Comparative Analysis of RDBMS and NoSQL Databases

Jam Jahanzeb Khan Behan[1], Ali Inam[2], Meesum Ali[2], Muhammad Talha Khan[2]

[1]Free University of Brussels
Bruxelles, Belgium
Email: jbehan@ulb.ac.be

[2]Institute of Business Administration
Karachi, Pakistan

Introduction (1/2)

- Big Data has been the subject of increased research since data has been termed the new oil for the 21st century
- Recently, smart grids have been used by energy providers to store the massive amount of data that is generated at regular time intervals
- We have obtained data from one such company that provides the residents of its City with electrical energy
- The company stores their data in a Not only Structured Query Language (NoSQL) database [1], since the smart grid data has a high volume, accelerated velocity, and tremendous variety
- Hence, we feel that we can provide an important comparison of NoSQL tools using this data. NoSQL tools have been actively used for storage purposes in the industry
- Companies like eBay, GitHub, and Amazon have been using these tools for storage and analytical purposes alike
- In this paper, we compare and analyze four different technologies: MySQL [2], MongoDB [3], MonetDB [4], and InfluxDB [5] using the data generated by the smart grids of KElectric

Introduction (2/2)

- The aim of this paper is to compare RDBMS and NoSQL technologies
- By analysing a relational data in a non-relational environment
- The company from which the data has been acquired uses MySQL as the main RDBMS technology for their storage and querying purposes

Data (1/2)

- The data comprises of meter readings from over 9000 smart meters spread throughout the city
- These smart meters are installed at consumer sites, on Pole Mounted Transformers (PMTs), and on distribution feeders
- The data is initially stored in an internal buffer, of each respective meter. The device then communicates with the server, based on configurable intervals (using the push/pull protocol)
- In case of any communication lapse, the infrastructure is designed to record the lost data over a period of seven days
- The data is recorded in Head End, the objective of which is to acquire meter data and monitor device parameters automatically, thus avoiding any human intervention
- The data being utilized for this project is collected over a period of three months, with an uncompressed size of approximately 45 GB
- The devices installed at the consumers end generate data over a 30 minutes interval, while the devices placed on distribution assets generate data after every 15 minutes
- As a first step, we aimed to understand the data on hand by (i) manually looking at a smaller chunk of manageable data and by (ii) asking the domain experts. We also had regular meetings with the employees of the company

Data (2/2)

- Once the data was analyzed, we gained the understanding of the fields provided in the data
- We imported the original data from the databases to use the data for querying purposes and then evaluate the query execution times
- Some fields have been highlighted as an essential part of the analysis. However, we omitted the fields: Description, Status, MeasuredUnit, and Scalar since these columns did not provide any information relevant to our analysis
- Moreover, a new field by the name of Timestamp was created by concatenation of the Time and Date fields

Field name	Field definition
DeviceID	The unique meter identification ID
Time	The time at which the reading was recorded at
Date	The date on which the reading was recorded at in MM/DD/YYYY format
Value	The profile value we obtain against the corresponding Result-TypeID
MeasuredUnit	The unit of measurement we obtain after multiplying Value with the number (10Ŝcaler)
Scaler	Represents the number (10Ŝcaler) to be multiplied with the Value to get the Value measured according to the units in MeasuredUnit
ResultTypeID	The profile for which the value has been generated.
Status	This is a 32 bit number to represent the status of the meter itself
Description	The description of the DeviceID. Not properly maintained

MongoDB

- In MongoDB, the data is stored in flexible, JavaScript Object Notation (JSON) like documents
- Fields can vary from document to document and data structure can be changed over time
- Ad hoc queries, indexing, and real-time aggregation provide powerful ways to access and analyze the data
- A distributed database, so high availability, horizontal scaling, and geographic distribution are built-in and easy to use while providing querying and indexing Functionalities
- We have selected MongoDB because it contains the best mixture obtained from RDBMS and NoSQL technologies
- It provides the data model flexibility, elastic scalability, and high performance of NoSQL databases while scaling on commodity hardware [6].

InfluxDB

- It is an open-source time-series database that is optimized for fast, high-availability storage, and retrieval of time series data
- It has no external dependencies and provides an SQL-like language with built in time-centric functions for querying
- Each point consists of several key-value pairs called the field set and a timestamp
- A series is defined when a set of key-value pairs are grouped together
- Finally, series are grouped together by a string identifier to form a measurement
- Points are indexed by their time and tag set
- Retention policies are defined on measurement and control of how data is down sampled and deleted
- Continuous queries run periodically, storing results in a target measurement.

MonetDB

- It is an open-source column-oriented database management system designed to provide high performance on complex queries against large databases, such as combining tables with hundreds of columns and millions of rows
- Its architecture is represented in three layers, each with its own set of optimizers
  - The front-end provides a query interface for SQL, where queries are parsed into domain-specific representations, like relational algebra for SQL, and optimized.
  - The middle or back-end layer provides several cost-based optimizers
  - The bottom layer is the database kernel, which provides access to the data stored in Binary Association Tables, where each table consists of an Object-identifier and value columns, representing a single column in the database
- Internal data representation also relies on the memory addressing ranges of contemporary CPUs using demand paging of memory mapped files and, thus, departing from traditional DBMS designs involving complex management of large data stores in limited memory
- We have selected MonetDB because it has been designed to provide high performance on complex queries against large databases and also because it has been applied in high-performance applications for better analytics

Queries, Results, and Conclusion

- We have written queries of different categories for each database and ran them on the AWS instances. Due to their complex nature, we were unable to perform UNION queries for MongoDB and InfluxDB [7]. The queries belong to one of the following categories:
    1. Simple Query
    2. Range Query
    3. Aggregated Query
    4. Nested Query
    5. UNION Query
- To provide an unbiased experimental runtime, the repeated the experiments 10 times. As it can be seen from the table, all the technologies were able to obtain the same results (in terms of the number of records) for identical queries. Hence, we compare the results based on the Runtime(s) columns in the table that correspond to the average time taken to obtain the results while computing on the given technology instance
- The results obtained from MySQL serve as a baseline for the NoSQL technologies, and it is safe to say that all the NoSQL technologies were able to obtain better results than the baseline. It can be observed that MonetDB was able to outperform MySQL and was still able to provide results for all categories of queries. It is also worth mentioning that InfluxDB outperformed all the systems in terms of computational time

Query Number	Result	Runtime (s)
MongoDB 1	84965 row(s) returned	0.65
MongoDB 2	781 row(s) returned	57.26
MongoDB 3	1 row(s) returned	0.04
MongoDB 4	697409 row(s) returned	0.88
MongoDB 5	33856 row(s) returned	854.75
MongoDB 6	33856 row(s) returned	874.18
MongoDB 7	0 row(s) returned	1054.73
InfluxDB 1	84965 row(s) returned	1.38
InfluxDB 2	781 row(s) returned	13.55
InfluxDB 3	1 row(s) returned	10.85
InfluxDB 4	697409 row(s) returned	0.08
InfluxDB 5	33856 row(s) returned	15.37
InfluxDB 6	33856 row(s) returned	10.37
InfluxDB 7	0 row(s) returned	58.88
MonetDB 1	84965 row(s) returned	8.54
MonetDB 2	781 row(s) returned	60.46
MonetDB 3	1 row(s) returned	57.19
MonetDB 4	697409 row(s) returned	1.88
MonetDB 5	33856 row(s) returned	78.64
MonetDB 6	33856 row(s) returned	76.63
MonetDB 7	0 row(s) returned	256.15
MonetDB 8	101444 row(s) returned	265.65
MonetDB 9	3 row(s) returned	9054.64
MySQL 1	84965 row(s) returned	134.70
MySQL 2	781 row(s) returned	121.69
MySQL 3	1 row(s) returned	117.76
MySQL 4	697409 row(s) returned	121.81
MySQL 5	33856 row(s) returned	148.98
MySQL 6	33856 row(s) returned	158.33
MySQL 7	0 row(s) returned	451.45
MySQL 8	101444 row(s) returned	473.36
MySQL 9	3 row(s) returned	18184.06

Related Work

- Hadjigeorgiou et al. [8] have compared the performance of MongoDB and MySQL when they are scaled and sharded. They conclude that the most important factor was the query type used as MongoDB handled more complex queries faster, due mainly to its simpler schema and performed better during insertions. MySQL performs better when deleting data since it performs better in simple search queries. The authors highlight that both databases have had a linear trend in the benchmarks
- Ansari et al. [9] selected Hbase, MongoDB, Cassandra, and Elasticsearch NoSQL technologies and compared them using structural column-based data from smart grids. They compared the databases on effectiveness and scalability. The results showed that Cassandra had the smallest average latency in both read and write processes
- Venkatraman et al. [10] discussed the four main data models of non-relational databases and compared them to SQL databases. They state that Couchbase processes more operations per second with lower average latency in reading and writing data than both MongoDB and Cassandra. Also, Cassandra is faster in writing than MongoDB, however, both have almost equal reading speed. The authors conclude that the flexible data modeling of NoSQL is well suited to support dynamic scalability and improved performance for Big Data analytics
- Santos et al. [11] have used Geographic Information Systems (GIS) data to compare PostGIS, MongoDB, and Neo4j-Spatial. The authors have performed different types of operations, where each operation contains a group of queries. Even though all groups include 20 parameterized queries, the parameter values vary within predefined ranges for each group. In the conclusion, the authors have highlighted that, since the spatial attributes are much more complex to handle as compared to strings, numbers, and other relational data types, evaluating and benchmarking spatial DBMS performances is not as simple as doing so in RDBMS

References

1. S. Edlich, “Your Ultimate Guide to the Non-Relational Universe!” http://nosql-database.org/, [Online; accessed April 23rd, 2020]
2. Oracle Corporation, “MySQL,” https://www.mysql.com/, [Online; accessed April 23rd, 2020]
3. MongoDB, Inc., “MongoDB,” https://www.mongodb.com/, [Online; accessed April 23rd, 2020]
4. MonetDB B.V., “MonetDB,” https://www.monetdb.org/Home, [Online; accessed April 23rd, 2020]
5. InfluxData Inc, “InfluxDB,” https://www.influxdata.com/, [Online; accessed April 23rd, 2020]
6. C. Kristina and D. Michael, “MongoDB: The Definitive Guide,” 2010
7. Joson Morn, “Issues with InfluxDB,” https://groups.google.com/forum/msg/ influxdb/jGVE3uDStNg/9KYxjY46AQAJ, [Online; accessed April 23rd, 2020]
8. C. Hadjigeorgiou et al., “RDBMS vs NoSQL: Performance and Scaling Comparison,” MSc in High, 2013
9. M. H. Ansari, V. T. Vakili, and B. Bahrak, “Evaluation of big data frameworks for analysis of smart grids,” J. Big Data, vol. 6, 2019, p.109
10. S. Venkatraman, K. Fahd, S. Kaspi, and R. Venkatraman, “SQL versus NoSQL movement with Big Data Analytics,” International Journal of Information Technology and Computer Science, vol. 8, no. 12, 2016, pp. 59–66
11. P. O. Santos, M. M. Moro, and C. A. D. Jr., “Comparative Performance Evaluation of Relational and NoSQL Databases for Spatial and Mobile Applications,” in Database and Expert Systems Applications - 26th International Conference, DEXA 2015, Valencia, Spain, September 1-4, 2015, Proceedings, Part I, ser. Lecture Notes in Computer Science, Q. Chen, A. Hameurlain, F. Toumani, R. R. Wagner, and H. Decker, Eds., vol. 9261. Springer, 2015, pp. 186–200.