Hadoop Distributed File System (HDFS)

Overview

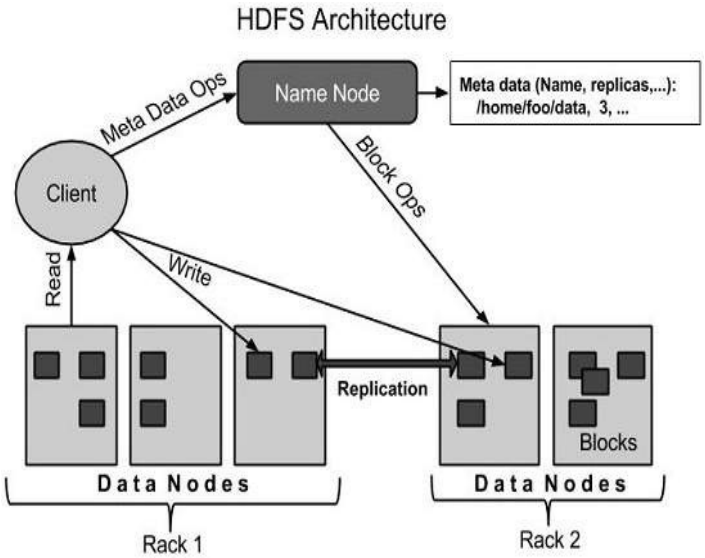
- Provides inexpensive and reliable storage for massive amounts of data.
- Designed for:
  - Big files (100 MB to several TBs file sizes).
  - Write once, read many times (Appending only).
  - Running on commodity hardware.
- Hierarchical UNIX style file systems.
  - UNIX style file ownership and permissions.

HDFS main design principles

- I/O pattern:
  - Append only → reduce synchronization.
- Data distribution:
  - File is split in big chunks (64 MB).
    → Reduce metadata size.
    → Reduce network communication.
- Data replication:
  - Each chunk is usually replicated in 3 different nodes.
- Fault tolerance:
  - Data node: re-replication.
  - Name node:
    - Secondary name node.
    - Standby, active name node.

HDFS Architecture

- Master/slave architecture.
- HDFS master: Name node.
  - Manage namespace and metadata.
  - Monitor data nodes.
- HDFS slaves: Data nodes.
  - Handle read/write the actual data (chunks).
  - Chunks are local files in the local file systems.



Functions of a Name Node

- Manage File System Namespace.
  - Maps a file name to a set of blocks.
  - Maps a block to the Data nodes where it resides.

- Cluster Configuration Management.
- Replication Engine for Blocks.

Name Node metadata

- Metadata in memory.
    - The entire metadata is in main memory.
    - No demand paging of metadata.
- Types of metadata:
    - List of files.
    - List of Blocks for each file.
    - List of Data nodes for each block.
    - File attributes, e.g., creation time, replication factor.
- A Transaction Log:
    - Records file creations, file deletions, etc.

Data Node

- A Block Server:
    - Stores data in the local file system (e.g., ext3).
    - Stores metadata of a block (e.g., CRC).
    - Serves data and metadata to Clients.
- Block Report:
    - Periodically sends a report of all existing blocks to the Name node.
- Facilitates Pipelining of Data:
    - Forwards data to other specified Data node.
- Heartbeat:
    - Data nodes send heartbeat to the Name node.
        - Once every 3 seconds.
    - Name node uses heartbeats to detect Data nodes failure.

Data Replication

- Chunk replacement.
    - Current Strategy.
        - One replica on local node.
        - Second replica on remote rack.
        - Third replica on the same remote rack.
        - Additional replicas are randomly placed.
    - Clients read from nearest replicas.
- Name node detects Data node failures:
    - Chooses new Data nodes for new replicas.
    - Balances disk usage.
    - Balances communication traffic to Data nodes.

Data Rebalance

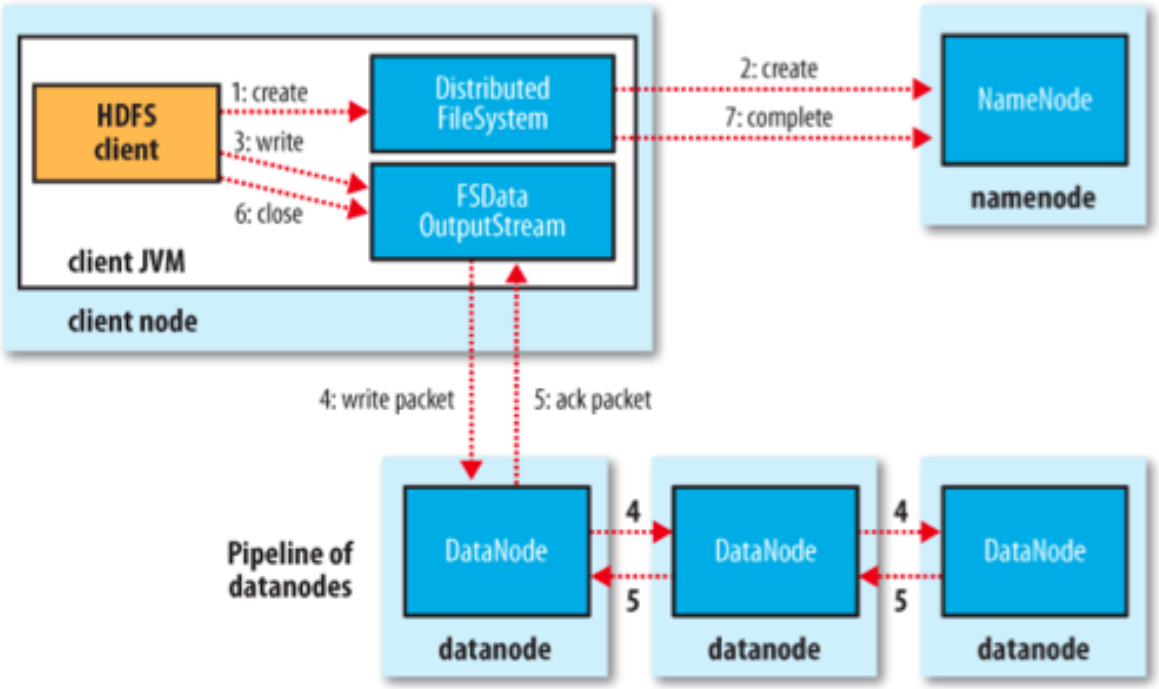
- Goal: % disk full on Data nodes should be similar.
    - Usually run when new Data nodes are added.
    - Cluster is online when Rebalancer is active.
    - Rebalancer is throttled to avoid network congestion.
    - Command line tool.

Data Correctness

- Use Checksums to validate data.
    - CRC32.
- File Creation:
    - Client computes checksum per 512 bytes.
    - Data node stores the checksum.
- File Access:
    - Client retrieves the data and checksum from Data node.
    - If validation fails, Client tries other replicas.

Data pipelining

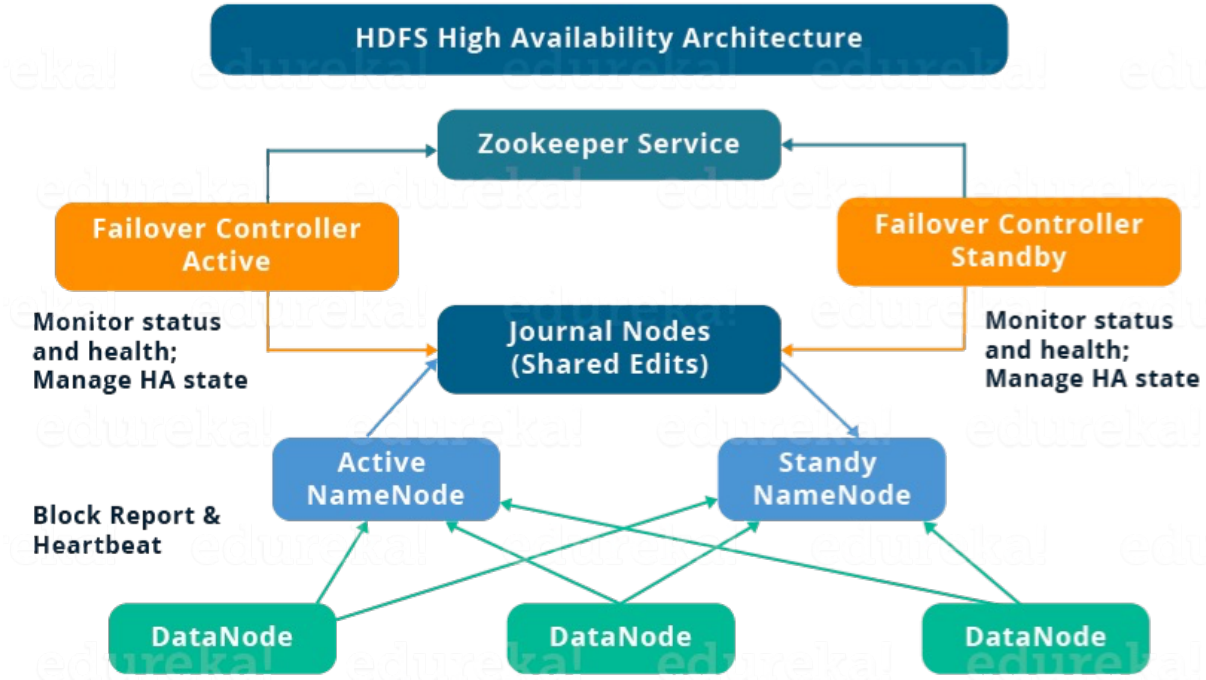
- Client retrieves a list of Data nodes on which to replace replicas of a block.
- Client writes block to the first Data node.
- The first Data node forwards the data to the next node in the Pipeline.
- When all replicas are written, the Client moves on to write the next block in file.

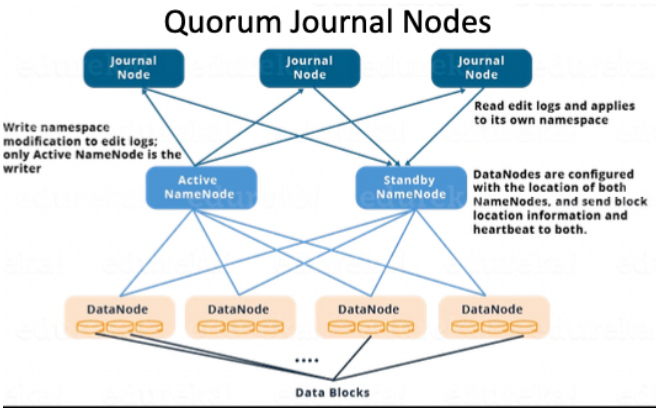


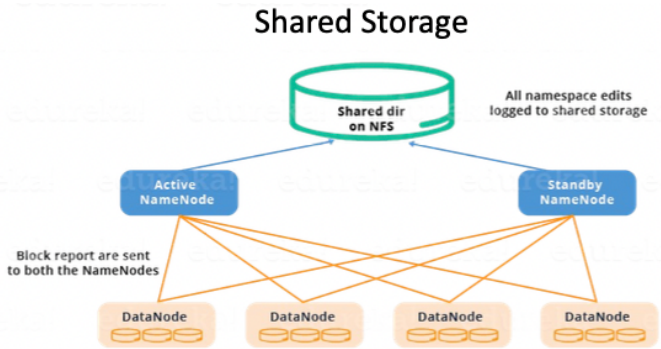
Secondary Name node

- Name node is a single point of failure.
- Secondary Name node:
    - Checkpointing latest copy of the Fs Image and the Transaction Log Files.
        - Copies Fs Image and Transaction Log from Name node to a temporary directory.
- When Name node restarted:
    - Merges Fs Image and Transaction Log into a new Fs Image in temporary directory.
    - Uploads new Fs Image to the Name node.
        - Transaction Log on Name node is purged.

Name node high availability (HA)







HDFS command-line interface

List Files	
hdfs dfs -ls /	List all the files/directories for the given hdfs destination path.
hdfs dfs -ls -d /hadoop	Directories are listed as plain files. In this case, this command will list the details of hadoop folder.
hdfs dfs -ls -h /data	Format file sizes in a human-readable fashion (eg 64.0m instead of 67108864).
hdfs dfs -ls -R /hadoop	Recursively list all files in hadoop directory and all subdirectories in hadoop directory.
hdfs dfs -ls /hadoop/dat*	List all the files matching the pattern. In this case, it will list all the files inside hadoop directory which starts with 'dat'.

Read/Write Files	
hdfs dfs -text /hadoop/derby.log	HDFS Command that takes a source file and outputs the file in text format on the terminal. The allowed formats are zip and TextRecordInputStream.
hdfs dfs -cat /hadoop/test	This command will display the content of the HDFS file test on your stdout .
hdfs dfs -appendToFile /home/ubuntu/test1 /hadoop/text2	Appends the content of a local file test1 to a hdfs file test2.

Upload, download files

Upload/Download Files	
hdfs dfs -put /home/ubuntu/sample /hadoop	Copies the file from local file system to HDFS.
hdfs dfs -put -f /home/ubuntu/sample /hadoop	Copies the file from local file system to HDFS, and in case the local already exits in the given destination path, using -f option with put command will overwrite it.
hdfs dfs -put -l /home/ubuntu/sample /hadoop	Copies the file from local file system to HDFS. Allow DataNode to lazily persist the file to disk. Forces replication factor of 1.
hdfs dfs -put -p /home/ubuntu/sample /hadoop	Copies the file from local file system to HDFS. Passing -p preserves access and modification times, ownership and the mode.
hdfs dfs -get /newfile /home/ubuntu/	Copies the file from HDFS to local file system.
hdfs dfs -get -p /newfile /home/ubuntu/	Copies the file from HDFS to local file system. Passing -p preserves access and modification times, ownership and the mode.
hdfs dfs -get /hadoop/*.txt /home/ubuntu/	Copies all the files matching the pattern from local file system to HDFS.
hdfs dfs -copyFromLocal /home/ubuntu/sample /hadoop	Works similarly to the put command, except that the source is restricted to a local file reference.
hdfs dfs -copyToLocal /newfile /home/ubuntu/	Works similarly to the put command, except that the destination is restricted to a local file reference.
hdfs dfs -moveFromLocal /home/ubuntu/sample /hadoop	Works similarly to the put command, except that the source is deleted after it's copied.

File management

File Management	
hdfs dfs -cp /hadoop/file1 /hadoop1	Copies file from source to destination on HDFS. In this case, copying file1 from hadoop directory to hadoop1 directory.
hdfs dfs -cp -p /hadoop/file1 /hadoop1	Copies file from source to destination on HDFS. Passing -p preserves access and modification times, ownership and the mode.
hdfs dfs -cp -f /hadoop/file1 /hadoop1	Copies file from source to destination on HDFS. Passing -f overwrites the destination if it already exists.
hdfs dfs -mv /hadoop/file1 /hadoop1	Move files that match the specified file pattern <src> to a destination <dst>. When moving multiple files, the destination must be a directory.
hdfs dfs -rm /hadoop/file1	Deletes the file (sends it to the trash).
hdfs dfs -rm -r /hadoop hdfs dfs -rm -R /hadoop hdfs dfs -rmr /hadoop	Deletes the directory and any content under it recursively.
hdfs dfs -rm -skipTrash /hadoop	The -skipTrash option will bypass trash, if enabled, and delete the specified file(s) immediately.
hdfs dfs -rm -f /hadoop	If the file does not exist, do not display a diagnostic message or modify the exit status to reflect an error.
hdfs dfs -rmdir /hadoop1	Delete a directory.
hdfs dfs -mkdir /hadoop2	Create a directory in specified HDFS location.
hdfs dfs -mkdir -f /hadoop2	Create a directory in specified HDFS location. This command does not fail even if the directory already exists.
hdfs dfs -touchz /hadoop3	Creates a file of zero length at <path> with current time as the timestamp of that <path>.

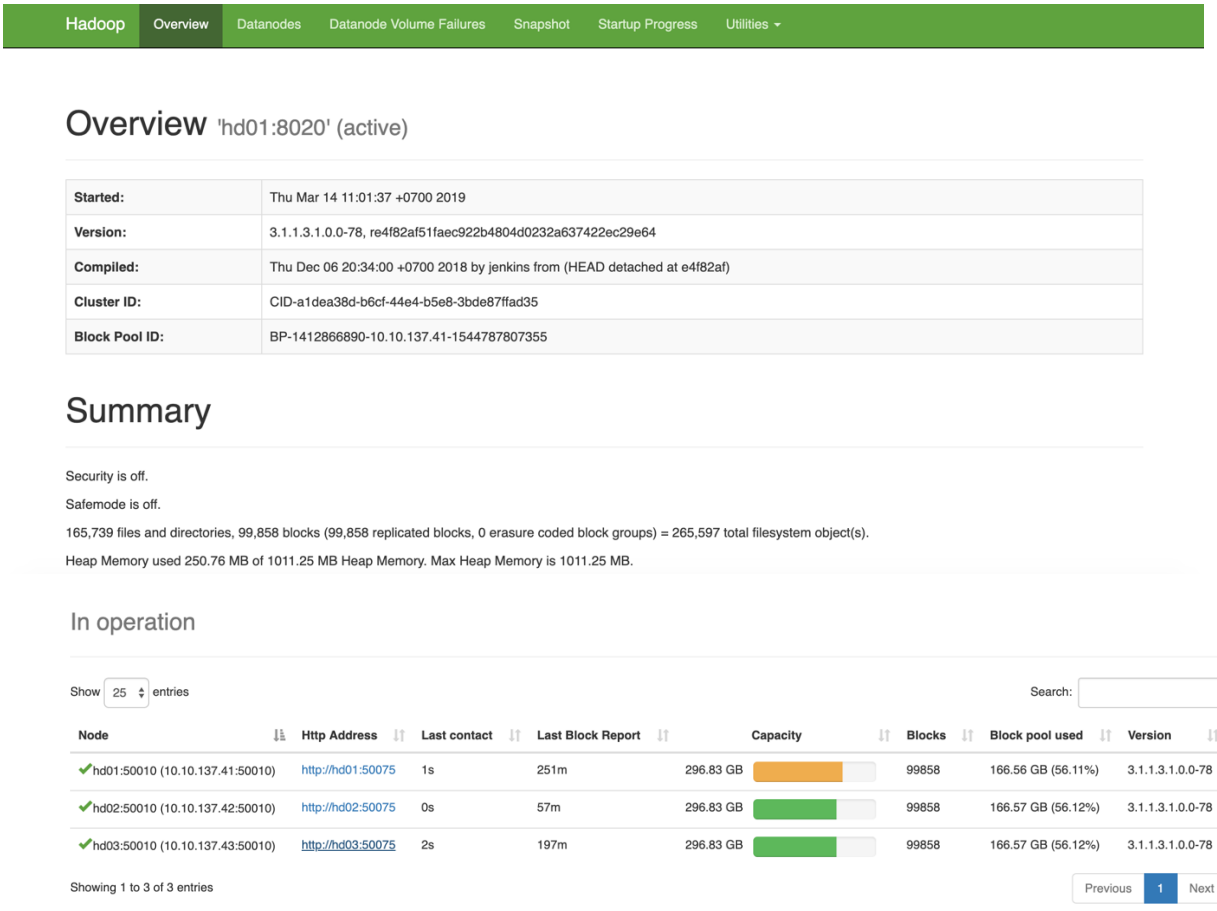
Ownership and validation

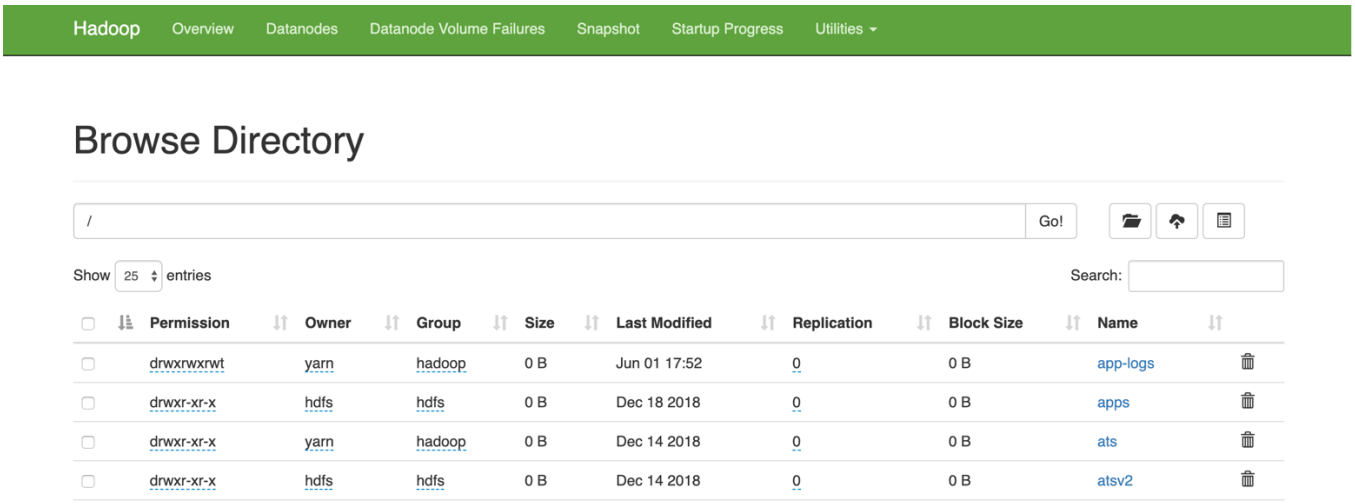
Ownership and Validation	
hdfs dfs -checksum /hadoop/file1	Dump checksum information for files that match the file pattern <src> to stdout.
hdfs dfs -chmod 755 /hadoop/file1	Changes permissions of the file.
hdfs dfs -chmod -R 755 /hadoop	Changes permissions of the files recursively.
hdfs dfs -chown ubuntu:ubuntu /hadoop	Changes owner of the file. 1st ubuntu in the command is owner and 2nd one is group.
hdfs dfs -chown -R ubuntu:ubuntu /hadoop	Changes owner of the files recursively.
hdfs dfs -chgrp ubuntu /hadoop	Changes group association of the file.
hdfs dfs -chgrp -R ubuntu /hadoop	Changes group association of the files recursively.
Filesystem	
hdfs dfs -df /hadoop	Shows the capacity, free and used space of the filesystem.
hdfs dfs -df -h /hadoop	Shows the capacity, free and used space of the filesystem. -h parameter Formats the sizes of files in a human-readable fashion.
hdfs dfs -du /hadoop/file	Show the amount of space, in bytes, used by the files that match the specified file pattern.
hdfs dfs -du -s /hadoop/file	Rather than showing the size of each individual file that matches the pattern, shows the total (summary) size.
hdfs dfs -du -h /hadoop/file	Show the amount of space, in bytes, used by the files that match the specified file pattern. Formats the sizes of files in a human-readable fashion.

Adminstration

Administration	
hdfs balancer -threshold 30	Runs a cluster balancing utility. Percentage of disk capacity. This overwrites the default threshold.
hadoop version	To check the vesrion of Hadoop.
hdfs fsck /	It checks the health of the Hadoop file system.
hdfs dfsadmin -safemode leave	The command to turn off the safemode of NameNode.
hdfs dfsadmin -refreshNodes	Re-read the hosts and exclude files to update the set of Datanodes that are allowed to connect to the Namenode and those that should be decommissioned or recommissioned.
hdfs namenode -format	Formats the NameNode.

HDFS Name Node UI





Other HDFS interfaces

- Java API.
- Thrift API.
- Fuse.
- WebDAV.

HDFS data format

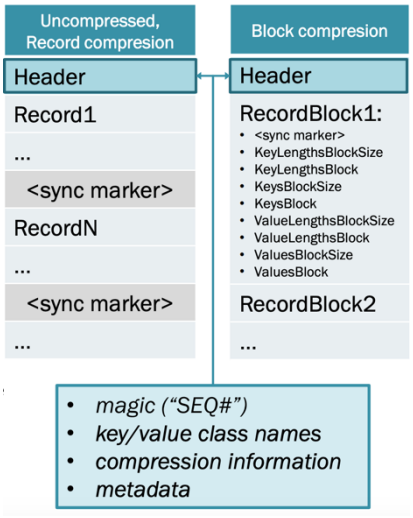
- Text.
- Sequence file.
- Avro.
- Parquet.
- Optimized Row Columnar (ORC).

Text file

- CSV, TSV, JSON records.
- Convenient format to use to Exchange between applications or scripts.
- Human readable and parse able.
- Do not support block compression.
- Not as efficient to query.
- Good for the beginning, but not good enough for real life.

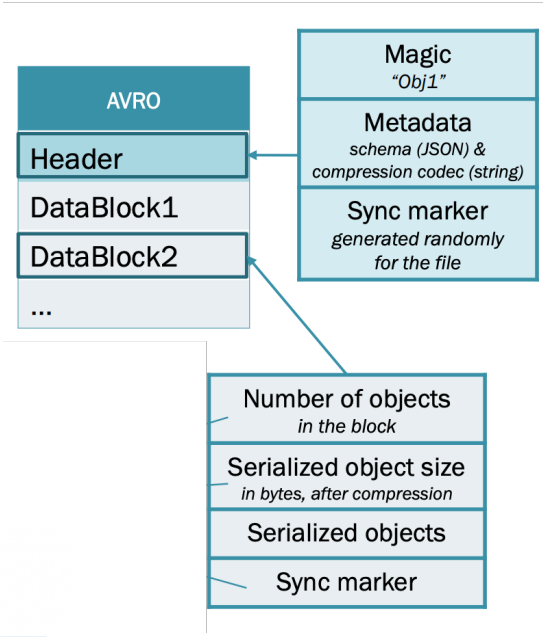
Sequence file

- Provides a persistent data structure for binary key-value pairs.
- Commonly used to transfer data between Map Reduce jobs.
- Can be used as an archive to pack small files in Hadoop.
- Row-based.
- Compression.
- Splitable.
    - Support splitting even when the data is compressed.



Avro

- Row-based.
- Supports (object) compression and splitting.
- Flexible data scheme.
  - Scheme (JSON) included to the file.
- Data types:
  - Primitive: null, Boolean, int, long, etc.
  - Complex: records, arrays, maps, etc.
- Binary and JSON data serialization.
- Data corruption detection.

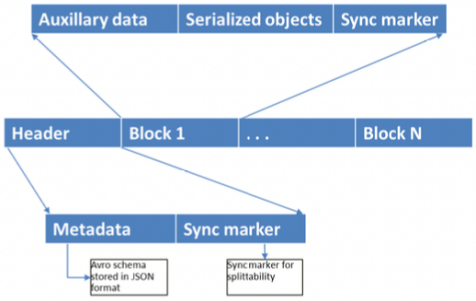


Arvo – File structure and example

Sample AVRO schema in JSON format

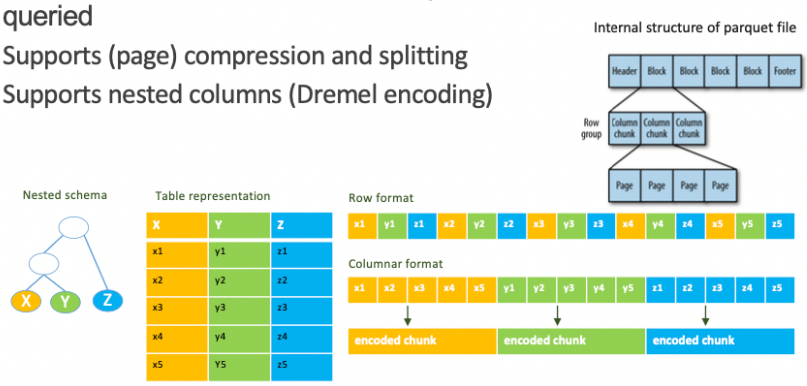
```
{
 "type" : "record",
 "name" : "tweets",
 "fields" : [ {
  "name" : "username",
  "type" : "string",
}, {
  "name" : "tweet",
  "type" : "string",
}, {
  "name" : "timestamp",
  "type" : "long",
} ],
 "doc:" : "schema for storing tweets"
}
```

Avro file structure



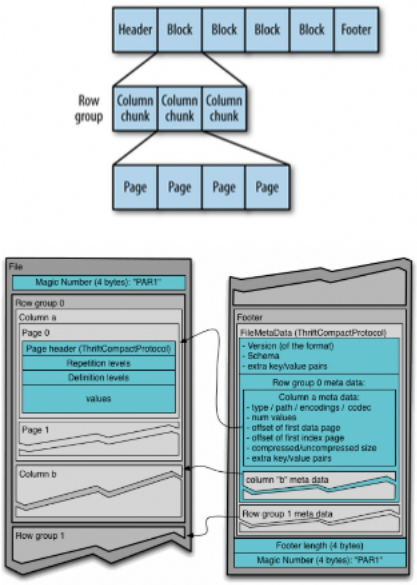
Parquet

- Column-oriented binary file format
- Efficient in terms of disk I/O when specific columns need to be queried
- Supports (page) compression and splitting
- Supports nested columns (Dremel encoding)



Parquet file structure and configuration

Internal structure of parquet file



Configurable parquet parameters

Property name	Default value	Description
parquet.block.size	128 MB	The size in bytes of a block (row group).
parquet.page.size	1MB	The size in bytes of a page.
parquet.dictionary.page.size	1MB	The maximum allowed size in bytes of a dictionary before falling back to plain encoding for a page.
parquet.enable.dictionary	true	Whether to use dictionary encoding.
parquet.compression	UNCOMPRESSED	The type of compression: UNCOMPRESSED, SNAPPY, GZIP & LZO

Parquet is optimized for high compression and high scan efficiency

Optimized row columnar (ORC)

- RCFile
  - Every column is compressed individually within the row group
- ORC File
  - Block-mode compression
  - Data type support
  - Ordered data store (within one stripe)
- Stores collections of rows and within the collection the row data is stored in columnar format
- Introduces a lightweight indexing that enables skipping of irrelevant blocks of rows
- Splittable: allows parallel processing of row collections
- Indices with column-level aggregated values (min, max, sum and count)

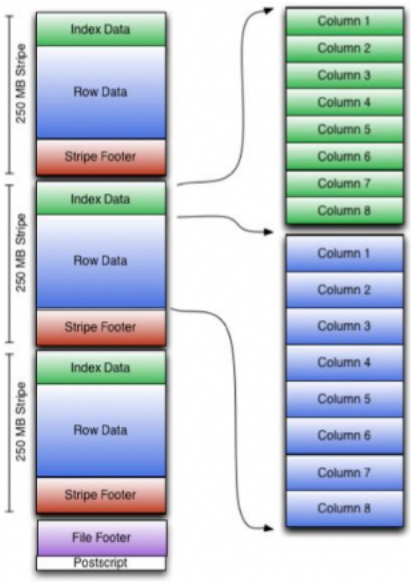


Table of Contents