Performance of NoSQL Databases in Application for Network Analysis on the QP

JaguarDB and MongoDB

Report Date: September 25, 2016

http://www.quantea.com

Summary:

Quantea performed a series of test with two NoSQL databases MongoDB and JaguarDB on their QP (Quantea Probe) network analysis systems on September 2016. The test involves inserting large amounts of network traffic (close representation to real world traffic data) to databases utilizing MongoDB and JaguarDB.

With the provided evaluation methodology, deployment conditions, use case, data type and usage, the overall conclusion is that JaguarDB outperforming MongoDB in all the tests done in this evaluation. Not only JaguarDB performed better than MongoDB, JaguarDB utilized significantly less resources.

Overview:

Each and every subsequent year, data driven applications, internet-enabled devices and their overall network usage continue to surge. Systems that perform network analysis continue to require more and more analytical capabilities at a higher throughput. The rise of NoSQL databases are rapidly becoming the data platform of choice when it comes to addressing the performance trend.

Quantea and the QP for years have been analyzing and recording billions of network packets for large telecommunication companies and enterprise; and with the rapidly developing trend, the QP will require utilization of a NoSQL data platform. Our goal for this evaluation is to determine which NoSQL data platform is best suited for the deployment conditions and use case of the QP.

Test Methodology and Configuration:

Quantea performed the benchmark by integrating it on their QP appliance which employs the industry standard x86 architecture on 24 logical core Intel E7 CPU and 64GB of DDR4 memory. The data IO is handled through the RAID based storage which is also an industry standard. The server runs on a RedHat based operating system with a standard configuration for the QP.

Each test is ran on a script that inserts information on an empty database (starting point). While the process is running, external applications were used to measure the performance and system resource usages for each test. For each trial, we run the data insertion, search and performance measurement using JaguarDB and then run the same methods using MongoDB. In order to reduce the discrepancy between each test, the system was restarted

The latest version available (as of Sept. 2016) of JaguarDB [v. 2.3.9] and MongoDB[v. 3.2.9] were evaluated for this report.

Test Methodology and Configuration (cont.):

About JaguarDB (excerpt from datajaguar.com):Jaguar is a scalable, fast NoSQL database for big data. It uses an innovative array-indexing technology in data access path. Compared to traditional B+Tree in relational database software, Jaguar employs plain arrays in data indexing. Arrays have excellent data locality, read-ahead, and caching properties. The core storage and indexing engine is 10X - 50X faster than B+Tree, therefore table join and range search are extremely fast.

Installation: JaguarDB has a precompiled binary package that is readily available to be copied and installed on the server. Server and client software was installed easily. Server software is started using the <jaguarstart> command.

```
./jag -u perfmon -p perfmon -h 10.192.168.71:8888

jaguar:eval_perfmon> create table rxstat (key asc: ts datetimenano, value: port0rxpktsec bigint, port0rxbytesec bigint, port1rxpktsec bigint, port1rxbytesec bigint, port2rxpktsec bigint, port2rxbytesec bigint, port3rxpktsec bigint, port3rxbytesec bigint, port4rxpktsec bigint, port4rxbytesec bigint, port5rxpktsec bigint, port5rxbytesec bigint, port6rxpktsec bigint, port6rxbytesec bigint, port7rxpktsec bigint, port7rxbytesec bigint, port8rxpktsec bigint, port8rxbytesec bigint, port9rxpktsec bigint, port9rxbytesec bigint, port10rxpktsec bigint, port10rxbytesec bigint, port11rxpktsec bigint, port11rxbytesec bigint);

jaguar:eval_perfmon> insert into rxstat (ts, port0rxpktsec, port0rxbytesec, port1rxpktsec, port1rxbytesec, port2rxpktsec, port2rxbytesec, port3rxpktsec, port3rxbytesec, port4rxpktsec, port4rxbytesec, port5rxpktsec, port5rxbytesec, port6rxpktsec, port6rxbytesec, port7rxpktsec, port7rxbytesec, port8rxpktsec, port8rxbytesec, port9rxpktsec, port9rxbytesec, port10rxpktsec, port10rxbytesec, port11rxpktsec, port11rxbytesec) values ('2016-09-24 16:49:14 -08:00.828462161', 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096, 0000001203012096);
```

About MongoDB (excerpt from en.wikipedia.org/wiki/MongoDB): MongoDB (from humongous) is a Free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB avoids the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (It calls the format BSON), making the integration of data in certain types of applications easier and faster.

Test Methodology and Configuration (cont.):

Installation: MongoDB was installed using a precompiled package. Installation of mongoDB required certain dependencies to be installed prior to completion. Server software is started using the <mongod> command.

/usr/bin/mongod -dbpath /vol1/part1/quantea/suite/data/db/

mongoimport --db PERFMON_STORE --collection perfmon_stat < [{"DATETIME":"09/24/2016 14:48:31.186253681"},{"PORT_NUM": "0","RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"},{"PORT_NUM": "1","RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"}, {"PORT_NUM": "2","RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"},{"PORT_NUM": "3","RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"},{"PORT_NUM": "4","RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"}, {"PORT_NUM": "5","RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"},{"PORT_NUM": "6","RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"},{"PORT_NUM": "7","RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"}, {"PORT_NUM": "8","RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"},{"PORT_NUM": "9","RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"},{"PORT_NUM": "10""RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"}, {"PORT_NUM": "11","RX_PACKETS_SEC": "0000001203012096","RX_BYTES_SEC": "0000001203012096"}] --jsonArray --maintainInsertionOrder

There are two main operations that are imperative to the functionality of the QP: intensive data insertion and query/search from a large data set.

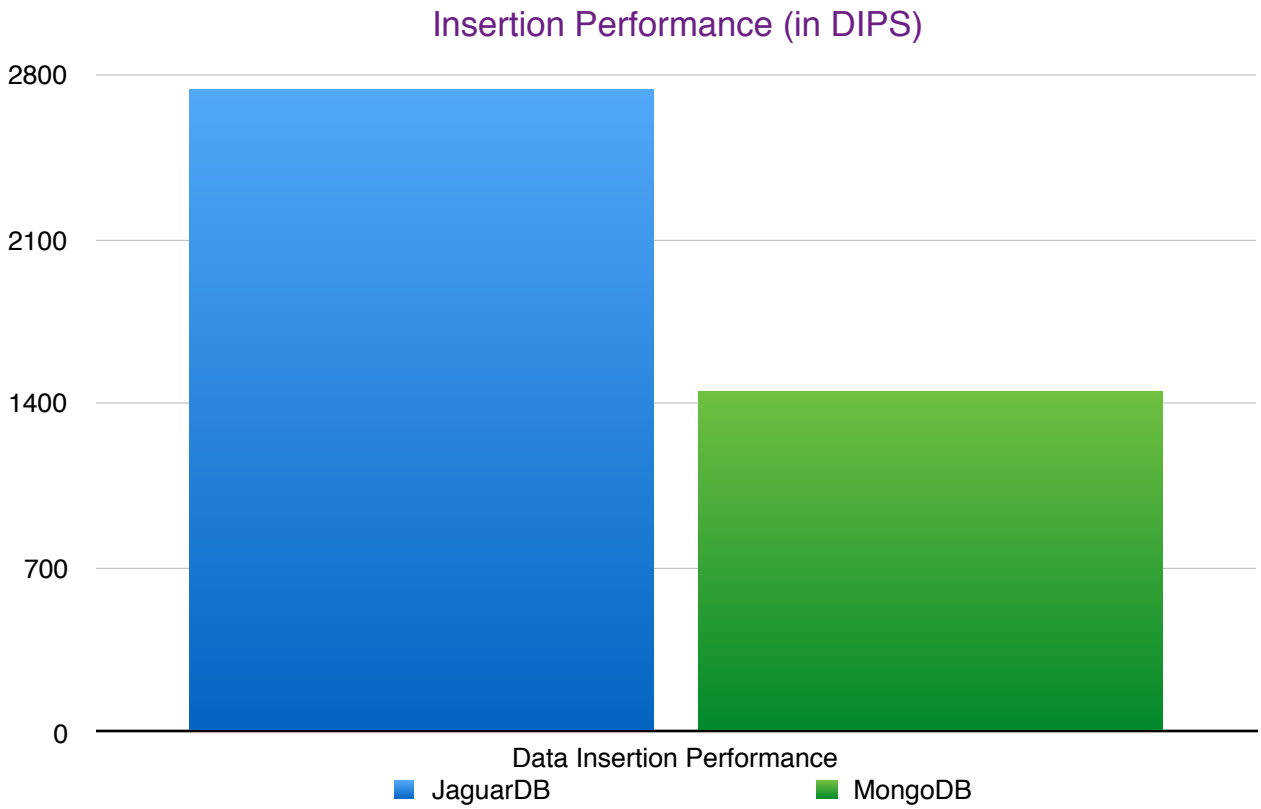
Both database platforms went through a data insertion test where the column structure described above is inserted at a maximum rate per second (data insertion per second - DIPS).

Insertion of data was performed for 48 hours for each platform and then after all the data is inserted, search query was performed for each platform.

While both databases are being tested, the memory usage and system load are concurrently measured to determine each platform's resource management. Resource usage is measured by iostat, htop/top and an internal utility used by Quantea.

[Test Results Begins on Page 5]

Test Results (Insertion):

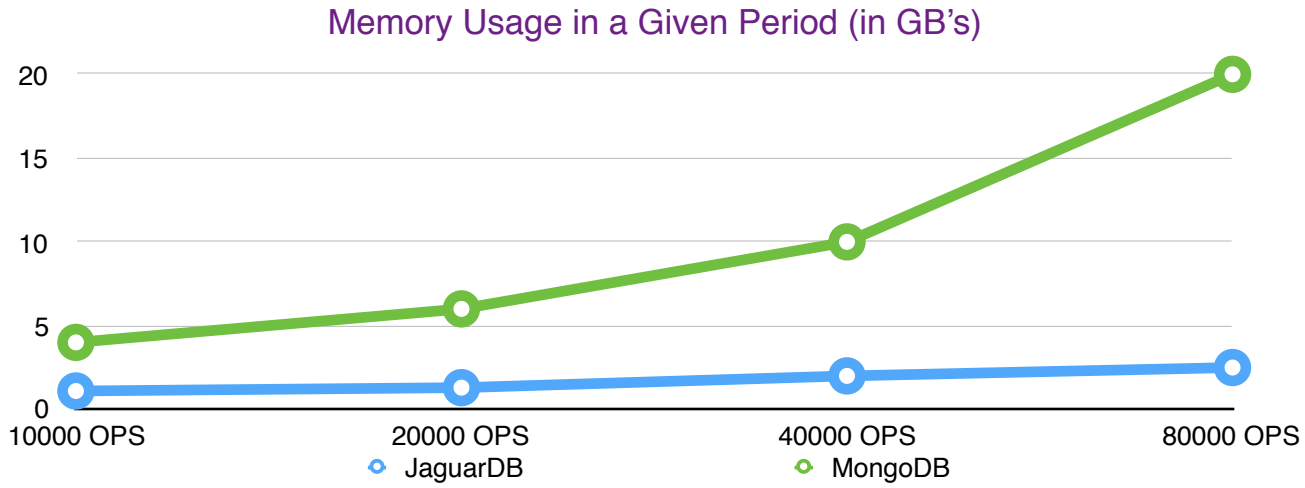


JaguarDB	2740
MongoDB	1455

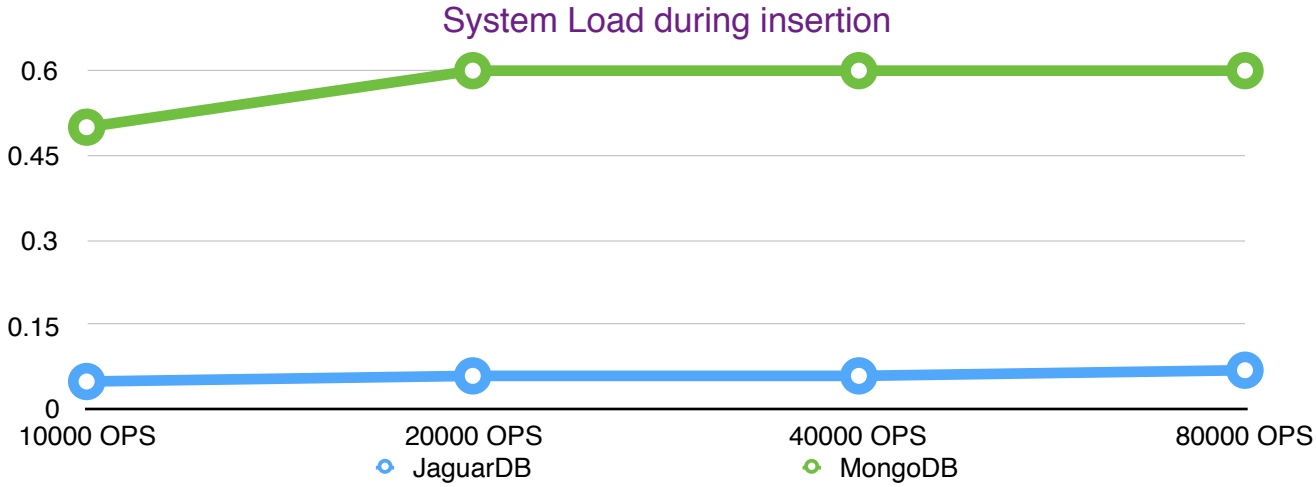
Burst mode insertion performance test was performed for both JaguarDB and MongoDB. JaguarDB's is close to 2x the performance compared to MongoDB.

[Resource Measurement Test Results on Page 6]

Test Results (Insertion Resource):



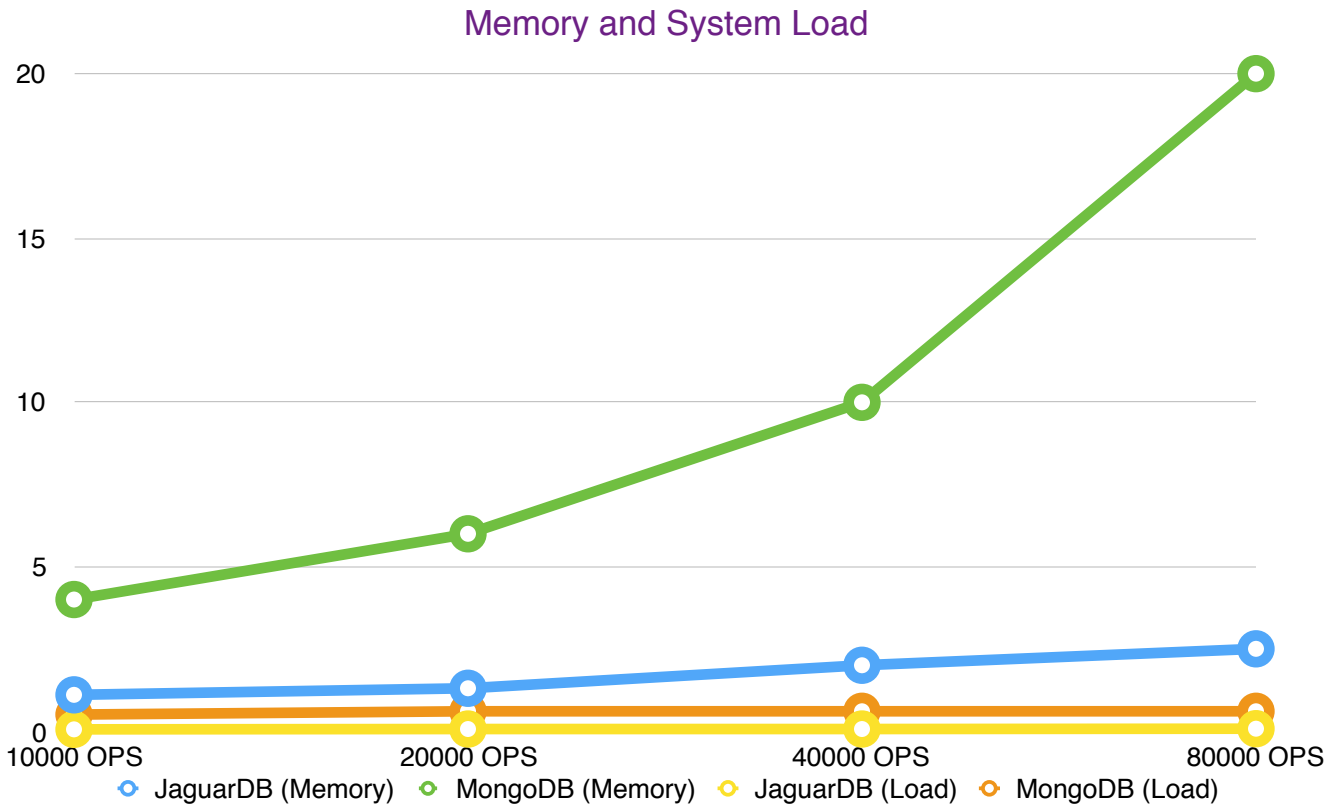
	10000 OPS	20000 OPS	40000 OPS	80000 OPS
JaguarDB	1.1GB	1.3GB	2GB	2.5GB
MongoDB	4GB	6GB	10GB	20GB



	10000 OPS	20000 OPS	40000 OPS	80000 OPS
JaguarDB	0.05	0.06	0.06	0.07
MongoDB	0.5	0.6	0.6	0.6

During the data insertion test, JaguarDB not only outperformed MongoDB but also used significantly less resources. Memory usage is almost 9x lower in JaguarDB than MongoDB and system load (logical core processing) is 10x less.

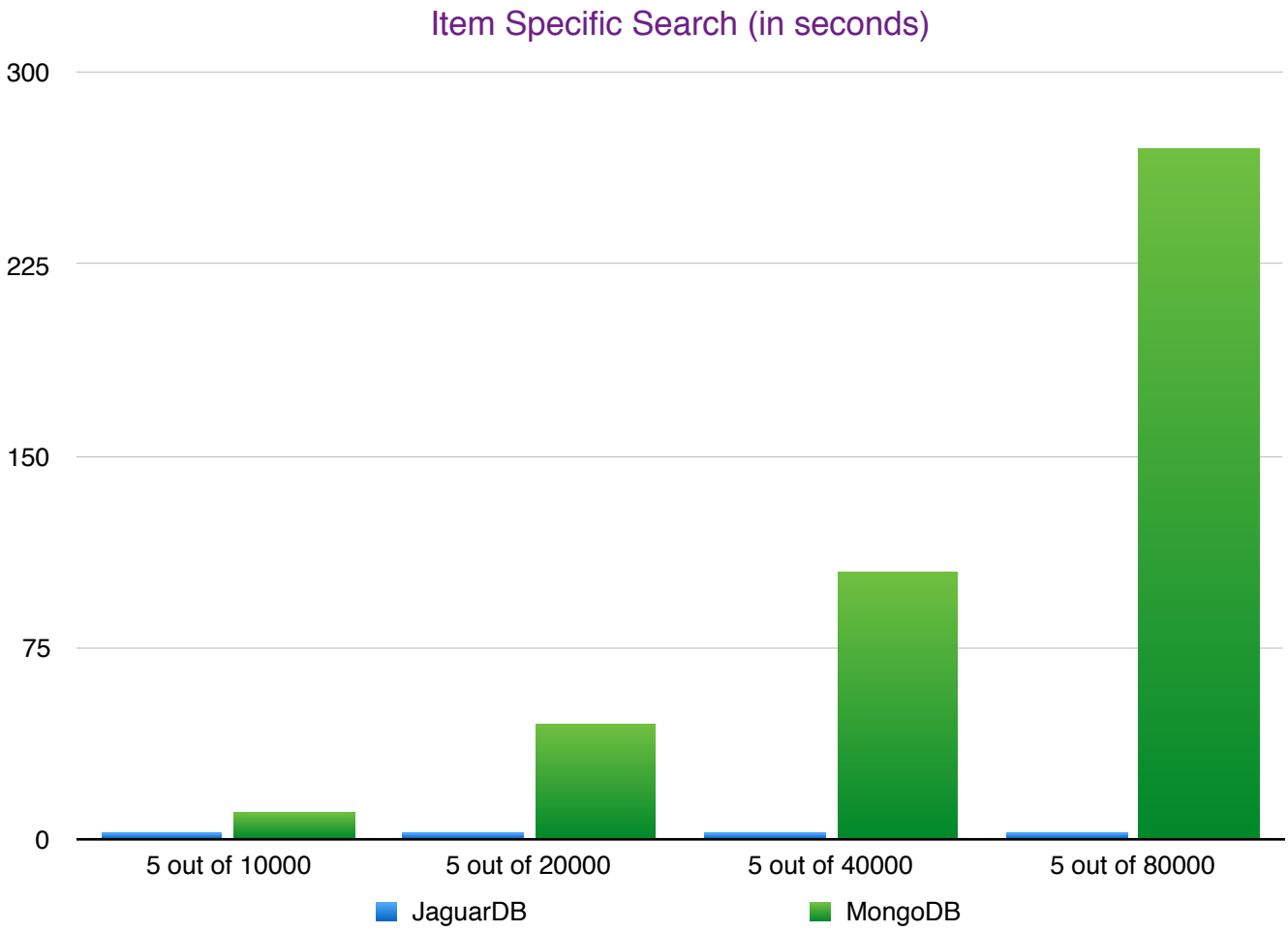
Test Results (Insertion Resource):



System Load and Memory Usage Consolidated Graph

[Search Query Performance Benchmark on Page 8]

Test Results (Search Query):

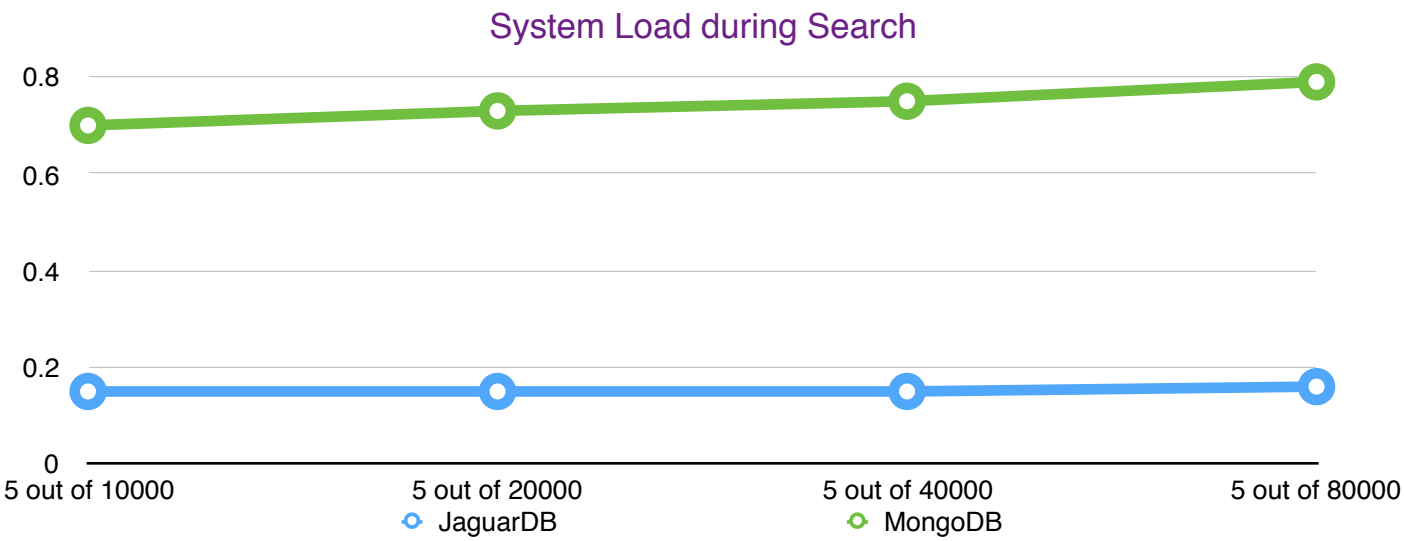


[Lower value is better]

	5 out of 10000	5 out of 20000	5 out of 40000	5 out of 80000
JaguarDB	2.5s	2.6s	2.6s	2.6s
MongoDB	11s	45s	105s	270s

The search test is where the JaguarDB greatly outperforms MongoDB. During the insertion process, JaguarDB indexes the entries right away whereas for MongoDB requires additional processing time to query information among a large data set. JaguarDB query performance remains consistent throughout the test even regarding different data set sizes and in the last test outperformed MongoDB almost 100x in query processing time.

Test Results (Search Query Resource):



	5 out of 10000	5 out of 20000	5 out of 40000	5 out of 80000
JaguarDB	0.15	0.15	0.15	0.16
MongoDB	0.7	0.73	0.75	0.79

During search, system load measurement was recorded. JaguarDB consistently has lower system load overall. MongoDB utilized all the logical cores in the system (24 cores) whereas JaguarDB used a single logical core.

Conclusion:

With the provided evaluation methodology, deployment conditions, use case, data type and usage, the overall conclusion is that JaguarDB outperforming MongoDB in all the tests done in this evaluation. Not only JaguarDB performed better than MongoDB, JaguarDB utilized significantly less resources. Having more system resources available leaves room for auxiliary applications that can run within the same server. With high performance capabilities and efficient system resource usage, Quantea has concluded that JaguarDB is the NoSQL database platform of choice for the QP network analysis system.

www.quantea.com