Sector vs. Hadoop

A Brief Comparison Between the Two Systems

Background

- Sector is a relatively "new" system that is broadly comparable to Hadoop, and people want to know what are the differences.
- Is Sector simply another clone of GFS/MapReduce? No.
- These slides compare most recent versions of Sector and Hadoop as of Nov. 2010.
  - Both software are still under active development.
- If you find any inaccurate information, please contact Yunhong Gu [yunhong.gu # gmail]. We will try to keep these slides accurate and up to date.

Design Goals

Sector

- Three-layer functionality:
  - Distributed file system
  - Data collection, sharing, and distribution (over wide area networks)
  - Massive in-storage parallel data processing with simplified interface

Hadoop

- Two-layer functionality:
  - Distributed file system
  - Massive in-storage parallel data processing with simplified interface

History

Sector

- Started around 2005 - 2006, as a P2P file sharing and content distribution system for extreme large scientific datasets.
- Switched to centralized general purpose file system in 2007.
- Introduced in-storage parallel data processing in 2007.
- First "modern" version released in 2009.

Hadoop

- Started as a web crawler & indexer, Nutch, that adopted GFS/MapReduce between 2004 – 2006.
- Y! took over the project in 2006. Hadoop split from Nutch.
- First "modern" version released in 2008.

Architecture

Sector

- Master-slave system
- Masters store metadata, slaves store data
- Multiple active masters
- Clients perform IO directly with slaves

Hadoop

- Master-slave system
- Namenode stores metadata, datanodes store data
- Single namenode (single point of failure)
- Clients perform IO directly with datanodes

Distributed File System

Sector

- General purpose IO
- Optimized for large files
- File based (file not split by Sector but users have to take care of it)
- Use replication for fault tolerance

HDFS

- Write once read many (no random write)
- Optimized for large files
- Block based (64MB block as default)
- Use replication for fault tolerance

Replication

Sector

- System level default in configuration
- Per-file replica factor can be specified in a configuration file and can be changed at run-time
- Replicas are stored as far away as possible, but within a distance limit, configurable at per-file level
- File location can be limited to certain clusters only

HDFS

- System level default in configuration
- Per-file replica factor is supported during file creation
- For 3 replicas (default), 2 on the same rack, the 3rd on a different rack

Security

Sector

- A Sector security server is used to maintain user credentials and permission, server ACL, etc.
- Security server can be extended to connect to other sources, e.g., LDAP
- Optional file transfer encryption
- UDP-based hole punching firewall traversing for clients

Hadoop

- Still in active development, new features in 2010
- Kerberos/token based security framework to authenticate users
- No file transfer encryption

Wide Area Data Access

Sector

- Sector ensures high performance data transfer with UDT, a high speed data transfer protocol
- As Sector pushes replicas as far away from each other as possible, a remote Sector client may find a nearby replica
- Thus, Sector can be used as content distribution network for very large datasets

HDFS

- HDFS has no special consideration for wide area access. Its performance for remote access would be close to a stock FTP server.
- Its security mechanism may also be a problem for remote data access.

In-Storage Data Processing

Sphere

- Apply arbitrary user-defined functions (UDFs) on data segments
- UDFs can be Map, Reduce, or others
- Support native MapReduce as well

Hadoop MapReduce

- Support the MapReduce framework

Sphere UDF vs. Hadoop MapReduce

Sphere UDF

- Parsing: permanent record offset index if necessary
- Data segments (records, blocks, files, and directories) are processed by UDFs
- Transparent load balancing and fault tolerance
- Sphere is about 2 – 4x faster in various benchmarks

Hadoop MapReduce

- Parsing: run-time data parsing with default or user-defined parser
- Data records are processed by Map and Reduce operations
- Transparent load balancing and fault tolerance

Why Sphere is Faster than Hadoop?

- C++ vs. Java
- Different internal data flows
- Sphere UDF model is more flexible than MapReduce
  - UDF dissembles MapReduce and gives developers more control to the process
- Sphere has better input data locality
  - Better performance for applications that process files and group of files as minimum input unit
- Sphere has better output data locality
  - Output location can be used to optimize iterative and combinative processing, such as join
- Different implementations and optimizations
- UDT vs. TCP (significant in wide area systems)

Compatibility with Existing Systems

Sector/Sphere

- Sector files can be accessed from outside if necessary
- Sphere can simply apply an existing application executable on multiple files or even directories in parallel, if the executable accepts a file or a directory as input

Hadoop

- Data in HDFS can only be accessed via HDFS interfaces
- In Hadoop, executables that process files may also be wrapped in Map or Reduce operations, but will cause extra data movement if file size is greater than block size
- Hadoop cannot process multiple files within one operation.

Conclusions

- Sector is a unique system that integrates distributed file system, content sharing network, and parallel data processing framework.
- Hadoop is mainly focused on large data processing within a single data center.
- They overlap on the parallel data processing support.

Our Recommendations

- Consider using Sector if:
  - You need a scalable, fault-tolerant, general purpose file system
  - You have data across multiple data centers
  - You have users who upload and download data from remote locations
  - You are a C++ programmer
  - You have many legacy applications and you don't want to re-write them to suit a new platform
  - You value the fact that Sphere is about 2 – 4 times faster than Hadoop

Our Recommendations (cont.)

- Consider using Hadoop if:
  - You are a Java programmer
  - It is important for you to have data semantic support such as HBase or Hive
  - You can benefit from reusing existing packages from the larger Hadoop community