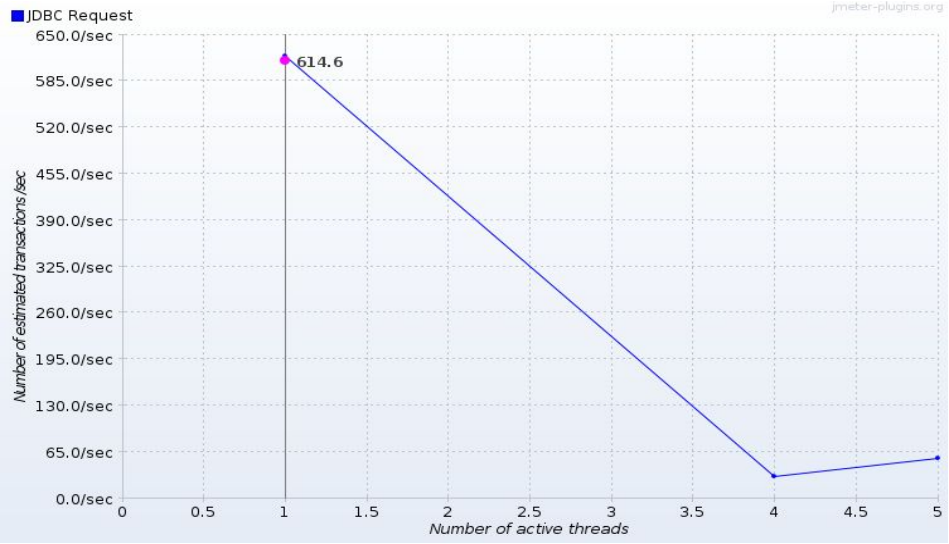


Figure 58. RocksDB(Estimated transactions over active threads)

- **Response Latencies V/s elapsed time :**

1. VM1:

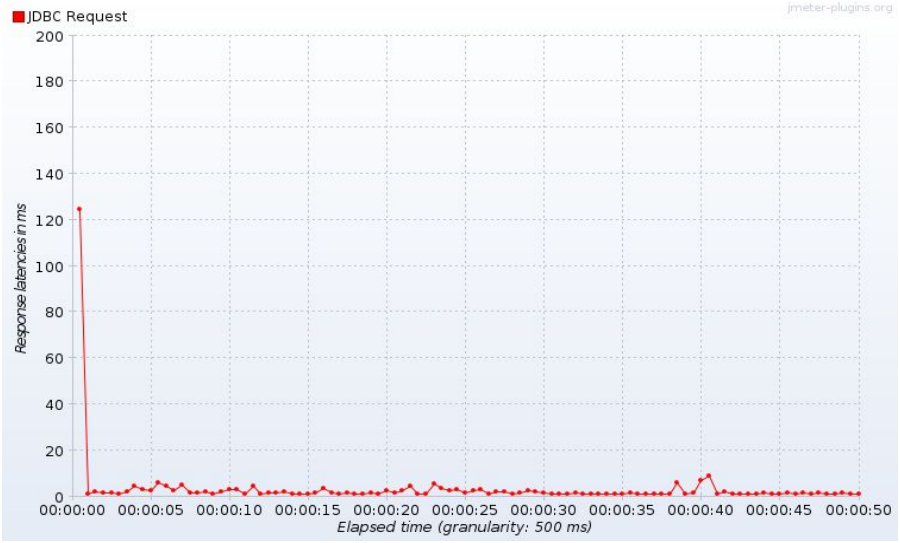


Figure 59. InnoDB(Response Latencies over elapsed time)

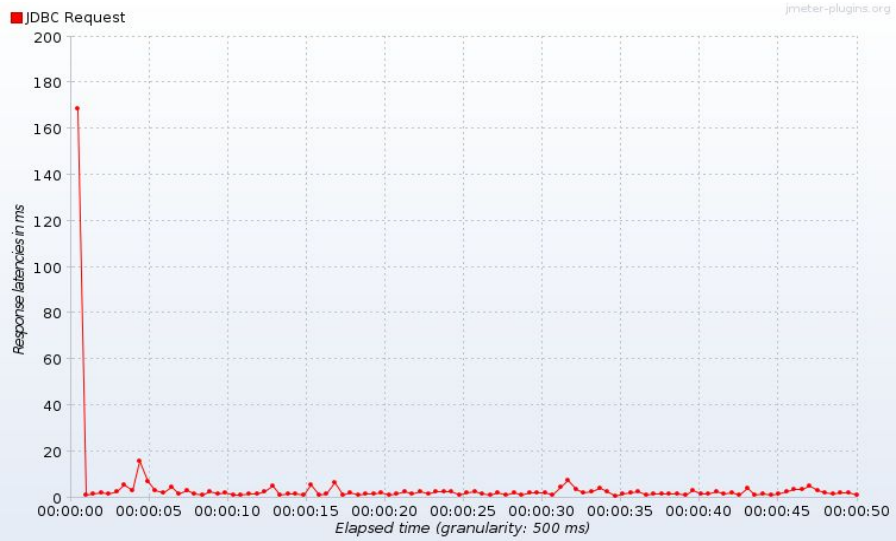


Figure 60. RocksDB(Response Latencies over elapsed time)

2. VM2:

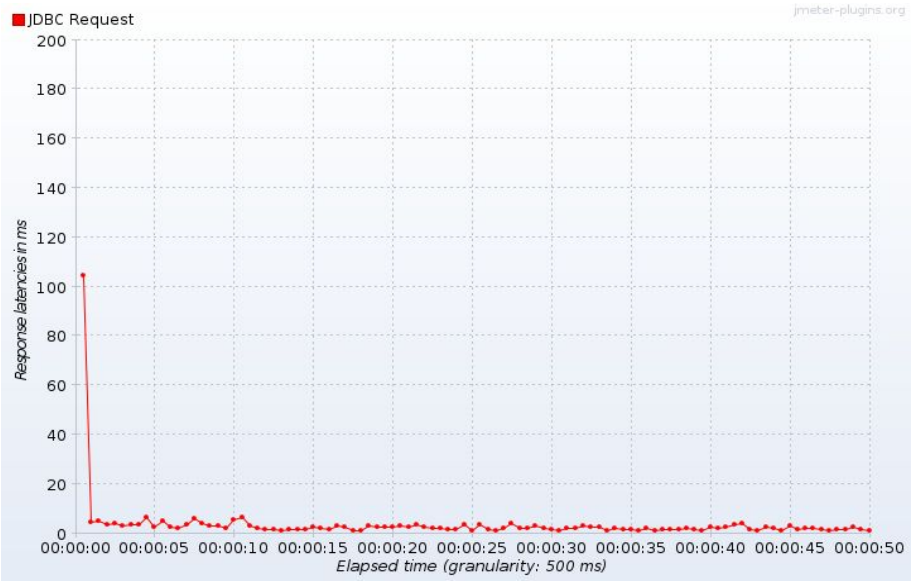


Figure 61. InnoDB(Response Latencies over elapsed time)

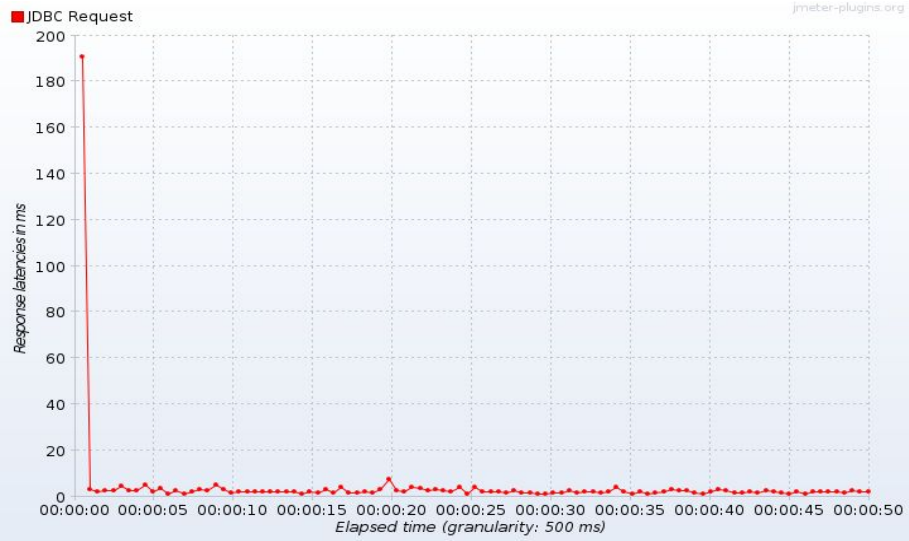


Figure 62. RocksDB(Response Latencies over elapsed time)

Performance Measurement of OpenEdx

1. **User Registration Phase**
    - **Active threads v/s Time**

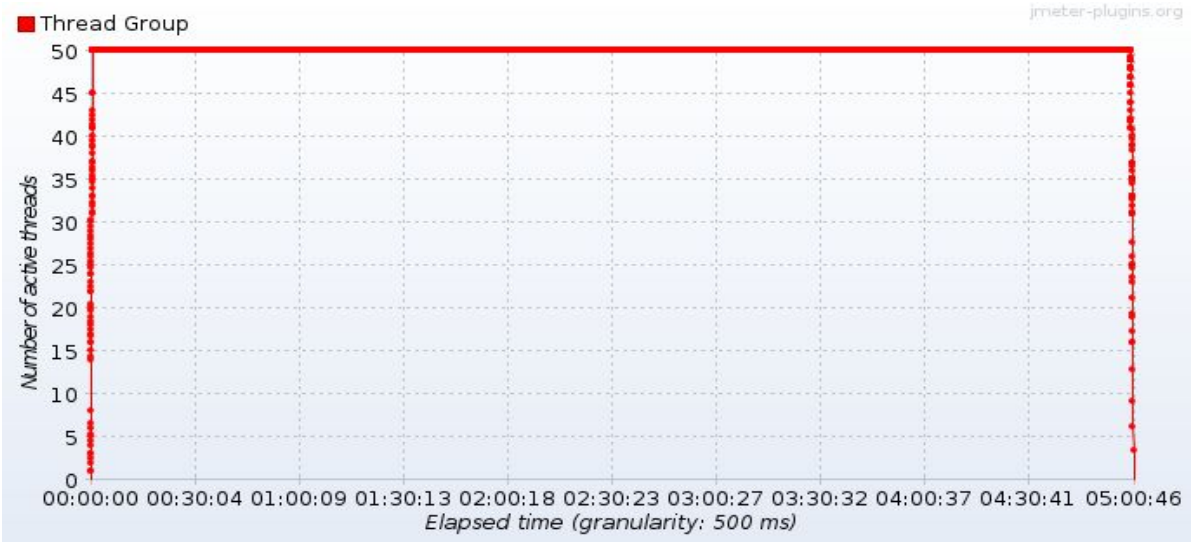


Figure 63. Bytes/sec over elapsed time

- **Bytes Throughput vs Time**

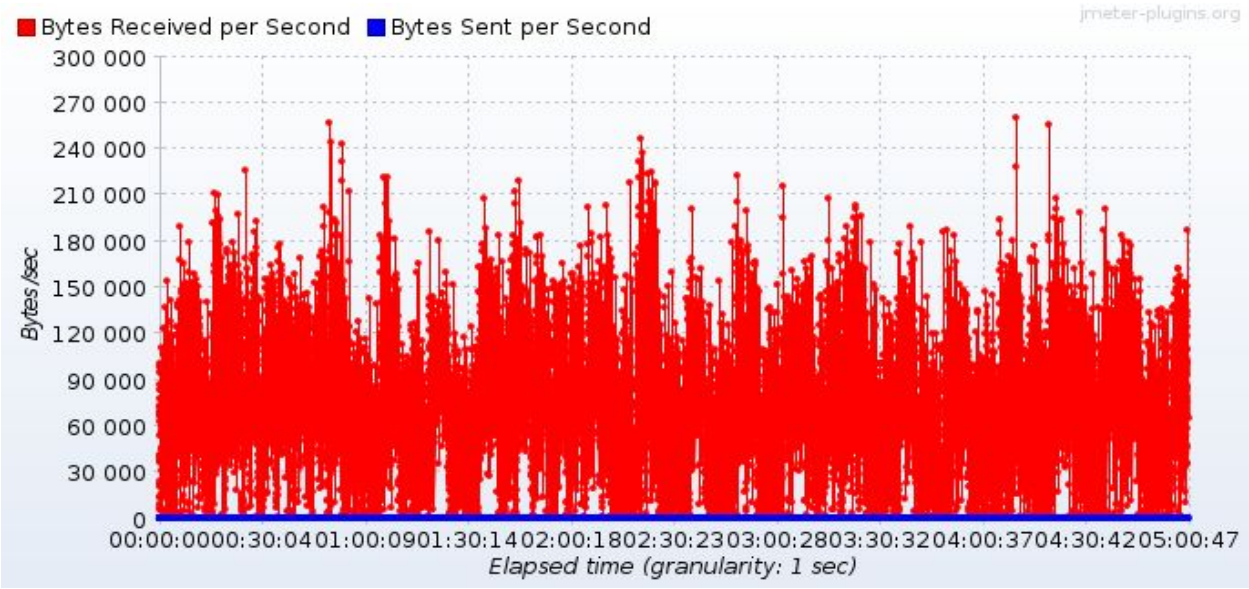


Figure 64. Bytes/sec over elapsed time

- **Response Latency**

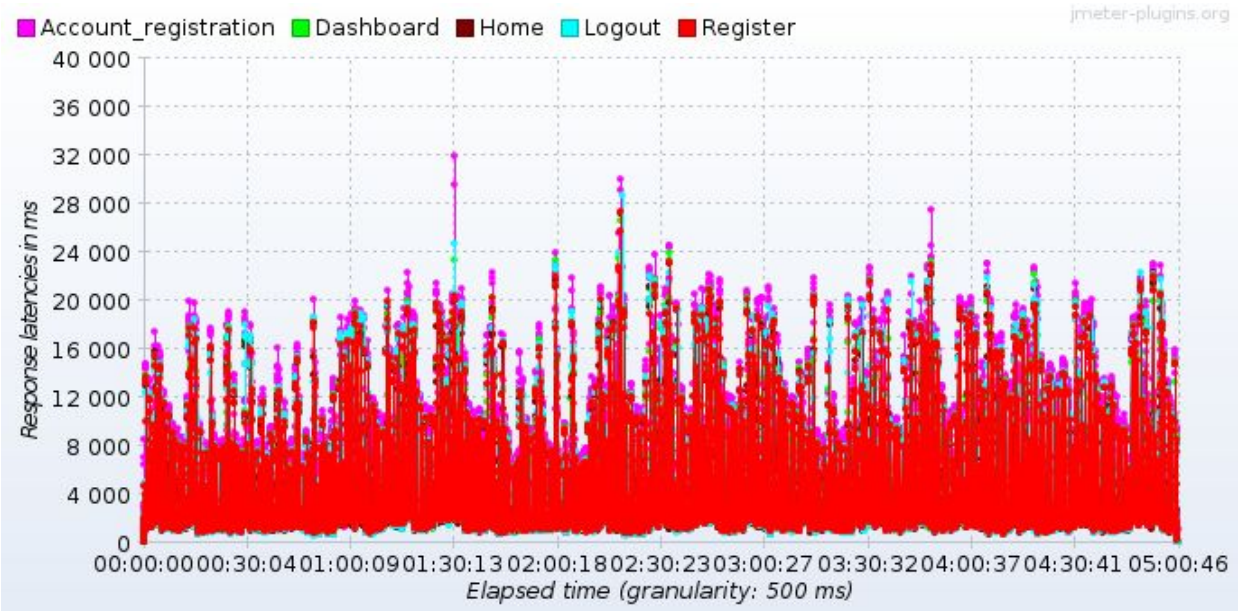


Figure 65. Response latencies over elapsed time

- **Transaction Throughput**

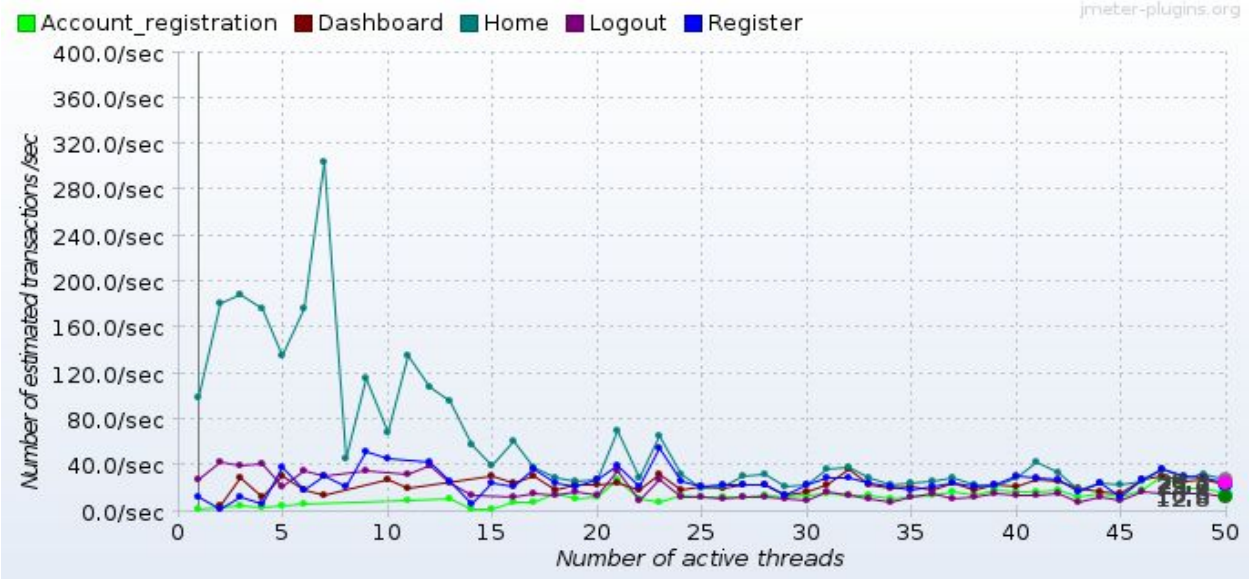


Figure 66. Estimated transactions over active threads

2. Login Phase

- **Active Threads vs Time**

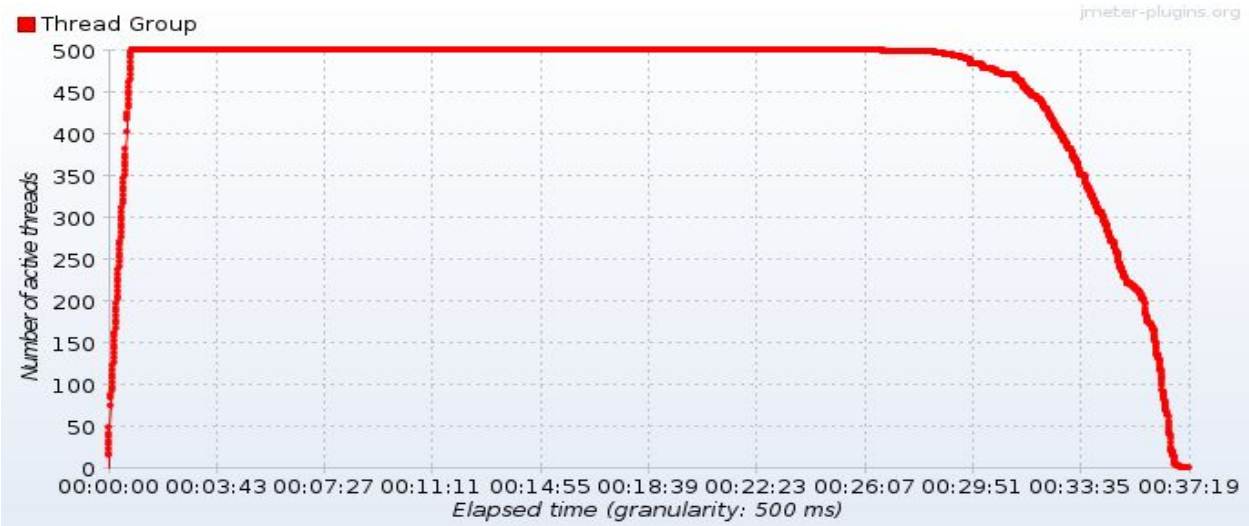


Figure 67. Active threads over elapsed time

- **Bytes throughput vs time**

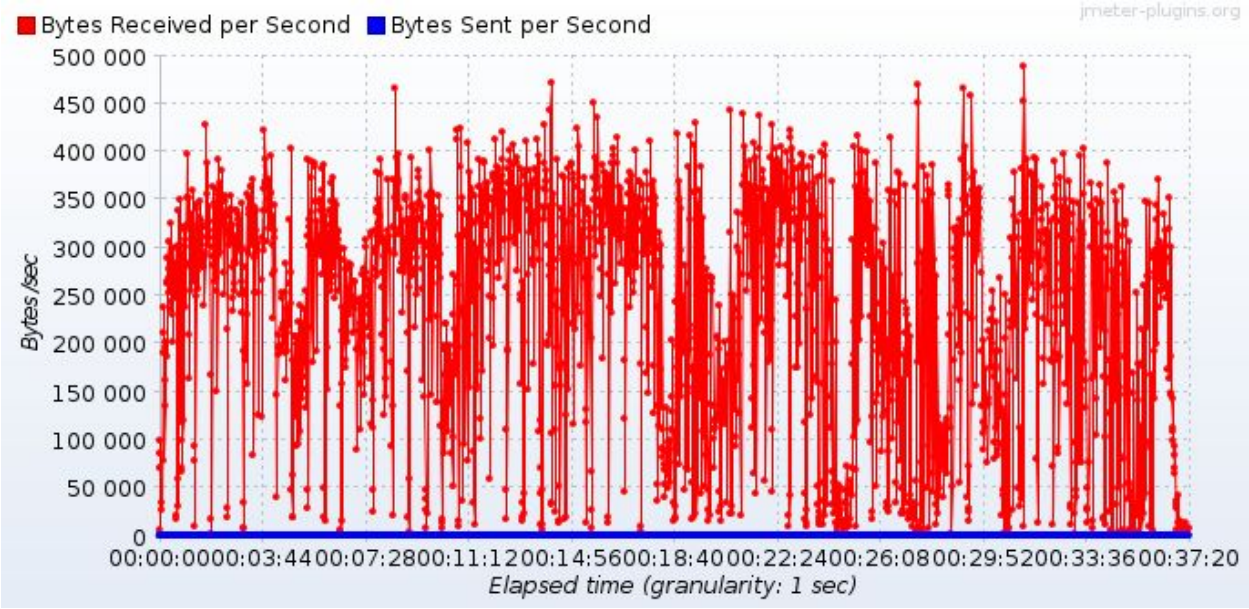


Figure 68. Bytes/sec over elapsed time

- **Response Latency**

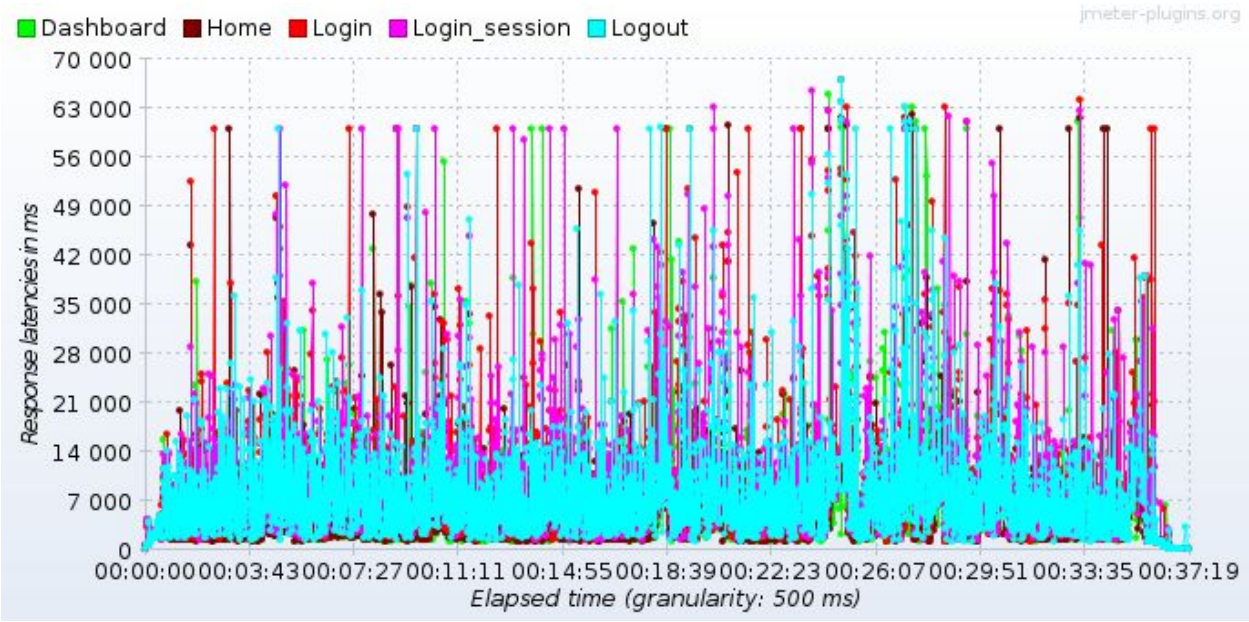


Figure 69. Response latencies over elapsed time

- **Transaction Throughput**

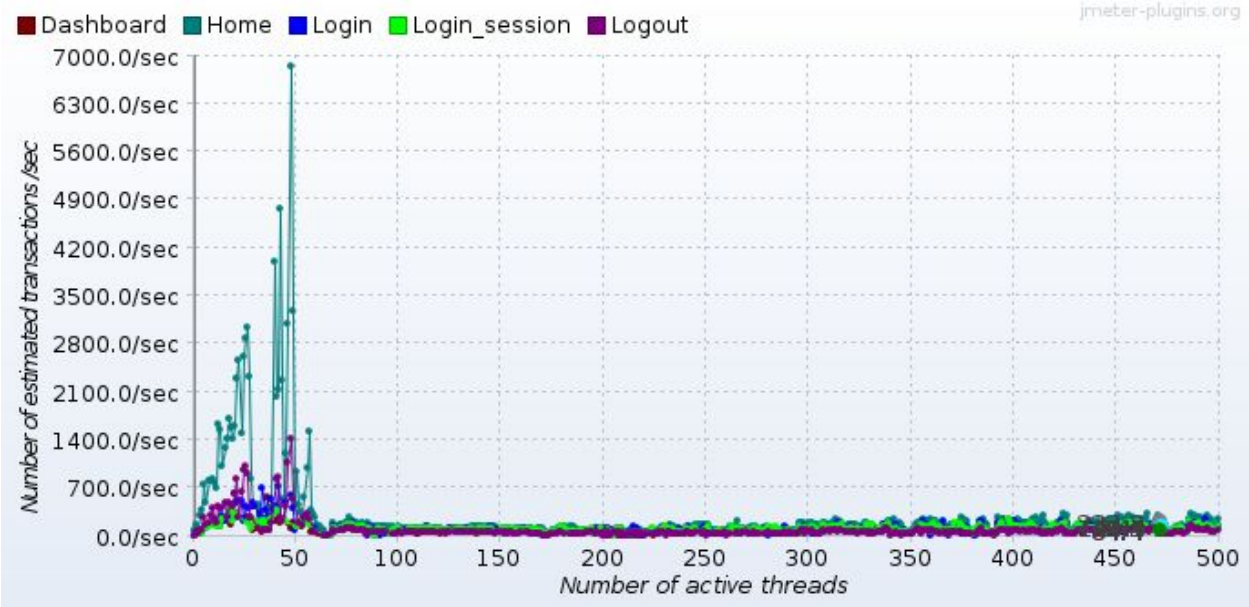


Figure 70. Estimated transactions over active thread

10. Detailed Analysis

10.1 Table for Insert Query Specifications

	InnoDB	RocksDB
Bytes Throughput	24500 Bytes/sec	28000 Bytes/sec
Response Latencies	500 ms	390 ms
Transaction Throughput	110517 Transaction/Thread	175525 Transaction/Thread
Transactions per second	2150 Transaction/sec	2700 Transaction/sec

10.2 Table for Update Query Specifications

	InnoDB	RocksDB
Bytes Throughput	110 Bytes/sec	110 Bytes/sec
Response Latencies	3 ms	3-4 ms
Transaction Throughput	649 Transaction/Thread	614 Transaction/Thread
Transactions per second	10 Transaction/sec	10 Transaction/sec

10.3 Table for Select Query Specifications

	InnoDB	RocksDB
Bytes Throughput	**27000**	**27000**
Response Latencies	**6-8**	**7-8**
Transaction Throughput	**440871 Transactions /Thread**	**436169 Transaction/Thread**
Transactions per second	**4700 Transaction/second**	**4500 Transaction/second**

10.4 Table for Delete Query Specifications

	InnoDB	RocksDB
Bytes Throughput	**110 Bytes/sec**	**110 Bytes/sec**
Response Latencies	**3 ms**	**2 ms**
Transaction Throughput	**630 Transaction/Thread**	**690 Transaction/Thread**
Transactions per second	**10 Transaction/sec**	**10 Transaction/sec**

Summary

- In case of insert operation,RocksDB gives 15% better bytes throughput as compared to InnoDB. In rest of the cases results are quite comparable.
- Response Latencies are low for RocksDB in case of insert and delete operations and high for select query as compared to InnoDB.

- Transaction Throughput of RocksDB is 37% more than InnoDB in case of insert operation and 10% more in case of delete operation.
- Transaction Throughput is about 5% higher for InnoDB in case of select and update queries.
- Transactions per second is somewhat 20% higher for RocksDB in case of insert operation.
- Rest all operations are quite comparable in case of InnoDB and RocksDB.

Foreign key support for RocksDB

To check for the support of Foreign Key, we created two tables, “Customer” and “Contact”, with “*id*” field of customer table as the foreign key in contact table. The fact that MyRocks did not support foreign key constraint was proved by the table.

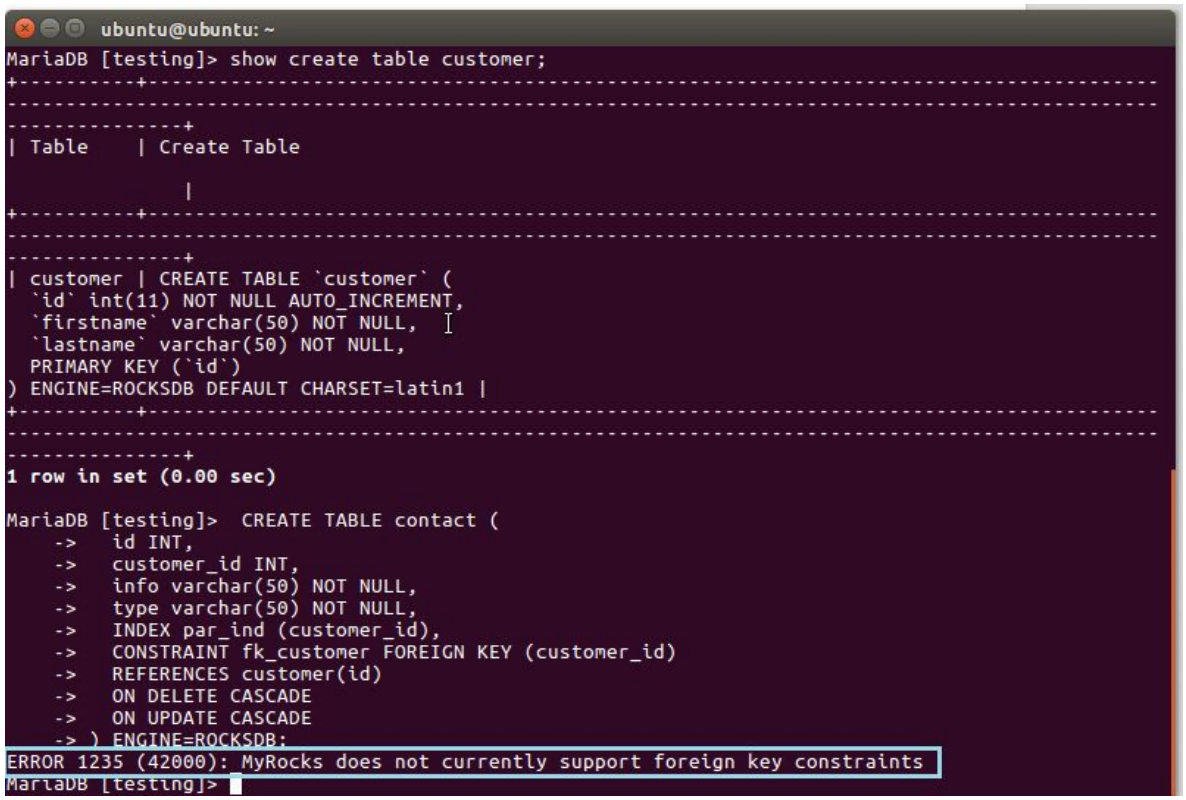


Figure 71. Check foreign key support in RocksDB

Database Compression Reports

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ sudo mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 10.2.6-MariaDB-10.2.6+maria~xenial-log mariadb.org binary distri
bution

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> select table_schema, sum((data_length+index_length)/1024/1024)
as MB from information_schema.tables group by 1;
+--------------------+---------------+
| table_schema       | MB            |
+--------------------+---------------+
| information_schema | 0.17187500    |
| mysql              | 0.64846515    |
| performance_schema | 0.00000000    |
| testing            | 0.00000000    |
| University         | 4540.54687500 |
| Universityrocks    | 1361.66603660 |
+--------------------+---------------+
6 rows in set (0.01 sec)

MariaDB [(none)]>
```

Figure 72. Compression Statistics

```
ubuntu@ubuntu: ~
as MB from informatio_schema.tables group by 1;
ERROR 1146 (42S02): Table 'informatio_schema.tables' doesn't exist
MariaDB [(none)]> select table_schema, sum((data_length+index_length)/1024/1024)
as MB from information_schema.tables group by 1;
+--------------------+---------------+
| table_schema       | MB            |
+--------------------+---------------+
| analytics-api      | 0.29724312    |
| dashboard          | 0.53357506    |
| ecommerce          | 5.93750000    |
| edxapp             | 124.12500000  |
| edxapp_csmh        | 0.03125000    |
| information_schema | 0.17187500    |
| mysql              | 2.44843198    |
| performance_schema | 0.00000000    |
| programs           | 0.81250000    |
| testing            | 0.00000000    |
| University         | 4611.51562500 |
| Universityrocks    | 1254.96938515 |
| xqueue             | 0.39062500    |
+--------------------+---------------+
13 rows in set (0.92 sec)

MariaDB [(none)]>
```

11. Challenges Faced

1. Issues with Hardware Virtualization in OpenEdX

There was an error while executing `VBoxManage`, a CLI used by Vagrant for controlling VirtualBox. The command and stderr is shown below.

Command: ["startvm", "e87b69f2-1f7c-4a81-a061-98c32e9173a2", "--type", "headless"]

Stderr: VBoxManage: error: VT-x is not available (VERR_VMX_NO_VMX)

VBoxManage: error: Details: code NS_ERROR_FAILURE (0x80004005), component Console Wrap, interface IConsole

This issue arises when Hardware Virtualization is Disabled in the machine. This can be resolved by enabling hardware virtualization in the BIOS setting. If running inside a Virtual Machine, kindly have a look at the following link https://forum.proxmox.com/threads/nested-virtualization.25996/ .

2. Issues with MySQL

- Error: 1045 SQLSTATE: 28000 (ER_ACCESS_DENIED_ERROR)

Message: Access denied for user '%s'@'%s' (using password: %s)

This issue arises due to incorrect installation of MySQL. This may be resolved by following this link :
https://stackoverflow.com/questions/489119/mysql-error-1045-access-denied

- ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock' (2)

This issue arises due to incorrect starting up of the MySQL server. This issue may be resolved by following this link:

https://ubuntuforums.org/showthread.php?t=804021

- Error: 1203 SQLSTATE: 42000

  (ER_TOO_MANY_USER_CONNECTIONS)

  Message: User %s already has more than 'max_user_connections' active connections

This issue arises due to the maximum number of outgoing connections from MySQL getting exceeded. This may be resolved by typing

"*SET GLOBAL max_connections=1024" in the MySQL console.*

If the error is still not resolved, then use the following link: https://www.digitalocean.com/community/questions/max_connections-will-not-change-in-ubuntu .

3. Issues with JMeter

- ERROR - jmeter.JMeter: Uncaught exception:

  java.lang.OutOfMemoryError: unable to create new native thread

This issue arises when the Heap Size of JMeter exceed the allotted memory size. This can be be resolved by following the given Instructions :

https://stackoverflow.com/questions/2286750/jmeter-outofmemoryerror

- ERROR 403 Forbidden

Enable Cookie Manager and Cache Manager and make the given changes to JMeter code.

- ERROR 502 Forbidden

Add the following lines to the *user.properties* file:

httpclient4.retryCount=1;

hc.parameters.file=hc.parameters

Add the following lines to the *hc.parameters* file

http.connection.stalecheck$Boolean=true

12. Future Scope

1. Currently, RocksDB does not have any support for **Foreign Key, Fulltext Search or Spatial Search.** OpenEdX databases implements foreign keys for checking the integrity of the data, something that is presently not possible using the current database schema for RocksDB
2. SAVEPOINT is not supported by RocksDB, thus it is difficult to process OLTP. This may be implemented in the future versions of RocksDB
3. ORDER BY DESC or ASC is slow. This is because of "Prefix Key Encoding" feature in RocksDB.

Since this is a relatively new database engine, it has its own inherent flaws. But, with a steadily increasing developer base, improvements and enhancements will take place in the near future. As a result, there may be scope, in the future, to use RocksDB as an alternative to InnoDB for all kinds of Database Tasks.

Currently, it is not possible to implement RocksDB as a suitable database engine for OpenEdX, although RocksDB provides marginally better utility in case of Insertions and Querying, when using relatively large databases.

13. References

- MySQL - Wikipedia (https://en.wikipedia.org/wiki/MySQL)
- MySQL Components (https://actimem.com/mysql/mysql-components/)
- MySQL 5.7 Reference Manual (https://dev.mysql.com/doc/refman/5.7/en/)
- RocksDB | A persistent key-value store(www.rocksdb.org)
- RocksDB Github (https://github.com/facebook/rocksdb)
- MariaDB Documentation (https://mariadb.com/kb/en/mariadb/documentation/)
- Optimizing Space Amplification in RocksDB - CIDR (http://cidrdb.org/cidr2017/papers/p82-dong-cidr17.pdf)
- LevelDB Github (http://leveldb.org)
- P. Wang, G. Sun, S. Jiang, J. Ouyang, S. Lin, C. Zhang, and J. Cong, *"An efficient design and implementation of LSM-tree based key-value store on open-channel SSD,"* in Proc. *9th European Conf. on Computer Systems. ACM, 2014, p. 16*.
- R. Sears and R. Ramakrishnan, *" b*LSM*: A general purpose log structured merge tree,"* in Proc. 2012 ACM SIGMOD Intl. Conf. on Management of Data, 2012, pp. 217–228
- E. H. Hallili, *Apache JMeter: A practical beginner's guide to automated testing and performance measurement for your websites.* Packt Publishing Ltd., 2008.
- Apache Jmeter User Manual (http:/jmeter.apache.org/usermanual/get-started.html)
- https://stackoverflow.com/
- https://askubuntu.com/
- The Log-Structured Merge-Tree (LSM-Tree) by Patrick O'Neil, Edward Cheng,Dieter Gawlick, Elizabeth O'Neil.