RasberryPI 3 ModelB Database Benchmarking
HarperDB and SQLite

HarperDB

Overview

The purpose of this benchmark test was to evaluate an HTAP use case commonly found in IoT architectures. The goal was to evaluate data writes and aggregate reads to demonstrate increasing data size and real-time analytical capability. For the scope of this test HarperDB was compared in contrast to SQLite[1], a popular IoT database on the same Rasberry PI 3 Model B[2]

Methodology

To evaluate both solutions simple apps were created using Node.js. In HarperDB the native REST API was utilized. In SQLite the sqlite3[3] and mydb.db [4]libraries were utilized. The app takes both databases through the same process. In each test the data is purged before the next test is run. A varying number of records are simultaneously inserted into each product. The results are then queried in aggregate using a SELECT COUNT SQL query. In each case the steps that make up the process are designed to benefit each product. The process appears as follows in each app:

HarperDB

1. create_schema
2. create_table
3. begin timer
4. insert
5. select count sql statement
6. end timer
7. drop table

SQLite

1. drop_table if exists
2. create_table
3. begin timer
4. prepared statement insert
5. select count sql statement
6. timer end

As you can see from the above, the insert and the select statements are the only elements of the test being timed. Each table is named “dog”. Each insert has the same four attributes/columns: id, name, age, and breed. The id column is a randomly generated UUID using the uuid/v4 package in Node.

Both SQL statements look as follows: “SELECT count(id) from dog”

In HarperDB the id column is defined as the hash column and the remaining columns are dynamically indexed upon insert. This is native to HarperDB and a feature that cannot be disabled. To reduce overhead on SQLite only the primary key is indexed and this is the column used in both SQL queries.

We performed the test inserting 1, 10, 100, 500, 1000, 2500, and 5000 records into the database. To avoid variance in the results each test was performed 5 times at each level. The results displayed below represent the average of each test.

The code for the tests can be downloaded and examined here:
https://github.com/HarperDB/harperdb-sqlite-benchmark

[1] https://www.sqlite.org/index.html
[2] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[3] https://www.npmjs.com/package/sqlite3
[4] https://www.npmjs.com/package/mydb

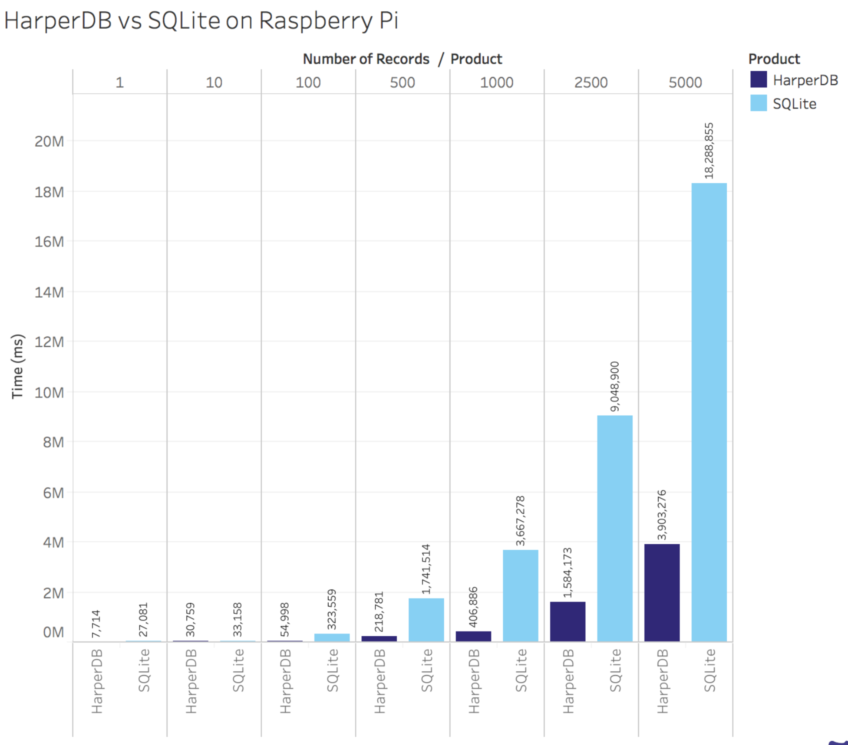
Test Results

HarperDB outperformed SQLite on all test cases, and on average was 581% faster. The greatest difference was seen at a 1,000 record insert where **HarperDB was 901.30% faster than SQLite** where the roundtrip was 20955.88ms for SQLite and 2305.06ms
or HarperDB. The averaged test results can be seen in the charts below.

Number of Records	*Product*	*Avg. Time (ms)*
1	SQLite	154.75
1	HarperDB	44.08
10	SQLite	189.48
10	HarperDB	175.76
100	SQLite	1848.91
100	HarperDB	314.27
500	SQLite	9951.51
500	HarperDB	1250.17
1000	SQLite	20955.88
1000	HarperDB	2325.06
2500	SQLite	51707.10
2500	HarperDB	9052.419
5000	SQLite	104507.74
5000	HarperDB	22303.44

Average time(ms) by number of records

Graphical Representation


HarperDB vs SQLite on Raspberry Pi
Number of Records / Product
Product
HarperDB
SQLite
Time (ms)
20M
18M
16M
14M
12M
10M
8M
6M
4M
2M
0M
HarperDB
SQLite