Analysis and comparison of “The Google File System”, “A Comparison of Approaches to Large-Scale Data Analysis” , and the Stonebraker Talk.

Charles Schmitz

March 06, 2017

Main Ideas of The Google File System

- File storage must be streamlined for large files and their replacements; small files and rewrites should be supported, but need not be efficient.
- Architecture is done in “chunks”.
- Caching is necessary for master, but not client.
- Namespace directory unlike other file systems; many locks to ensure proper file management operations
- Garbage collection is done less rigidly, reduces strain
- Resilient to faults and has many tools to perform diagnostics

Implementation of The Google File System

- Since GFS and the applications are run by the same developer, they work together better
- Workloads on chunkservers depend on number of clients
- Appends are more efficient than overwrites
- Reads are *very* efficient

The Google File System : Analysis

- By allocating space for large files and appending to those large files inside the allocation, read/write efficiency is increased
- Less time spent managing files means more time handling them for the user’s needs
- Client gets only what it needs, when it needs it; reduces load
- Ensuring files are handled properly using lockouts helps with sharing
- Expecting hardware faults reduces data loss

Main ideas of "A Comparison of Approaches to Large-Scale Data Analysis (CALSDA)

- MapReduce and DBMS can serve many of the same tasks, but do serve very different purposes
- MapReduce handles everything in key/value pairs while DBMS use row formats
- DBMS are far more rigid in how they handle data (tables of rows)
- MR is more flexible, but requires the programmer to create the structure, indexes, and searches, whereas SQL has all of this built in for its standard.
- MR is more fault tolerant than SQL

Implementation, CALSDA

- Hadoop MR uses Java with non-default data sizes for block, heap, and node, runs best without extra features.
- Hadoop has a command line and file utility to load files
- DBMS-X performed reasonably using compression
- Uses LOAD SQL command that requires reorganization later
- Vertica runs well when only one query is sent at a time
- Vertica has a COPY command that reorganizes and sorts data according to the design specified
- Overall, each system had an individual fastest task, further pointing specializations

Analysis, CALSDA

- Each structure handles data differently
- Specific needs may cause one data structure to be more relevant than others
- Each structure responds at different speeds for different tasks
- DBMS and Vertica’s age have afforded them many developed advantages in speed over MapReduce

Comparison of “The Google File System” and CALSDA

- GFS runs very similarly to Hadoop in how it handles file access
- Also uses the same multi-node configuration as Hadoop
- Is supposedly faster than Hadoop
- GFS is maintained and run by its developers, not third parties.

Main ideas of the Stonebraker talk

- Attempts made to make RDBMS universally useful. This is not the case.
- Column store will be the new standard for certain markets; 2 orders of magnitude faster.
- NoSQL market is growing astoundingly fast
- Complex analytics will require faster means of generating information
- Streaming market comes down to programmer preference (Streaming/OLTP), nothing to do with row-store at all
- Graph analytics is easier in column store/array engine
- New tech will make new, faster implementations realistic
- Transition will be difficult, old implementations will attempt to adapt

Advantages/disadvantages: Google File system as compared to CALSDA and Stonebraker

- Similarity to MapReduce allows for more flexibility and less server load
- Requires attention to failed hardware; maintenance load may be high
- GFS expects to be good for storing/handling/updating files; given its focused purpose, it coincides with Stonebraker’s philosophy of specialized architecture.
- Its focused purpose reduces its ability to handle more broad tasks