School of Computer Science

COMP 47470

Project 1

Performance Testing of Two Different Database Management Systems

Name -	Kamran Azmat
Student Number -	1620491

Introduction

MySQL and MongoDB represent two sides of an argument that has been raging recently concerning data storage – the tried and tested relational database vs. non-relational or NoSQL database. They are both open-source products distributed under a version of the GNU GPL, and both are also available as commercial versions offering many more features and corporate support.

Relational database management systems took a leadership and for years the choice was obvious, either **MySQL**, PostreSQL or Oracle. MySQL is one world's most popular open source database developed, distributed and supported by Oracle Corporation.

As the challenge faced by various organizations - to handle such a large dataset that relational data stores often become a bottleneck, thus imposing the limits on overall system scalability. Hence the industry was actively looking into a cheaper ways to build complex distributed systems, making use of horizontal scalability instead. Than the growth of NoSQL has begun. **MongoDB** is an open-source NoSQL database developed by MongoDB, Inc. MongoDB stores data in JSON-like (BSON) documents that can vary in structure.

This project is an attempt to design and run some performance tests on the two different database systems.

For relative performance test, we will be using IMDB's database. It's is not huge but still contains millions of films, series and individuals. This will be enough to test the (relative) performance of different platforms and tools.

Requirements

- Extract **IMDB**'s dataset and populate MySQL database using tools like JMDB or **IMDbPY**
- Write a script (in any language) to extract data from MySQL database and then insert it into **MongoDB**.
- Use any open source testing tools for performance testing of the two database systems. (eg. Apache **JMeter**)

Architecture and Design

I used **IMDbPY** API to scrape the IMDB dataset. In this API, there is a script - **imdbpy2sql.py** which converts IMDB movie dataset to a MySQL database. This populated **21** tables with '**id**' as primary key for every table.

Tables:

aka_name, aka_title, cast_info, char_name, comp_cast_type, company_name, company_type, complete_cast, info_type, keyword, kind_type, link_type, movie_companies, movie_info, movie_info_idx, movie_keyword, movie_link, name, person_info, role_type, title

The EER diagram generated is:

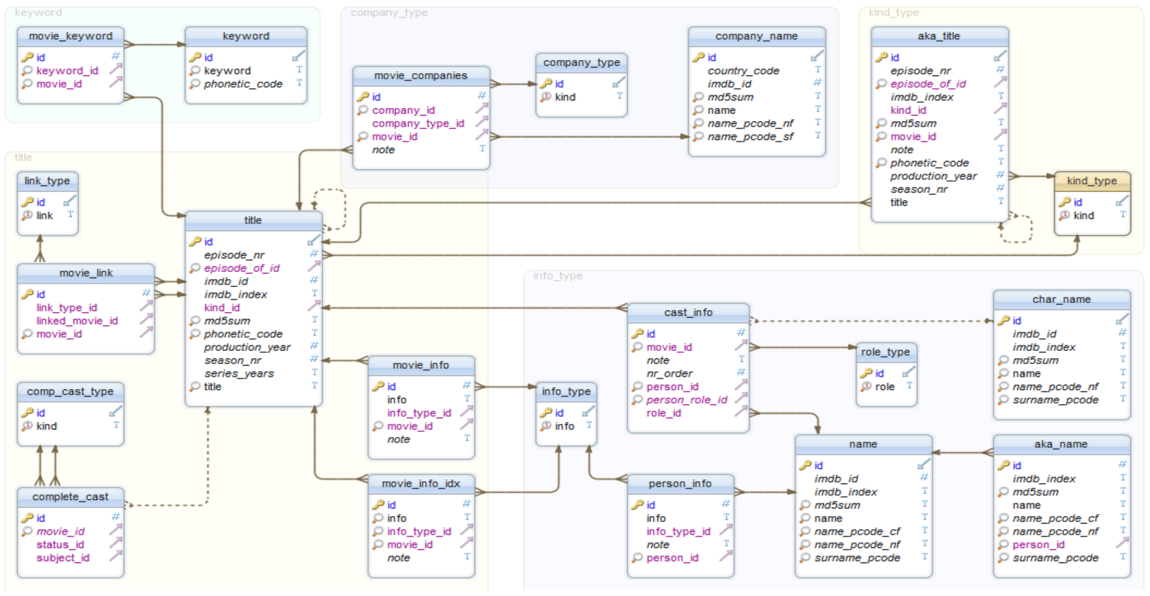
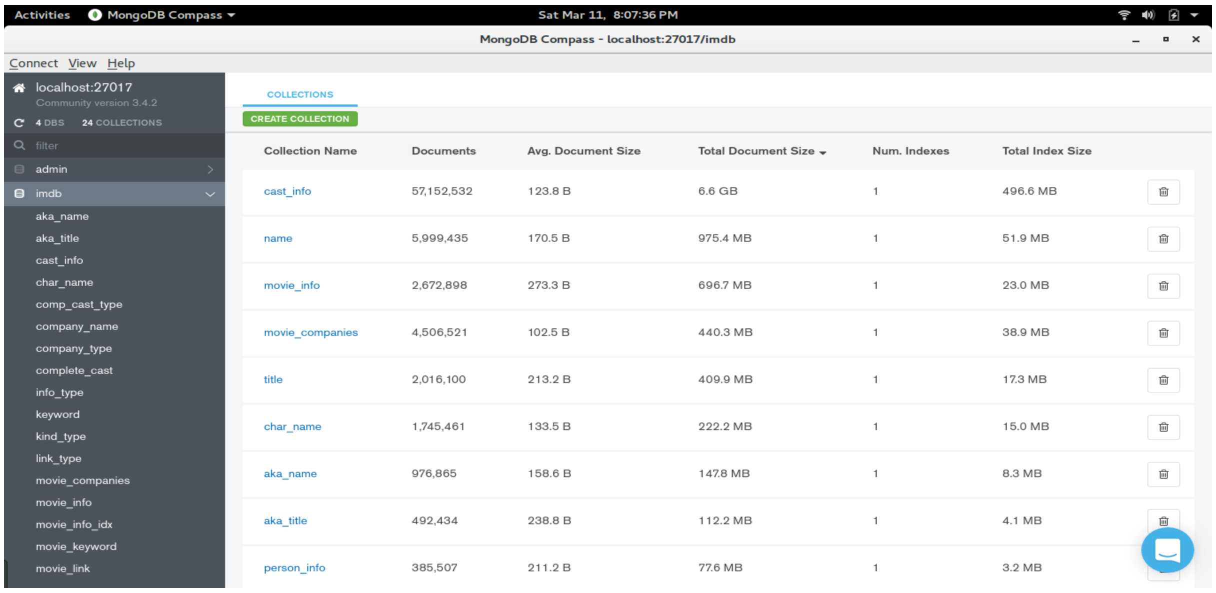


Fig 1: Entity Relation Diagram

Next, I wrote a python script to populate MongoDB. Below is the screen shot of all the collections in MongoDB. 21 tables in MySQL and 21 collections in MongoDB. MongoDB doesn't support join hence, I decided to convert all the tables into different collections.



Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size
cast_info	57,152,532	123.8 B	6.6 GB	1	496.6 MB
name	5,999,435	170.5 B	975.4 MB	1	51.9 MB
movie_info	2,672,898	273.3 B	696.7 MB	1	23.0 MB
movie_companies	4,506,521	102.5 B	440.3 MB	1	38.9 MB
title	2,016,100	213.2 B	409.9 MB	1	17.3 MB
char_name	1,745,461	133.5 B	222.2 MB	1	15.0 MB
aka_name	976,865	158.6 B	147.8 MB	1	8.3 MB
aka_title	492,434	238.8 B	112.2 MB	1	4.1 MB
person_info	385,507	211.2 B	77.6 MB	1	3.2 MB

Fig 2: Graphical view of MongoDB schema using MongoDB Compass

To test the performance, I can easily retrieve documents from both MongoDB and MySQL. E.g.

```
SELECT * FROM movie_info;
db.movie_info.find().pretty();
```

Project Timeline and Challenges

Report: 7th March 2017

- mysql and connection done using python (7th), downloaded all the IMDB dataset using **wget** command.
- faced problem during json.dumps, then enoded to 'latin-1'
- tried inserting data to mongodb, successful !!!
- next automating the script

Report: 9th March 2017

- I can make the primary key of a table as '_id' in mongodb but the problem will arise when there are more than one column as primary key
- making all 'null' value as 0 (even if it is a string)
- using mongoDB compass to get better visualisation of the database
- all the tables have 'id' as primary key, so this can be made '_id'
- the current script failed, it took around 4-5 hours just to loop around 2 tables
- next I will try to run one by one for each table

Report: 11th March 2017

- SHOW VARIABLES LIKE 'connect_timeout'; --- it was 30 secs, I changed it to 240 secs
- then it started working (error: connection timeout problem !!)
- I tried to access the MySQL database and then retrived one table at a time. Then, I looped around all the row in the table and also inserted it to the MongoDB database at the same time. This was effective from previous attempt.
- table 'movie_info' took 23:22:07715 minutes to transfer data from MySQL database to MongoDB
- similarly, 'movie_company' took 37:12:689009 minutes
- 'cast_info' is left and this is the largest database, this may take upto 2-3 hours
- screenshot added 'mongoDB.png', showing the current status of MongoDB (using MongoDB Compass)

Report: 11th March 2017 8:10 PM

- the previous script didn't worked for 'cast_info'. So, I then insted of calling all the rows at the same time, I loop around and called 100000 rows evry time and then inserted into mongoDB one-by-one
- this took me around 7-8 hours
- now cleaning up
- next performance testing

For a in-depth picture of the report with all the relevant changes to the python script, please visit my GitHub repo: https://github.com/kamranazmat/UCD-Big-Data-Programming (this was private till 11th March)

Summarizing:

- Populating MySQL database: Data with millions of record is not easy to populate, it took around 12-13 hours to populate MySQL database. (thanks to developers of IMDbPY).

- MongoDB Schema design: Since MongoDB does not support any join, so I created same number of schemas. This becomes relevant for easy data access.
- Populating MongoDB Collection: For populating data into Mongodb , I wrote several versions of python script to achieve this task. First I tried to convert all the data to JSON format and then insert it into MongoDB but this didn't worked. I tried to automate the process but the process **killed** itself several times. I have to reboot several times because the system freezes. At last I manually ran 21 different scripts to populate 21 collections. The largest database took me around 7-8 hours to insert into MongoDB.

Performance Testing

Performance testing is done using an open source tool such as Apache JMeter. For this project, I tested two queries on both the database system and then analyzed the report.

Query 1: Count number of documents in table / collection in 'char_name'

```
SELECT COUNT(*) FROM char_name;
db.getCollection("char_name").count()
```

Thread Properties: (for both database system)
No of users – 100
Ramp up period(in secs) - 1
Loop count – 10

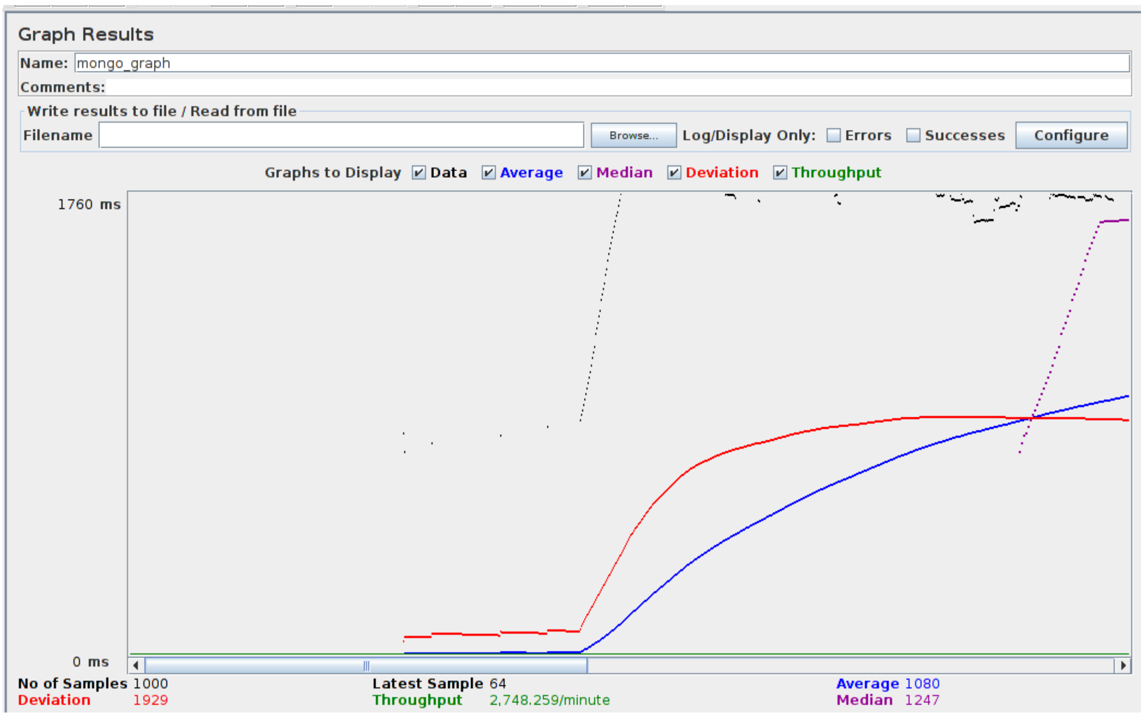
MongoDB report:

sampler_label	aggregate_report_count	average	aggregate_report_min	aggregate_report_max	aggregate_report_stddev	aggregate_report_error%	aggregate_report_rate	aggregate_report_bandwidth	average_bytes
MongoDB Script	1000	1075	0	20876	1897.1126410814	0.4	46.5809577045	1.5011441448	33
TOTAL	1000	1075	0	20876	1897.1126410814	0.4	46.5809577045	1.5011441448	33

MySQL report:

sampler_label	aggregate_report_count	average	aggregate_report_min	aggregate_report_max	aggregate_report_stddev	aggregate_report_error%	aggregate_report_rate	aggregate_report_bandwidth	average_bytes
JDBC Request	1000	10509	1122	12658	1540.0431311749	0.405	8.8520643014	0.3360153705	38.87
TOTAL	1000	10509	1122	12658	1540.0431311749	0.405	8.8520643014	0.3360153705	38.87

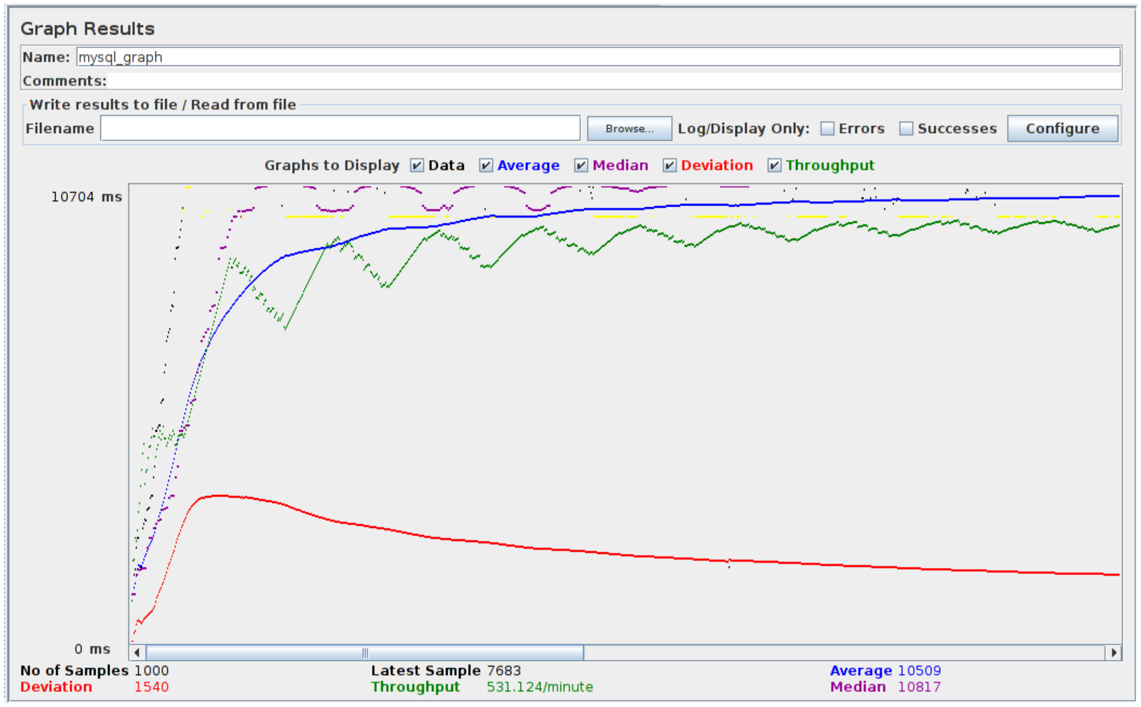
Mongo Graph:


Graph Results
Browse...
Configure
Graphs to Display ✓ Data ✓ Average ✓ Median ✓ Deviation ✓ Throughput
1760 ms
0 ms

Name: mongo_graph
Comments:
Write results to file / Read from file
Filename
Log/Display Only: Errors Successes

No of Samples 1000
Deviation 1929
Latest Sample 64
Throughput 2,748.259/minute
Average 1080
Median 1247

MySQL Graph:


Graph Results
Browse...
Configure
Graphs to Display ✓ Data ✓ Average ✓ Median ✓ Deviation ✓ Throughput
10704 ms
0 ms

Name: mysql_graph
Comments:
Write results to file / Read from file
Filename
Log/Display Only: Errors Successes

No of Samples 1000
Deviation 1540
Latest Sample 7683
Throughput 531.124/minute
Average 10509
Median 10817

Query 2: Show all the names with gender female

SELECT id, name, gender FROM name WHERE gender = 'f' LIMIT 1000;
db.name.find({gender: 'f'}, {'_id': 0, 'name': 1, 'gender': 1}).limit(10)

Thread Properties: (for both database system)
No of users – 100
Ramp up period(in secs) - 1
Loop count – 10

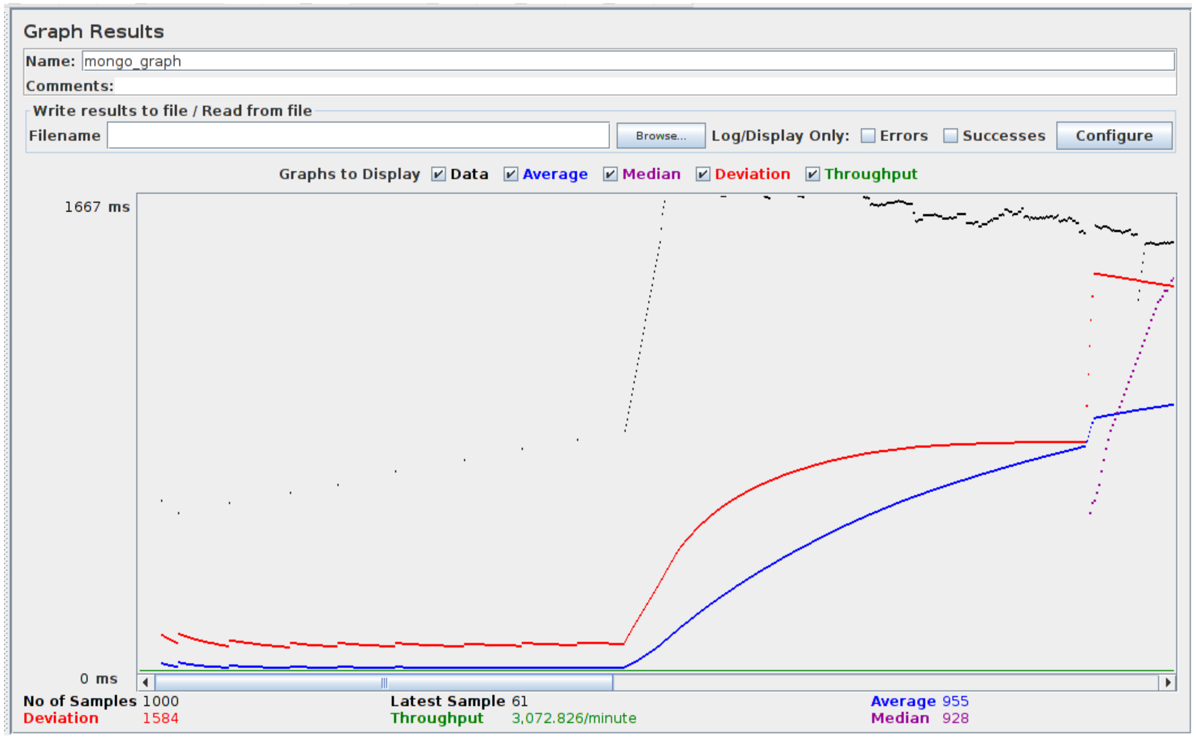
MongoDB report:

sampler_label	aggregate_report_count	average	aggregate_report_min	aggregate_report_max	aggregate_report_stddev	aggregate_report_error%	aggregate_report_rate	aggregate_report_bandwidth	average_bytes
MongoDB Script	1000	955	0	18946	1584.6305404652	0.42	51.2137662604	19.1441459464	382.78
TOTAL	1000	955	0	18946	1584.6305404652	0.42	51.2137662604	19.1441459464	382.78

MySQL report:

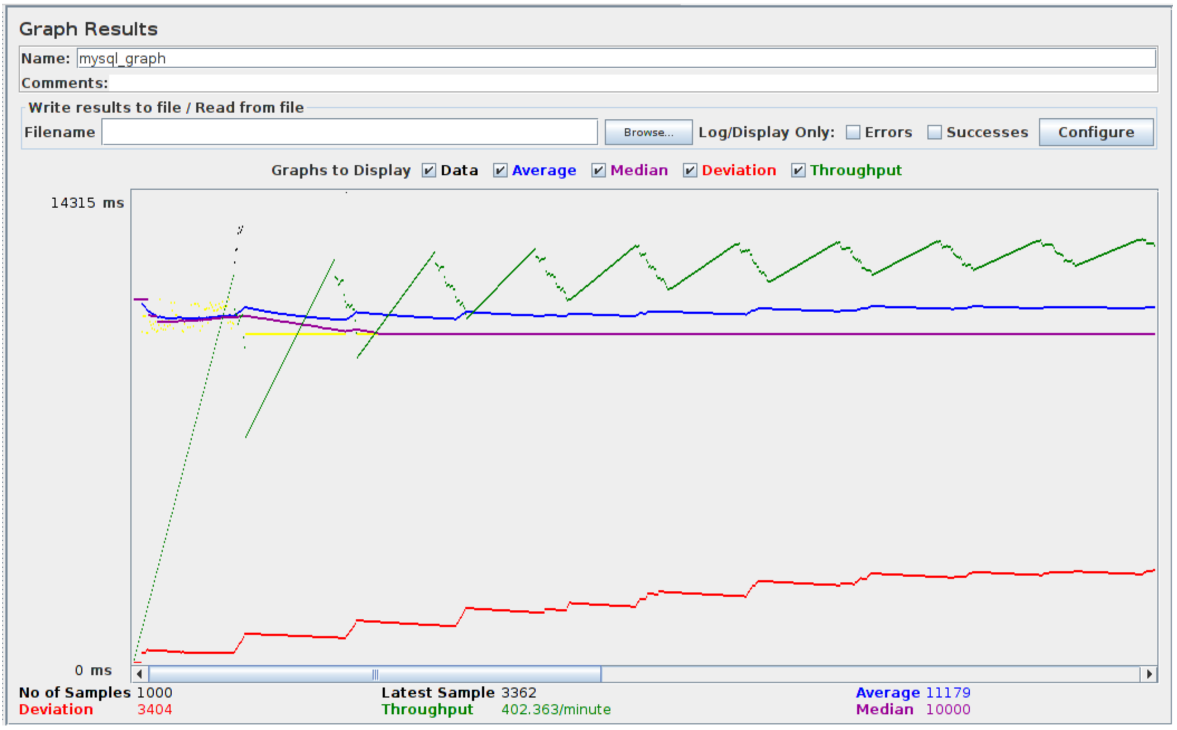
sampler_label	aggregate_report_count	average	aggregate_report_min	aggregate_report_max	aggregate_report_stddev	aggregate_report_error%	aggregate_report_rate	aggregate_report_bandwidth	average_bytes
JDBC Request	1000	11179	3362	24572	3404.4691899195	0.887	6.7060535545	18.8915553601	2884.7
TOTAL	1000	11179	3362	24572	3404.4691899195	0.887	6.7060535545	18.8915553601	2884.7

MongoDB Graph:


Graph Results
Name: mongo_graph
Comments:
Write results to file / Read from file
Filename
Log/Display Only: ☐ Errors ☐ Successes
Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 1000
Deviation 1584
Latest Sample 61
Throughput 3,072.826/minute
Average 955
Median 928

MySQL Graph:


Graph Results

Name: mysql_graph

Comments:

Write results to file / Read from file

Filename

Log/Display Only: ☐ Errors ☐ Successes

Graphs to Display ✓ Data ✓ Average ✓ Median ✓ Deviation ✓ Throughput

No of Samples 1000 Latest Sample 3362 Average 11179

Deviation 3404 Throughput 402.363/minute Median 10000

Conclusion

For query 1: (comparison between MongoDB and MySQL)

- Sample tested – 1000
- No. of requests can handle - **Throughput**
  MongoDB - 2748.259/minute
  MySQL – 531.124/minute
  This proves, MongoDB performs better w.r.t handling server request
- **Error rate**
  MongoDB – 0.4%
  MySQL – 0.405%
  In this case, they performed equally well. (less failed test)
- **Minimum** time spend by sample
  MongoDB – 0
  MySQL – 1122
  Therefore, MongoDB was able to count the number of documents faster as compared to MySQL
- Talking about the overall performance result, let's consider **Average** response time:
  MongoDB – 1080 ms
  MySQL – 10509 ms
  The result clearly shows that MongoDB was better w.r.t MySQL for this query.

Now, let's discuss the result for second query

For query 2: (comparison between MongoDB and MySQL)

- Sample tested – 1000
- No. of requests can handle - **Throughput**
  MongoDB - 3072.826/minute
  MySQL – 402.363/minute
  Again this proves, MongoDB performs better w.r.t handling server request. There is a big difference for both the queries.
- **Error rate**
  MongoDB – 0.42%
  MySQL – 0.887%
  In this case, MongoDB was better. MySQL gave error : "Could not create enough Components to service your request (Timed out)"
- **Minimum** time spend by sample
  MongoDB – 0
  MySQL – 3362 (had to search many records to find female actors)
  MongoDB way better in this regard. For both the query, it gave very quick answers.
- Talking about the overall performance result, let's consider **Average** response time:
  MongoDB – 995 ms
  MySQL – 11179 ms
  The result clearly shows that MongoDB was better w.r.t MySQL for this query. I am going for MongoDB.

From this small exercise, this is clear that MongoDB is very good in dealing with large amount of data. It's fast, it scale better !!