Performance Analysis of RocksDB and its Integration into Open EdX

FUNDAMENTAL RESEARCH GROUP, IIT BOMBAY

Our Team

- **Principle Investigator :** Prof Deepak B Phatak
- **Project in-Charge :** Mr. Nagesh Karmali & Miss Firuza Aibara
- **Mentor :** Nithin S
- Project Interns: Abhishek Chattopadhyay

  Ekta Agrawal

  Simran Goyal

Primary Objective

- Performance improvement of OpenEdX using Facebook’s RocksDB engine.
- Performance Comparison between InnoDB and RocksDB and feasibility study.

Software Prerequisites

- OpenEdX
- JMeter
- MariaDB with RocksDB Plugin

JMeter

Apache JMeter is an open source software, Java application designed to load test functional behavior and measure performance.

OpenEdX

OpenEdX is an open source MOOC Platform for providing online courses.

It consists of 2 components:

- LMS
- CMS

MySQL is in the LMS to store User Content

MariaDB

An Enhanced Drop-In replacement for MySQL, which is faster and has a richer eco-system of storage engines and plugins as compared to MySQL

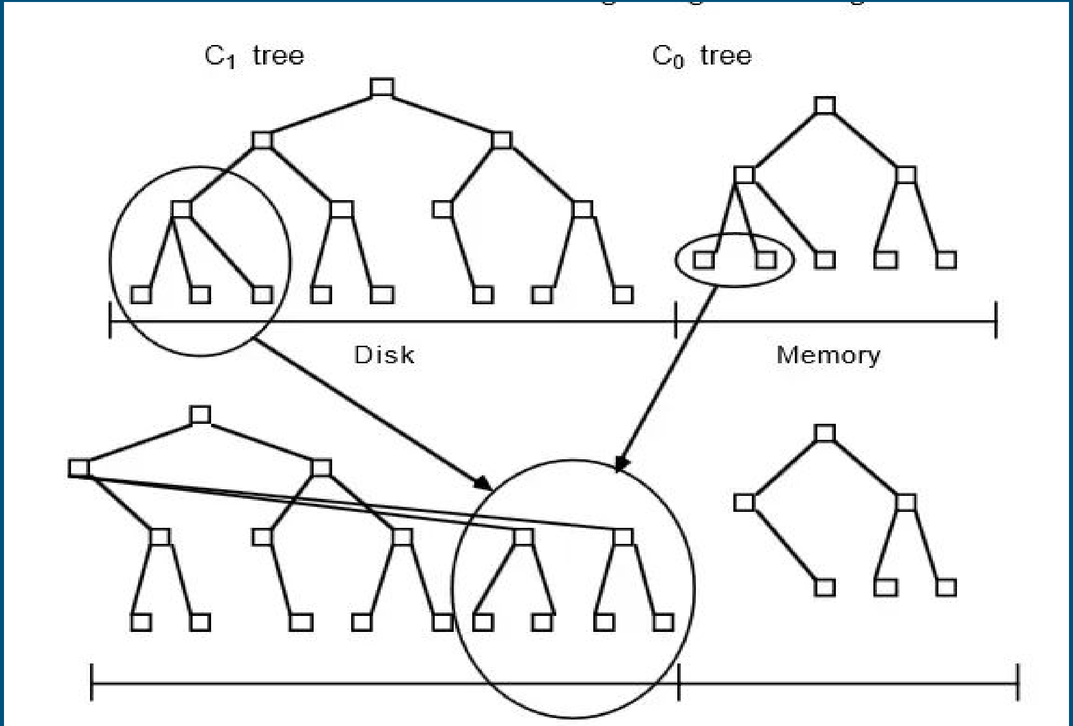
RocksDB

- RocksDB is an embedded, high performance, persistent key-value storage engine developed at Facebook Inc.
- RocksDB uses Log-Structured Merge trees (LSM) to obtain significant space efficiency and better write throughput.
- It is used by major organisations like Facebook, Yahoo!, LinkedIn and Netflix

Compaction in Log-Structured Merge Tree

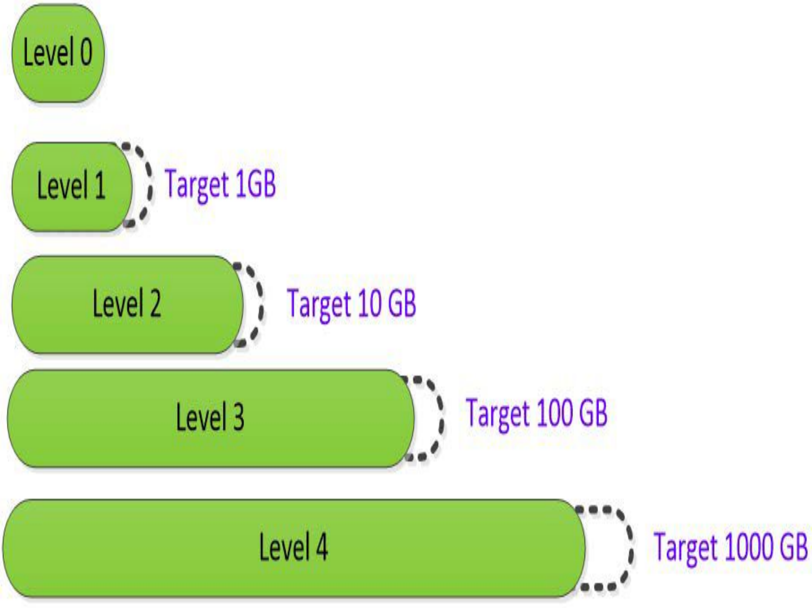
- The LSM Tree is an N-level tree with levels C_0-C_N , where C_0 is always present in Memory and C_1 to C_n in the disk.
- Insertion request to database enters into the C_0 level and is subsequently transferred to C_1 and the corresponding lower levels.
- This transfer happens through the process of a “Rolling Merge”, similar to Merge Sort.
- Each merge reads a leaf node of the $(l+1)^{th}$ level buffered in the block, merges it with the entries of the current level (l) and creates a newly merged node of the $(l+1)^{th}$ level, which are returned to new blocks so that the old block is not overwritten. This is the compaction process in LSM.

Compaction Procedure



LSM Compaction

- For each level, data is sorted by key
- LSM Amplification depends on the stale data yet to be collected.
- In the worst case, LSM-tree space amplification for data is 1.11, considering that size of all levels until the last level amounts to only 11.11% of the last level. (Default Case)
- The level size multiplier is a tunable parameter which can be used to moderate the space, read and write amplification.



Machine Specifications

- Remote testing of two Virtual Machines (VM1 and VM2) using JMeter.
- Configuration of the VM’s:
  - 4 cores x 4 sockets
  - 2GB RAM
  - 230 GB Hard Drive
  - Ubuntu Server 16.04 LTS

Test Parameters

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

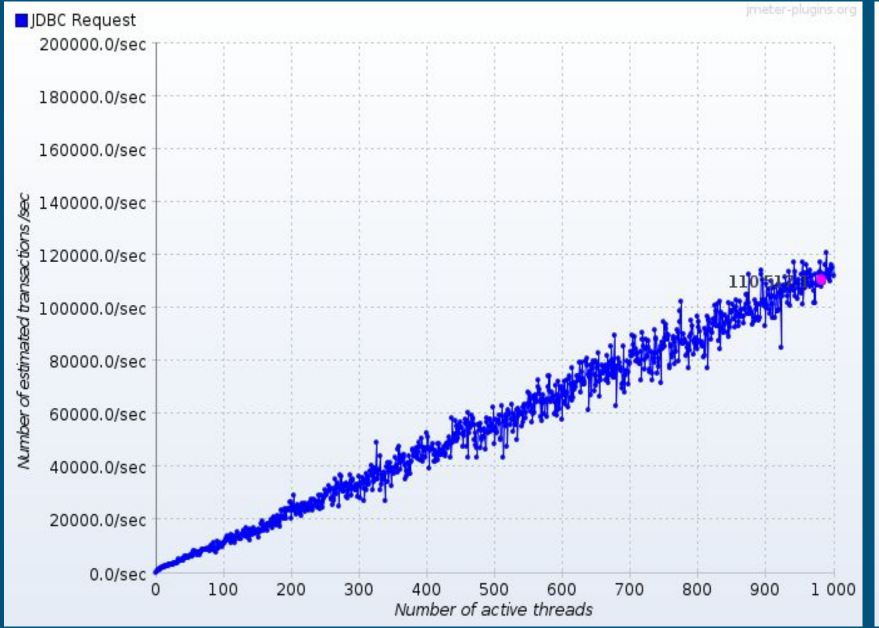
Test Procedure

Set Up and Tune JMeter -> Make Changes to *‘my.cnf’* file in VM -> Add required test plan -> Set up JDBC Configuration -> Select the type of JDBC Request -> Save and Run Test Plan in Command Line Mode

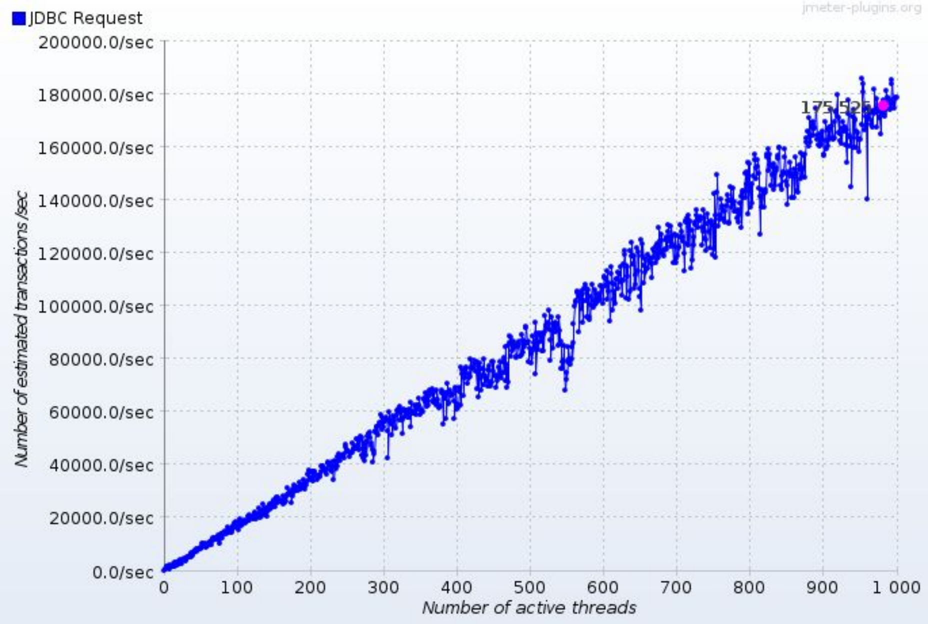
Open JMeter (GUI) -> Add Listeners to Workbench -> Analyse results

Graph Analysis for Insert Query

Number of Threads : 1000   Ramp-Up Time : 100



InnoDB



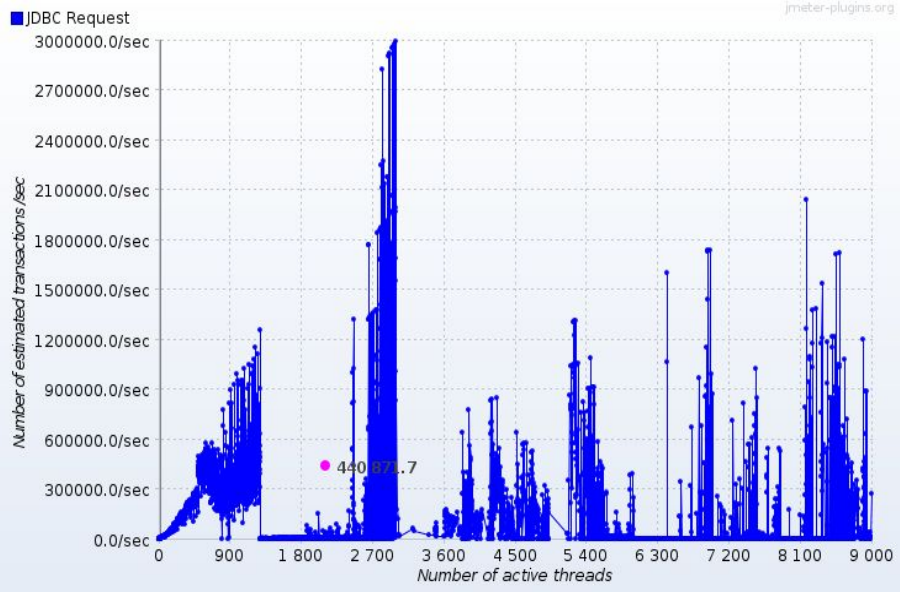
RocksDB

Insert Query Specifications

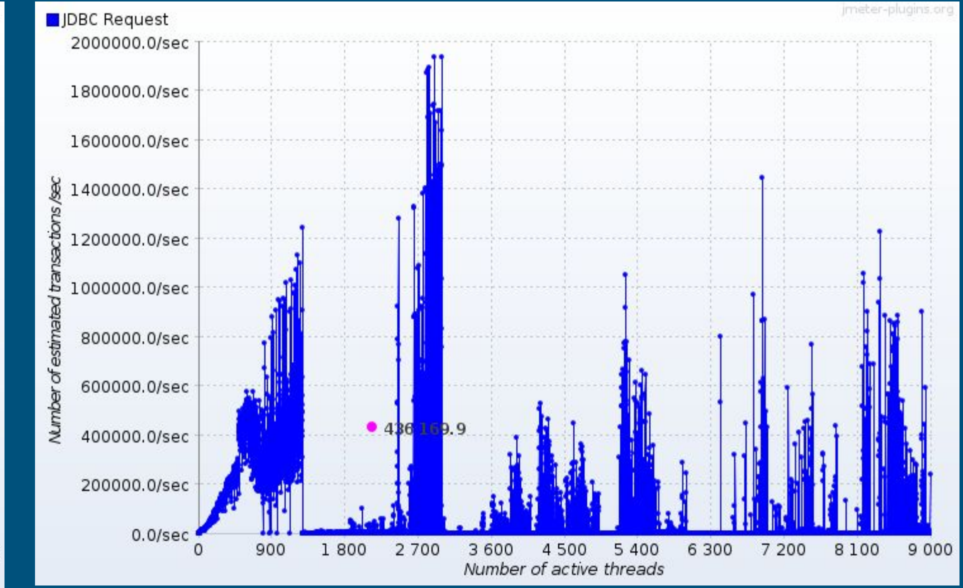
Basis	InnoDB	RocksDB
Bytes Throughput	24500 Bytes/Sec	28000 Bytes/Sec
Response Latencies	500 ms	390 ms
Transaction Throughput	110517 Transactions/thread	175525 Transactions/thread
Transactions per Second	2150 Transactions/sec	2700 Transactions/sec

Graph Analysis For Select Query

Number of Threads : 9000   Ramp-Up Time : 300



InnoDB



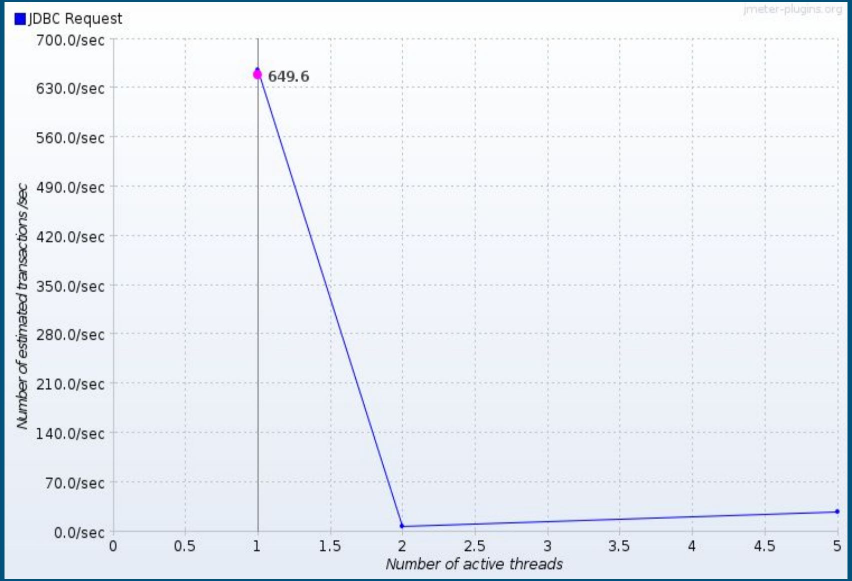
RocksDB

Select Query Specifications

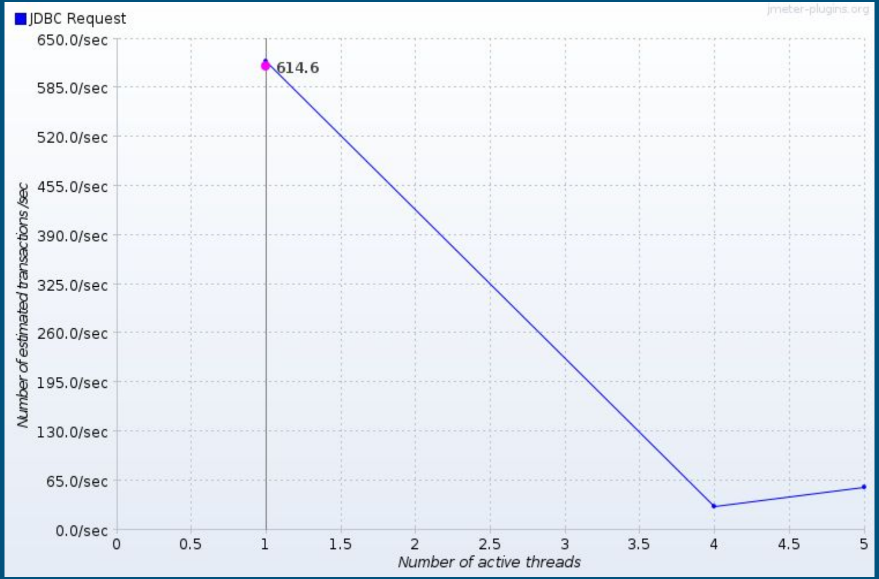
Basis	InnoDB	RocksDB
Bytes Throughput	27000	27000
Response Latencies	6-8	7-8
Transaction Throughput	440871 Transactions/ threads	436169 Transactions/ threads
Transactions per Second	4700 Transactions/sec	4500 Transactions/sec

Graph Analysis For Update Query

Number of Threads : 500 Ramp-Up Time : 50



InnoDB



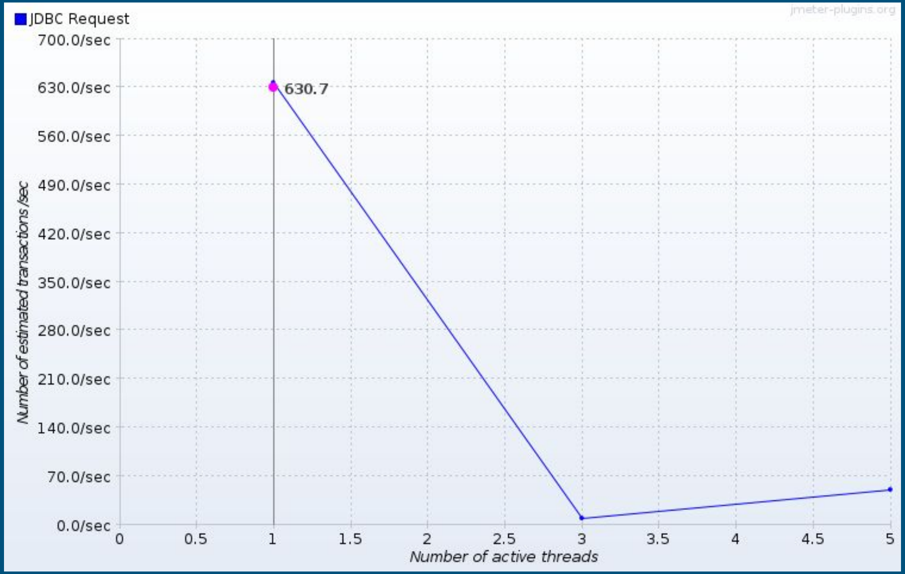
RocksDB

Update Query Specifications

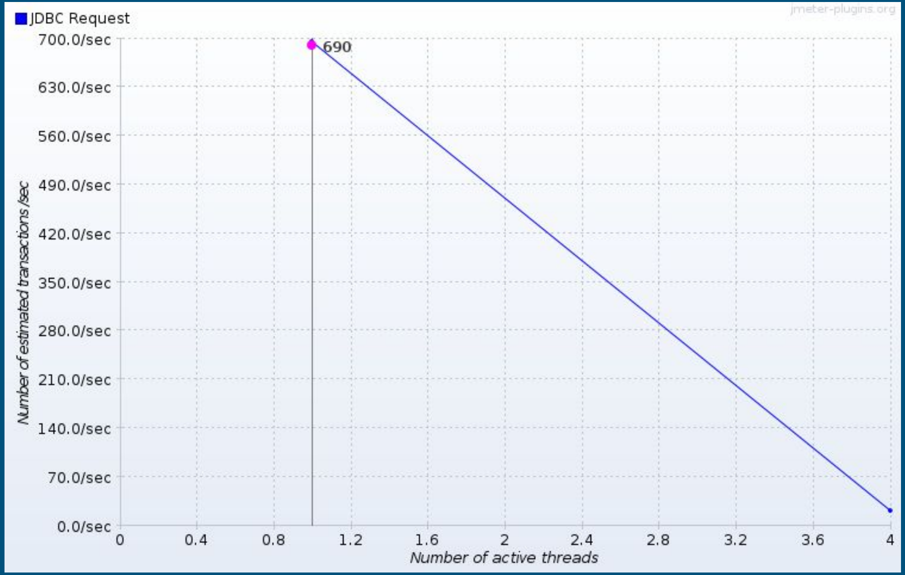
Basis	InnoDB	RocksDB
Bytes Throughput	110 Bytes/sec	110 Bytes/sec
Response Latencies	3 ms	3-4 ms
Transaction Throughput	649 Transactions/thread	614 Transactions/thread
Transactions per Second	10 Transactions/sec	10 Transactions/sec

Graph Analysis For Delete Query

Number of Threads : 500 Ramp-Up Time : 50



InnoDB



RocksDB

Delete Query Specifications

Basis	InnoDB	RocksDB
Bytes Throughput	110 Bytes/sec	110 Bytes/sec
Response Latencies	3 ms	2 ms
Transaction Throughput	630 Transactions/ thread	690 Transactions/ thread
Transactions per Second	10 Transactions/sec	10 Transactions/sec

Relative ORDER BY clause time frame

Database Name	Time Taken	Throughput
University	50 sec	9.9 Threads/sec
Universityrocks	10 min 33 sec	0.8 Threads/sec

Number of Threads : 500   Ramp-Up Time : 50

Relative Database Sizes

Database Name	Database Size (MB)
University	4540.54687500
Universityrocks	1361.6663660

VM1

Database Name	Database Size (MB)
University	4611.51562500
Universityrocks	1254.96938515

VM2

Test Analysis

- High Compression Efficiency (~66% Smaller Database)
- Improved Performance in INSERT and DELETE queries by RocksDB
- Comparable Performance in simple SELECT and UPDATE queries
- Lags behind in ORDER BY due to underlying data structure.
- Lack of FOREIGN KEY support in RocksDB

Final Results

- There is noticeable improvement in performance for RocksDB over InnoDB with regards to Storage Space efficiency and higher average Transaction Throughput for queries which relate to write operations

- Integration of RocksDB with OpenEdX is currently not feasible, since it does not support FOREIGN KEY constraint.

Future Works

- Implementation of FOREIGN KEY in RocksDB
- RocksDB compilation with MySQL
- Implementation of SAVEPOINT in MySQL
- Improving the performance of ORDER BY Queries
- Integration of OpenEdX with RocksDB

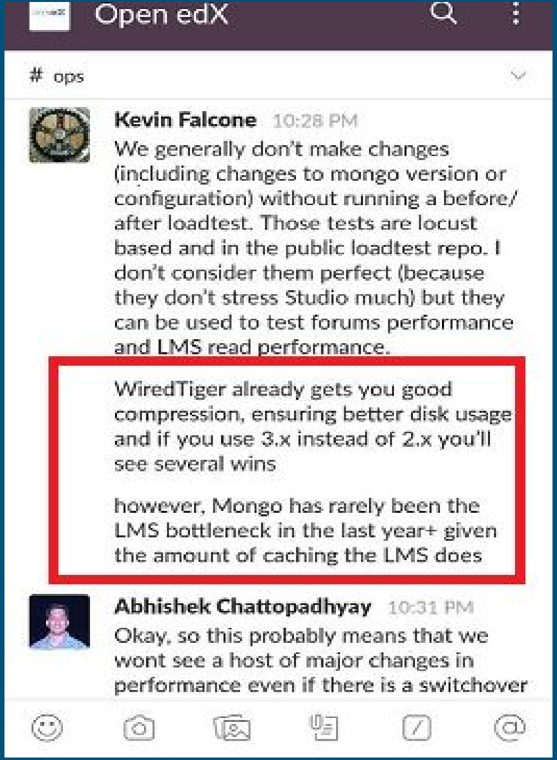
- For detailed analysis and graphs, kindly refer to our GitHub repository:

  https://github.com/fresearchgroup/performance-improvement-of-openedx

  https://github.com/fresearchgroup/performance-improvement-of-openedx/wiki

Thank you!

Absence of bottleneck issues in MongoDB



Kevin Falcone 10:28 PM

We generally don't make changes (including changes to mongo version or configuration) without running a before/after loadtest. Those tests are locust based and in the public loadtest repo. I don't consider them perfect (because they don't stress Studio much) but they can be used to test forums performance and LMS read performance.

WiredTiger already gets you good compression, ensuring better disk usage and if you use 3.x instead of 2.x you'll see several wins

however, Mongo has rarely been the LMS bottleneck in the last year+ given the amount of caching the LMS does

Abhishek Chattopadhyay 10:31 PM

Okay, so this probably means that we wont see a host of major changes in performance even if there is a switchover