bankmark UG (haftungsbeschränkt)
Bahnhofstraße 10 | 94032 Passau | Germany
www.bankmark.de | info@bankmark.de
T +49 851 205 490 490 | F +49 851 205 490 499

NoSQL Performance Test

In-Memory Performance Comparison of SequoiaDB, Cassandra, and MongoDB

White Paper

by bankmark UG (haftungsbeschränkt)

December 2014

CONTENTS

1 INTRODUCTION

In view of the fast development of innovative IT-technologies, **NoSQL** technology is increasingly utilized in big data and real-time web application in recent years. Because NoSQL stores allow for a more agile development process and execution, they can replace **traditional relational database management systems (RDBMS)** in a large number of industrial application fields. NoSQL technology significantly improves both database scalability and usability by softening RDBMS features, such as consistency and relational model.

In this report, bankmark reports on a large series of benchmark experiments to compare publicly available NoSQL store products with SequoiaDB in in different workload scenarios. For this purpose, the bankmark team used the **Yahoo Cloud Serving Benchmark (YCSB)** suite as testing platform. The bankmark team used preset settings for all systems wherever possible and only adapted settings that caused major performance bottlenecks. For all databases official documentation as well as information from other publicly available sources was utilized. All major adaptations are documented in this report, a full report is available on request that contains all configuration settings.

In the present report, bankmark focused on the performance of each database for different use cases and ensured a maximum of comparability between different results. One aim of the experiments was to get out-of-the-box performance. On the other hand, the distributed environment required some amount of optimization to get the systems running in a clustered environment. All systems were configured for clustered use and some optimization regarding partitioning / sharding took place to get competitive results for all systems.

All tests were implemented by the bankmark team. All important details concerning the physical environment and the testing settings are specified in this testing report, a full report ensuring repeatability of all experiments is available on request.

2 RESULTS SUMMARY

In our experiments, three systems were compared, SequoiaDB[1], Cassandra[2], and MongoDB[3]. All systems were tested on a 10 node cluster in an in-memory (raw data size ¼ of total RAM) or mostly-in-memory (raw data size ½ of total RAM) setup. We used the widely accepted YCSB suite as benchmarking platform. In all experiments, all data was replicated 3 times for fault tolerance. The workloads tested were all using skewed workloads (with Zipfian or latest distribution). The detailed configuration can be seen below and in an extended report that is available on request.

The results do not show a clear winner across all experiments. Our tests with the mostly-in-memory setup show that Cassandra uses most memory and thus has to perform much more disk I/O in read heavy workloads, leading to a highly decreased performance. In this case, SequoiaDB outperforms the other systems in most cases, except for write heavy workloads, which are dominated by Cassandra. In a pure in memory setup (raw data size is ¼ of total RAM), the performance of Cassandra and SequoiaDB with SequoiaDB being faster for read requests and Cassandra being faster for write request. MongoDB is slowest in almost all test cases.

[1] http://www.sequoiadb.com/en/

[2] http://cassandra.apache.org/

[3] http://www.mongodb.org/

3 HARDWARE AND SOFTWARE CONFIGURATION

In this section, the software and hardware used in all experiments will be described. The tests were performed on a cluster located at SequoiaDB's lab. All experiments were executed on physical hardware without any virtualization layer. The base system as well as the basic MongoDB and SequoiaDB installation were installed by trained professionals. bankmark had full root access to the cluster and reviewed all settings. Cassandra was installed by bankmark.

3.1 CLUSTER HARDWARE

All experiments were performed on a 10 node cluster (five Dell PowerEdge R520 servers and five Dell PowerEdge R720 servers) for the database system and five HP ProLiant BL465c blades as YCSB clients. The hardware configuration is listed below:

3.1.1 5x Dell PowerEdge R520 (server)

- 1x Intel Xeon E5-2420, 6 cores/12 threads, 1.9 GHz
- 47 GB RAM
- 6x 2 TB HDD, JBOD

3.1.2 5x Dell PowerEdge R720 (server)

- 1x Intel Xeon E5-2620, 6 cores/12 threads, 2.0 GHz
- 47 GB RAM
- 6x 2 TB HDD, JBOD

3.1.3 5x HP ProLiant BL465c (clients)

- 1x AMD Opteron 2378
- 4 GB RAM
- 300 GB logical HDD on a HP Smart Array E200i Controller, RAID 0

3.2 CLUSTER SOFTWARE

The cluster consists of Dell PowerEdge R520, Dell PowerEdge R720 and HP ProLiant BL465c blades as physical systems, all of which are equipped with different software. All information concerning the software in use and the corresponding software versions are listed below.

3.2.1 Dell PowerEdge R520 and R720 (used as server)

- OS: Red Hat Enterprise Linux Server 6.4
- Architecture: x86_64
- Kernel: 2.6.32
- Apache Cassandra: 2.1.2
- MongoDB: 2.6.5
- SequoiaDB: 1.8
- YCSB: 0.1.4 master (brianfrankcooper version at Github) with bankmark changes (see 4.5)

3.2.2 HP ProLiant BL465c *(used as client)*

- OS: SUSE Linux Enterprise Server 11
- Architecture: x86_64
- Kernel: 3.0.13

YCSB: 0.1.4 master (brianfrankcooper version at Github) with bankmark changes (see 4.5)

4 Setup Procedure

Three systems were benchmarked using YCSB, namely Apache Cassandra, MongoDB and SequoiaDB. In the following sections, it is described how those systems were installed. The systems running at the cluster were tested with a replication factor of three and usage of three separate disks. Compression was activated if the system had support for it.

4.1 Cluster Kernel Parameters

The following parameters where changed for all systems:

- vm.swappiness = 0
- vm.dirty_ratio = 100
- vm.dirty_background_ratio = 40
- vm.dirty_expire_centisecs = 3000
- vm.vfs_cache_pressure = 200
- vm.min_free_kbytes = 3949963
- vm.max_map_count = 131072

4.2 Apache Cassandra

Apache Cassandra was installed according to the official documentation[4] on all servers. It was configured with the recommended production settings[5]. Commit log and data were assigned to different disks (disk1 for the commit log and disk 5 and disk 6 for data).

4.3 MongoDB

MongoDB was installed by trained professionals. To use all three data disks and to perform replication on the cluster, a complex schema was implemented on the systems that follows the official documentation for cluster setups[6]. Config servers were started on three of the cluster nodes. On all ten servers, a mongos instance (for sharding) was started. Each shard was added to the cluster. To use all three disks and to have three replicas, ten replica sets were distributed according to the following table (columns are cluster nodes):

	node1	node2	node3	node4	node5	node6	node7	node8	node9	node10
disk3	dg0	dg0	dg0	dg1	dg1	dg1	dg2	dg2	dg2	dg3
disk4	dg3	dg3	dg4	dg4	dg4	dg5	dg5	dg5	dg6	dg6
disk5	dg6	dg7	dg7	dg7	dg8	dg8	dg8	dg9	dg9	dg9

As MongoDB provides no mechanism to start the sharded cluster automatically, the manual startup procedure was implemented into the YCSB kit specifically for the 10 node cluster.

4.4 SequoiaDB

SequoiaDB was installed by trained professionals according to the official documentation[7]. The setup was executed according to the documentation for cluster setups[8]. SequoiaDB is able to start all instances through a cluster manager, the preinstalled init script "sdbcm" could be used to start all services. Three of

[4] http://www.datastax.com/documentation/cassandra/2.1/cassandra/install/installRHEL_t.html

[5] http://www.datastax.com/documentation/cassandra/2.1/cassandra/install/installRecommendSettings.html

[6] http://docs.mongodb.org/manual/tutorial/deploy-shard-cluster/

[7] http://www.sequoiadb.com/en/document/1.8/installation/server_installation/topics/linux_en.html

[8] http://www.sequoiadb.com/en/document/1.8/installation/configuration_start/topics/cluster_en.html

the system's nodes were chosen as catalog nodes. Three instances of SequoiaDB were started on each node, each accessing its own disk.

4.5 YCSB

YCSB has several shortcomings. It is not well suited to run multiple YCSB instances on different hosts, long running high OPS workloads, on machines with many physical cores. Furthermore, it is not very actively maintained. bankmark made several extensions and modifications to the 0.1.4 version in the main repository. Below are the most important changes:

- Add scripting to automate tests
- Cassandra driver from jbellis (https://github.com/jbellis/YCSB)
- MongoDB driver from achille (https://github.com/achille/YCSB)
  - Add batch insert function (provided by SequoiaDB)
  - Updated driver interface implementation to MongoDB 2_12 and added property flag to activate "unordered inserts" in batch mode.
- SequoiaDB driver from SequoiaDB
- Changes for multi-node setups and bulk load option

5 Benchmark Setup

The following generic and specific parameters were chosen for the benchmark run:

- Ten servers (R520 and R720) hosted the database systems and five blades as clients
- Use the sixth blade as system running the control script
- Each database system wrote data to three independent disks
- All experiments were run with replication factor 3

bankmark's YCSB kit provides workload files according to the tests defined in the statement of work:

workload1	warmup	Single load	Zipfian distribution	100% read
workload1		bulk load (1k records)	Zipfian distribution	100% read
workload2	warmup	Single load	Zipfian distribution	50% read, 50% update
workload2		bulk load (1k records)	Zipfian distribution	50% read, 50% insert
workload3	warmup	Single load	Zipfian distribution	5% read, 95% update
workload3		bulk load (1k records)	Zipfian distribution	5% read, 95% update
workload4	warmup	Single load	Zipfian distribution	95% read, 5% update
workload4		bulk load (1k records)	Zipfian distribution	95% read, 5% update
workload5	warmup	Single load	latest distribution	95% read, 5% update
workload6		bulk load (1k records)	latest distribution	95% read, 5% insert

For the data load, either the workload[1-5]-warmup or the workload[1-5] file can be used, depending on the desired load type. Each of the five workloads has been divided into a workload file for the final result and a warmup file, which is performed before the measured run. To avoid complications with YCSB's internal handling of accessing records, no inserts were performed during the warmup. Using a thread scaling experiment, we determined that 64 threads per YCSB instance worked best across all systems.

These were the remaining parameters used in the test:

- We used compression where possible
- Thread count per YCSB clients: 64
- Generate
  - 200 Million (20 Million per node) records for Test Case I and
  - 100 Million (10 Million per node) records for Test Case II test

  Each record consists of one key "user<ID>" and ten fields "Field<ID>". Default record size of YCSB (100 byte) was used, which results in an average of 1128 Bytes per record (10 fields + field names + key)
- General benchmarking procedure for each key value store:
  - Start database servers
  - Iterate over the five workloads defined in the provided workloads files:
    - Perform the data single load (no time limit, workload file workload[1-5]-warmup)
    - Pause for 30 minutes to give each system time to perform any clean up etc.
    - Run a 30 minutes warmup of the workload (workload file workload[1-5]-warmup)
    - Run workload for 30 minutes (workload file workload[1-5])
  - Stop database servers

5.1 Guidelines / Procedures

All systems performed a single load, a warmup and a measured run. For systems which support a bulk load operation, namely MongoDB and SequoiaDB, an additional bulk load test was performed after the test completed.

5.2 Configuration Matrix

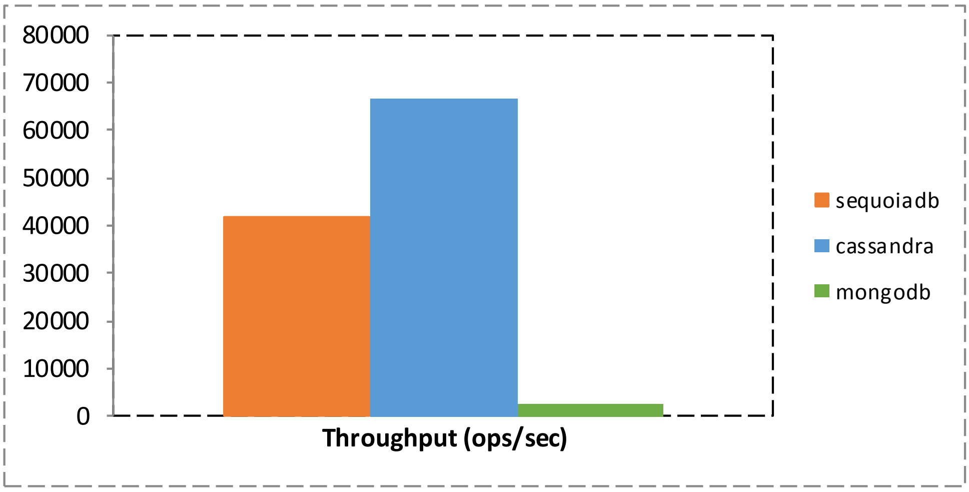
Database Options	Cassandra	MongoDB	SequoiaDB
Nodes	10 instances (1 per node)	10 "mongos" instances (1 per node) 30 "mongod" replica instances (3 per node) 3 configuration Servers (every 3rd node)	10 SequoiaDB instances 30 Replica instances (3 per node)
Disks	Log: disk 1 Data: disk5, disk6	Replicas: disk3, disk4, disk5	Replicas: disk3 disk4 disk5
Sharding /Replication	3 replicas (on db creation)	10 shards with 3 replicas each	10 shards with 3 replicas each
Compression	Yes	No (not supported)	Yes
Consistency	Read/write/scan/ delete: ONE	Read preference: nearest Write concern: Journaled	Write concern: Journaled (not changeable)
Bulk	No	Yes (1k records per batch)	Yes (1k records per batch)

6 Benchmark Results

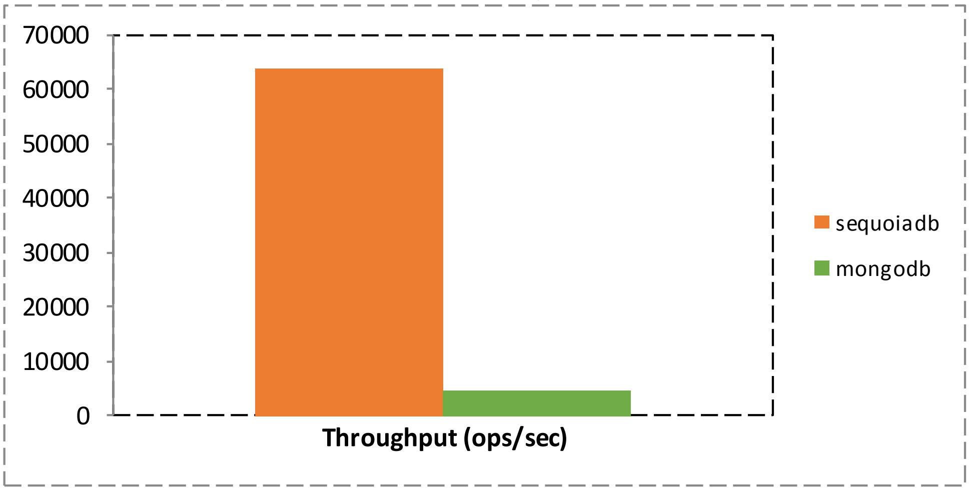
6.1 Test Case I (200 Million Records / 20 Million Records per Node)

In this experiment, the raw data size is about 45% of the system's total RAM.

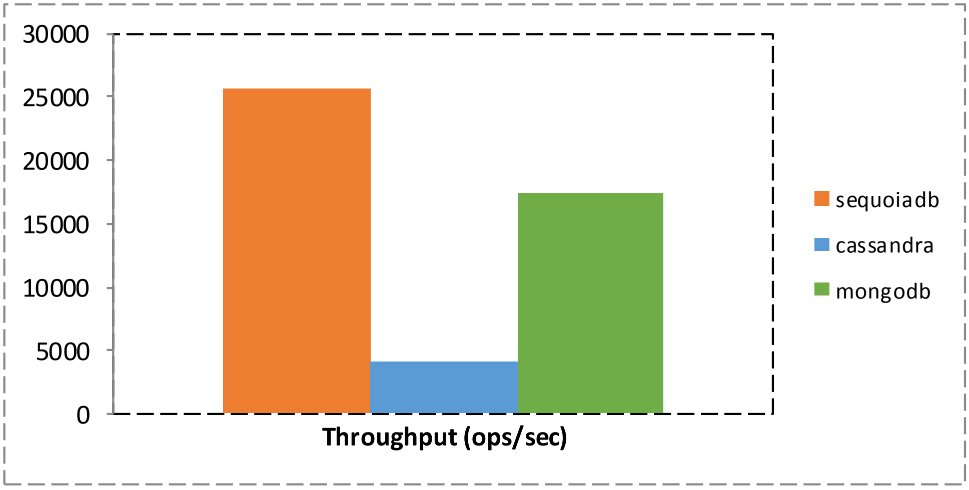
6.1.1 Load



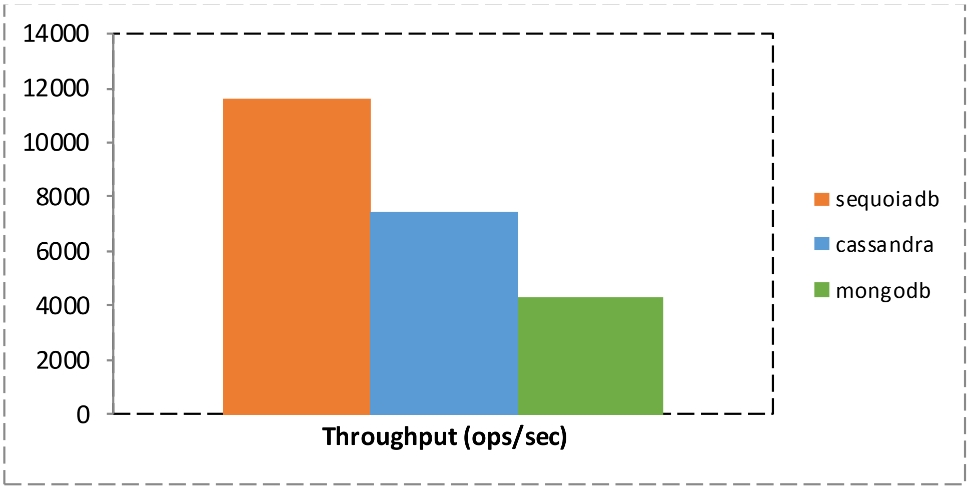
6.1.2 Bulkload (Batches each 1000 records)



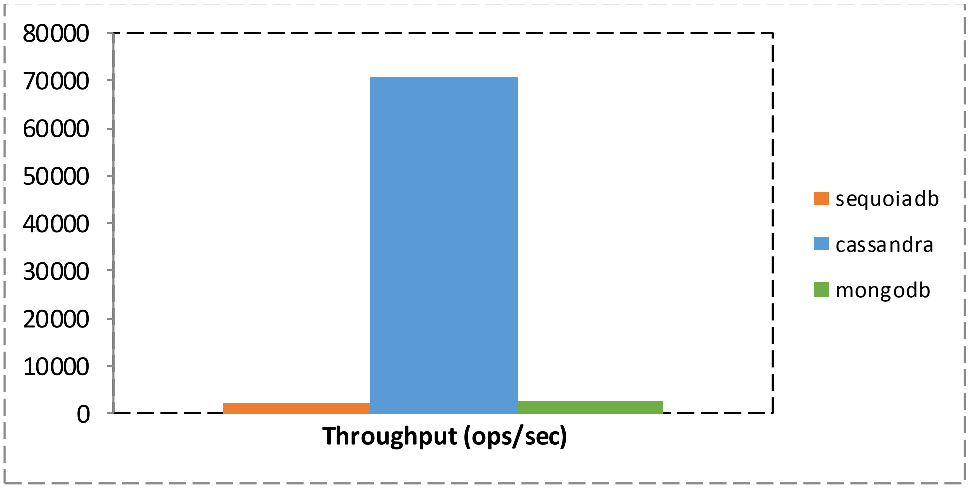
6.1.3 Workload 1, zipfian, 100% read



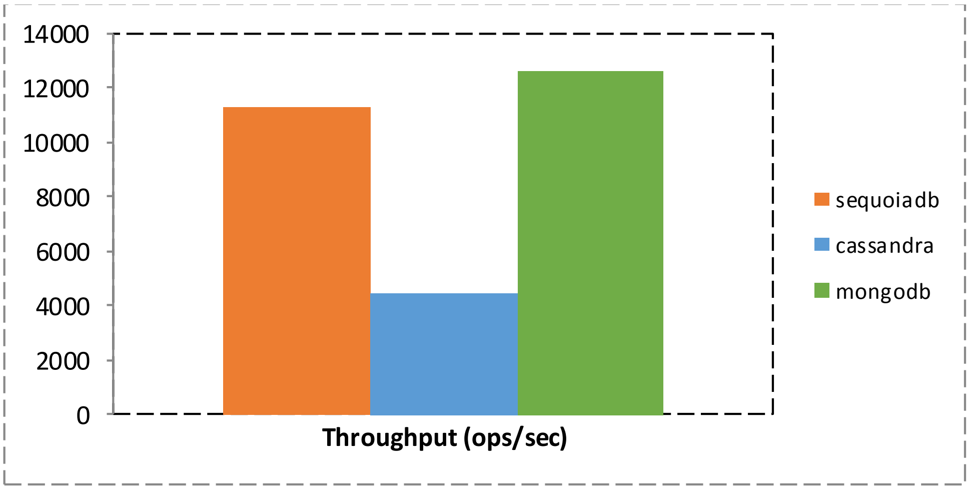
6.1.4 Workload 2, zipfian, 50% read, 50% update



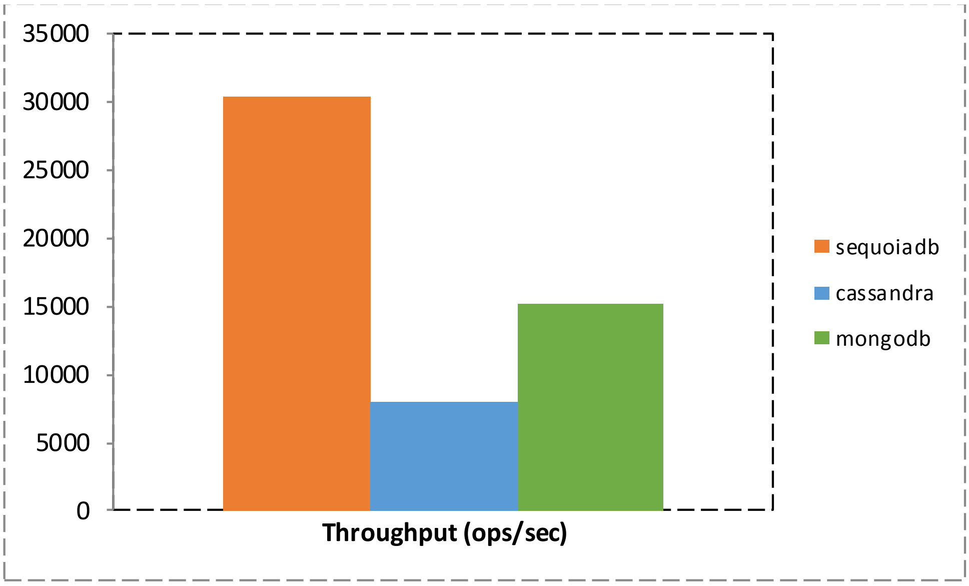
6.1.5 Workload 3, zipfian, 5% read, 95% update



6.1.6 Workload 4, zipfian, 95% read, 5% update



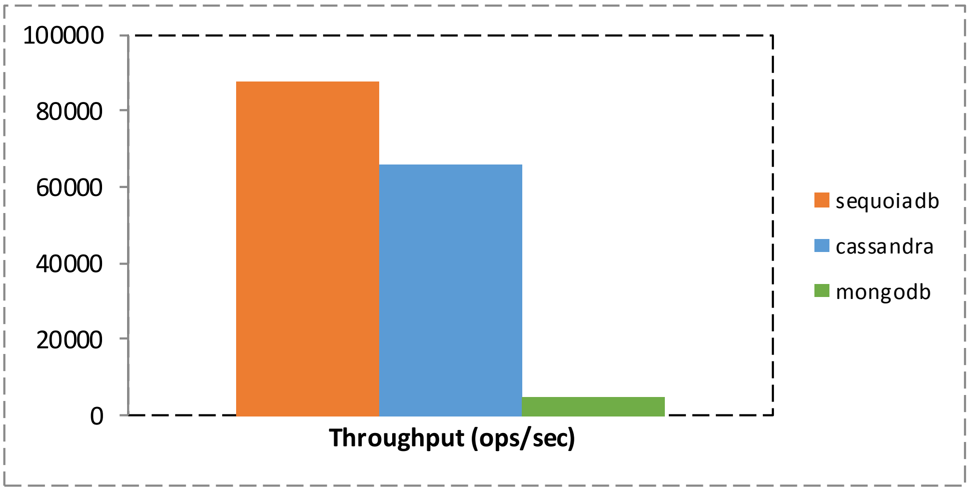
6.1.7 Workload 5, latest distribution, 95% read, 5% insert



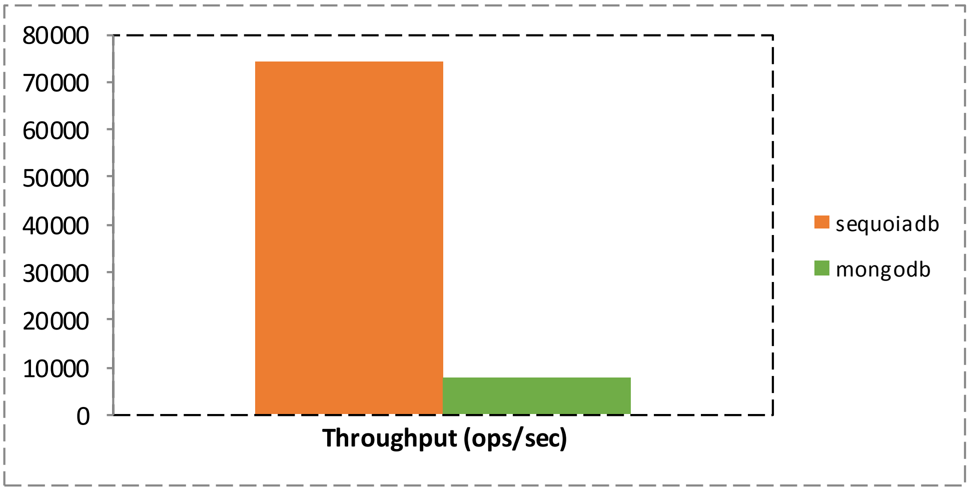
6.2 Test Case II (100 Million Records / 10 Million Records per Node)

In this experiment, the raw data size is about 22% of the system's total RAM.

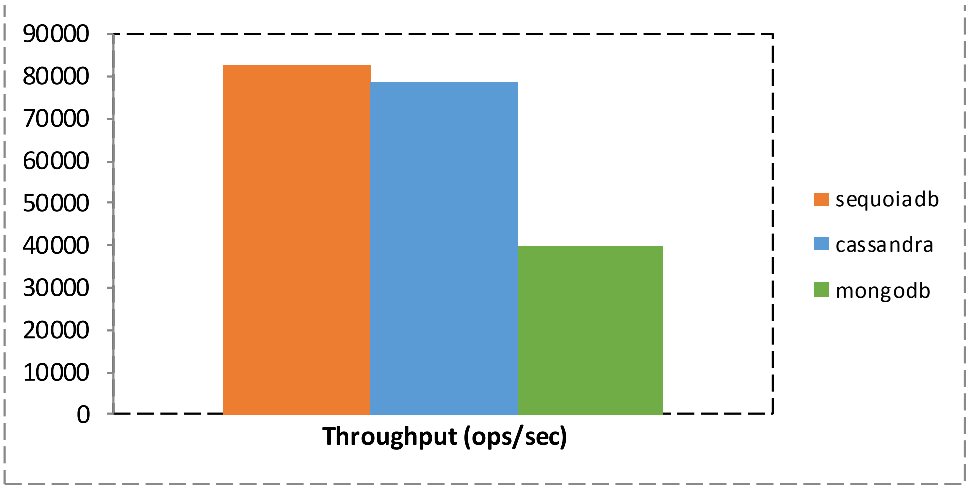
6.2.1 Load



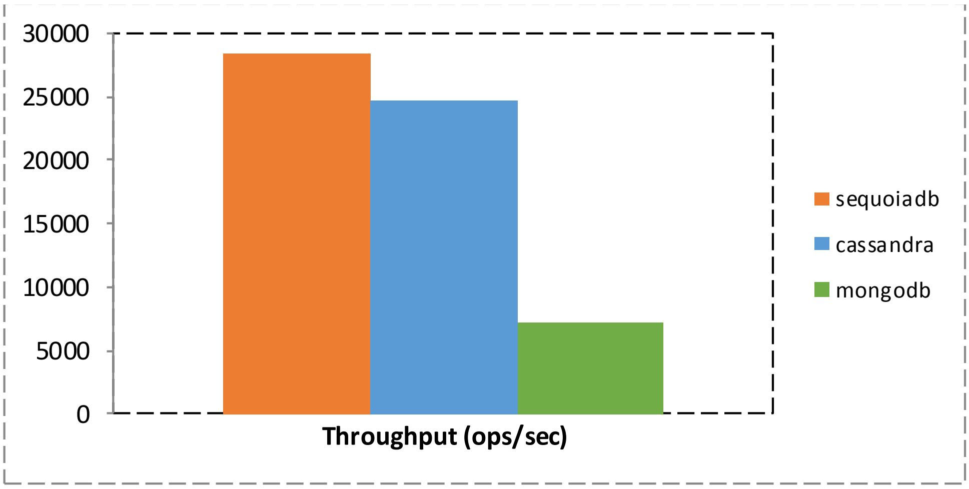
6.2.2 Bulkload



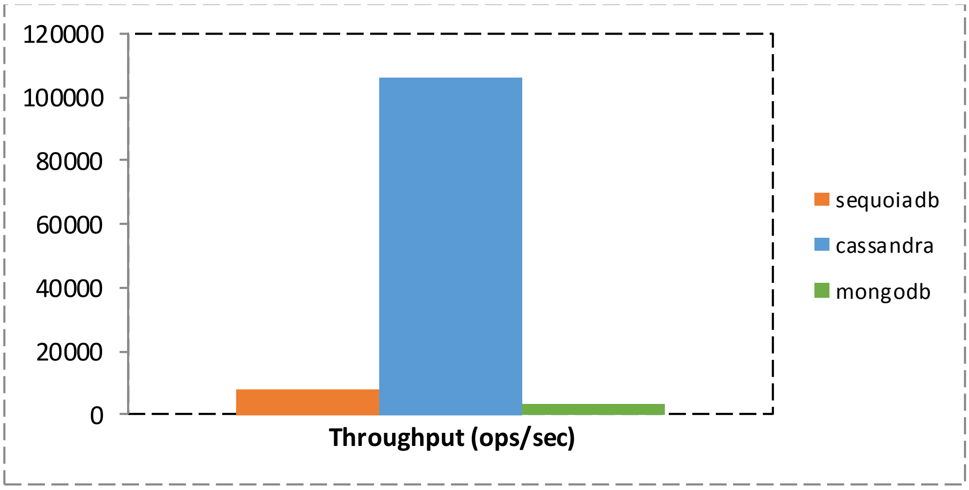
6.2.3 Workload 1, zipfian, 100% read



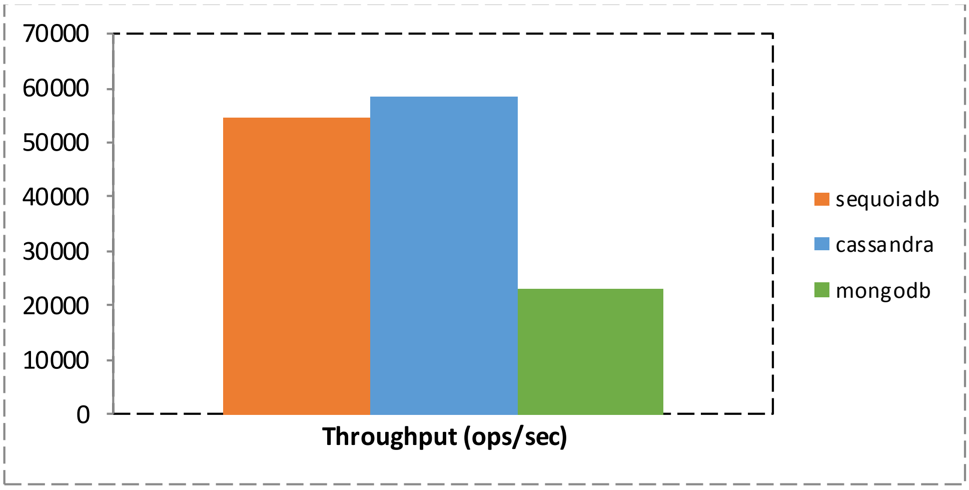
6.2.4 Workload 2, zipfian, 50% read, 50% update



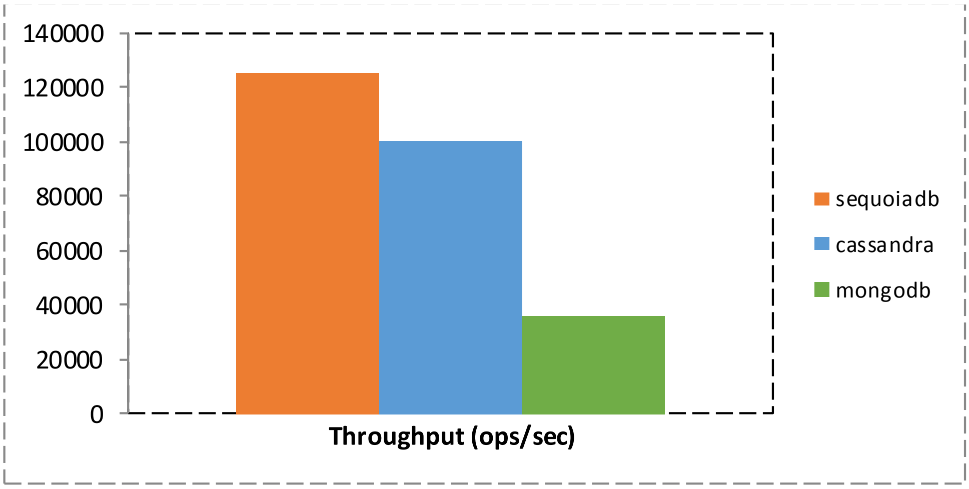
6.2.5 Workload 3, zipfian, 5% read, 95% update



6.2.6 Workload 4, zipfian, 95% read, 5% update



6.2.7 Workload 5, latest distribution, 95% read, 5% insert



7 About the Authors

Tilmann Rabl is a Postdoctoral fellow at the University of Toronto and CEO of bankmark. His research focusses on big data benchmarking and big data systems. Michael Frank is CTO of bankmark, he is the core developer of the Parallel Data Generation Framework (PDGF), which is basis for industry standard benchmarks. Manuel Danisch is COO of bankmark. He is one of the main developers of the BigBench big data analytics benchmark and the data generator for the Transaction Processing Performance Council's (TPC) benchmark TPC-DI.

bankmark is an independent benchmark institution whose mission is to revolutionize big data benchmarking. Driven by innovative technology, bankmark creates performance and quality tests as well as proof of concept systems and completely simulated productive systems efficiently and cost-effectively. Techniques based on cutting edge scientific research allow for unprecedented quality and speed.

bankmark is independent member of the industry benchmark standardization organizations SPEC and TPC and its technology is basis for benchmarks such as TPC-DI and BigBench.