Hardware performance analysis between Postgres and MariaDB

Roy Shadmon
rshadmon@ucsc.edu

Summary

- Postgres vs MariaDB
- Input set is self generated and can be found via in each test director from the following link:
  - https://github.com/royshadmon/Perf-Tracing-Postgres-MariaDB (this is also the project link)
- Time
  - MariaDB has better performance than Postgres on indexed data
  - Postgres has better performance than MariaDB on transactional queries
  - MariaDB’s partition table optimization allows it to scale and perform faster than Postgres
- Main summary perf metrics
  - MariaDB has a higher IPC than Postgres in regards to queries on indexed tables and when partitioning of tables optimization takes place
  - Postgres has a higher IPC than MariaDB on transactional queries

Applications

- Postgres
    - https://github.com/postgres/postgres
- MariaDB
    - https://github.com/MariaDB/server

Postgres vs MariaDB

- Both open source relational database management systems (RDBMS)
- Postgres is implemented in C and MariaDB is implemented in C and C++
- Both support ACID transactions, concurrency, foreign key, immediate consistency
- MariaDB has in-memory capabilities, while Postgres does not
- MariaDB has a partition tables feature, while Postgres does not (discussed in next slide)
- Postgres was first released in 1989, while MariaDB was released in 2009

Postgres vs MariaDB cont

MariaDB

- Partitioned tables
  - Allows for a table to be split in smaller subsets, as well as both data and indexes are partitioned
  - Improves performance on very big tables

Input sets

To create/generate input sets, I wrote a bash script that inserts “x” amount of rows into both a Postgres and MariaDB instance, where “x” is the amount of rows to insert.

In addition, in each script comparable tables are created for the respective database. This allows for for the opportunity to compare comparable results.

Note that the syntax may be different as SQL syntax is not universal and dependent on the respective database.

Input sets cont

The input set bash scripts can be found for the following types of tests:

- Primary key lookup, index lookup, and full scan
  - https://github.com/royshadmon/Perf-Tracing-Postgres-MariaDB/blob/master/primary-key-index-test/populate_both_dbs.sh
- MariaDB partitioned table vs normal Postgres table
  - https://github.com/royshadmon/Perf-Tracing-Postgres-MariaDB/blob/master/key-partition-test/populate_both_dbs.sh
- Postgres & MariaDB transaction test (the test is the input set)
  - https://github.com/royshadmon/Perf-Tracing-Postgres-MariaDB/blob/master/transaction-test/populate_both_dbs.sh
- Inserting data test
  - https://github.com/royshadmon/Perf-Tracing-Postgres-MariaDB/blob/master/insert-data-test/populate_mariaDB.sh
  - https://github.com/royshadmon/Perf-Tracing-Postgres-MariaDB/blob/master/insert-data-test/populate_postgres.sh

Machine specs

All tests were run on a single machine with the following specs:

- Machine: HP Pavilion TouchSmart
- Operating system: Ubuntu 18.04.4 LTS (64-bit)
- Memory: 12 GiB
- Processor: Intel Core i5-4210U CPU @ 1.70GHz x 4
- Graphics: Intel Haswell Mobile
- HD: 1TB

*Note that the machine does not have an SSD

Controlled aspects

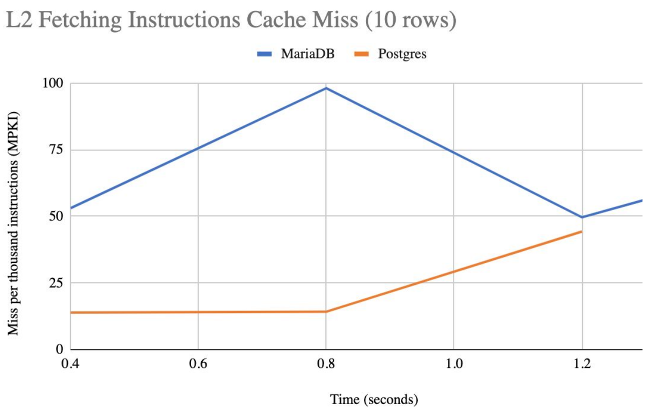
- I disabled address space randomization (ASLR)
    - ASLR is a computer security technique that prevents the exploitation of memory corruption vulnerabilities. By randomizing the arrangement of address space positions of key data areas of a process, it becomes difficult for an attacker to exploit a particular function in memory.
    - Disabling this feature ensures more consistent results since it better controls the memory placement of instructions/data for the CPU
- Manually dropped the file system cache prior to every query
- Set process priority to high so that the process gets the full CPU time.
    - perf stat sudo nice -n -5 taskset -c 1 sudo [mysql or psql command]
- Used InnoDB setting for MariaDB

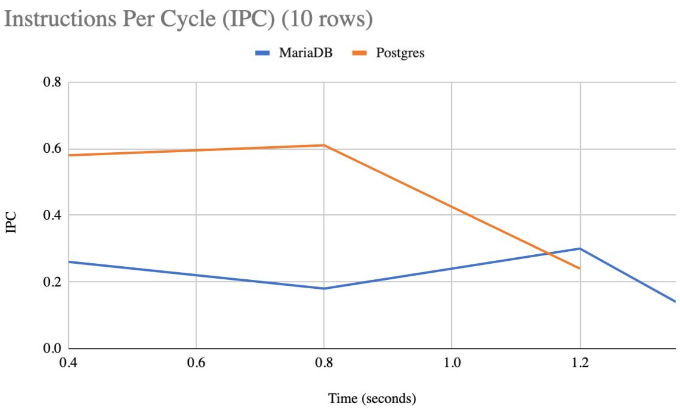
Tracing Results Observations

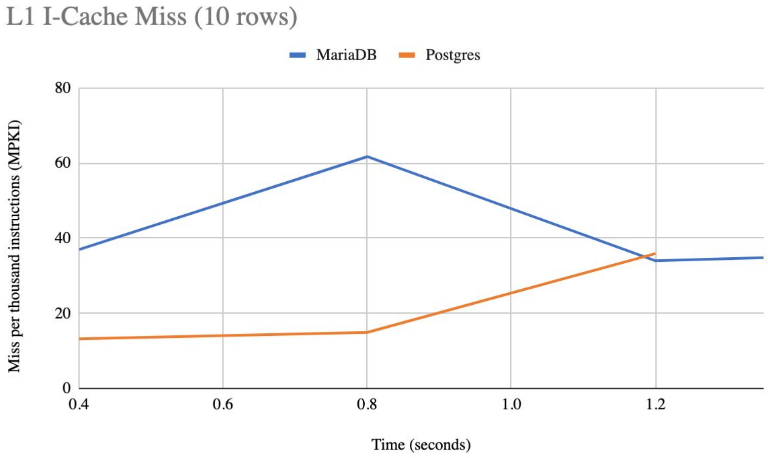
The graphs in the following 2 slides trace the performance of inserting data into a Postgres and MariaDB database. The conclusion for this test:

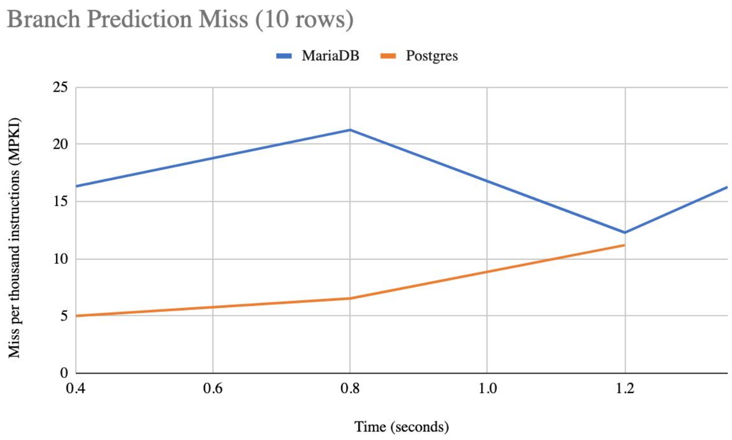
- MariaDB has more L1 i-cache misses and L2 instruction fetch miss than Postgres
- IPC in Postgres is higher than MariaDB
- MariaDB has more branch prediction misses than Postgres
- MariaDB completes the execution of inserting database rows faster than Postgres

Tracing Results


L2 Fetching Instructions Cache Miss (10 rows)

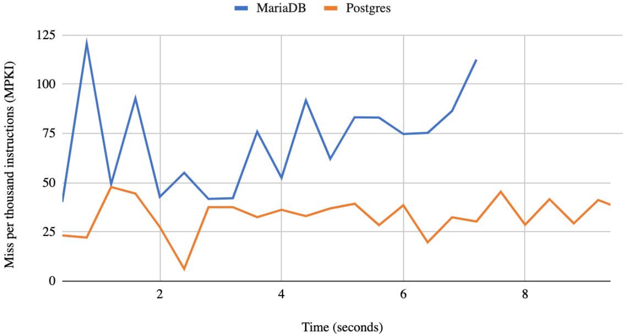

Instructions Per Cycle (IPC) (10 rows)


L1 I-Cache Miss (10 rows)

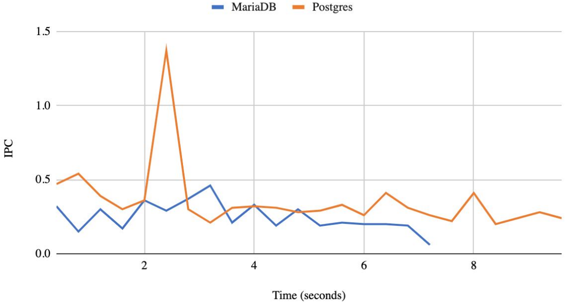

Branch Prediction Miss (10 rows)

Tracing Results cont.

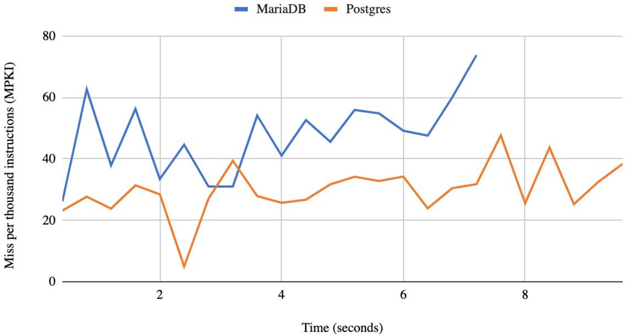
L2 Fetching Instructions Cache Miss (100 rows)



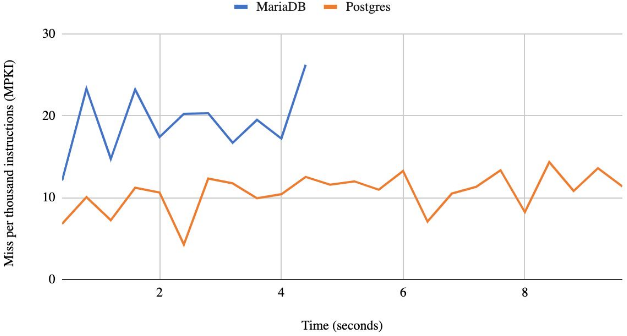
Instructions Per Cycle (IPC) (100 rows)



L1 I-Cache Miss (100 rows)

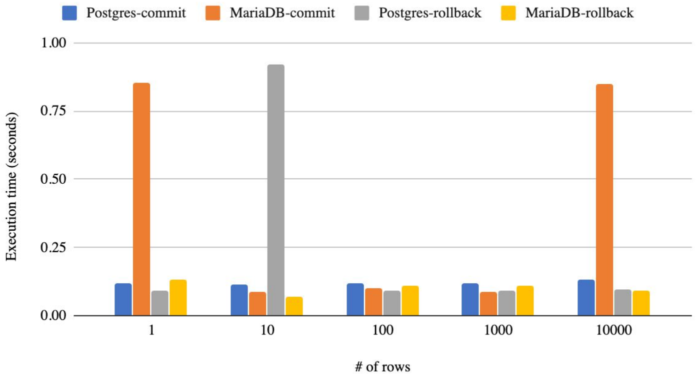


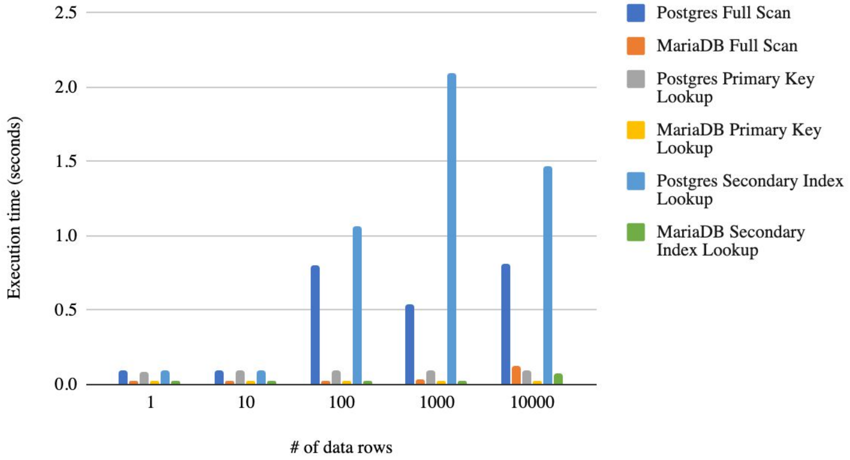
Branch Prediction Miss (100 rows)

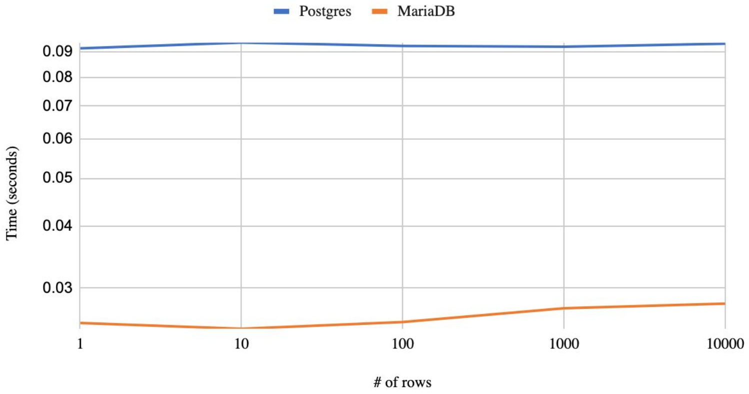


The following slides illustrate the hardware performance of specific RDBMS queries. The point is to show which DB better performs the other over various query types.

Execution Time


Execution Time - Transaction Test (logscale)

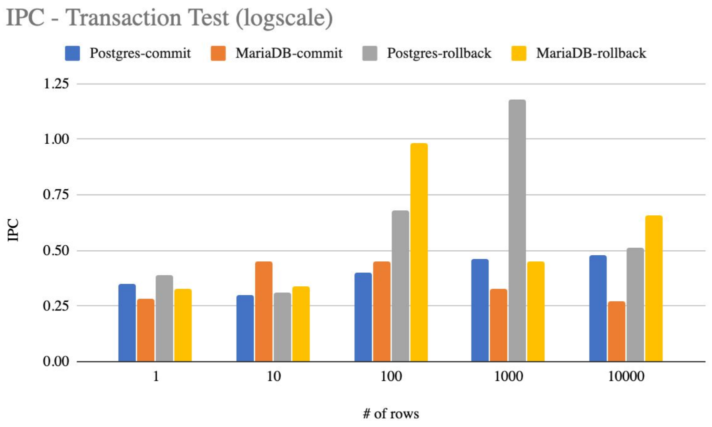

Execution time - Index Test

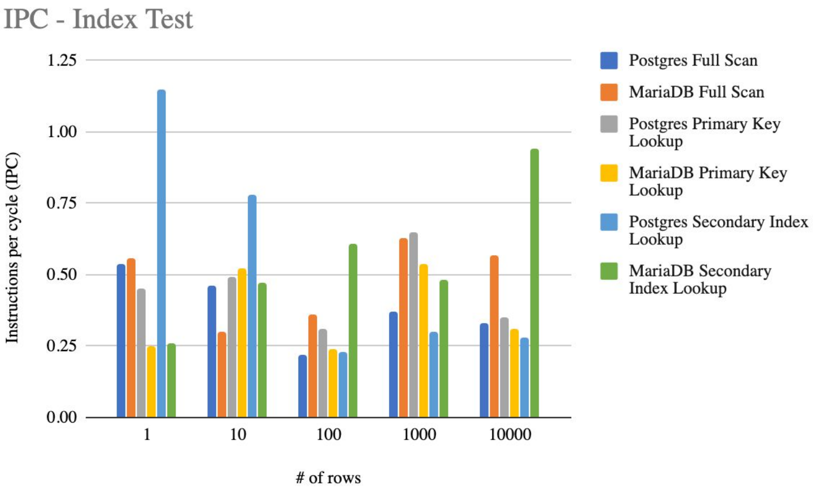

Execution Time - Partition Table Test (logscale)

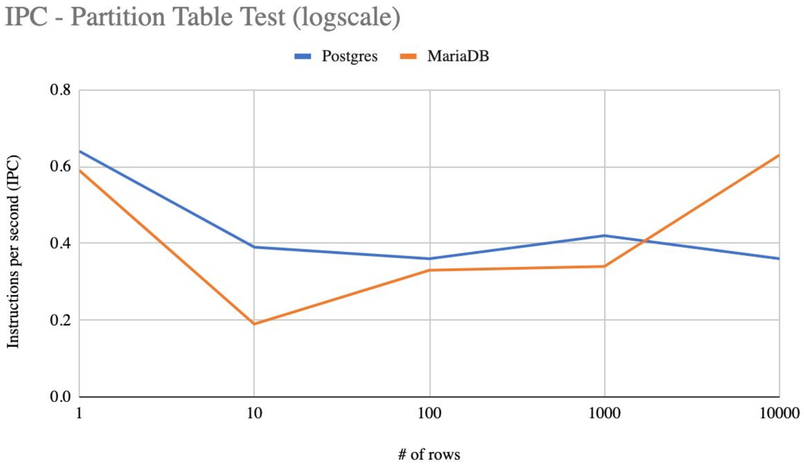
Main Observations - Execution Time

- MariaDB performs better than Postgres in regards to SELECT statements on indexed data
  - Primarily because MariaDB is able to search indexed data more efficiently than Postgres
- Postgres performs better overall on transactional queries
  - A transactional query only commits or rollbacks (undo) after all the queries in the transaction finish executing (atomicity)
- MariaDB's unique partition table optimization results in faster query execution time
  - This is expected because the same query is run on a smaller subset of data since the global table is partitioned into multiple smaller tables

IPC


IPC - Transaction Test (logscale)


IPC - Index Test

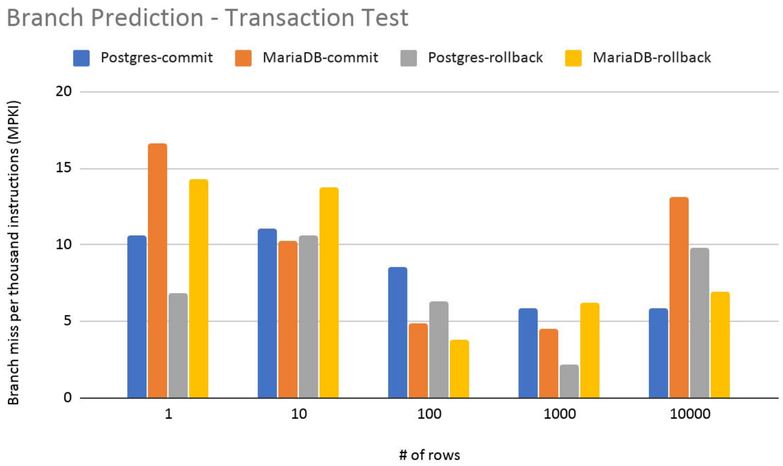

IPC - Partition Table Test (logscale)

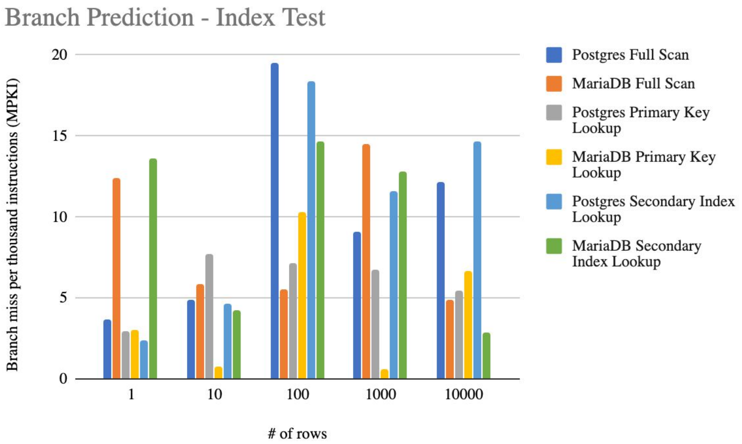
Main Observations - IPC

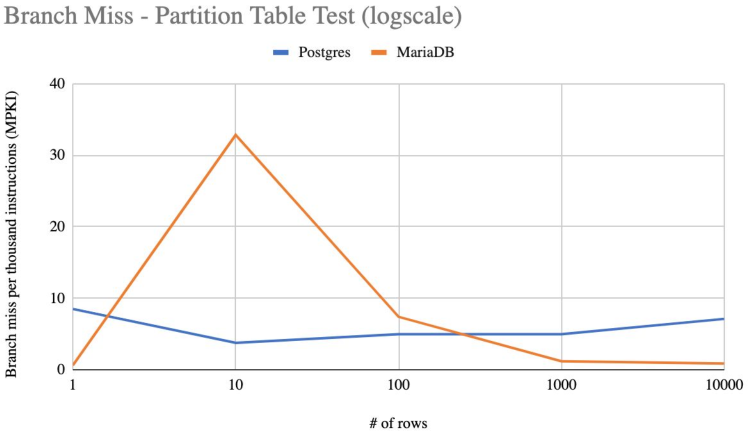
- MariaDB IPC is better than Postgres in regards to SELECT statements on indexed data
  - MariaDB's algorithm is more efficient--they also take advantage of in memory operations (Postgres does not)
- The IPC in query transactions in Postgres is better than in MariaDB
- Queries in MariaDB will outperform Postgres when leveraging partitioned table optimization as more data is stored in the database
  - Expected, as queries on a smaller subset of data will outperform the same query on the global data set

Branch Prediction (MPKI)

MPKI = (total branch miss)/(total instructions) * 1000


Branch Prediction - Transaction Test


Branch Prediction - Index Test

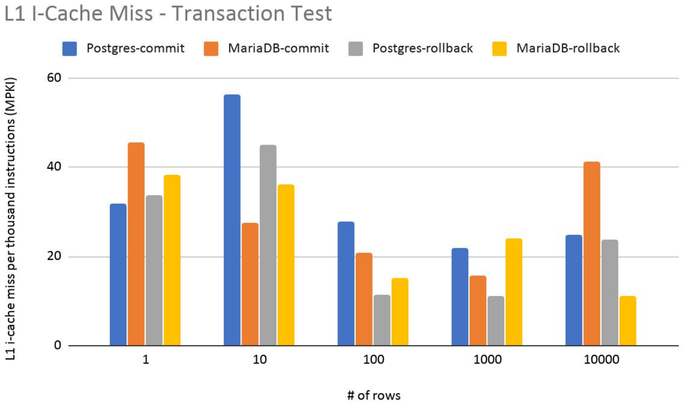

Branch Miss - Partition Table Test (logscale)

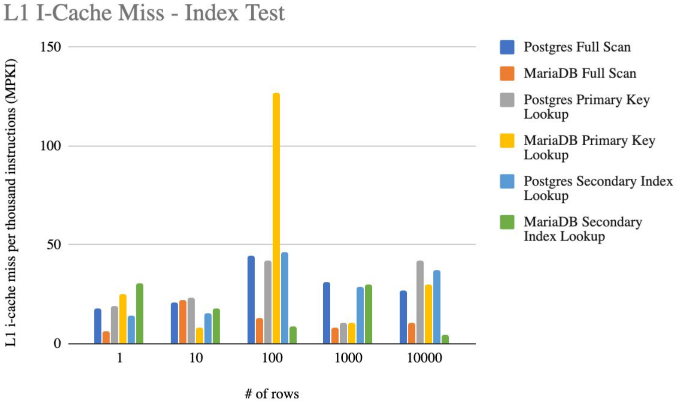
Main Observations - Branch Prediction

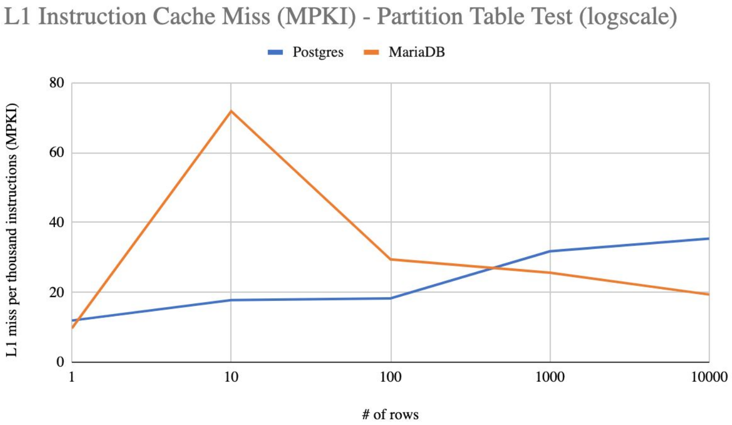
- As more rows are added, there are more branch prediction misses in Postgres than in MariaDB for queries on indexed data
  - Also, a reason why the execution time in Postgres is slower
- There are more branch prediction misses in MariaDB on transactional queries than in Postgres
  - Explains why the execution time in MariaDB is slower than Postgres for this test
- As there are more rows, there are fewer branch misses in MariaDB than in Postgres on a partitioned table
  - This is expected because queries are run on a significantly smaller table than a full Postgres table

L1 Instruction Cache (MPKI)

MPKI = (L1-icache-miss)/(total instructions)*1000





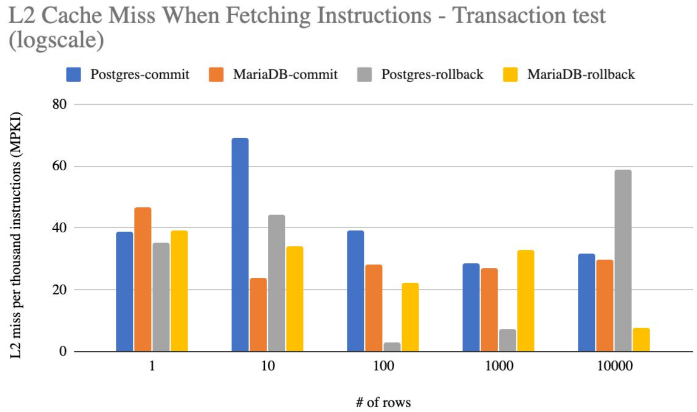


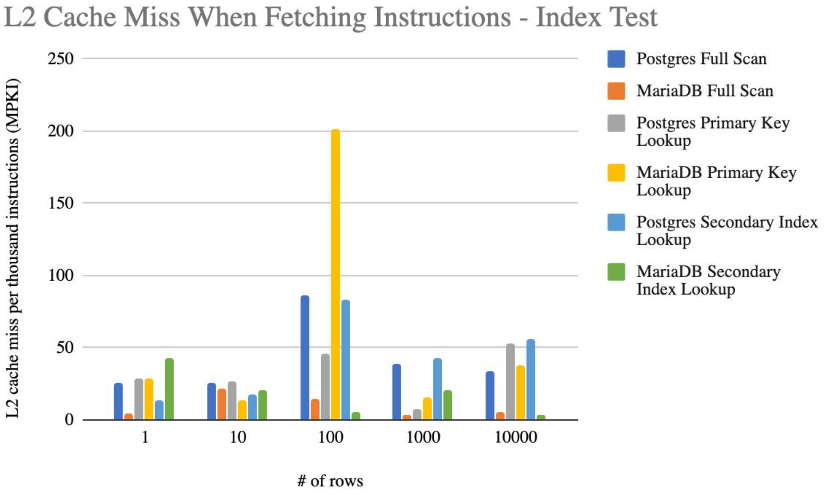
Main Observations - L1 Instruction Cache Miss

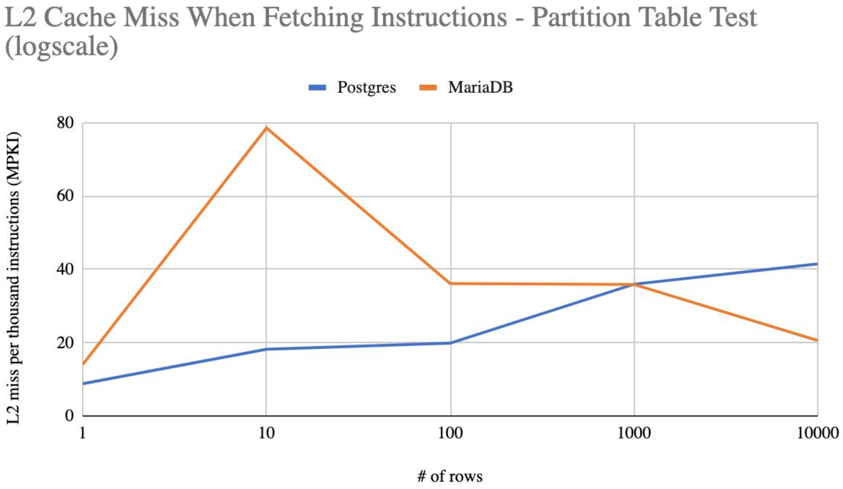
- Over time, it seems like MariaDB has fewer L1 i-cache misses for the index test
- It seems like MariaDB has more L1 i-cache miss in the transaction test
- Over time, MariaDB has fewer L1 i-cache miss when partitioning tables optimization takes place

L2 Cache Miss (MPKI)

MPKI = (L2-rqsts.code_rd_miss)/(total instructions)*1000







Main Observations - L2 Fetch Instruction Cache Miss

- MariaDB has fewer L2 fetch instruction cache misses than Postgres for the index test
- When transactions are committed or rolled back, it’s inconclusive which one has more L2 fetch instruction cache miss for the transaction test
- MariaDB has fewer L2 fetch instruction cache miss as more rows are added in the partition table test

How to reproduce results

1. Get a machine running Ubuntu 18.04 (64-bit)
2. Git clone this repository:
   a. https://github.com/royshadmon/Perf-Tracing-Postgres-MariaDB.git
3. Run the “setup_MariaDB_Ubuntu.sh” script
4. Run the “setup_Postgres_Ubuntu.sh” script
5. Run the “setup.sh” script
6. Cd into a test directory in the repository and run the “run_[test name]_test.sh” script
   a. Results will be stored in directories 1,10,100,1000,1000, where each directory name resembles the amount of rows in each database at the time of the test
7. If you have any troubles, feel free to reach out to me via email rshadmon[at]ucsc.edu

Resources Used

1. https://db-engines.com/en/system/MariaDB%3bPostgreSQL
2. https://stackoverflow.com/questions/25701265/how-to-generate-a-list-of-all-dates-in-a-range-using-the-tools-available-in-bash
3. https://www.mssqltips.com/sqlservertip/5745/compare-sql-server-mysql-and-postgresql-features/
4. https://easyperf.net/blog/2019/08/02/Perf-measurement-environment-on-Linux
5. http://www.nilinfobin.com/mysql/how-to-login-in-mariadb-with-os-user-without-password/
6. https://dba.stackexchange.com/questions/83164/postgresql-remove-password-requirement-for-user-postgres