it's about time

Kdb+ Transitive Comparisons

15 May 2018

Hugh Hyndman, Director, Industrial IoT Solutions

Introduction

Last summer, I wrote a blog discussing my experiences running kdb+ on a Raspberry Pi, in particular making use of published benchmark content from InfluxData to generate test data, perform ingestion, and invoke a set of benchmarking queries. As a result of kdb+'s excellent performance, I concluded that it would be a perfect fit for small platform or edge computing.

I felt that I owed it to the Kx community to take things a step further: to run performance tests against all of the products that InfluxData documented, including Cassandra, ElasticSearch, MongoDB, and OpenTSDB – and go beyond the Raspberry Pi and use a variety of other server configurations.

The difficulty with doing this is that I didn't have time to install and configure these technologies (let alone on the Raspberry Pi), so I decided to take a different approach and exploit the old transitivity argument, where if *a* is greater than *b*, and if *b* is greater than *c*, then it follows that *a* is greater than *c*.

So, using this logic and taking InfluxData's benchmark results at face value, I concluded that all I had to do was run the tests on my hardware and compare my results with theirs to get a broad comparison across all the other technologies. Moreover, as InfluxDB had pretty much outperformed all the other databases in their tests, I reckoned that if kdb+ outperformed InfluxDB, then by transitivity, kdb+ was the fastest of them all!

This article summarizes the data, queries and hardware environment that I used and the resulting performance figures.

Data

The raw data for the tests was based on capturing nine categories of system and application metrics (CPU, memory, disk, disk I/O, kernel, network, Redis, PostgreSQL, and Nginx) over a 24-hour period on a standard server environment. Depending on the test being undertaken the data was extrapolated to varying numbers of servers (from 100 to 1,000) and different time periods (from 4 hours to 4 days).

All data sets were based on 100 measurements every 10 seconds yielding quite small data sets (by kdb+ standards anyway as kdb+ can easily support trillions of data points) ranging from roughly 150 million to 850 million entries. Because of this small size, I chose not to spread the data across multiple disks and partitions to benefit from the parallelism inherent in kdb+.

Queries

The table below summarizes the queries that InfluxData ran, and that I correspondingly ran on kdb+, to compare performance across other technologies. Note that the queries were not identical across each technology in recognition of the fact that they are not all times-series databases (in particular, Cassandra, MongoDB and ElasticSearch are not) so the tests were attuned to gauge the effects of concurrency and other performance characteristics that yielded best results for each technology.

Query	Definition	Compared Against	Data Spanning
Query 1	Return maximum value, by minute, in a 1-hour time frame, for 1 host	InfluxDB vs Cassandra	1 day
Query 2	Return maximum value, by minute, in a 12-hour time frame, for 1 host	InfluxDB vs Cassandra	1 day
Query 3	Return maximum value, by minute, in a 12-hour time frame, for 8 hosts	InfluxDB vs Cassandra	1 day
Query 4	Return maximum value, by minute, in a 1-hour time frame, for 1 host	InfluxDB vs ElasticSearch	4 days
Query 5	Return maximum value, by minute, in a 1-hour time frame, for 1 host	InfluxDB vs MongoDB	6 hours
Query 6	Return maximum value, by minute, in a 1-hour time frame, for 8 hosts	InfluxDB vs OpenTSDB	4 hours

Unlike the tests run in my previous blog, this time I ran the kdb+ queries between a test-harness client and the kdb+ server, which provides a more apples-to-apples comparison of performance and introduces network latency.

Hardware

I ran Queries 1 to 5 on kdb+ over three different platforms, small to large, including a Raspberry Pi, my personal MacBook Pro, and a fairly modest server. Their configurations and that of the InfluxData servers are detailed as follows.

Platform	CPU	Memory	Storage	OS	Database
Raspberry Pi	1.2Ghz quad-core ARM Cortex-A53	1GB DDR2-900 MHz	32GB Micro SDHC	Raspbian	kdb+ (32-bit)
MacBook Pro (mid-2014)	3Ghz Intel Core i7 (2 cores)	16GB DDR3-1600 MHz	500GB SSD Flash	MacOS 10.13.2	kdb+ (64-bit)
Kx Server*	3.2Ghz quad-core E5-2667v3 Xeon (20MB cache)	32GB DDR4-2133 MHz	300GB SAS 10K	CentOS 7.3.1611	kdb+ (64-bit)
InfluxData Server*	3.6Ghz quad-core E5-1271v3 Xeon (8MB cache)	32GB DDR3-1600 MHz	1.2TB NVMe SSD	Ubuntu 16.04 LTS	InfluxDB

* denotes similar server configurations for head-to-head comparisons

The configuration for Query 6 was different as the OpenTSDB tests and corresponding InfluxData tests were run in the Amazon Cloud on a 2-core m4.xlarge EC2 instance. I ran my tests on the same instance type.

Results

The table below summarizes the simple comparison of kdb+ versus other technologies by running the six queries on my Raspberry Pi, my MacBook Pro and three different specifications of the Kx Server described above (i.e., using 1, 4, and 8 cores). The three rightmost columns indicate how much faster kdb+ is than InfluxDB and, by transitivity, the other technologies.

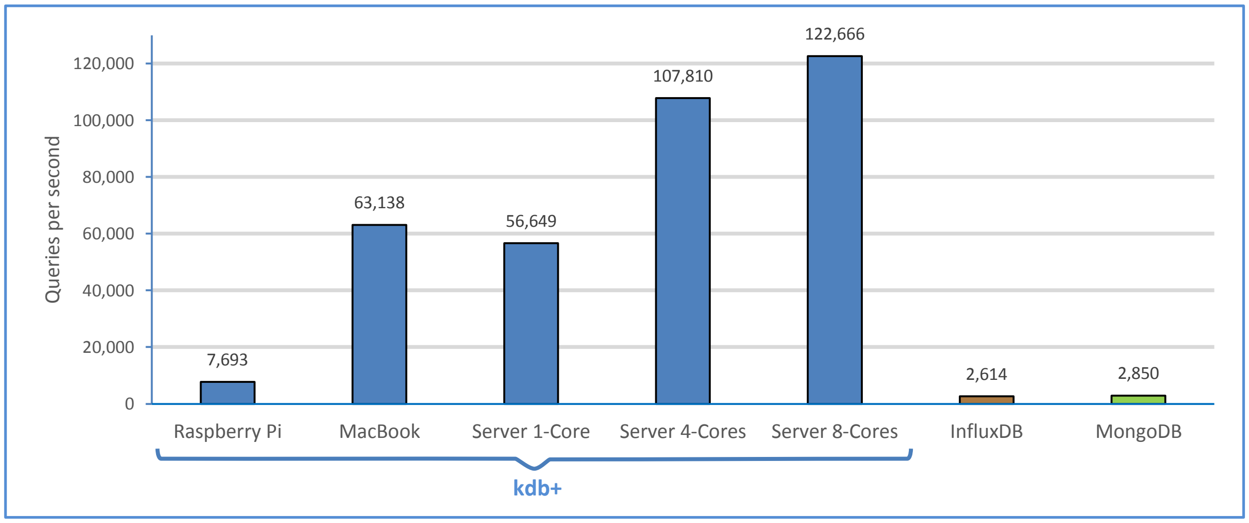
Let's look at Query 6. If kdb+ is faster than InfluxDB by 32.5 times, and InfluxDB is 3.8 times faster than OpenTSDB (i.e., 400÷106=3.8), then by transitivity we can claim that kdb+ is 123 times faster than OpenTSDB.

	Kdb+					InfluxDB		Transitive Comparisons	
Query	Raspberry Pi	MacBook Pro	Server 1-core	Server 4-cores	Server 8-cores	Server 4-cores	How much faster is kdb+?	Technology: queries/sec	How much faster is kdb+?
1	4,741	48,055	25,061	55,578	79,084	2,606	21.3×	Cassandra: 1,912	29x
2	457	4,487	3,442	12,019	21,087	714	16.8×	Cassandra: 442	27×
3	54	531	534	1,101	1,918	192	5.7×	Cassandra: 66	17×
4	1,333	24,266	12,455	34,905	53,682	3,600	9.7×	ElasticSearch: 79	442×
5	7,693	63,138	56,649	107,810	122,666	2,614	41.2×	MongoDB: 2,850	38×
6	875	7,804	5,366	13,018	17,090	400	32.5×	OpenTSDB: 106	123×

Note: units above are in queries per second

similar server configurations

Perhaps a more dramatic way of presenting these numbers is by charting one of the queries. The chart below shows the result of running Query 5 on kdb+ versus InfluxDB and MongoDB. The bars in blue are the results of my tests and the two rightmost bars are the results from the original InfluxDB tests.



Query Rate: kdb+ vs InfluxDB vs MongoDB

As the kdb+ Server 4-Cores environment most closely resembles that of the InfluxDB server, we can use its results for our quick comparison. In this case, the processing of 107,810 queries per second by kdb+ compared to 2,614 by MongoDB represents a 41.2 times faster performance. Similar charts for each of the other queries are presented in the Appendix.

Summary

Kdb+ is well-known as the world's fastest time-series database. We have industrial clients running kdb+ powered systems where up to 30-million sensor readings are being ingested per second, and over 10TB of compressed data being stored daily – all of this happening while multiple analytical queries and CEP are run against the database and inbound data streams.

I have to admit that the design of the benchmark that InfluxData published does not accurately mimic real-world IIoT applications. The test database schema is simplistic, the volumes are small and the queries are rudimentary. Because the correct technology choice is so important, and because are so many vendors out there with often lavish claims on their processing capabilities, at Kx we always impress upon our clients how important it is to base any technology choice on its performance at scale, with representative data volumes, actual ingestion load, and complex with multi-table time-series queries. This is the only way to accurately assess if the claims live up to reality and if the technology can truly serve the business. Any solution that avoids such scrutiny should be similarly avoided itself.

I am sure that it is apparent that the results are neither a scientific nor an independent assessment of kdb+'s performance capabilities, but my demonstrating that kdb+ outperform other time-series databases by one or two orders of magnitude should give readers pause for thought.

For a more rigorous view, I would suggest you visit Mark Litwintschik's blog discussing the Billion Taxi Ride Benchmarks (http://tech.marksblogg.com/billion-nyc-taxi-kdb.html).

For completely independent and audited performance benchmarks, the STAC Benchmark Council has a number of tests comparing low-latency, high volume technologies; kdb+ features well in STAC's results. You can visit STAC at https://stacresearch.com.

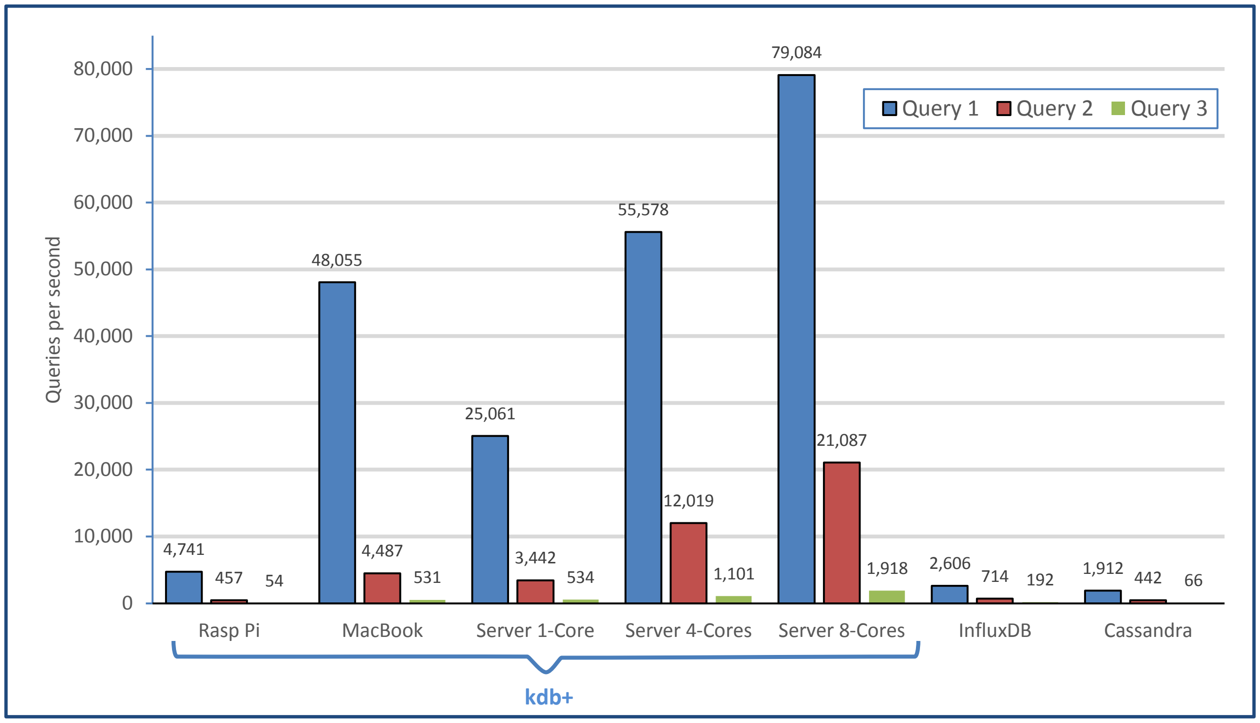
About Hugh Hyndman

Hugh Hyndman is the Director of Industrial IoT Solutions at Kx, based out of Toronto. Hugh has been involved with high-performance big data computing for most of his career. His current focus is to help companies supercharge their software systems and products by injecting Kx technologies into their stack. If you are interested in OEM or partnership opportunities, please contact Hugh. You can reach him through sales@kx.com.

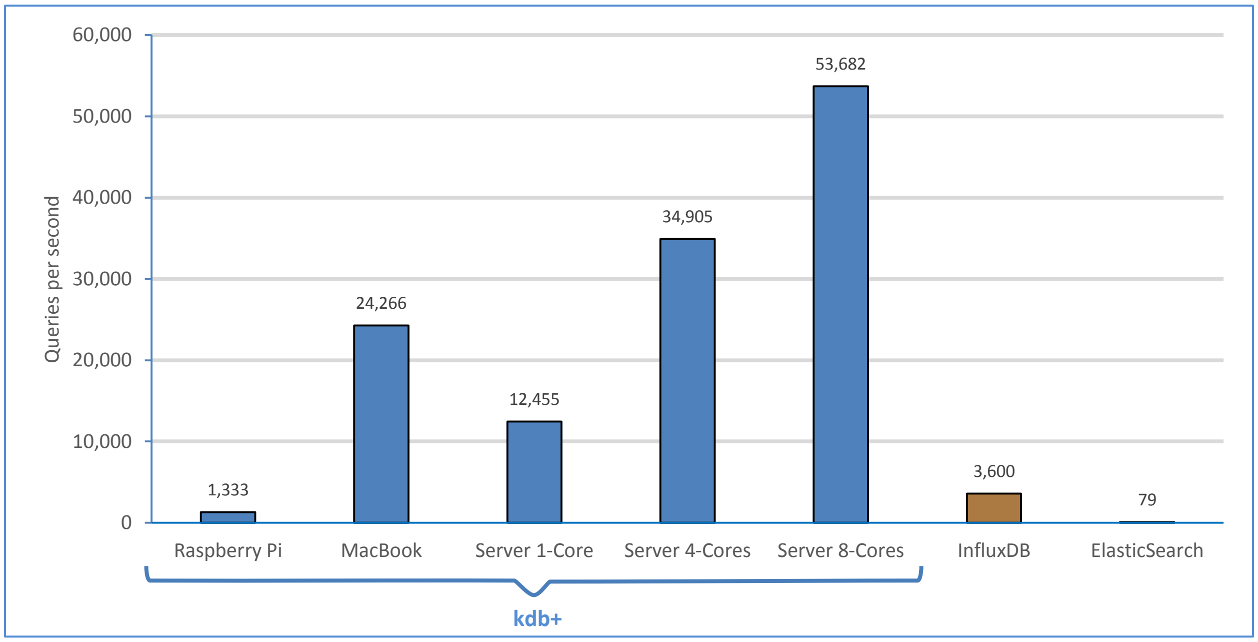
Appendix

The following charts provide a graphic depiction of the performance differences amongst the various other database products.

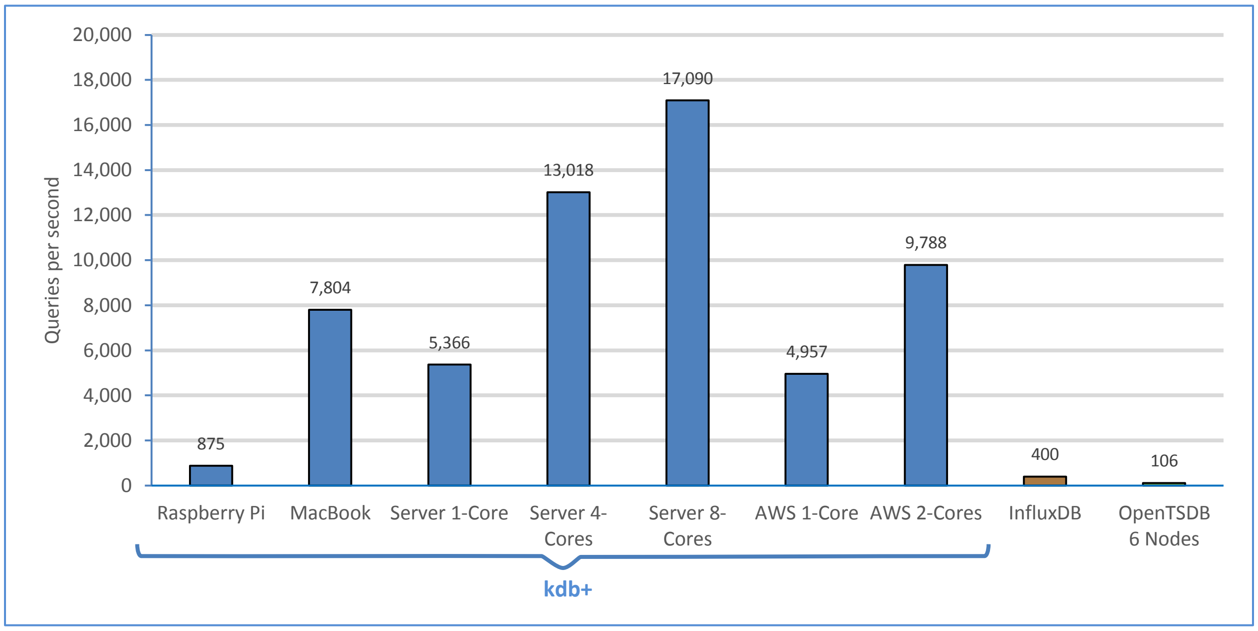
Queries 1, 2 and 3: Kdb+ vs InfluxDB vs Cassandra



Query 4: Kdb+ vs InfluxDB vs ElasticSearch



Query 6: Kdb+ vs InfluxDB vs OpenTSDB


Queries per second
Raspberry Pi
MacBook
Server 1-Core
Server 4-Cores
Server 8-Cores
AWS 1-Core
AWS 2-Cores
InfluxDB
OpenTSDB 6 Nodes
kdb+