MapReduce: Simplified Data Processing on Large Clusters

Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters." . Google, Inc., n.d. Web. 7 Dec 2014. <http://static.googleusercontent.com/media/research.google.com/en/us/archive/mapreduce-osdi04.pdf>.

A Comparison of Approaches to Large-Scale Data Analysis

Abadi, Daniel, David DeWitt, Samuel Madden, and Erik Paulson. "A Comparison of Approaches to Large-Scale Data Analysis." . SIGMOD, n.d. Web. 7 Dec 2014. <http://database.cs.brown.edu/sigmod09/benchmarks-sigmod09.pdf>.

Cody Eichelberger

December 6th 2014

Map Reduce

- Map Reduce – essentially a framework to write programs for Hadoop clusters of workstations to simultaneously process and execute a program on a massive amount of data in parallel distributing tasks and workloads evenly across the cluster.
- In the first paper, Google offers a test of it's own C++ iteration of MR within its typical environment of a cluster of roughly 1,800 low-level workstations.
- Tasks are divided up, inputs analyzed, intermediary keys are created, and later counted and sorted according to the user-defined program.
- Aims to address Issues of how to parallel simultaneous actions, distribute and otherwise manage executions on very large data sets.
- Overall, it touts its ease of use for new programmers, easy conceptual translation of problems as Map Reduce computations, and high scalability

Map Reduce: How it works

- Suitability depends on the environment, map calls are distributed automatically across different machines. Inputs and results are partitioned but keyed in order to be compiled afterwards
- Splits input files into small pieces (16-64 MB); the program is started on cluster machines, map task reads contents and assigns and sorts key/value pairs
- These pairs are buffered and then written into local memory and then passed to the master workstation
- A reduce worker then reads the locations of the local pair lists and then goes to them and tallies the number of iterations of these keys and then passes each unique key and intermediate values to the user defined reduce function
- It is then compiled into a final file (one per reduce task)
- When a worker fails, it resets and reassigns the specified task for that machine to another idle, available machine
- Atomic storage of R outputs
- Establishes a schedule for backup executions of In progress tasks that are taking too long
- Comprehensive status information to monitor the executions
- Makes it possible to write a simple program and run it simultaneously on thousands of machines to greatly speed up computations which allows for even more rapid development and prototyping.
- Streamlining, reduces amount of necessary code

Analysis: Map Reduce

- When considering the Google report, I question the practice of using so many low-end workstations instead of relatively fewer, high powered machines.
- However, it is apparently easy to use, user friendly, hides the details, applicable to many different applications due to its very open structure and is easily scalable.
- OPEN SOURCED (At least the JAVA version) which allows for low up-front costs
- BUT, it is not easily transferable to other business applications. (Code must be rewritten and adjusted for each operation)
- Redundant executions allows the system to sort of self check and fix faults and failures which will prove to be a salient advantage of Map Reduce over traditional DBMSs
- Also, it seems unnecessary that it would need to scan the entire input before calculating

Map Reduce VS DBMS test paper

- The second paper presented a similar runtime environment on a much more scaled version of Google's test, although they second team asserts that the difference in running between 100 workstations and more is negligible
- Google's test was with many more nodes
- Despite parallel cluster computing seeming like a new concept, parallel SQL has existed for over 20 years
- When using MR, particular conditions must be known and adhered to by programmers or it risks being corrupted by bad data. Constraints in SQL DBMSs allow integrity to be independent of programmer and transferrable to many other applications via a schema.
- Filtration is done after statistics for all data is created with MR which seems costly and wasteful
- A DBMS doesn't save intermediary results and so it takes longer to restart the work when there is a node failure with a DBMS
- OVERALL, the common theme seemed to be that for short queries, Hadoop MR is limited by its start up times and costs as for nearly every test conducted, the majority of the time it took for MR to complete a task was dedicated to preliminary loading and readying
- Hadoop was quite easy to set up, DBMS was quite complicated to set up and configure.
- However, MR requires tuning occasionally to optimally suit individual tasks.
- Cold (MR) vs. warm (DBMS) start
- It is possible to write nearly any task as map reduce jobs or database queries
- Both parallel by partitioning datasets
- Relational tables appear connected over a cluster to the end user anyway
- DBMS requires data fit nicely into the row and column paradigm

Map Reduce – Pros/cons

PROS

- Provides a simple model that users use to express sophisticated distributed programs
- According to both papers, Map Reduce and Hadoop handle node failure very well and the disruption to execution progress is minimal when compared to the recovery of a DBMS to a node failure
- Hadoop and Map Reduce are open-source while traditional DBMSs are a substantial, costly investment
- Both offer runtime support to assist in debugging ,although java is one of the most common languages among programmers
- Map Reduce will take any type of data and is free from the usual row/column format
- This allows Map Reduce to more effectively handle analyze that is not conducive to the general row/column format such as social media data.

CONS

- Needs to read the entire input file before conducting any calculations; this causes MR to waste vast amounts of resources
- Not many extensions or integrated tools within MR while there exists a wealth of DBMS extensions such as data visualization, mining, intelligence, replication etc. However this may change as the user base of Hadoop and MR increases
- SQL is being commonly taught and is portable between different DBMSs with only minor changes to syntax or keywords
- Much more hardware involved with clusters
- DBMSs use a high level language so they can be applied to many different types of machines regardless of storage details etc. where as Map Reduce must be customized and tweaked fro each different environment
- MR programmers need to custom program support for complex data structures such as compound keys