The Google File System

Kimberly Tobias

Dec 8, 2014

GHEMAWAT, SANJAY, HOWARTHE GOOGLE FILE SYSTEM D GOBIOFF, AND SHUN-TAK LEUNG. "THE GOOGLE FILE SYSTEM." ACM SIGOPS OPERATING SYSTEMS REVIEW 37.5 (2003): 29. WEB.

MAIN IDEA

- The Google File System is a scalable distributed file system for large distributed data-intensive applications
  - Provides fault tolerance
  - Runs on inexpensive commodity hardware
  - Delivers high aggregate performance to a large number of clients
- Has been designed to meet the rapidly growing demands of Google's data processing needs
  - Goals
    - 1. Performance
    - 2. Scalability
    - 3. Reliability
    - 4. Availability
- In the design space, Google has explored radically different points

IMPLEMENTATION

- Google understands that their inexpensive machines will fail on a regular basis
  - Constant monitoring, error detections, fault tolerance, and automatic recovery are key to the system
- Files tend to be very large
  - All files are broken up into 64 MB chunks, each receiving their own id
- Optimized the system by appending files rather than rewrites
  - Overwriting files is rare and inefficient, this has resulted in adding data to the end of files
- Three components of its architecture
  - A Single Master: maintains operation log, watches the metadata
  - Multiple Chunkservers: store each 64 MB chunk
  - Multiple Clients: anyone/anything making a file request

MY ANALYSIS

- The Google File System is a well thought out, inexpensive design that is efficient and reliable
- I am very surprised that the components used are not some high end costly machine
  - Instead, there are three replicas produced from each chunk server, this provides a high level of redundancy
- Also I would have thought that having such large chunks would do more damage than good by wasting space.. BUT
  - Having a large chunk size reduces interaction with the master as reads and writes do not involve the master
  - Also, with more performance per chunk, the network overhead is reduced by consistent TCP connection
  - Last, it reduces the size of the metadata stored in the master

COMPARISONS

- The Large-Scale Data Analysis paper focuses primarily on the MapReduce paradigm and then comparing that to the parallel database management systems
- They both handle large scale data processing
- MapReduce is great because of its simplicity, having only two functions: map and reduce
  - Map: from an input file reads, transforms, then outputs a new key/value pair
  - Reduce: combines the records assigned and then writes the records to an output file
- Parallel DBMSs have all tables partitioned over the nodes in a cluster, and the system uses an optimizer that translates SQL into execution over multiple nodes

ADVANTAGES VS. DISADVANTAGES

- Advantages
  - Inexpensive machines make up the system
  - Large chunk sizes
  - Fabulous fault tolerance
  - High level of redundancy
- Disadvantages
  - Components fail on a normal basis
  - Not optimized for small scale data processing
  - Three replicas +/- can result in all replicas not being updated
  - Relies on appending rather than updating