Data Paradigms Comparison

Client Server Technology

Pradeep Raja Mohan

Illinois Institute of Technology

Abstract

David J. Dewitt is a computer science professor at the University of Wisconsin -Madison. He is well known for his research on parallel databases, benchmarking, object oriented databases and XML databases. His research interests include database design, implementation, and evaluation. He is member of National academy of engineering and received ACM SIGNMOD award for his contribution to the database system field. He is recognized by ACM for his contributions to Gamma Parallel database system project. IEEE awarded him for fundamental contribution to the architecture, algorithms, and implementation of innovative database systems. He has published over more than 120 technical papers. One such of activity is a blog on database column which is about MapReduce. He states that it is a good idea to use MapReduce only for certain general purpose computations and compares this with the database community in this blog.

MapReduce and Its Problems

David J. Dewitt posits views on database technology and MapReduce. He explains about map reduce and reinforces the methods used in database technology.

MapReduce

MapReduce is a framework for writing applications which is used for processing enormous amount of data in parallel on large clusters of system. It reads the data set from the input file and applies the necessary filtering and transformations and outputs the set of records in the form of key, value. the split function splits the data set into independent buckets in a parallel manner. It partitions the record into M disjoint by applying functions (which is usually hash function) to the key of output record. The file is written to the disk when the bucket fills. The framework will sort the output of the maps and this will be the feed to the reduce task. The input and output of the job are stored in the same file system. The framework handles the scheduling of task, monitoring, and re-executes the failed tasks.

There are different nodes of a computer cluster in which multiple instances of map program is running on it. There will be N*M files, if there are N nodes in map and M files on disk. $F_{i,j}$, $1 \leq i \leq N$, $1 \leq j \leq M$.

The next phase is to reduce the M instances R_j, $1 \leq j \leq M$. The output with the same hash value will be consumed by the reduce even If the input has come from different map instance.

The below points support the database community and explains what MapReduce is lagging from.

- Schema is good. Schema is a skeletal structure in a database represents the logical view of the entire database. It shows the relationship amongst the data which is stored in an organized manner. To uncover the structure, it can be queried using SQL.
- Separation of schema from application
- High level language access

They provide a single command which will operate on the whole/multiple files at once. The tasks include.

- Select records from the file which will be retrieved only if the condition is true
- It gives the values and named fields from the records retrieved
- Derive additional values from the records
- Sort the records
- Present the output in desired title format and headings

MapReduce which uses low level language, a single statement tends to be simple. but to process the task more statements are needed which makes the low-level language difficult to learn and use. This is what the codasyl view says presenting an algorithm for data access. If the schema is not available in the application, then the structure must be uncovered and the source code for the application should be known. Maybe the dataset targeted by MapReduce has no schema but while extracting input from the key set the map function is relying on the existence of at least one data field in each input record.

MapReduce poor implementation

DBMS uses B-tree indexes to access the data. It stores all the values in order. Each leaf is the same distance from the root. The search will start from the root node it doesn't have to scan the whole table so by this way it speeds up the access. If we are looking for subset of records, then we can use the index search for optimizing.

In contrast, MapReduce has no indexes and therefore uses brute force for processing option. Brute force search enumerates all the possible candidates and check whether each one satisfies the problem statement.

MapReduce may provide parallel execution but this feature was included in the DBMS during 80's itself. It has multiple prototype built using gamma, bubba, grace.

Challenge with MapReduce is that skew issue is overlooked. This issue occurs when there is wide distribution of data with the same key.so this instances will take much longer time to run compared to others. Example, MapReduce programming model have large data processing which is parallelizable across large data sets. Map reduce program is the join operation of two tables. One problem which occurs in the join is the skewed values. If the values occur disproportionately then with these skewed values a single reducer process handling this value will be swamped with huge number of records.

Another hindrance with MapReduce is that since there are N*M files. When the reduces are running simultaneously it is possible that two or more instances of a reduce will try to read the input files from the same map node simultaneously. This will end up in reducing the disk transfer rate.so this the reason parallel system does not use push for the split files and use pull instead.

MapReduce is not novel

The author says that MapReduce is following the idea of partitioning larger data set into smaller portions which was already proposed earlier in the following

- Application of Hash to Data Base Machine and Its Architecture – in this the application of dynamic clustering feature of hash to relational database is discussed. The partitioning of relation using hash, load reductions in join and set operations is also discussed.
- Multiprocessor Hash-Based Join Algorithms – David J Dewitt has researched on hash algorithms to multiprocessor architecture. Implementation of join algorithms are also measured.

Gerber technique

He demonstrated the algorithm and performance evaluation of multidimensional clustering technique.

The author says that all techniques were earlier used in the database community and even Teradata has been selling DBMS techniques were more than 20 years. He claims that only the MapReduce function differentiates its software from SQL implementation

MapReduce missing feature

The following features are missing in the MapReduce which is present in the DBMS **Bulk Loader** - it loads the data in the desired format into the database. It is a program which forms the correct arguments and create database auxiliary files and invokes database utility, Indexing, Updates, Transactions, Integrity constraints, Referential Integrity, and Views

The author states that MapReduce provides only the narrow portion of technology which was used in the modern DBMS.

MapReduce is incompatible with the DBMS tools

The following tools are available in DBMS for performing end-end task and it is missing MapReduce. It will be difficult to perform these kinds of task in MapReduce unless these tools are SQL compatible or it has tools of its own kind.

- Report writers, Business intelligence tools Data mining tools, Replication tools and Database design tools

Mark C. Chu-Carroll view

Mark C. Chu-Carroll is google employee and he wrote his counter argument on the blog for which dewitt wrote.

MapReduce is a giant step backward - MapReduce is feasible for large dataset intensive applications. He states that MapReduce is not intended to replace the database but can be used for computations that are complex to run it takes only lesser amount of time to run on a single machine. He gives an example of Fractals which implemented MapReduce. In this example, there are millions of points to trace through. So, the points are divided into subsets and farm them out in map. The result was the map set of pairs. The points were and keys and number of crossings were values.

The result was transformed into the desired format with ease with the help of MapReduce. He states that this cannot be done in relational database. The input here is points rather than rows due to which it cannot be done.

- **Indexing.** If the data is tabular then indexing can be used, but it the task is not relational, that where computation comes in. MapReduce makes it easy to write the program that does the computation in reasonable amount of time.
- **It is novel.** the author states that MapReduce was never claimed to be novel. it actually said it was inspired from functional programming languages. MapReduce is leap for computation and not a replacement for DBMS.
- **Missing features in DBMS.** MapReduce is used for massive parallel/computing, meanwhile DBMS tools cannot be used for this purpose. If the requirement is a relational database problem then DBMS tools can be used, else there is no point in comparing.
- **Incompatible with tools.** It is the recursion of the statement which has already been stated above. The database oriented tools cannot work for MapReduce programming. They both ate meant for different purposes and not on the point to be compared.

My-View

Depending on the requirement analysis gathered the suitable software should be used. Importance of the usage of the software depends on the requirements. I will explain cerayin instances where the importance of the usage depends on the type of environment, input format and the desired output result.

Single node failure DBMS is not suited for this. If there are big queries or jobs running parallelly on DBMS , it will have to stop , stored and restarted upon node failure. But MapReduce can shift the jobs to other nodes if one fails. MapReduce can run of thousands of nodes.

Unstructured data DBMS suits for the writing queries on relational data, it struggles for the unstructured data or social media data. MapReduce can handle the complex data which is designed to overcome the difficulty faced by DBMS.

Complex query Complex queries are feasible in MapReduce, while the aggregation in SQL is not able to process multiple set data flows.

Querying It can be queried efficiently using DBMS than in MapReduce. The task can be performed repeatedly and effective for large datasets.

Structured data if the data is structured and iterative queries it is advantageous to use in DBMS. MapReduce is used for one time query while DBMS is used for repetitive queries

Cost It costs more to handle unstructured data when there is a node failure and if there are complex analytics. We can handle the structured data in DBMS with ease.

Conclusion

It solely depends on the organization's resources and requirements as stated above. In any case, the two ought to be essentially thought to be inconsistent with each other, as likely some type of combination will deliver comprehensive, all data management system.

Reference

David DeWitt on January 17, 2008 4:20 PM, retrieved from,

http://homes.cs.washington.edu/~billhowe/mapreduce_a_major_step_backwards.html

Mark C. Chu-Carroll on January 22, 2008, retrieved from,

http://scienceblogs.com/goodmath/2008/01/22/databases-are-hammers-mapreduc/