
Analyzing Non Functional aspect of transactions, soft lock, hard lock over normal Read/Write Operations in different data stores

BITS ZG628T: Dissertation Final Submission

by

Saurabh Araiya

2017HT12225

Dissertation work carried out at

Flipkart, Bangalore



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)**

February 2019

Analyzing Non Functional aspect of transactions, soft lock, hard lock over normal Read/Write Operations in different data stores

BITS ZG628T: Dissertation Final Submission

by

Saurabh Araiya

2017HT12225

Dissertation work to be carried out at

Flipkart, Bangalore

Under the Supervision of
Vishwajith Bharadwaj, SDE-3
Flipkart Bangalore



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)**

April, 2019

**Birla Institute of Technology & Science, Pilani
Work-Integrated Learning Programmes Division
Second Semester 2018-2019**

BITS ZG628T : Dissertation End Sem Submission

ID No. : 2017HT12225

NAME OF THE STUDENT : Saurabh Araiya

EMAIL ADDRESS : araiyaurabh@gmail.com

**STUDENT'S EMPLOYING
ORGANIZATION & LOCATION** : Flipkart

SUPERVISOR'S NAME : Vishwajith Bharadwaj

**SUPERVISOR'S EMPLOYING
ORGANIZATION & LOCATION** : Flipkart

SUPERVISOR'S EMAIL ADDRESS : vishwajith.b@flipkart.com

DISSERTATION TITLE : Analyzing Non Functional aspect of
transactions, soft lock, hard lock over normal Read/Write Operations in different data stores

Acknowledgements:

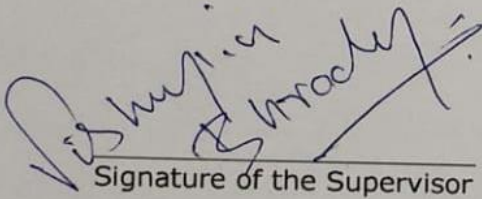
I would like to thank BITS Pilani and Faculty mentors provided by BITS Pilani WILP division and Prof R Gururaj for giving me this opportunity to work on this formal project which may help the readers in decision making for making the right choice of Database Management System. Without their input, motivation and guidance, the project would not have been possible at all.

I would like to thank my organization for providing me with an opportunity of working on multiple data stores and informed colleagues who are able to help me in case of any roadblocks.

I would like to thank Arun Thulsidharan, Vishwajith Bharadwaj for helping in shaping this progress report of dissertation work, without their critical feedback and guidance this would not be possible

CERTIFICATE

This is to certify that the Dissertation entitled **Analyzing Non Functional aspect of transactions, soft lock, hard lock over normal Read/Write Operations in different data stores** and submitted by **Saurabh Araiya** having ID-No. **2017HT12225** for the partial fulfillment of the requirements of M.Tech. Software Systems degree of BITS, embodies the bonafide work done by him/her under my supervision.



Signature of the Supervisor

Place : Bangalore

Date : 09/04/2019

Name, Designation and Location: Vishwajith Bharadwaj, SDE-3, Flipkart



BITS
PILANI

Table of Contents

1.Chapter 1: Background	1
2.Chapter 2: Objective	2
3.Chapter 3: Scope of Work	3
4. Chapter 4: Main Text	4
4.1 Reason for Selecting	4
4.1.1 Different MySQL Engines (Table 1)	5
4.2 Test Bench Description:	7
4.3 Trigger Interface:	7
4.4 Technologies used:	8
4.5 Platform:	8
4.5 General Configuration used for Test:	8
4.6 Algorithms for load generation:	9
5. Observations.....	10
5.1 MySQL Test (Table 2-5).....	10
5.1.1 InnoDB.....	10
5.1.2 MyISAM.....	11
5.2 RMQ Test (Table6-9).....	11-12
5.3 Elasticsearch (Table 10-11).....	13
6. Conclusion	13
7. Literature References.....	14

Abstract:

Today's industry is ruled by data, and how we handle data. There are battles over data and whole companies built around it. In layman's terms data stores can be classified into two categories: OLTP and OLAP. Before we jump on to any database choices, we need to skin down our requirements and look for the specific set of features we may be needing to be supported by a data store. For example a data store like redis can provide thousands of read per second, but it would not be useful when we try to make reservation systems with it. On the other hand, MySQL may provide very accurate transaction processing.

This work is mostly around getting a set of data to help us decide on what to choose when.

Technical Keywords: Java, Elasticsearch, ORM, Hibernate, Dropwizard, REST, NFR, Load Test

Chapter 1: Background:

- a. As per the definition, Non-Functional attributes are the quality attributes of a system. One of the quality attributes is queries per second we are able to execute on a given infrastructure.
- b. Transactions are one of the most important operations which we perform on a data store. And because of the constraints, particularly ACID properties, in case of a single system, problems are not so difficult. But when we consider distributed systems, another level of problem comes.

There are many data stores which natively support transaction like innodb engine of MySQL, transaction channel RMQ etc (however in RMQ, full ACID properties are not supported as of date). And some very widely used data stores like HBase doesn't even support transaction properties. So using a distributed locking

Considering importance of transaction, it is very important to compare different data stores in order to trigger the appropriate decision-making metrics.

Chapter 2: Objective:

Main objective of this activity is to explore the limiting factor (in terms of system resources like CPU, Memory) in three scenarios:

- a. Normal Read/Write operations
- b. Read/Write operations with locking. Locking can be of two types:
 - i. Hard lock, where if a row is locked then it is not available for reading as well
 - ii. Soft lock, a lock doesn't block read access to the data store

To achieve the above goals, there would be a need to automate, learn interactions with different data stores, and explore workarounds to enable some properties of a transaction in data stores which do not support it natively. Also since with the increase in scale of systems, distributed transactions also come into picture, I would attempt to simulate and try to tackle those problems to some extent.

Chapter 3: Scope of Work:

This would be limited to automation of load test bench in form of Java microservices, metrics collection, information gathering from load test result and finding a logical relation among throttling factors for our experiment.

This applications covers the applications and data stores in breadth with shallow load testing. The application is written in a way that complex queries could be supported with minimal changes to current code.

Github Link: <https://github.com/saurabhsar/dissertation/>

Chapter 4: Main Text:

1. Choosing the data stores to test, learning interactions with them and automate the load generation process to verify
 - a. The work is based on automating load generation on MySQL, RabbitMQ, Elasticsearch

Reasons for selecting:

1. MySQL: It is the most popular data store used currently. It provides many kinds of engine for different use cases.
Those are explained separately
2. RabbitMQ: This is a distributed Queue.
3. Elasticsearch: This is a no-SQL document store it supports REST based queries, This is written on top of Apache Lucene. Currently it is one of the most popular document based free data store in the industry.

I tried to chose the data stores which describe cover a wide spectrum of use cases

2. Exploring the transaction aspect of those data stores, if they do not support transaction then try to find workaround for them. If partial support for ACID properties are there, create a test bench to test those partial properties.

- a. Of the above stores, Only MySQL supports full ACID property in one engine.
 - b. RabbitMQ supports transaction channel, by this it means that in case of multiple queue publish it would help. But it doesn't support basic ACID properties.
 - c. Elasticsearch is primarily designed for search based purposes so this doesn't support any ACID properties.
- 3. Find, compare the data with and without transaction
 - 4. If possible, try to simulate above problems in different environment like SSD, Rotating Disk etc. (If possible because some data stores like aerospike are meant only for SSD and do not support rotating disk)
 - 5. For distributed transaction, try to simulate a failure scenario and if possible implement a workaround

Observe and summarize so that it can be helpful for a proper decision making matrix.

1.a.1: Different MySQL Engines

1.a.1.1: **InnoDB**: This is the default storage engine in most versions of MySQL. InnoDB is a (ACID compliant) data store. Operations supported by InnoDB are commit, rollback, and crash-recovery capabilities. InnoDB supports row-level locking and Oracle-style consistent nonlocking reads which

helps increase in multi-user concurrency and performance. InnoDB stores user data in clustered indexes which helps in reducing I/O for common queries based on primary keys. InnoDB also supports FOREIGN KEY referential-integrity constraints which helps in maintaining data integrity.

1.a.1.2: **MyISAM:** These tables have a small memory footprint as compared to InnoDB. This engine is preferred in read-only or read-heavy workloads in Web and data warehousing configurations. Table lock, which is supported by MyISAM prevents transactions from accessing a particular table.

1.a.1.3: **Memory:** This engine stores all the data in RAM for fast access in environments needing quick lookups of non-critical data. Earlier, this engine was known as HEAP engine.

1.a.1.4: **CSV:** They are text files with comma-separated values. CSV tables let us import or dump data in CSV format. Since CSV tables are not indexed, it is a general practice to keep in InnoDB tables during normal operation, and only using CSV tables during the import or export stage.

1.a.1.5: **Blackhole:** This engine accepts the data but does not store, its behavior is similar to the *nix /dev/null device. Since no data is stored, search

queries would always return an empty set. These tables are generally used in replication with query-based replication where DML statements are sent to slave servers.

There are many other engines like Archive, NDB, Merge, Federated, Example

Table 1: As a summary, source[Table 16.1, <https://dev.mysql.com/doc/>]

Feature	MyISAM	Memory	InnoDB	Archive
<i>B-tree indexes</i>	Yes	Yes	Yes	No
<i>Backup/point-in-time recovery</i>	Yes	Yes	Yes	Yes
<i>Cluster database support</i>	No	No	No	No
<i>Clustered indexes</i>	No	No	Yes	No
<i>Compressed data</i>	Yes	No	Yes	Yes
<i>Data caches</i>	No	N/A	Yes	No
<i>Encrypted data</i>	Yes	Yes	Yes	Yes
<i>Foreign key support</i>	No	No	Yes	No
<i>Full-text search indexes</i>	Yes	No	Yes (note 6)	No
<i>Geospatial data type support</i>	Yes	No	Yes	Yes
<i>Geospatial indexing support</i>	Yes	No	Yes (note 7)	No
<i>Hash indexes</i>	No	Yes	No (note 8)	No
<i>Index caches</i>	Yes	N/A	Yes	No
<i>Locking granularity</i>	Table	Table	Row	Row
<i>MVCC</i>	No	No	Yes	No

<i>Replication support</i>	Yes	Limited (note 9)	Yes	Yes
<i>Storage limits</i>	256TB	RAM	64TB	None
<i>T-tree indexes</i>	No	No	No	No
<i>Transactions</i>	No	No	Yes	No
<i>Update statistics for data dictionary</i>	Yes	Yes	Yes	Yes

Test Bench Description:

All the tech stacks/design patterns used in the development of this test bench are generally followed in industry. This bench is made in such a manner so that new databases are easily onboardable. And if needed, this can be used to benchmark different Databases as per the use case.

Trigger Interface: REST API's

Technologies used: Java, Dropwizard, Hibernate for ORM Based connection, Jersey Rest API's

Platform: Mac OS X

General Configuration used for test: threads: 5, test ran for 5 seconds

Sample Request curl for triggering test:

```
curl -i -X POST \
  -H "Content-Type:application/json" \
  -d '{ "threads" : 5, "load" : 1, "timeInMilis" : 1000,
"transactional" : false, "durable" : true, "requestType":
"READ"}' \
  'http://localhost:1730/test/gen-load/mysql'
```

Algorithms for load generation:

For RabbitMQ and Elasticsearch:

Depending on the request, desired number of threads are created and then it hits data-store parallelly. Since RabbitMQ supports transaction channels, the request has a boolean value isTransactional and tries to write to queues on the basis of this configuration.

For MySQL:

Hibernate ORM of MySQL works on SessionContext. So every request which comes to the application is bound by sessionContext. And all the requests would be written to database on successful response from the application. So this would be effectively giving the first phase of commit which has happened in ORM. The Second phase commit (actual commit to DB) which is costly and involves disk operation would be delayed, hence not giving the

proper number.

To counter the above problem, the load generator, instead of calling MySQL directly, it calls an internal API which is bound by SessionContext (for transactional capabilities) and writes to MySQL on a successful return.

The observations are marked in yammer metrics.

Observations:

Table 2: MySQL Write: (InnoDB)

Test #	Time (in ms)	versioned	Count of operation
1	5005	No	3680
2	5003	No	3318
3	5000	Yes	4705
4	5008	Yes	4158

Table 3: MySQL Read: (InnoDB)

Test #	Time (in ms)	versioned	Count of operation
1	1005	No	2599
2	1003	No	2621
3	1003	Yes	2253
4	1004	Yes	2104

Table 4: MySQL Write: (MyISAM)

Test #	Time (in ms)	versioned	Count of operation
1	5000	No	5046
2	5001	No	4935
3	5000	Yes	5602
4	5001	Yes	5274

Table 5: MySQL Read: (MyISAM)

Test #	Time (in ms)	versioned	Count of operation
1	5005	No	3512
2	5003	No	3204
3	1003	Yes	2451
4	1004	Yes	2395

Table 6: RMQ READ (Without disk persistence):

Test #	Time (in ms)	transactional	Count of operation
1	10001	No	33274
2	10905	No	40901

Table 7: RMQ Write (Without disk persisted messages):

Test #	Time in ms	transactional	Count of operation
1	10010	No	323,419
2	10009	No	315,017
3	10100	Yes	17,336
4	10054	Yes	18,547

Table 8: RMQ Write (With disk persistence):

Test #	Time (in ms)	transactional	Count of operation
1	10200	No	153,964
2	10108	No	156,398
3	10093	Yes	354
4	10110	Yes	392

Table 9: RMQ Read (disk persisted):

Test #	Time (in ms)	transactional	Count of operation
1	10001	No	30560
2	10905	No	31294

Table 10: Elasticsearch Write: (With HTTP Client)

Test #	Time (in ms)	Count of operation
1	10007	689
2	10041	676
3	10105	678
4	20041	1303

Table 11: Elasticsearch Read: (With HTTP Client)

Test #	Time (in ms)	Count of operation
1	1004	72
2	1001	70
3	1010	82
4	1031	78

Conclusion and Direction of Future work

The main focus of this project was to enable the load testing of different databases in different scenarios. Currently this is focused on insertion and lookup. As the code has been open sourced on github, we can add more databases and more scenarios.

Looking into the data, wherever durability was involved there was a clear difference in normal query per second and durable query per second. In case of MySQL, although the engines are made for different purposes, but at the small scale of test which I was performing showed very minimal difference, so for small use cases having a MySQL would not hurt and transaction guarantees would be a bonus. For me, the strangest part was getting low qps from Elasticsearch. Elasticsearch is designed for environments needing different kinds of querability support like range based, absolute etc. But in the test environment it was outperformed by all the databases, even MySQL. For testing it's true potential, an appropriate distributed environment may be required with multiple shards and nodes. Also range of data was limited to some thousands which might have led to hotspotting.

As this is very relevant to my work environment, I would be maturing the test bench to at least one-two more databases and introduce the query involving altering current data for currently onboarded databases.

Literature References:

1. <https://dev.mysql.com/doc/refman/8.0/en/>
2. <https://www.rabbitmq.com/semantics.html>
3. <https://www.aerospike.com/docs/architecture/acid.html>
4. https://en.wikipedia.org/wiki/Distributed_transaction
5. <https://lucidworks.com/2013/08/23/understanding-transaction-logs-softcommit-and-commit-in-sorlcloud/>
6. <https://docs.oracle.com/javacomponents/jmc-5-4/jfr-runtime-guide/about.htm#JFRUH170>
7. [https://en.wikipedia.org/wiki/ACID_\(computer_science\)](https://en.wikipedia.org/wiki/ACID_(computer_science))
8. https://en.wikipedia.org/wiki/Non-functional_requirement
9. <https://dzone.com/>
10. <https://stackoverflow.com/>
11. <https://docs.oracle.com/javase/7/docs/technotes/guides/management/jconsole.html>
12. <https://www.dropwizard.io/1.3.9/docs/>
13. <https://hibernate.org/orm/>
14. <https://www.elastic.co/>
15. <https://www.elastic.co/guide/index.html>

Particulars of Supervisor and Examiner

	Supervisor	Examiner
Name	Vishwajith Bharadwaj	Arun Thulsidharan
Qualification	B-Tech	B-Tech
Designation	SDE-3	SDE-3
Employing Organization and Location	Flipkart, Bangalore	Hevo, Bangalore
Phone No	8861982583	9632244339
Email address	vishwajith.b@flipkart.com	arun@hevodata.com
Date	08/04/2019	08/04/2019

This checklist is to be attached as the last page of the report.

This Checklist is to be duly completed, verified and signed by the student.		
1.	Is the final report neatly formatted with all the elements required for a technical Report?	Yes / No ✓
2.	Is the Cover page in proper format as given in Annexure A?	Yes / No ✓
3.	Is the Title page (Inner cover page) in proper format?	Yes / No ✓
4.	(a) Is the Certificate from the Supervisor in proper format? (b) Has it been signed by the Supervisor?	Yes / No ✓ Yes / No ✓
5.	Is the Abstract included in the report properly written within one page? Have the technical keywords been specified properly?	Yes / No ✓ Yes / No ✓
6.	Is the title of your report appropriate? The title should be adequately descriptive, precise and must reflect scope of the actual work done. Uncommon abbreviations / Acronyms should not be used in the title	Yes / No ✓
7.	Have you included the List of abbreviations / Acronyms?	Yes / No ✓
8.	Does the Report contain a summary of the literature survey?	Yes / No ✓
9.	Does the Table of Contents include page numbers? (i). Are the Pages numbered properly? (Ch. 1 should start on Page # 1) (ii). Are the Figures numbered properly? (Figure Numbers and Figure Titles should be at the bottom of the figures) (iii). Are the Tables numbered properly? (Table Numbers and Table Titles should be at the top of the tables) (iv). Are the Captions for the Figures and Tables proper? (v). Are the Appendices numbered properly? Are their titles appropriate	Yes / No ✓ Yes / No ✓ N/A Yes / No ✓ Yes / No N/A Yes / No ✓
10.	Is the conclusion of the Report based on discussion of the work?	Yes / No ✓
11.	Are References or Bibliography given at the end of the Report? Have the References been cited properly inside the text of the Report? Are all the references cited in the body of the report	Yes / No ✓ N/A Yes / No ✓ Yes / No ✓
12.	Is the report format and content according to the guidelines? The report should not be a mere printout of a Power Point Presentation, or a user manual. Source code of software need not be included in the report.	Yes / No ✓

I certify that I have properly verified all the items in this checklist and ensure that the report is in proper format as specified in the course handout.

Signature of the Student

Name: Saurabh Araiya