Database Management System Design for the Environment with Big Data

Zhihao Wang

Cooper Union

ECE 464 Databases

Prof. Sokolov

1. Introduction **3**

2. Advantages and Disadvantages **4**

3. Physical Data Modeling and Storage **5**

4. Data Management and Scalability **6**

5. Queries **7**

6. Robustness and Recovery **8**

7. Conclusion and Benchmarking **8**

8. References **9**

1. Introduction

Concerning the question “How much data can be considered as Big Data?”, different eras have given different answers. In the early 1980s, big data was very large data that needed to be stored on tens of millions of tapes. In the 1990s, big data was data that exceed the storage capacity of a single desktop. Big data is data that is difficult to store in a relational database and cannot be processed by independent data analysis and statistical tools. For organizations of all sizes, data management has become a key differentiator that can determine the market winners. Companies and government agencies are beginning to benefit from the innovations of Internet pioneers. These organizations are defining new plans and re-evaluating existing strategies to study how to use big data to transform their businesses. In the process, they learned that big data is not a single technology, technique or initiative. Instead, this is a trend that crosses many areas of business and technology.

One ideal Big Data DBMS design is to utilize NoSQL with MapReduce programming. The main example that will be demonstrated below is MongoDB with Hadoop. With the DBMS designed using these mechanism and model to adapt to Big Data, organizations are able to build new applications that were not possible before, to quantify and satisfy user demands, and to reduce costs.

2. Advantages and Disadvantages

Although the NoSQL system is not yet mature, it is still a more effective solution to solve massive data in comparison to traditional SQL supporting the ACID properties. The main advantage the NoSQL can provide for Big Data is to avoid unnecessary complexity. Relational databases provide a variety of features and strong consistency, but many features can only be used in some specific applications, and most functions are rarely used. NoSQL systems provide less functionality to improve performance. NoSQL DBMS also provides High throughput. The throughput of some NoSQL data systems is much higher than traditional relational data management systems. For example, Google uses MapReduce to process 20PB of data stored in Bigtable every day. NoSQL also supports high level expansion capability and low-end hardware cluster. NoSQL data systems can scale horizontally well. Unlike relational database clustering methods, this kind of expansion does not require a large price. The design concept based on low-end hardware saves a lot of hardware overhead for users who adopt NoSQL data systems. It then avoids expensive object-relational mapping. Many NoSQL systems can store data objects, which avoids the cost of transforming the relational model in the database and the object model in the program. Overall, NoSQL provides more economic and efficient system for data management.

However, as previously stated, NoSQL is still immature with a lot of disadvantages for improvements. The data model and query language have not been mathematically verified. The query structure of SQL based on relational algebra and relational calculus has a solid mathematical guarantee. Even if a structured query itself is complex, it can obtain all the data that meets the conditions. Because NoSQL systems do not use SQL, some models used do not have a sound mathematical foundation. This is also one of the main reasons for the disorderness of NoSQL systems. ACID properties are not supported in NoSQL. This brings advantages and disadvantages to NoSQL. After all, transactions are still required in many situations. The ACID feature enables the system to ensure that online transactions can be accurately executed in the event of an interruption.

With most NoSQL systems provide simplified functions, which increases the burden on the application layer. For example, if the ACID feature is implemented at the application layer, the task will be difficult for the developers writing the code.

3. Physical Data Modeling and Storage

MongoDB implements data storation as documents with Binary JSON(BSON). Documents that tend to share a similar structure are organized as collections. It may be helpful to think of collections as being analogous to a table in a relational database: documents are similar to rows, and fields are similar to columns. MongoDB documents tend to have all data for a given record in a single document, whereas in a relational database information for a given record is usually spread across many tables. With the MongoDB document model, data is more localized, which significantly reduces the need to JOIN separate tables. The result is dramatically higher performance and scalability across commodity hardware as a single read to the database can retrieve the entire document containing all related data. MongoDB BSON documents are closely aligned to the structure of objects in the programming language. This makes it simpler and faster for developers to model how data in the application will map to data stored in the database.

MongoDB with Hadoop implements the Hadoop Distributed File System (HDFS): A distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster. HDFS provides a write-once-read-many, append-only access model for data and is optimized for sequential reads of large files (64MB or 128MB blocks by default). Also, HDFS is designed for high-throughput, rather than low-latency, which matches perfectly with MongoDB.

4. Data Management and Scalability

Any machine will have physical limitations: memory capacity, hard disk capacity, processor speed, etc. We need to make a trade-off between these hardware limitations and performance. For example, the memory read speed is much faster than the hard disk, so the memory A database has better performance than a hard disk database, but a machine with 2GB of memory cannot put all 100GB of data into the memory. Perhaps a machine with 128GB of memory can do it, but when the data is increased to 200GB, it is powerless. MongoDB provides horizontal scale-out for databases on low cost, commodity hardware or cloud infrastructure using a technique called sharding. Sharding allows MongoDB deployments to address the hardware limitations of a single server, such as bottlenecks in RAM or disk I/O, without adding complexity to the application. MongoDB automatically balances the data in the sharded cluster as the data grows or the size of the cluster increases or decreases. Unlike relational databases, sharding is automatic and built into the database. There are multiple kinds of sharding that allows MongoDB to deliver high scalability across diverse set of workloads.

5. Queries

The method to connect MongoDB with Hadoop provides the users the capability to perform complex and retroactive queries for Big Data analysis. MapReduce queries will be used in this design. MapReduce queries execute complex data processing that is expressed in JavaScript and executed across data in the database. MongoDB automatically optimizes queries to make evaluation as efficient as possible. Evaluation normally includes selecting data based on predicates, and sorting data based on the sort criteria provided. The query optimizer selects the best index to use by periodically running alternate query plans and selecting the index with the best response time for each query type. The results of this empirical test are stored as a cached query plan and are updated periodically. Developers can review and optimize plans using the powerful explain method and index filters.

6. Robustness and Recovery

A backup and recovery strategy is necessary to protect the essential data against catastrophic failure, such as unexpected disasters in a data center, or human error, such as code errors or accidentally dropping collections. Applying regular backups offers users the advantage to restore operations without data loss. A cloud based backup system should be maintained continuously. For example, MongoDB contains built-in high availability and data replication which provides such convenience.

7. Conclusion and Benchmarking

The database benchmark refers to a set of specifications to evaluate and compare different database systems. According to the new benchmarking standard designed for database proposed in (Jin, 2014), the old "one-size-fits-all" idea for DBMS does not apply to the complex Big Data environment. It is intrinsic that due to the size and characteristics of Big Data, different data types needs different DBMS designed and modified, meaning that there is no single database management structure which can fit all applications. Still, this combination of MongoDB and Hadoop provides powerful tools for organizations to apply to Big Data environments with all the advantages illustrated above.

8. References

Big Data: Examples and Guidelines for the Enterprise Decision Maker. (2016).

Florescu, D., Levy, A., & Mendelzon, A. (1998). Database techniques for the World-Wide Web. *ACM SIGMOD Record*, *27*(3), 59–74. doi: 10.1145/290593.290605

JIN, C.-Q., QIAN, W.-N., ZHOU, M.-Q., & ZHOU, A.-Y. (2014). Benchmarking Data Management Systems: from Traditional Database to Emergent Big Data. *CHINESE JOURNAL OF COMPUTERS*, *37*.

MongoDB Architecture Guide. (2015).