MapReduce: Simplified Data Processing on Large Clusters

Ryan Fredericks

5/8/15

Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters." (2004): n. pag. Print.

MapReduce: Simplified Data Processing on Large Clusters: Main Idea

- The main idea of MapReduce is to use the map functions to create an intermediate key/value pair and combines intermediate values that have the same keys.
- Uses text inputs, but can add new input types by creating a reader interface.
- It is very useful for large-scale indexing, as it can create complex data structures without the code being too difficult to understand and use.
- It is very scalable and can be used over parallel systems.
- Google has been using this method to enhance and improve their web search data.

MapReduce: Simplified Data Processing on Large Clusters: Implementation

- MapReduce is performed by splitting data into a group of M splits, which can be anywhere from 16 to 64 megabytes in size, and it runs in parallel throughout many machines.
- A master copy is created to assign work to worker copies when they are idle, such as map tasks and reduce tasks.
- Map workers map a key/value pair to a user-defined map function in order to get the desired output, and then buffered into memory.
- When necessary, the buffering will put data into reduce regions, where reduce workers will take that new data and group data based on the same key values.
- When sorted together, the reduce worker then passes every key/value combination for every unique intermediate key. When all data is passed, the key/values are appended to the final output file.

MapReduce: Simplified Data Processing on Large Clusters: Analysis

- The MapReduce function has a high usage in the normal workplace because, it is very efficient without being too complicated. A normal user without too much prior knowledge of this code can use it.
- It can be used in a parallel system without too many issues.
- Users can optimize different parts of the system by reading data from local disks.
- It is most efficient when used for large scale indexing.
- It can easily handle machine failures by creating checkpoints throughout the master data structures.
- Since there is only one master, there is a low chance of failure.

A Comparison of Approaches to Large-Scale Data Analysis: Main Idea

- This tries to compare and contrast large scale data analysis methods, talking about MapReduce and parallel DBMSs.
- The MapReduce section goes over the basic elements that I explained above.
- Parallel DBMSs use two aspects to have parallel computing:
  - Tables are partitioned over the nodes in a cluster .
  - System uses an optimizer that translates SQL commands into a query plan whose execution is divided amongst multiple nodes.

A Comparison of Approaches to Large-Scale Data Analysis: Implementation

- Parallel DBMSs use the relational paradigm of rows and columns in order to use the data.
- MapReduce does not need to do this because the data does not be adhere to any specific schema.
- DBMSs use B-tree indexes to have faster access to data.
- MapReduce, on the other hand, do not use any built in indexes during the process.
- Programmers can query the indexes to gain information on the indexes in the system.

A Comparison of Approaches to Large-Scale Data Analysis: Analysis

- These two systems have a number of distinct advantages over other different systems, including each other.
- DBMS uses data stored in columns and rows instead of in a free-form structure instead of
- It can create standard ways to create indexes in system between different programmers.
- DBMS states what the user wants, but does not create an algorithm on how to achieve receive the desired data from the system.
- Even though SQL can be difficult to use at first, it can be very useful to use.

Comparisons of Approaches to Large Scale Data Analysis vs MapReduce

- Part of the second paper dealt with MapReduce, so the same arguments are used in the comparison paper as the MapReduce paper.
- MapReduce and DBMSs would have the same amount of performance for the same amount of nodes, but MapReduce uses much more power in order to achieve the same results.
- Although MapReduce has an easier interface and is easier to code, it can cause more problems for the system in the long run.
- The pros of MapReduce is that since it creates so many redundant copies, it is significantly safer than DBMS, and there is a much smaller chance .
- Compression is more standard in size and method in MapReduce than DBMS
- DBMS parses at load time instead of pushing it off like MapReduce does.

One Size Fits All: An Idea Whose Time Has Come And Gone: Main Idea

- Data Warehouses use column stores, which are faster than row stores.
  - Rows stores are becoming redundant and useless.
- OLTP uses lightweight transactions and main memory for transactions.
- NoSQL uses key-value, record, and JSon stores and BigTable clones.
- Steaming markets can be used in conjunction with OLTP. This is easier than adding persistence to streaming engines.

One Size Fits All vs MapReduce

- MapReduce can be used in conjunction with column stores in order to be as efficient and safe as possible, as MapReduce does not need any specific structure.
- MapReduce helps with the idea of high availability brought up in the One Size Fits All since there are so many redundant copies in MapReduce.
- MapReduce can fit the mold of some systems since it focuses on high availability and takes up much more energy.
- Streaming market is increasing to the point that it can be more efficient than MapReduce.
  - Similarly to this, many systems of the systems discussed are more efficient energy-wise than MapReduce.