WhiteDB YCSB tests

Draft version

Introduction

WhiteDB is compared to popular NoSQL databases redis and MongoDB, using the Yahoo cloud-serving benchmark. The measurements were made of the latency of database operations, depending on the load, as well as overall throughput under single-user and parallel usages. The engines were deployed as a memory database, with the data stored in RAM during the tests.

Tested databases

- whitedb-0.7-alpha (memory database of n-tuples)
- redis-2.6.14 (a popular memory database)
- mongodb-linux-x86_64-2.4.5 (a popular document store)

All databases were using their default settings, with the following exceptions: 1. the snapshot function in redis was turned off and MongoDB was also launched with the --nojournal parameter; 2. the MongoDB database was stored on a RAM disk to eliminate the impact from disk I/O. These adjustments were made because WhiteDB in its default configuration does not use the disk at all.

Test system

AMD Opteron, 4 cores

The benchmark

The Yahoo cloud-serving benchmark is implemented in Java and runs predefined workloads against the chosen database engine, using one or multiple threads. A more detailed description can be found at: https://github.com/brianfrankcooper/YCSB/wiki.

This test used the workloads bundled with the benchmark:

- Workload A: 50% reads, 50% updates
- Workload B: 95% reads, 5% updates
- Workload C: 100% reads
- Workload F: simulates read-modify-write of records
- Workload D: inserts records, reads recently inserted
- Workload E: inserts records, queries ranges of records

Each workload performs 1000 operations and measures the average latency by operation type, as well as total throughput over all threads.

Overview of the tests

Each workload was tested with 1, 2, 4 and 8 client threads; each of these combined with an overall load of 100, 500, 2500, 12500 and unthrottled operations per second; and each of these combinations repeated 5 times.

Databases were tested with up to 170k records. YCSB by default creates 1k size records, resulting in memory requirements in the range from 500MB (WhiteDB) to 1.1GB (MongoDB). A memory database of this size could still be served comfortably from a typical VPS; applications where WhiteDB is currently deployed also have databases roughly this size.

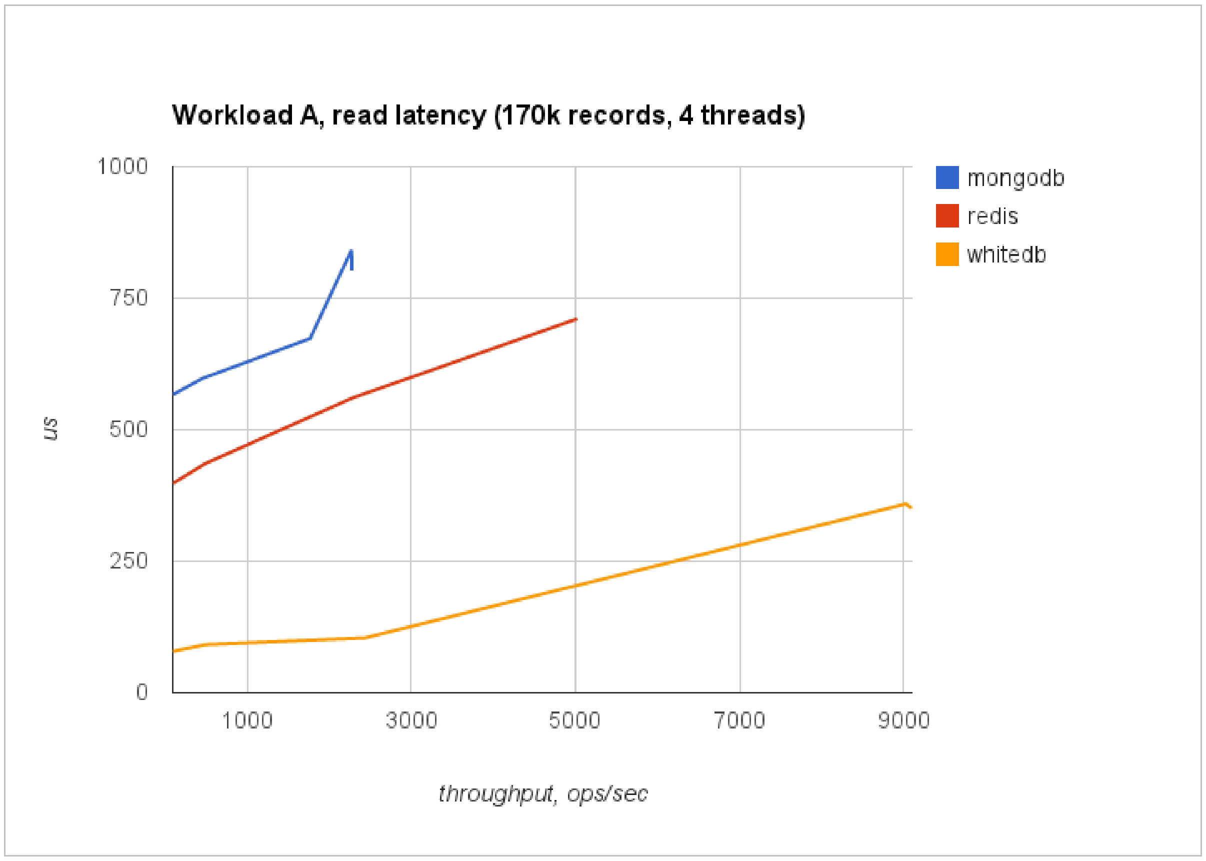
Two types of measurements are reported here: 1.) latency vs throughput with 4 client threads. This utilizes all the cores of the test system and describes how increasing the number of queries affects the average response time of a database operation; 2.) maximum unthrottled throughput by number of threads. This shows how the performance of the engine is affected by how many clients in parallel are accessing it.

The literal values of the measured latencies and throughputs should not be given too much weight, as each of the read, update etc operations generally reads or writes the entire 1kB record, pre- or postprocessing it using Java data structures. That results in a significant overhead that may not be present in real applications. Therefore, the only relevant interpretation of the tests should be comparing the results of different engines.

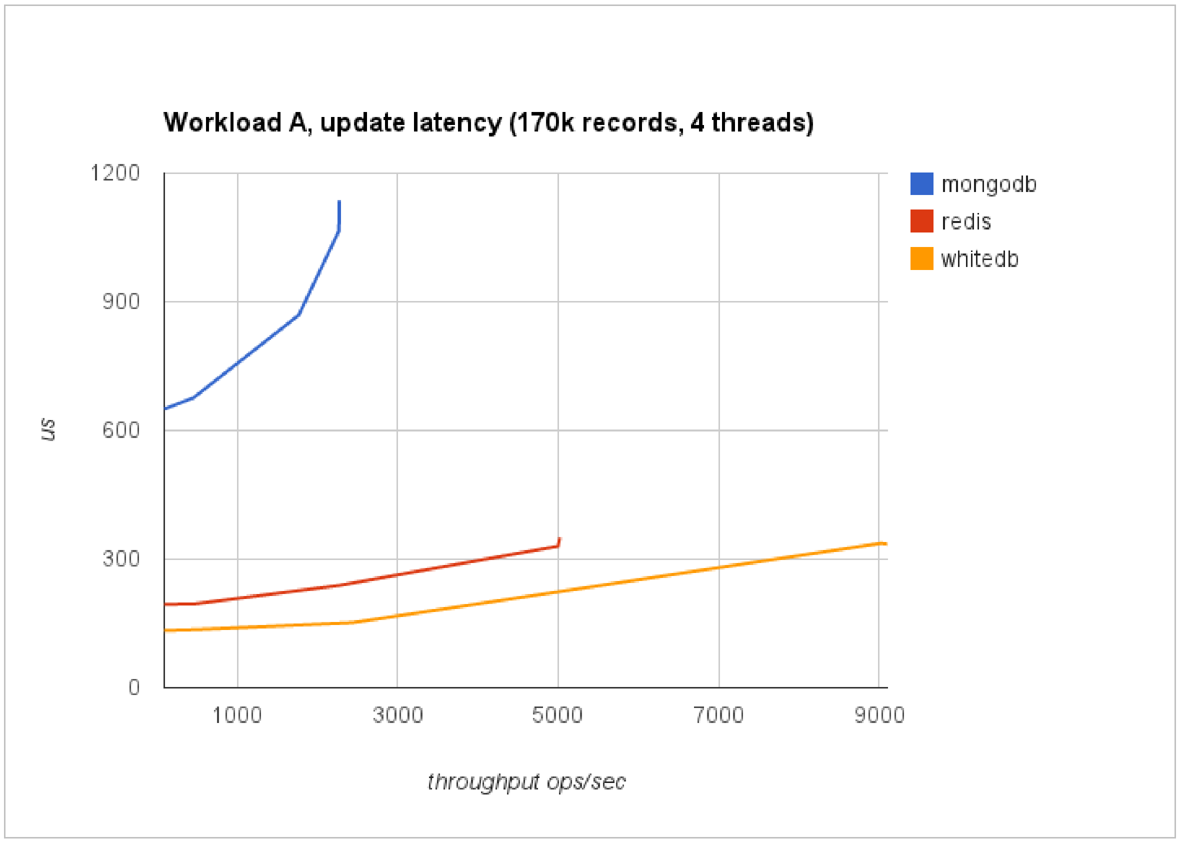
Latency vs throughput

In each of the latency graphs, lower is better. Latency is the average measured response time of a given database operation, in microseconds.

The workloads A,B,C are similar in that they each combine read and update operations with a varied ration. The workload A, described as “update heavy” is shown here as it may potentially be most revealing - in general mixing parallel reads and writes should be more demanding than just parallel reads.

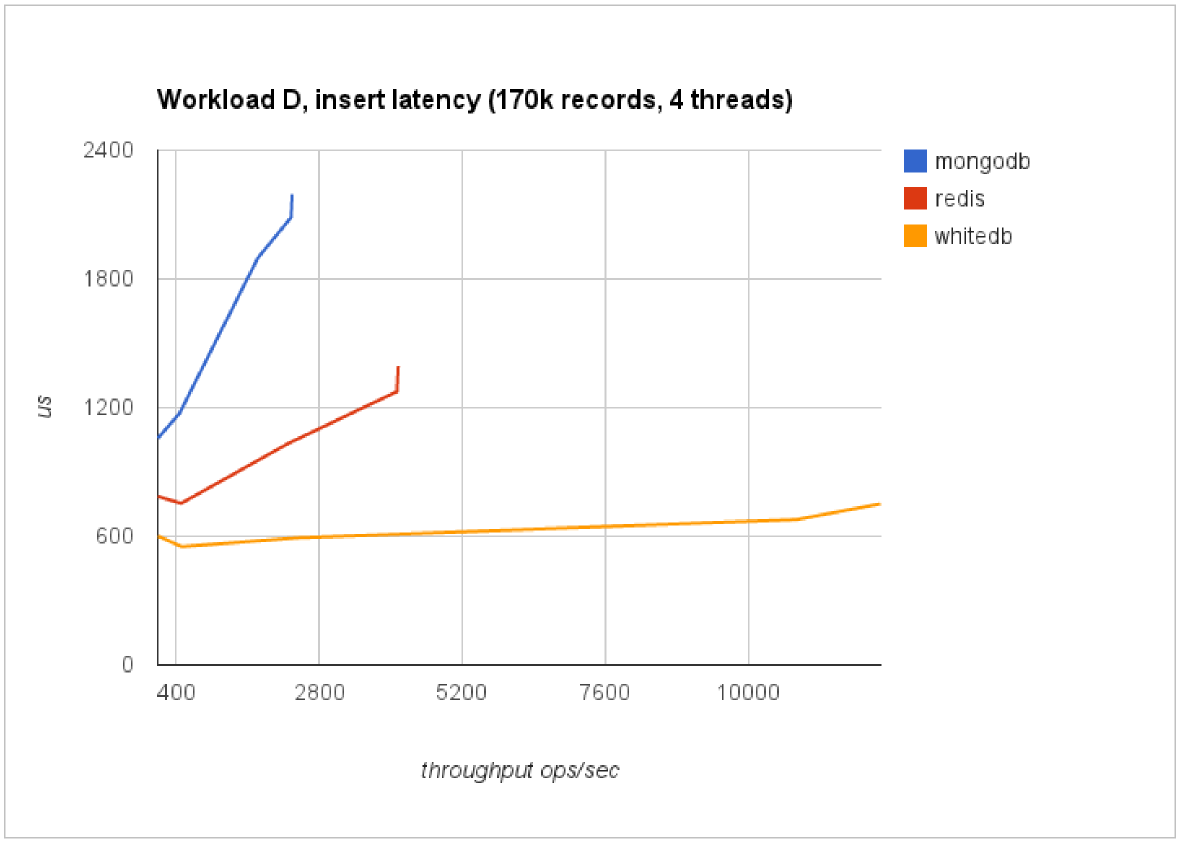


WhiteDB has the superior read latency, peaking at 350 us when reaching 9000 operations per second. The higher response time of the other databases also results in a lower maximum number of operations they could reach.



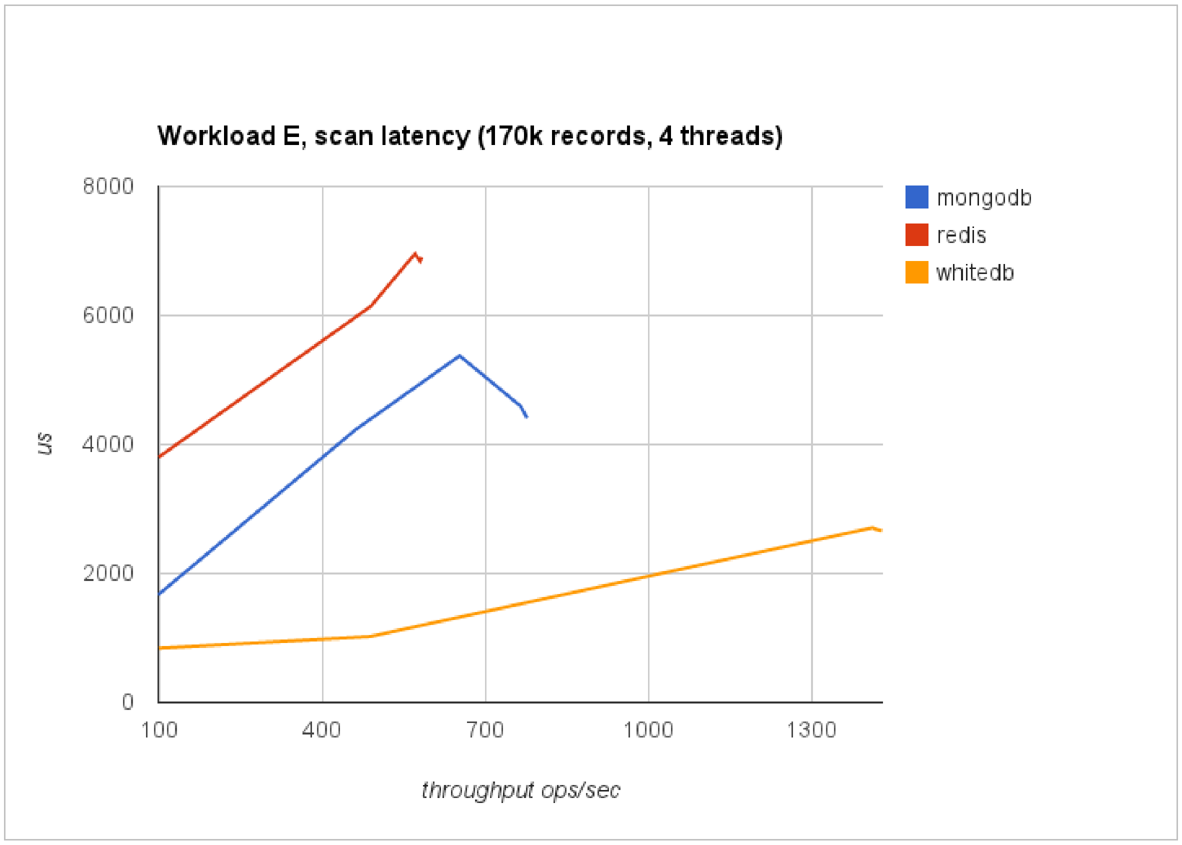
An interesting trend is revealed when examining update latencies. Redis is obviously optimized for writes and can perform them faster than reads even when the database is not heavily contended (compare the values for redis under 1000 ops/sec load on this and previous graph). WhiteDB nevertheless remains slightly faster.

Workload D contains inserts, which differ slightly from updates in the benchmark in that the database engine also needs to create the record. This test has a ratio of 5% of inserts and 95% of reads.



Compared to updating, all engines have higher latencies. Also, it appears that redis is somewhat less tolerant to increasing number of requests, while WhiteDB's response time remains stable.

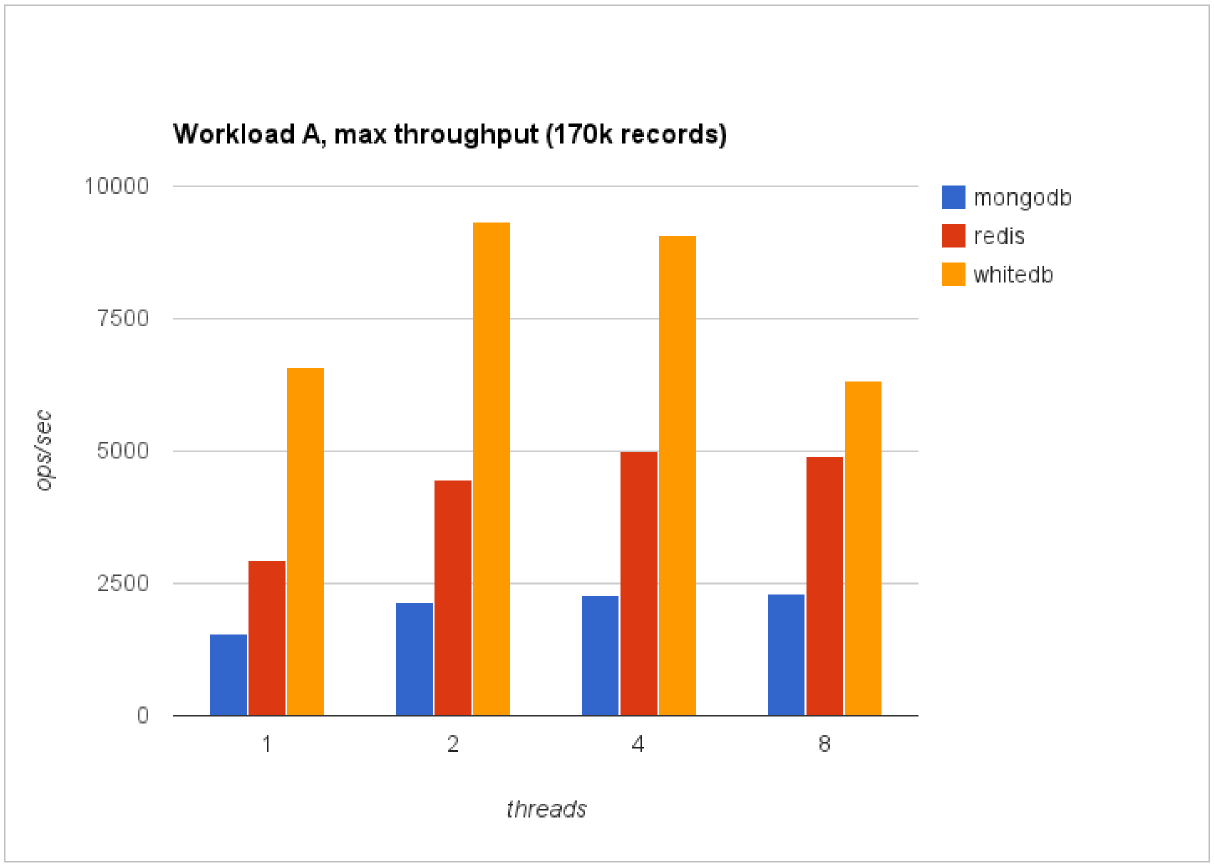
Workload E performs a simulation of range queries, retrieving (up to 100) consecutive records starting from a given key and reading their fields. These are mixed with insert operations.



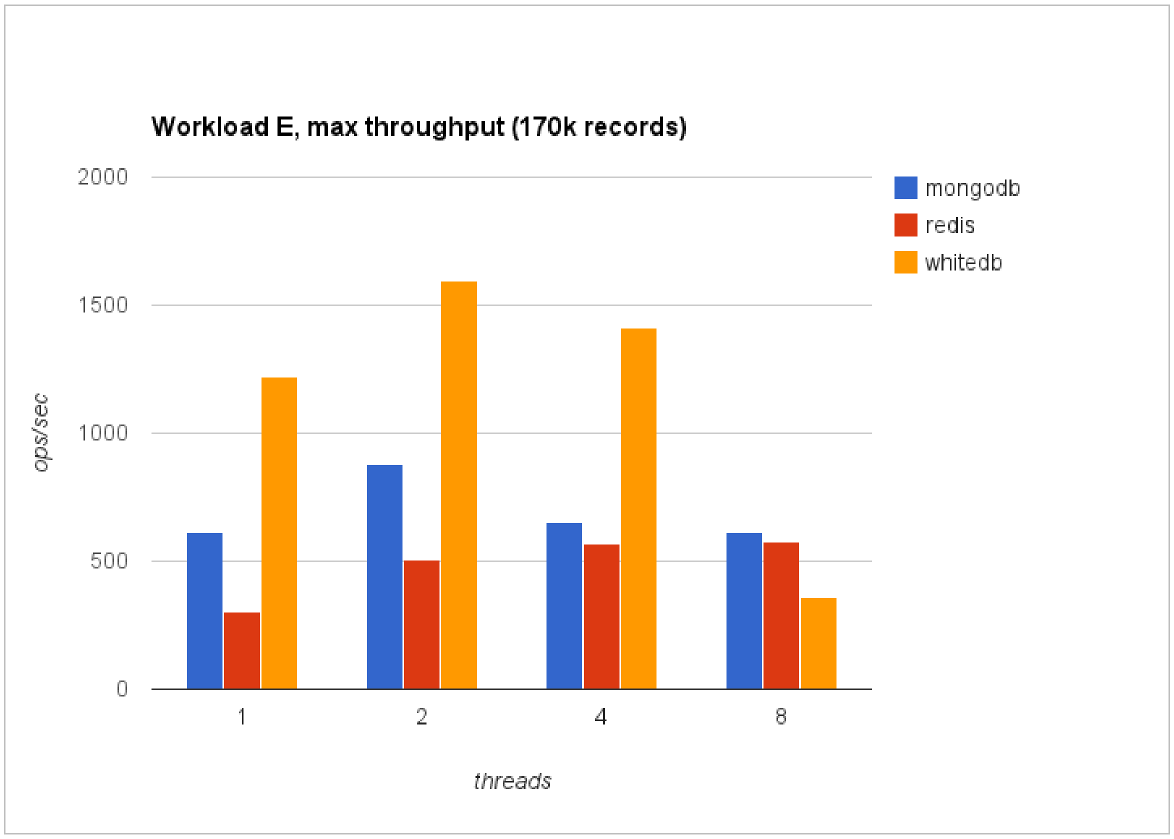
MongoDB, which has been consistently slowest with workloads that consist of single-record operations, is somewhat vindicated here. Due to the nature of the simulation, WhiteDB is able to leverage it’s query rowlimit feature and thus only reads as many records as the scan would access. This again translates to the lowest response time and highest maximum throughput.

Throughput by number of clients

In the throughput graphs, higher is better. The measured value was the unthrottled throughput (meaning that each client was constantly making queries) when accessed from 1 client thread to 8 client threads.



While WhiteDB has the superior throughput when the database is uncontended, it can be observed that it has the worst scaling behaviour when we add clients. All engines increase throughput initially, but the more mature databases are able to maintain performance when the number of client threads exceeds the number of available CPU cores (4 on the test system).



In range query tests the results are similar, except that WhiteDB's performance drops off with 8 threads. MongoDB, while faster than redis with a single client, seems also to be less tolerant to heavier concurrency.

Conclusions

The benchmark shows that WhiteDB is capable of competing with the popular NoSQL databases MongoDB and redis. With the in-memory database sizes tested, it is expected to outperform them by a large margin when doing simple, single-record database accesses.

It was also evident that the more mature engines still handle heavily concurrent load better than WhiteDB - their throughput is less affected by adding clients beyond the number of available CPU cores.