MinervaDB

Performance comparison of SQL based Columnar Database Systems

(ClickHouse, GreenPlum, Vertica, MariaDB ColumnStore)

By

Shiv Iyer

Disclaimer : This is not an sponsored benchmarking program by any of the vendors and we have no vested commercial interests in Greenplum, Vertica, MariaDB ColumnStore and ClickHouse. The objective of this exercise is just limited to benchmarking the performance (strictly by Response Time only) of above mentioned columnar database systems. We have not fine tuned any of the database systems for this benchmarking initiative .

What is a columnar database ?

Columnar database systems are built for Data Analytics, Let me explain you how analytical queries get benefitted by storing data in a column-oriented database against traditional row-based transactional database systems like MySQL, PostgreSQL etc. .
Imagine, for example, that you wanted to know the average sales of an retail company across the stores. Instead of looking up the sales from each row by row (in the case of row-based database), you can simply jump to the area where the "sales" data is stored and read just the data you need. So when querying, columnar database systems lets you skip over all non-relevant data very quickly, compared to row-based database systems . I have made the diagram below to explain difference between row-based and columnar database. The "**Diagram 1"** is actual logical schema of the table "SALES" . The **"Diagram 2"** represents how any typical row-based database system's physical structure layout looks like, If you see carefully the data is so well placed for transactional activities (*serialized each row of data*) . In a row-oriented indexed system, the primary key is the rowid that is mapped from indexed data.The "**Diagram 3**" is columnar database systems physical structure layout, The columnar databases boost performance by reducing the amount of data that needs to be read from disk, both by efficiently compressing the similar columnar data and by reading only the data necessary to answer the query. In the column-oriented system, the primary key is the data, which is mapped from rowids.

Logical Schema of table "SALES"

DATE	ITEM_ID	ITEM_NAME	COST	QUANTITY	TOTAL COST
01/01/2018	11011	Levis Jeans	$80	2	$160
01/01/2018	32451	GAP T-Shirt	$40	5	$200
01/01/2018	34865	iPhone X	$1300	1	$1300
01/01/2018	68432	DSLR	$3000	1	$3000

(Diagram 1)

Row-based database systems physical structure layout

Block 1	01/01/2018	11011	Levis Jeans	$80	2	$160
Block 2	01/01/2018	32451	GAP T-Shirt	$40	5	$200
Block 3	01/01/2018	34865	iPhone X	$1300	1	$1300
Block 4	01/01/2018	68432	DSLR	$3000	1	$3000

(Diagram 2)

Columnar database systems physical structure layout

Block 1	01/01/2018	01/01/2018	01/01/2018	01/01/2018
Block 2	11011	32451	34865	68432
Block 3	Levis Jeans	GAP T-Shirt	iPhone X	DSLR
Block 4	$80	$40	$1300	$3000
Block 5	2	5	1	1
Block 6	$160	$200	$1300	$3000

(Diagram 3)

Most popular columnar database systems

There are several columnar database systems available today with really strong community and enterprise support, Most of them are really good and have equally compelling propositions but in this whitepaper we have restricted our choice only to ClickHouse, Greenplum, Vertica and MariaDB Columnstore. I have explained below how each of these columnar database systems have their own respective compelling features.

ClickHouse (https://clickhouse.yandex)

- Open source columnar database system from Yandex
- Built for Massively Parallel Processing Systems , Large / complex queries can be run in parallel with minimal or no effort, The modern hardware infrastructure ready !
- Data compression - ClickHouse support data compression and this improves query performance .
- Horizontally scalable columnar database system - ClickHouse is built for web-scale data analytics, Data can be replicated across several ClickHouse Shards. ClickHouse is distributed database analytics ready columnar database system .
- In ClickHouse, data is not just stored by columns, but is also processed by vectors to achieve high CPU performance .
- Web-Scale data analytics ready - Primary keys are allowed, The data extraction for specific clients through Metrica counter over a specific time range makes low latency query analytics possible .
- Flexible aggregation - Aggregate functions for partial data with approximated calculation (minimal data retrieval option). Random keys aggregation instead of all keys for higher accuracy using minimal resources .
- Maximum availability and self healing - Asynchronous multi-master replication with auto failover capabilities .
- SQL based - ClickHouse supports SQL, JOINS, subqueries including FROM, IN, JOIN clauses; and scalar subqueries are allowed . Correlated subqueries are not allowed .

Greenplum (http://greenplum.org)

- Built on PostgreSQL source .
- Open source columnar database system from Yandex
- Hybrid storage and execution - Supports both row-based and columnar storage.
- Supports both B-tree and Bitmap indexes
- Domain specific data types and functions available .
- Store and analyze a mixture of structured, semi-structured and unstructured data in a single database engine .

- Massive parallel processing support - Efficiently use all available compute servers for parallel query processing .
- Parallel query optimizer makes building an optimal execution plan (using cost based optimizer) for both SQL and MapReduce across the cluster. Greenplum is capable of building an global execution plan for several nodes across the cluster for optimal data movement across the cluster .
- Greenplum support compression benefitting query performance and disk I/O.
- Multi-level self-healing fault tolerance - Greenplum internally uses log shipping and segment-level replication to achieve redundancy, and provides automated failover and fully online "self-healing" resynchronization .

Vertica (https://www.vertica.com)

- Built from PostgreSQL source .
- Columnar data storage and supports compression to leverage performance, I/O and storage efficiency .
- Shared nothing architecture, support Massive Parallel Processing (MPP), horizontally scalable, reliable and fault tolerant .
- Indexes and materialized views are not supported - A projection in Vertica is like a materialized view. You can create multiple projections on the same table if you need to optimize for a particular query .
- Standard SQL interface available, Seamless integration with Apache Hadoop, Kafka and Spark is possible with little or no effort .
- Flex tables support - Flex tables are like normal tables, Creating flex tables is similar to creating other tables, except column definitions are optional. When you create flex tables, with or without column definitions, HP Vertica implicitly adds a special column to your table, called __raw__. This is the column that stores loaded data. The __raw__column type is LONG VARBINARY, and its default maximum width is 130000 bytes (with an absolute maximum of 32000000 bytes) .

MariaDB ColumnStore (https://mariadb.com/kb/en/library/mariadb-columnstore/)

- MariaDB ColumnStore is a columnar storage system built by porting InfiniDB 4.6.7 to MariaDB 10.1 .
- MySQL compatible interface .
- Columnar storage reduces disk I/O, making it much faster for read-intensive analytic workloads on large datasets .
- Built in redundancy and high availability .
- Massively Parallel Processing (MPP) support .
- Online schema change, add new columns without impacting running queries .

- MariaDB ColumnStore support compression (enabled by default), This reduces disk I/O and improve SQL efficiency .
- MariaDB ColumnStore support complex statistical functions like distribution, percentile, lag and lead .
- Built for web-scale data analytics - Automatic horizontal partitioning and linear scalability
- No index, views and manual partition tuning required

Performance comparison of ClickHouse, Greenplum, Vertica and MariaDB ColumnStore

- **Dataset size - 5 Billion records**
- **Run number - Cold cache**
- **Performance is measured by " Response Time " in seconds(lower is better)**

Query	ClickHouse (1.1.53960)	Greenplum (5.1)	Vertica (8.1.x)	MariaDB ColumnStore (1.1)
SELECT sum(total_invoice) FROM booking_table	0.00174 s.	0.00851 s.	0.01629 s.	0.01827 s.
SELECT count(booking) FROM journey_table GROUP BY destination	0.00359 s.	0.04719 s.	0.04173 s.	0.04261 s.
SELECT name, passport_no, travel_from, travel_to FROM passenger_det_table WHERE travel_date >=toDate('2018-01-01') AND travel_date<=toDate('2018-01-08')	0.01478 s.	0.02753 s.	0.03172 s.	0.02641 s.
SELECT name, travel_from, gender, age, booking_status FROM holiday_table WHERE package_name = 'DUBAI'	0.15825 s.	0.25179 s.	0.19713 s.	0.19183 s.
SELECT count() AS Booking FROM package_holiday_table GROUP BY destination	1.4719 s.	2.5137 s.	1.6281 s.	1.8259 s.

Query	ClickHouse (1.1.53960)	Greenplum (5.1)	Vertica (8.1.x)	MariaDB ColumnStore (1.1)
SELECT avg(age) FROM passanger_table GROUP BY gender	1.4718 s.	2.7384 s.	2.4158 s.	2.6151 s.
SELECT name, age, gender FROM passanger_table ORDER BY nationality DESC	2.7128 s.	4.8392 s.	3.5169 s.	3.6192 s.
SELECT name, count(booking) FROM journey_table GROUP BY card_type	1.3712 s.	1.8216 s.	2.1618 s.	2.7153 s.
SELECT count(booking) FROM package_holiday_table GROUP BY package_name	1.0046 s.	1.4628 s.	1.7169 s.	2.7183 s.

Conclusion

ClickHouse performance was better compared to Vertica, Greenplum and MariaDB ColumnStore. What we observed during this benchmarking exercise was ClickHouse was naturally running queries parallely with no external hints or efforts. ClickHouse stores very compactly by supporting constant-length values. We also liked how Vertica, Greenplum and MariaDB ColumnStore handled Massively Parallel Processing (MPP), compression and linear scalability