HDFS- Deep Dive

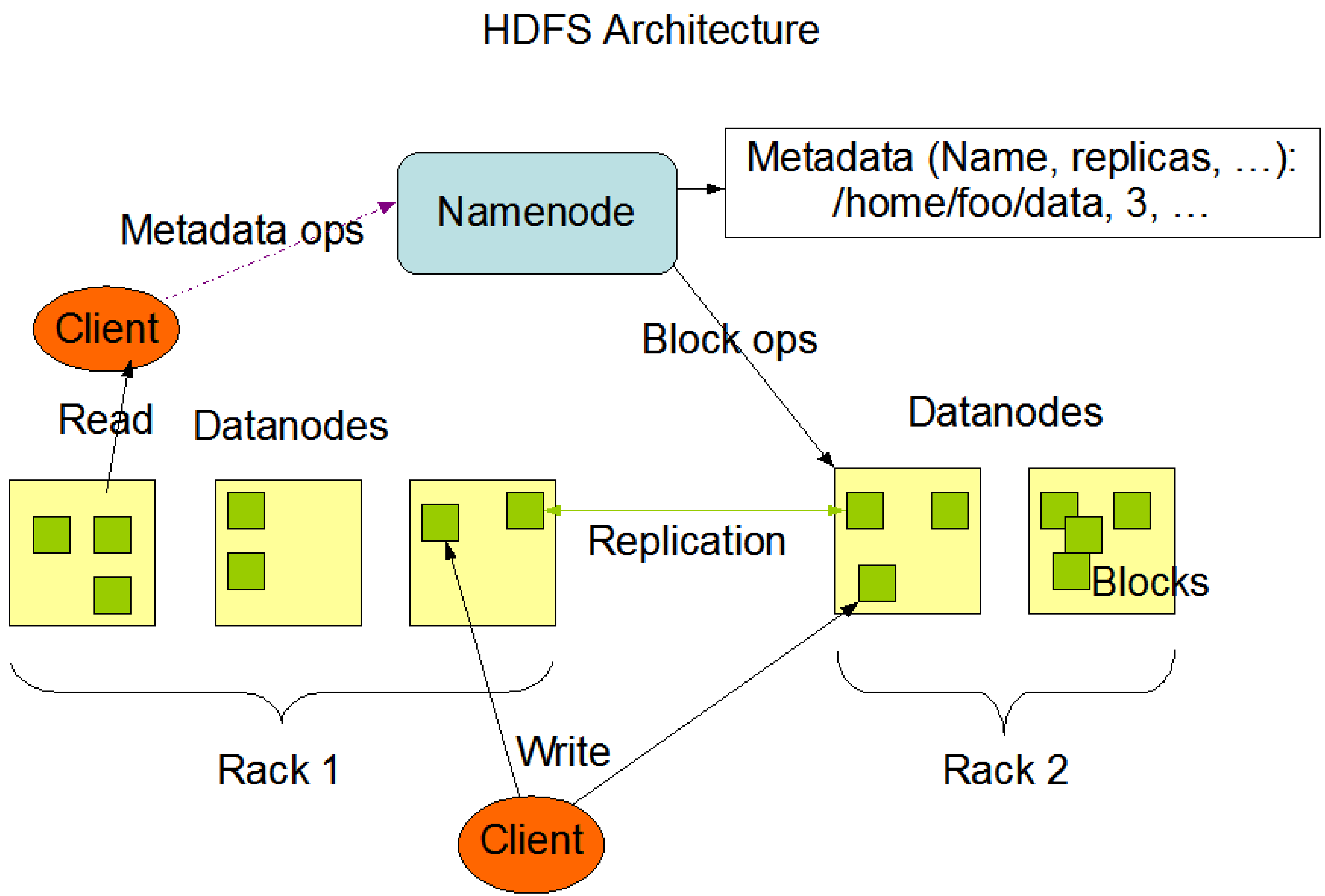
Agenda

- **HDFS**
- **Master/Slave architecture in HDFS**
- **The File System Namespace**
- **Data Replication**
- **Replica Selection**
- **Blockreport**
- **The Persistence of File System Metadata- EditLogs and FSImage**
- **Checkpoint**
- **Replication Pipelining**
- **Staging**
- **Space Reclamation**

HDFS

- **HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware.**
- **HDFS provides high throughput access to application data and is suitable for applications that have large data sets.**

Master/Slave architecture in HDFS



NameNode

- A master server that manages the file system namespace and regulates access to files by clients.
- The NameNode executes file system namespace operations like opening, closing, and renaming files and directories.
- It also determines the mapping of blocks to DataNodes.

DataNodes

- Manage storage attached to the nodes that they run on.
- Responsible for serving read and write requests from the file system's clients.
- Perform block creation, deletion, and replication upon instruction from the NameNode.

The File System Namespace

- **The NameNode maintains the file system namespace. Any change to the file system namespace or its properties is recorded by the NameNode.**
- **Traditional hierarchical file organization.**
- **Exa:- "/user/mydir/demo.csv"**
- **A user or an application can create and remove files, move a file from one directory to another, or rename a file.**
- **HDFS does not yet implement user quotas.**
- **HDFS does not support hard links or soft links.**
- **Stores Replication Factor Information(number of replicas of a file that should be maintained by HDFS).**

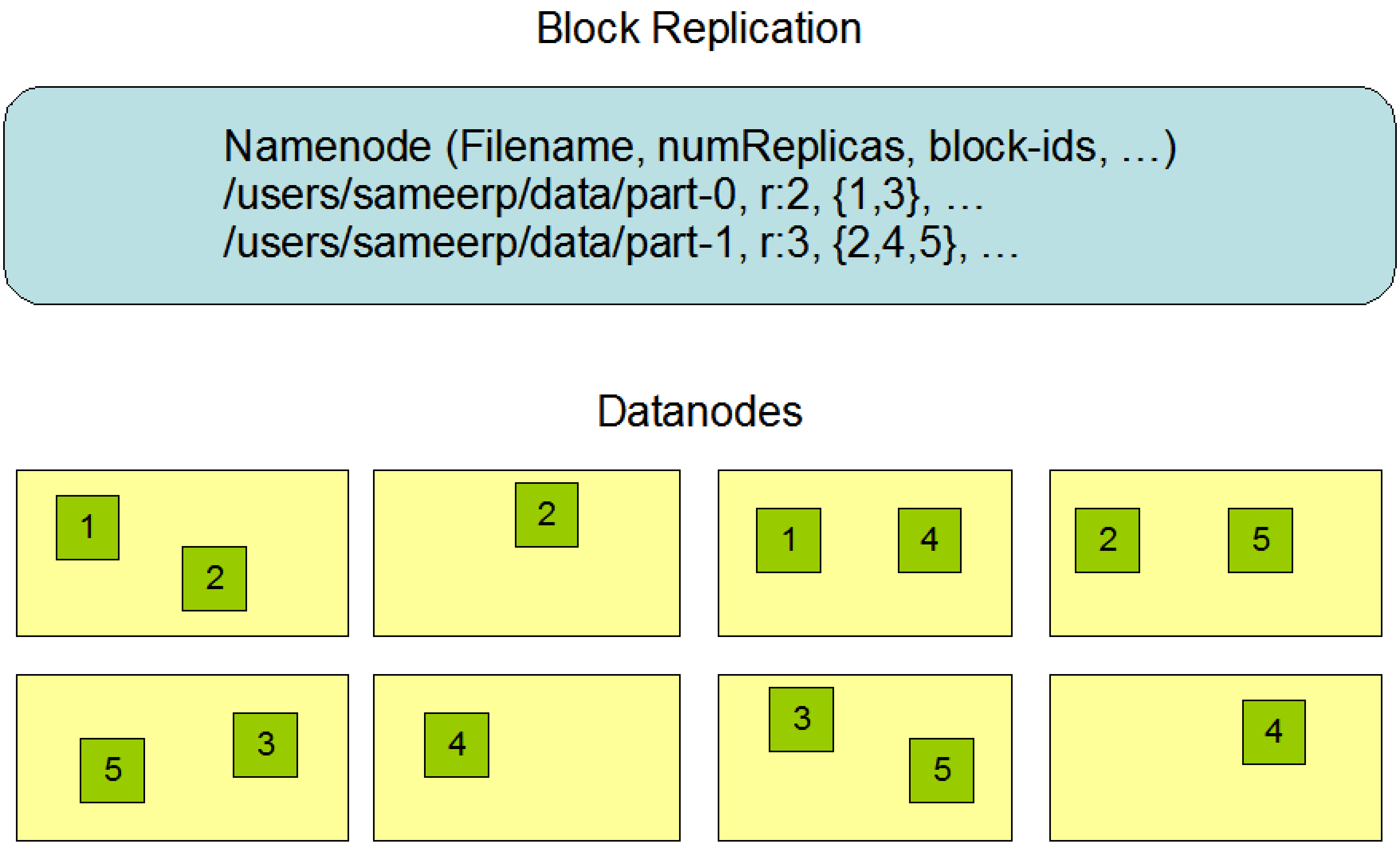
Data Replication

- **HDFS stores each file as a sequence of blocks; all blocks in a file except the last block are the same size.**
- **The blocks of a file are replicated for fault tolerance.**
- **The block size and replication factor are configurable per file. An application can specify the number of replicas of a file.**
- **The replication factor can be specified at file creation time and can be changed later.**

Data Replication

- **Files in HDFS are write-once and have strictly one writer at any time.**
- **The NameNode makes all decisions regarding replication of blocks.**
- **It periodically receives a Heartbeat and a Blockreport from each of the DataNodes in the cluster.**
- **Receipt of a Heartbeat implies that the DataNode is functioning properly. A**
- **Blockreport contains a list of all blocks on a DataNode.**

Data Replication



Replica Selection

- **To minimize global bandwidth consumption and read latency, HDFS tries closest replica selection.**
- **If there exists a replica on the same rack as the reader node, then that replica is preferred to satisfy the read request.**
- **If angg/ HDFS cluster spans multiple data centers, then a replica that is resident in the local data center is preferred over any remote replica.**

Safemode

- **On startup, the NameNode enters a special state called Safemode.**
- **Replication of data blocks does not occur when the NameNode is in the Safemode state.**
- **The NameNode receives Heartbeat and Blockreport messages from the DataNodes.**
- **Each block has a specified minimum number of replicas.**

Safemode

- A block is considered safely replicated when the minimum number of replicas of that data block has checked in with the NameNode.
- After a configurable percentage of safely replicated data blocks checks in with the NameNode (plus an additional 30 seconds), the NameNode exits the Safemode state.
- It then determines the list of data blocks (if any) that still have fewer than the specified number of replicas.
- The NameNode then replicates these blocks to other DataNodes.

The Persistence of File System Metadata-EditLog

- The HDFS namespace is stored by the NameNode.
- The NameNode uses a transaction log called the EditLog to persistently record every change that occurs to file system metadata.
- For example, creating a new file in HDFS causes the NameNode to insert a record into the EditLog indicating this.
- Similarly, changing the replication factor of a file causes a new record to be inserted into the EditLog.
- EditLog are stored as a file in local host OS file system.

FsImage

- **The entire file system namespace, including the mapping of blocks to files and file system properties, is stored in a file called the FsImage.**
- **The FsImage is stored as a file in the NameNode's local file system too.**
- **The NameNode keeps an image of the entire file system namespace and file Blockmap in memory.**
- **This key metadata item is designed to be compact, such that a NameNode with 4 GB of RAM is plenty to support a huge number of files and directories.**

Checkpoint

- **When the NameNode starts up,**
  - it reads the FsImage and EditLog from disk,
  - applies all the transactions from the EditLog to the in-memory representation of the FsImage, and
  - flushes out this new version into a new FsImage on disk.
- **It can then truncate the old EditLog because its transactions have been applied to the persistent FsImage.**
- **This process is called a checkpoint.**
- **In the current implementation, a checkpoint only occurs when the NameNode starts up.**
- **Work is in progress to support periodic checkpointing in the near future.**

DataNode Storage

- The DataNode stores HDFS data in files in its local file system.
- The DataNode has no knowledge about HDFS files.
- It stores each block of HDFS data in a separate file in its local file system.
- The DataNode does not create all files in the same directory. Instead, it uses a heuristic to determine the optimal number of files per directory and creates subdirectories appropriately.
- It is not optimal to create all local files in the same directory because the local file system might not be able to efficiently support a huge number of files in a single directory.

Blockreport

- When a DataNode starts up, it scans through its local file system, generates **a list of all HDFS data blocks** that **correspond to each of these local files** and sends this report to the NameNode: this is the Blockreport.

The Communication Protocols in HDFS

- All HDFS communication protocols are layered on top of the TCP/IP protocol.
- A client establishes a connection to a configurable TCP port on the NameNode machine.
- It talks the ClientProtocol with the NameNode.
- The DataNodes talk to the NameNode using the DataNode Protocol.
- A Remote Procedure Call (RPC) abstraction wraps both the Client Protocol and the DataNode Protocol.
- By design, the NameNode never initiates any RPCs. Instead, it only responds to RPC requests issued by DataNodes or clients.

Staging- File Creation Operation in HDFS

- **A client request to create a file does not reach the NameNode immediately.**
- **In fact, initially the HDFS client caches the file data into a temporary local file.**
- **Application writes are transparently redirected to this temporary local file.**
- **When the local file accumulates data worth over one HDFS block size, the client contacts the NameNode.**
- **The NameNode inserts the file name into the file system hierarchy and allocates a data block for it.**

Staging- File Creation Operation in HDFS

- The NameNode responds to the client request with the **identity of the DataNode and the destination data block**.
- Then the **client flushes the block** of data from the local temporary file to the specified DataNode.
- When a file is closed, the remaining un-flushed data in the temporary local file is transferred to the DataNode.
- The client then tells the NameNode that the file is closed.
- At this point, the NameNode commits the file creation operation into a **persistent store**.
- If the NameNode dies before the file is closed, the file is lost.

Replication Pipelining- Writing data to HDFS from one DataNode to another DataNode

- **When a client is writing data to an HDFS file, its data is first written to a local file.**
- **Suppose the HDFS file has a replication factor of three.**
- **When the local file accumulates a full block of user data, the client retrieves a list of DataNodes from the NameNode.**
- **This list contains the DataNodes that will host a replica of that block.**
- **The client then flushes the data block to the first DataNode.**

Replication Pipelining- Writing data to HDFS from one DataNode to another DataNode

- The first DataNode starts receiving the data in small portions (4 KB), writes each portion to its local repository and transfers that portion to the second DataNode in the list.
- The second DataNode, in turn starts receiving each portion of the data block, writes that portion to its repository and then flushes that portion to the third DataNode.
- Finally, the third DataNode writes the data to its local repository.
- Thus, a **DataNode can be receiving data from the previous one in the pipeline and at the same time forwarding data to the next one in the pipeline**.
- Thus, the data is pipelined from one DataNode to the next.

Space Reclamation- What happens when you remove something?

- **There is two techniques to reclaim space in HDFS architecture:-**
  - File Deletes and Undeletes
  - Decrease Replication Factor

File Deletes

- When a file is deleted by a user or an application, it is not immediately removed from HDFS.
- Instead, HDFS first renames it to a file in the /trash directory.
- The file can be restored quickly as long as it remains in /trash.
- A file remains in /trash for a configurable amount of time.
- After the expiry of its life in /trash, the NameNode deletes the file from the HDFS namespace.
- The deletion of a file causes the blocks associated with the file to be freed.
- Note that there could be an appreciable time delay between the time a file is deleted by a user and the time of the corresponding increase in free space in HDFS.

File Undeletes

- A user can Undelete a file after deleting it as long as it remains in the /trash directory.
- If a user wants to undelete a file that he/she has deleted, he/she can navigate the /trash directory and retrieve the file.
- The /trash directory contains only the latest copy of the file that was deleted.
- The /trash directory is just like any other directory with one special feature: HDFS applies specified policies to automatically delete files from this directory.
- The current default policy is to delete files from /trash that are more than 6 hours old. In the future, this policy will be configurable through a well defined interface.

Decrease Replication Factor

- When the replication factor of a file is reduced, the NameNode selects excess replicas that can be deleted.
- The next Heartbeat transfers this information to the DataNode.
- The DataNode then removes the corresponding blocks and the corresponding free space appears in the cluster.
- There might be a time delay between the completion of the setReplication API call and the appearance of free space in the cluster.

Resources

- **https://hadoop.apache.org/docs/current1/hdfs_design.html**
- **Hadoop: The Definitive Guide**
  - Tom White (Author)
  - O'Reilly Media; 4th Edition.