

HDFS Architecture

Gregory Kesden, CSE-291 (Cloud Computing) Fall 2016

Based Upon: <http://hadoop.apache.org/docs/r3.0.0-alpha1/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>

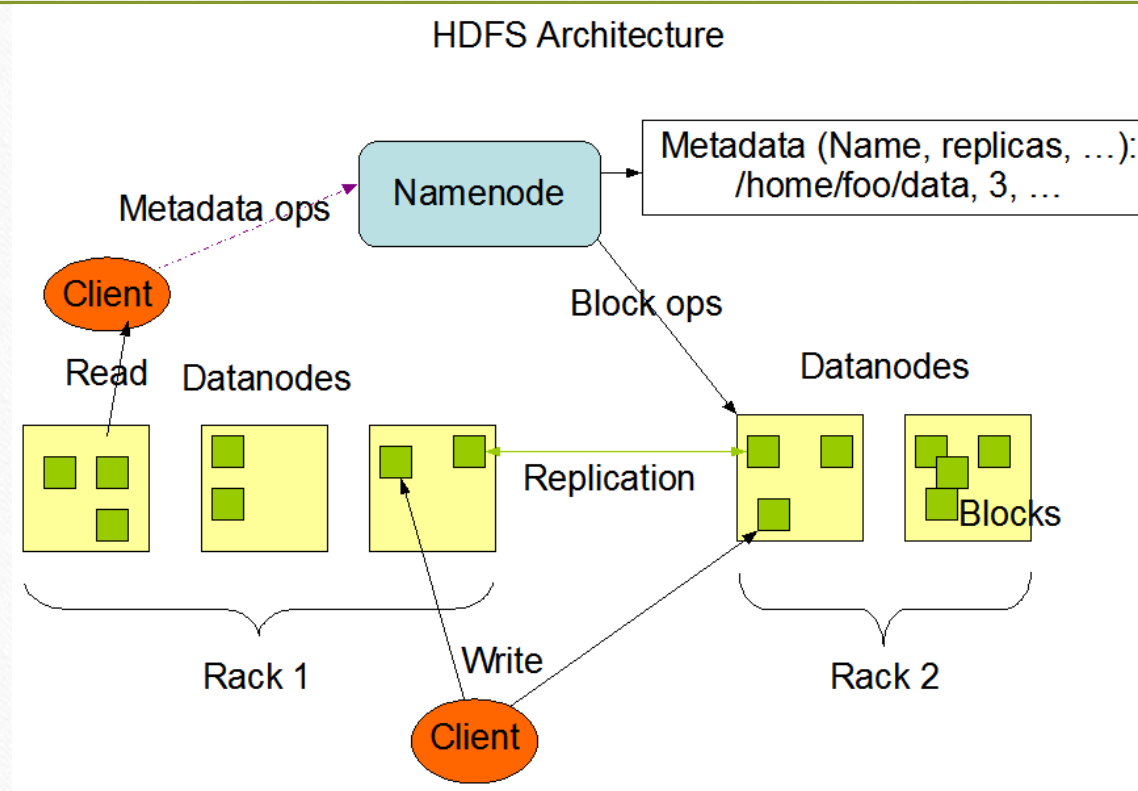
Assumptions

- At scale, hardware failure is the norm, not the exception
 - Continued availability via quick detection and work-around, and eventual automatic full recovery is key
- Applications stream data for batch processing
 - Not designed for random access, editing, interactive use, etc
 - Emphasis is on throughput, not latency
- Large data sets
 - Tens of millions of files many terabytes per instance

Assumptions, *continued*

- Simple Coherency Model = Lower overhead, higher throughput
 - Write Once, Read Many (WORM)
 - Gets rid of most concurrency control and resulting need for slow, blocking coordination
- “Moving computation is cheaper than moving data”
 - The data is huge, the network is relatively slow, and the computation per unit of data is small.
 - Moving (Migration) may not be necessary – mostly just placement of computation
- Portability, even across heterogeneous infrastructure
 - At scale, things can be different, fundamentally, or as updates roll-out

Overall Architecture



NameNode

- Master-slave architecture
- 1x NameNode (coordinator)
 - Manages name space, coordinates for clients
 - Directory lookups and changes
 - Block to DataNode mappings
- Files are composed of blocks
 - Blocks are stored by DataNodes
- Note: User data never comes to or from a NameNode.
 - The NameNode just coordinates

DataNode

- Many DataNodes (participants)
 - One per node in the cluster. Represent the node to the NameNode
 - Manage storage attached to node
 - Handles read(), write() requests, etc for clients
 - Store blocks as per NameNode
 - Create and Delete blocks, Replicate Blocks

Namespace

- Hierarchical name space
 - Directories, subdirectories, and files
- Managed by NameNode
- Maybe not needed, but low overhead
 - Files are huge and processed in entirety
 - Name to block lookups are rare
 - Remember, model is streaming of large files for processing
 - Throughput, not latency, is optimized

Access Model

- (Just to be really clear)
- Read anywhere
 - Streaming is in parallel across blocks across DataNodes
- Write only at end (append)
- Delete whole file (rare)
- No edit/random write, etc

Replication

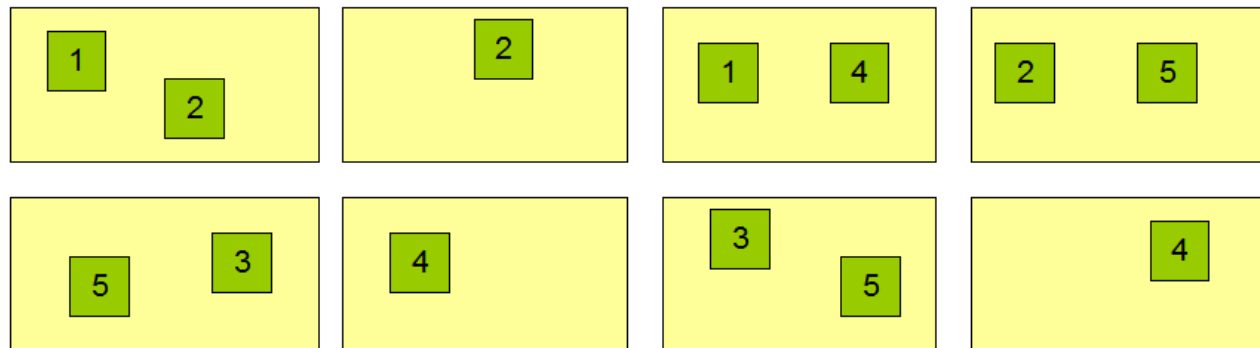
- Blocks are replicated by default
 - Blocks are all same size (except tail)
 - Fault tolerance
 - Opportunities for parallelism
- NameNode managed replication
 - Based upon heartbeats, block reports (per dataNode report of available blocks), and replication factor for file (per file metadata)

Replication

Block Replication

Namenode (Filename, numReplicas, block-ids, ...)
/users/sameerp/data/part-0, r:2, {1,3}, ...
/users/sameerp/data/part-1, r:3, {2,4,5}, ...

Datanodes



Location Awareness

- Site + 3-Tier Model is default

Replica Placement and Selection

- Assume bandwidth within rack greater than outside of rack
- Default placement
 - 2 nodes on same rack, one different rack (Beyond 3? Random, below replicas/rack limit)
 - Fault tolerance, parallelism, lower network overhead than spreading farther
- Read from closest replica (rack, site, global)

Filesystem Metadata Persistence

- EditLog keeps all metadata changes.
 - Stored in local host FS
- FSImage keeps all FS metadata
 - Also stored in local host FS
- FSImage kept in memory for use
 - Periodically (time interval, operation count), merges in changes and checkpoints
 - Can truncate EditLog via checkpoint
- Multiple copies of files can be kept for robustness
 - Kept in sync
 - Slows down, but okay given infrequency of metadata changes.

Failure of DataNodes

- Disk Failure, Node Failure, Partitioning
 - Detect via heartbeats (long delay, by default), blockmaps, etc
 - Re-Replicate
- Corruption
 - Detectable by client via checksums
 - Client can determine what to do (nothing is an option)
- Metadata

Datablocks, Staging

- Data blocks are large to minimize overhead for large files
- Staging
 - Initial creation and writes are cached locally and delayed, request goes to NameNode when 1st chunk is full.
 - Local caching is intended to support use of memory hierarchy and throughput needed for streaming. Don't want to block for remote end.
 - Replication is from replica to replica, "Replication pipeline"
 - Maximizes client's ability to stream