22nd - DEC - 2016

Comparative Analysis of Databases using kdb+ as a case study

By:
Saurav Saha
Intern, ACS Lab
IIT Mandi

22nd DEC 2016

Databases & DBMS

- A database is an organized collection of related data and the way it is organized.
- A database management system (DBMS) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data.
- DBMS is designed to allow the definition, creation, querying, update, and administration of databases.

Types of DBMS

- RDBMS
  * based on the relational model
- NoSQL
  * to meet the increasing volume, velocity, and variety of data
    - Wide Column Store such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.
    - Document Store such as MongoDB where central concept of a document store is the notion of a "document".
- TimeSeries
  * optimized for handling time series data: each entry is associated with a timestamp & dealing with real-time data.

Evolution : RDMBS -> NoSQL

- Drawbacks of RDBMS :
  - Not good for large volume (Petabytes) of data with variety of data types (eg. images, videos, text)
  - Cannot scale-up, limited by memory and CPU capabilities
  - Cannot scale for large data volume
  - Sharding (break database into pieces and store in different nodes) causes operational problems (e.g. managing a shared failure)

NoSQL for Handling Big Data

NoSQL databases are more scalable and provide superior performance.

NoSQL databases address the challenges that the relational model does not by providing the following solution:

- A scale-out, shared-nothing architecture, capable of running on a large number of nodes
- A non-locking concurrency control mechanism so that real-time reads will not conflict writes
- Scalable replication and distribution – thousands of machines with distributed data
- An architecture providing higher performance per node than RDBMS
- Schema-less data model

22nd DEC 2016

Cassandra

Cassandra: Wide-column store based on ideas of BigTable and DynamoDB.

Key characteristics:

- High availability
- Incremental scalability
- Eventually consistent
- Trade-offs between consistency and latency
- Minimal administration
- No SPF (Single point of failure) – all nodes are the same in Cassandra
- AP on CAP

Good for:

- Simple setup, maintenance code
- Fast random read/write
- Flexible parsing/wide column requirement
- No multiple secondary index needed

Not good for:

- Secondary index
- Relational data
- Transactional operations (Rollback, Commit)
- Primary & Financial record
- Stringent and authorization needed on data
- Dynamic queries/searching on column data
- Low latency

Usage Case: Twitter, Travel portal

22nd DEC 2016

MongoDB

MongoDB: It is a document oriented database.All data in mongodb is treated in JSON/BSON format.

Key characteristics:

- Schemas to change as applications evolve (Schema-free)
- Full index support for high performance
- Replication and failover for high availability
- Auto Sharding for easy Scalability
- Rich document based queries for easy readability
- Master-slave model
- CP on CAP

Good for:

- RDBMS replacement for web applications
- Semi-structured content management
- Real-time analytics and high-speed logging, caching and high scalability
- Web 2.0, Media, SAAS, Gaming

Not good for:

- Highly transactional system
- Applications with traditional database requirements such as foreign key constraints

Usage Case: Craigslist, Foursquare

HBase

Hbase: Wide-column store based on Apache Hadoop and on concepts of BigTable.

Key characteristics:

- Distributed and scalable big data store
- Strong consistency
- Built on top of Hadoop HDFS
- CP on CAP

Good for:

- Optimized for read
- Well suited for range based scan
- Strict consistency
- Fast read and write with scalability

Not good for:

- Classic transactional applications or even relational analytics
- Applications need full table scan
- Data to be aggregated, rolled up, analyzed cross rows

Usage Case: Facebook message

Time Series Database

- NoSQL is a good for BigData but it is sufficiently fast ?
- NoSQL supports real time analysis using HBase, yet it is fast enough for financial sectors ?

Time Series Database overcomes these limitations of NoSQL databases.

Kdb+ is one the most stable and reliable Time Series Database out there.

kdb+

- What is kdb+ ?

kdb+ is a high-performance column-oriented database designed, developed and marketed by Kx Systems with in-memory capabilities for storing and processing large amount of time-series data. The most important feature of kdb+ is that the commonly accessed data is cached into the main memory thereby speeding up access time manifold in comparison to traditional disk access for data.

Financial institutions use kdb+ to analyze time series data such as stock or commodity exchange data. The database has also been used for other time-sensitive data applications including energy trading, telecommunications, sensor data, log data, and machine and network usage monitoring.

Why kdb+ ?

a. It's a proprietary software and so highly stable and reliable.

b. It also supports SQL like queries using q-SQL which is a big advantage for the non-programmers as they don't need to learn kdb+ and the intricacies of it's internal working.

c. It has an in-memory component called the RDB(real time database) which holds the current day's data. The historical data is stored in the historical database(HDB). The in-memory component makes it possible for kdb+ to deal with billions of real-time data entries which is the most important factor in financial institutions to deal with stock market data, commodity exchange data etc. kdb+ is known for low-latency, in-databases analysis, from both a real-time and historical perspective.

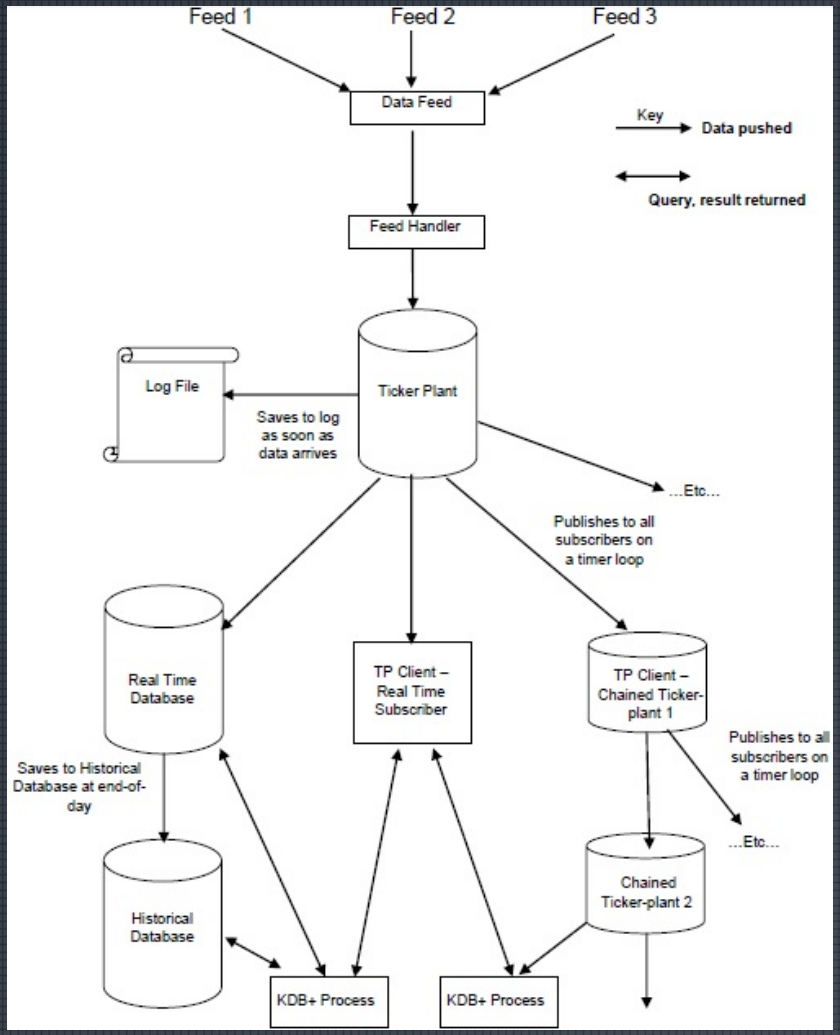
d. It also uses the map-reduce like constructs to manage and high performance querying speed of very large datasets(HDB) and low-latency.

e. Though it is built in q programming language yet it includes interfaces for all the popular languages as in C/C++, Java, C# and Python.

f. It implements a hybrid in-memory/disk architecture that is optimized to use whatever resources are available which, in our view is a better solution i.e, main memory for real-time analysis and disk for historical data analysis as in stream processing.

g. It's native support for temporal and bi-temporal features is a rare capability amongst all databases.

Kdb+ Architecture



How powerful is kdb+ ?

a. Single inserts, updates, joins and selects – millions per second per core. Consistent performance with 10s of billions of inserts per day.

b. Bulk inserts, updates, joins and selects: up to 10s of millions of bulk inserts per second which sums up to Trillions per day.

c. In-memory table scans of unrivalled speed measured in milliseconds across trillions of records.

d. Supports thousands of concurrent time series queries involving billions of rows of data.

e. Publish/subscribe mechanisms which can update hundreds of subscribers or a messaging bus in real-time.

f. Historical databases allow users to access terabytes of records in seconds.

g. Nanosecond timestamps.

h. kdb+ is 40 to 400 times faster than traditional VCF analysis (Genomic Analysis : Variant Calling) depending on the nature of the query on a Intel Xeon E5-2670 v2 processor, CPU 8 core with 61 Gigs of RAM and 2*800 Gigs of SSD.

i. kdb+ delivers a performance increase of up to 2.8x on the latest Intel® architecture (Intel Xeon E7-8890), compared to Intel systems that are a few years old (Intel Xeon E7-4890) which reveals the fact the kdb+ and Kx Systems has better support for better and latest hardwares.

Why should I learn q ? Why not SQL for kdb+ ?

kdb+ comes with its built-in functional vector processing programming language that is known as q.

Since learning a new language has it's own steep-leaning-curve, Kx Systems has built the q language keeping in mind the similarity with SQL and also extending capabilities of dealing with real-time database.

It incorporates a superset of standard SQL which is extended for time-series analysis and offers many advantages over the standard version.

Advantanges of q

a. Minimization of data traffic : Because q can operate on data directly, there is no need to first read data, then export to an external routine for analysis. Event processing can be done immediately, as data is received.

b. Virtually unlimited scalability : Lists, dictionaries, and tables are primitive data types, and the core primitives are designed for database operations — for example, doing arithmetic on tables. An operation can work just as easily on a million records, as on a single record.

c. Built-in time data types : Queries are highly optimized for time-series data.

d. Optimized performance : Data attributes such as sorted, parted, grouped, and unique can be applied to columns.

e. Query recognizability : q has database queries that are similar to counterparts in SQL, as well as functionality that goes far beyond traditional SQL.

f. Immediate feedback : The q interactive environment provides immediate feedback for rapid development.

g. Only a few lines of code : The brevity of data structures is actually one of the attributes that gives q its ability to express concisely what would take many lines in other languages.

BigData Concern : Hadoop vs kdb+ ?

The Apache Hadoop and Kx technologies are at their best when used to solve Big Data analytics. kdb+ also uses MapReduce functionality internally to deal with big data as done by Hadoop.So which one is the best?

Put simply, there is no “one-size-fits-all” answer to the question “which is best?”

Hadoop vs kdb+

1. kdb+ : high-performance, columnar-oriented database combining a functional vector processing language called q. This made it easy to rapidly compute massive amounts of time-series, streaming data. Created with financial institutions in mind, the database was developed as a central repository to store time series data that supports real-time analysis of billions of records.

   Hadoop : By Hadoop we mean the Hadoop Ecosystem consisting of HDFS, YARN, MapReduce, HBase, Hive, Pig etc. It was derived from a need to allow use cases such as web searches/queries to run significantly quicker than ever before. A big problem at that time (early 2000) was a finite amount of capability within existing server infrastructures. Specific key elements of this contribution were MapReduce and GFS from Google which resulted in Apache Hadoop.

Hadoop vs kdb+

2. kdb+ : It's especially optimized for trading market tick data, for real-time, intra-day databases and ticker plants. This model was highly adaptive to any form of time-series streaming and historical data.

   Hadoop : The elegance of the Apache Hadoop approach is that it allows a construct such as a keyword search
   or sort to work extremely quickly and has drastic improvement in fetching query results in search-engines.

3. kdb+ : kdb+ has an in-memory component called the RDB(real time database) which holds the current day's data. The historical data is stored in the historical database(HDB).

   Hadoop : Hadoop does not have any real time component. It uses the HDFS(Hadoop Distributed File System) to store the data. Hadoop doesn't have any concept of in-memory computation. However, Apache HBase, a key-value NoSQL databases which runs on the top of HDFS supports real-time querying of Big Data.

Hadoop vs kdb+

4. kdb+ : q-SQL is a SQL like language to query the kdb+ database which operates directly on the data.

   Hadoop : In the Hadoop world, the corresponding language is Hive which allows business or analyst folks to query the database without prior knowledge of Hadoop's HDFS.

5. kdb+ : kdb+ stores data as hybrid structure i.e, the real-time database in stored in RAM whereas the historical database is stored in disk. Since in-memory access if faster, performance is good in real-time and time-series analysis.

   Hadoop : Hadoop stores data in hard disk of the cluster as HDFS and brings computation to data. Speed of disk access and computation directly at the data node of cluster has trade-off. So, parallel processing is achieved in Hadoop compensating the slower speed of access the disks compared to memory.

Hadoop vs kdb+

6. kdb+ : kdb+ is a proprietary software by Kx Systems so the initial setup cost with be significant.

Hadoop : Apache Hadoop, on the other hand, is an open-source software framework by Apache Software Foundation licensed under Apache License 2.0, so the entry cost of set-up is nil leaving aside the hardware cost for cluster.

Conclusion

So, it's totally dependant on the task at hand and Infrastructure available.

Kdb+ has proved it's reliablility and stability and so 19/20 major financial banks use it.

So, pros and cons are always there so best is a subjective term which is to be decided by looking at the task at hand.

Questions

Feel free to ask out questions :)

THANKS