Pre-Registration

Performance Test Report

For

Execution of

Demographic_Detail – 100 users

Date: 29 March 2019

Author: Shankar N

Summary

This report presents the observations and findings of the load test conducted for a load of 100 users accessing demographic API Endpoint running for a duration of 30 minutes.

The objective of this load test was to observe and record the behavior of the application and the response time when users enter the demographic details.

Below are the scenario details:

Sprint/Report Name	Demographic_detail
Run Date	29-March -2019
Period	03:00 PM to 03:30 PM
Number of concurrent users	100
Ramp up	1 user per second
Run Duration	30 minutes
Ramp down	1 user per second

The aggregate report:

API Name	No of Requests	Average Response Time	90% line(ms)	Min(ms)	Max(ms)	Error %	Throughput/ Sec
Create form data	3100	456	676	179	7424	0.00 %	1.58

Test Environment

The DEV-Preregistration app server environment is used for test execution.

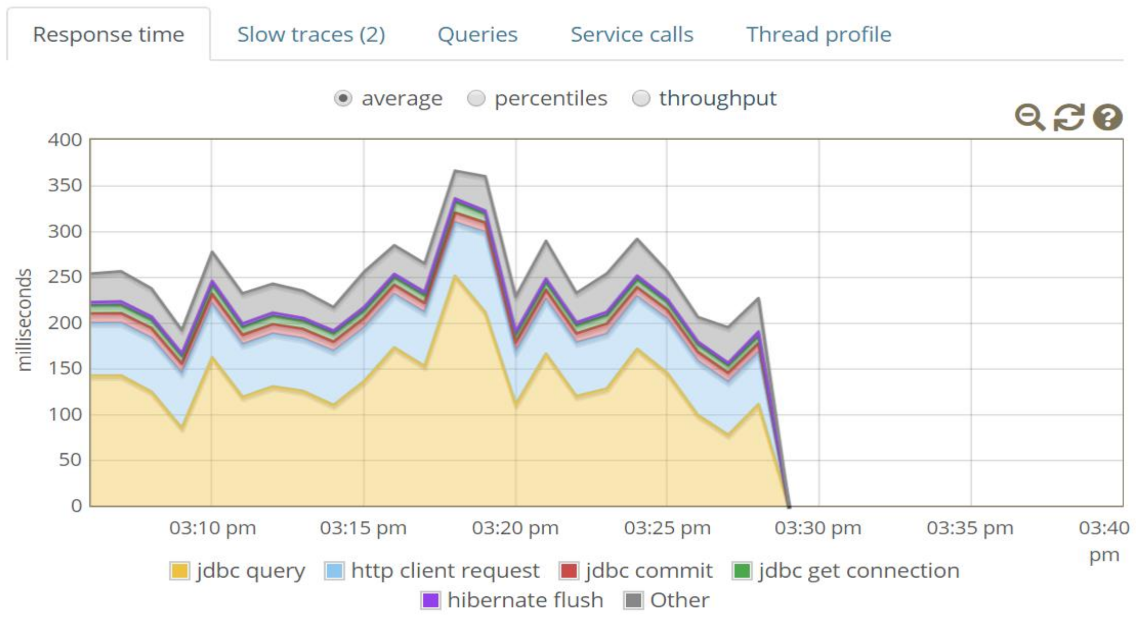
CPU cores: 2

Memory: 8GB

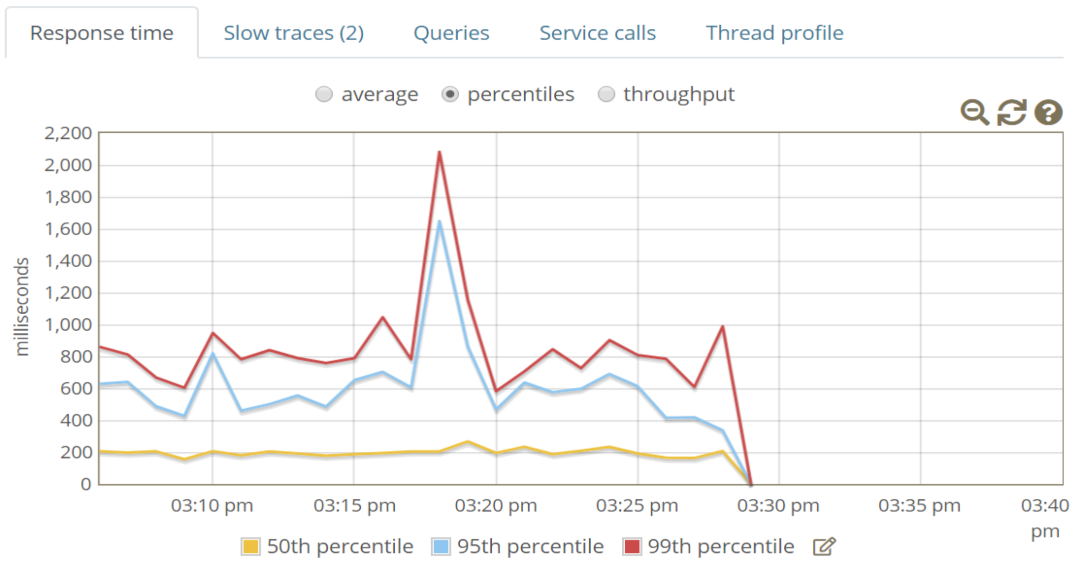
Below are the observations from glowroot profiling tool.

Response Time Graph

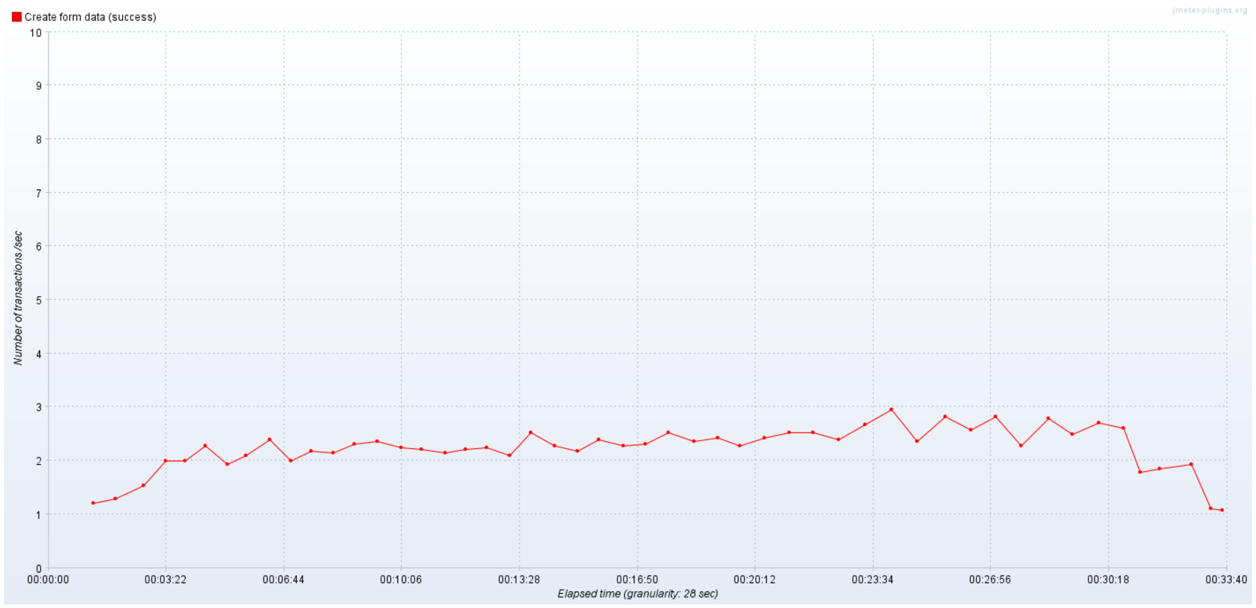
There were very high response time noticed during the test.



Percentile Response times:



Transactions per second:



Test Observations:

This is the retesting of the load test conducted on 20th March. There was 10+ seconds response time due to a JSON service call in the earlier test. Hence, a JIRA bug was raised (MOS-19823) to fix the high response time.

During the re-load test, the average response time is less than 0.5 seconds and throughput is achieved nearly 1.5 per second.

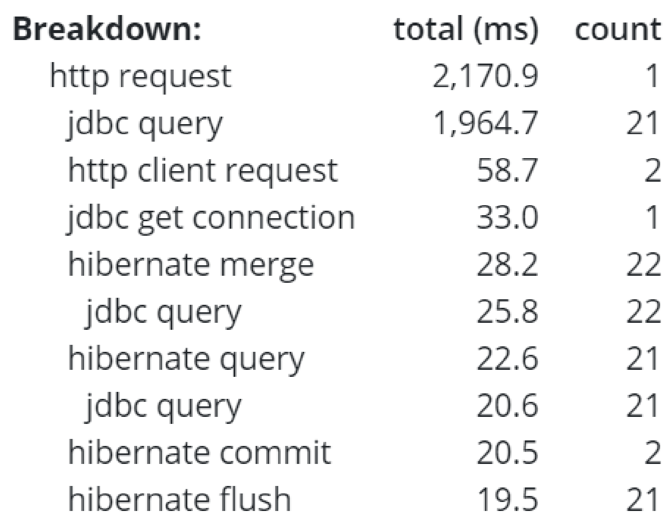
There were no service calls taking high response time. However, there are two JDBC calls which took more time to respond.

There were no error found throughout the load test duration.

JDBC Query: select pridsequen0_.seq_no as seq_no1_2_, pridsequen0_.cr_by as cr_by2_2_, pridsequen0_.cr_dtimes as cr_dtime3_2_, pridsequen0_.del_dtimes as del_dtim4_2_, pridsequen0_.is_deleted as is_delet5_2_ from prereg.prid_seq pridsequen0_ where pridsequen0_.seq_no=(select max(pridsequen1_.seq_no) from prereg.prid_seq pridsequen1_) for update of pridsequen0_

No of slow traces: 2

JDBC Response time breakdown:



Breakdown:	total (ms)	count
http request	2,170.9	1
jdbc query	1,964.7	21
http client request	58.7	2
jdbc get connection	33.0	1
hibernate merge	28.2	22
jdbc query	25.8	22
hibernate query	22.6	21
jdbc query	20.6	21
hibernate commit	20.5	2
hibernate flush	19.5	21

Conclusion and Next Steps:

The report will be shared with concerned architect and development team for further analysis of JDBC query.