Break

15 minute break before the HDFS presentation

VLS HDFS Content

A brief overview of the HDFS architecture and how it works.

By Stephen Hunt

September 2021

Key Terms

- HDFS

The Hadoop Distributed Files System. A distributed file system designed to be highly fault tolerant and able to handle very large amounts of data.

- Name node

The core piece of the HDFS file system. The Name node stores the metadata, such as the directory tree, of all files stored in HDFS. The Name node is responsible for managing where in the cluster the file data is kept. The Name node does not store any of the file data itself.

- Data node

The component of HDFS that handles storing the file data. A functional HDFS system has more than one Data node, with the data replicated across them. Data in these nodes are stored as blocks, with a default block size of 128MB. The Data node periodically sends a list of blocks it is storing to the Name node.

- Block

A block is a file segment for the files stored on HDFS. When a file is added to HDFS, it is divided into 128MB blocks that are then stored in the Data nodes.

- Bastion Host

A host whose purpose is to provide access to a private network from an external network, such as the Internet. Because of its exposure to potential attack, a bastion host must minimize the chances of penetration.

aws

Core Aspects of HDFS

- ## Streaming Data Access

Applications running on HDFS require streaming access to the data. HDFS is designed more for batch processing than interactive with an emphasis on high throughput rather than low latency data access.

- ## Large Data Sets

HDFS is designed to support large files. A typical file in HDFS is expected to be gigabytes to terabytes in size. HDFS is designed to provide high aggregate data bandwidth, scale to hundreds of nodes in a cluster and support tens of millions of files in a single instance.

- ## Simple Coherency Model

HDFS applications need a write-once-read-many access for files. Once a file is created, written and closed it is not expected to be changed aside from appends and truncates. Appending content to the end of files is supported, but files cannot be updated at arbitrary points. This assumption simplifies data coherency issues and enables high throughput data access.

- ## Portability Across Platforms

HDFS is designed to be easily portable from one platform to another to facilitate widespread adoption.

aws

Poll – Which of these is a function of the NameNode

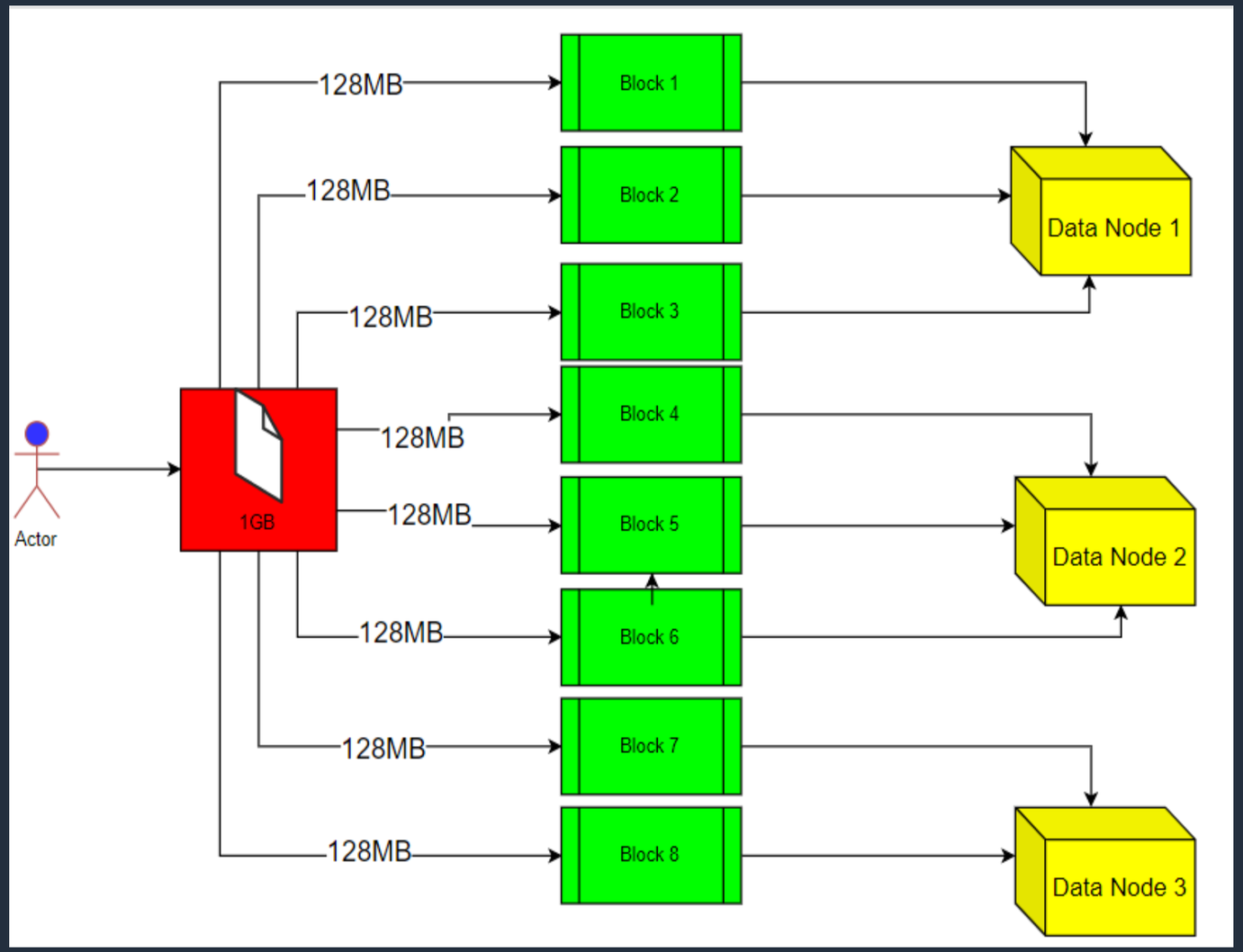
1. Stores blocks
2. Allows a secure connection to a private network
3. Run HDFS applications
4. Store filesystem metadata

How HDFS Stores Files in Blocks

1. Actor writes a 1GB files to HDFS

2. The file is broken into blocks of 128MB

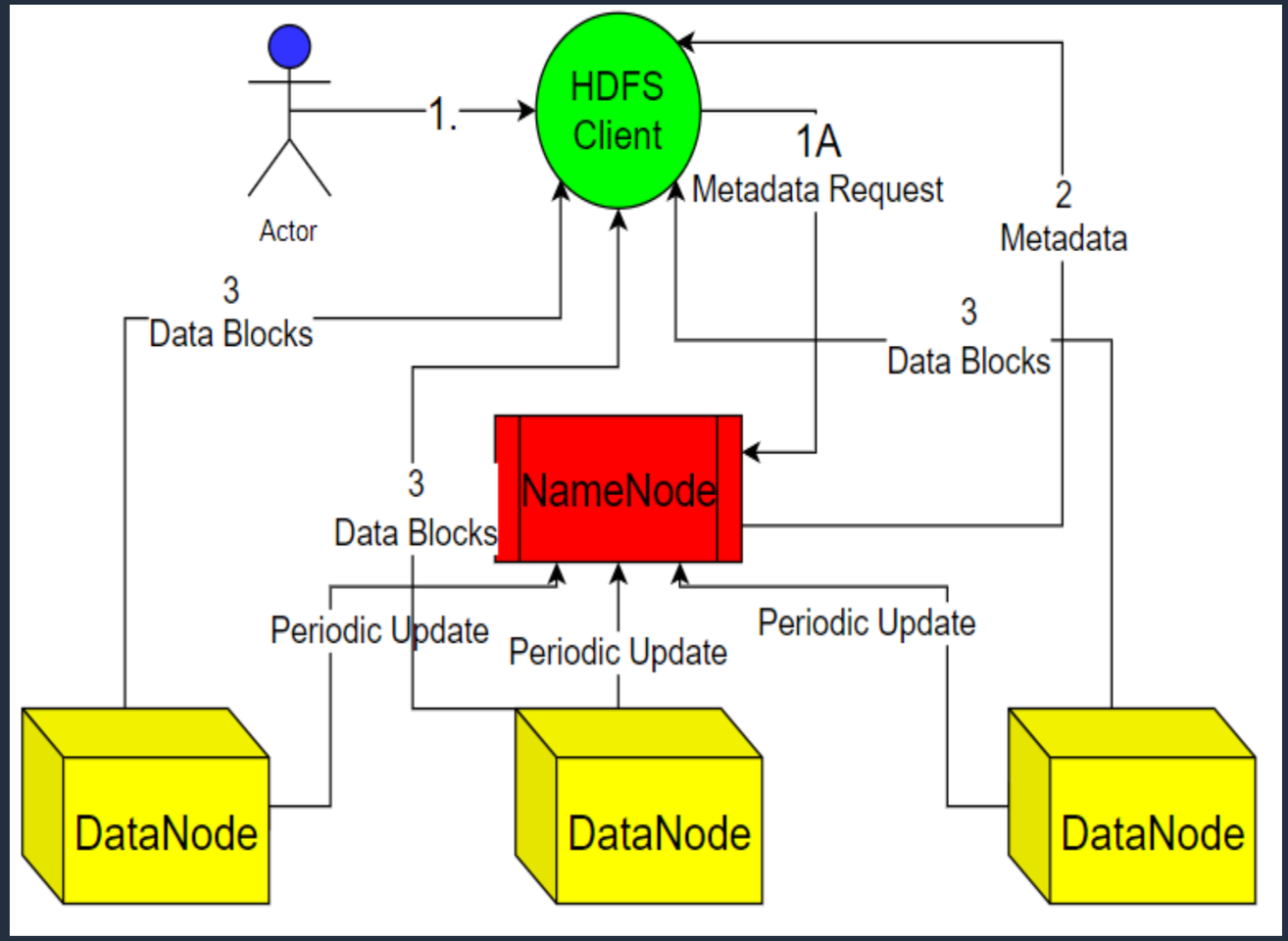
3. These blocks are distributed across all Data Nodes



How A Read/Write is Handled in HDFS

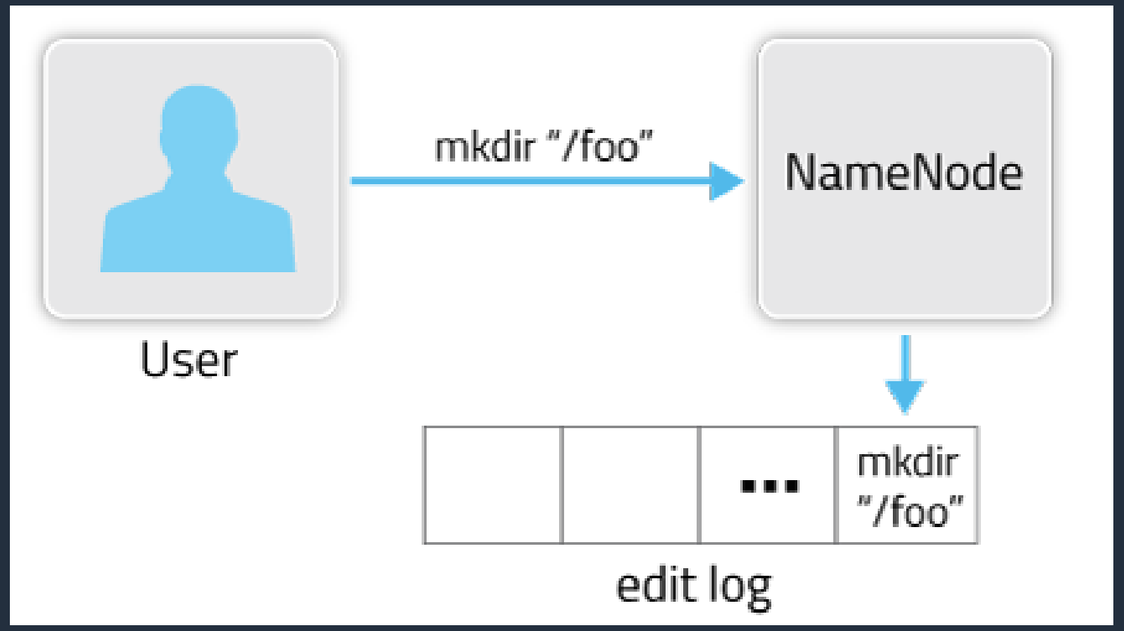
Client Application tries to read/write a file on HDFS

1. Client requests file system metadata.
2. NameNode sends Client the metadata, which includes a list of relevant Data nodes where the data lives.
3. Client application now communicates directly with the DataNodes to read/write data



Persistence of File System Metadata

- HDFS namespace stored by the NameNode.
- NameNode uses a transaction log called the EditLog to persistently record every change that occurs to the file system metadata.
- Includes reads, writes, changes to replication factor etc. The entire namespace, including mapping of blocks to files and file system properties, is stored in a file called the FsImage.
- Both the EditLog and FsImage are stored on the NameNode's local file system.
- The NameNode keeps an image of the entire namespace and file Blockmap in memory.



Poll – How does HDFS ensure that blocks are not prematurely replicated when loading a previous state?

1. Blocks all actions while starting
2. Blocks are not replicated during start-up
3. Only allows read-only access during start-up
4. Creates a duplicate of the metadata and uses that to restore state

HDFS Safemode

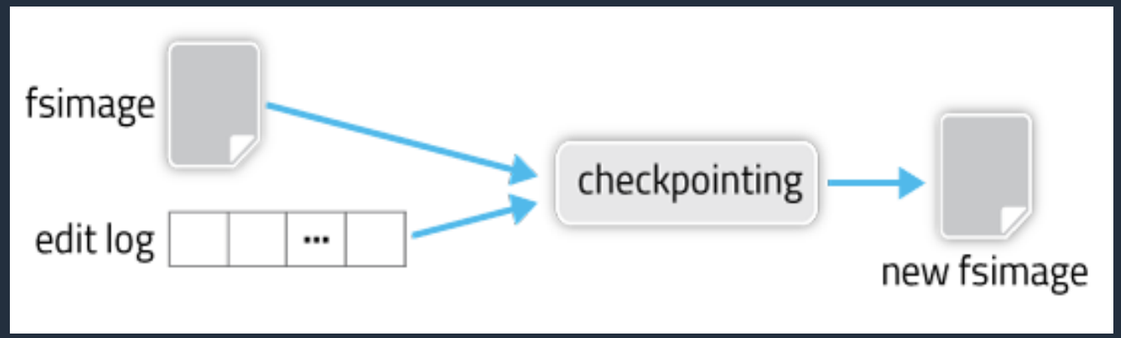
- Read-only mode.
- Used when loading file system state.
- Prevents premature replication of blocks when waiting for DataNodes to report their blocks.

- hdfs dfsadmin –safemode enter

NameNode Start-up and Checkpointing

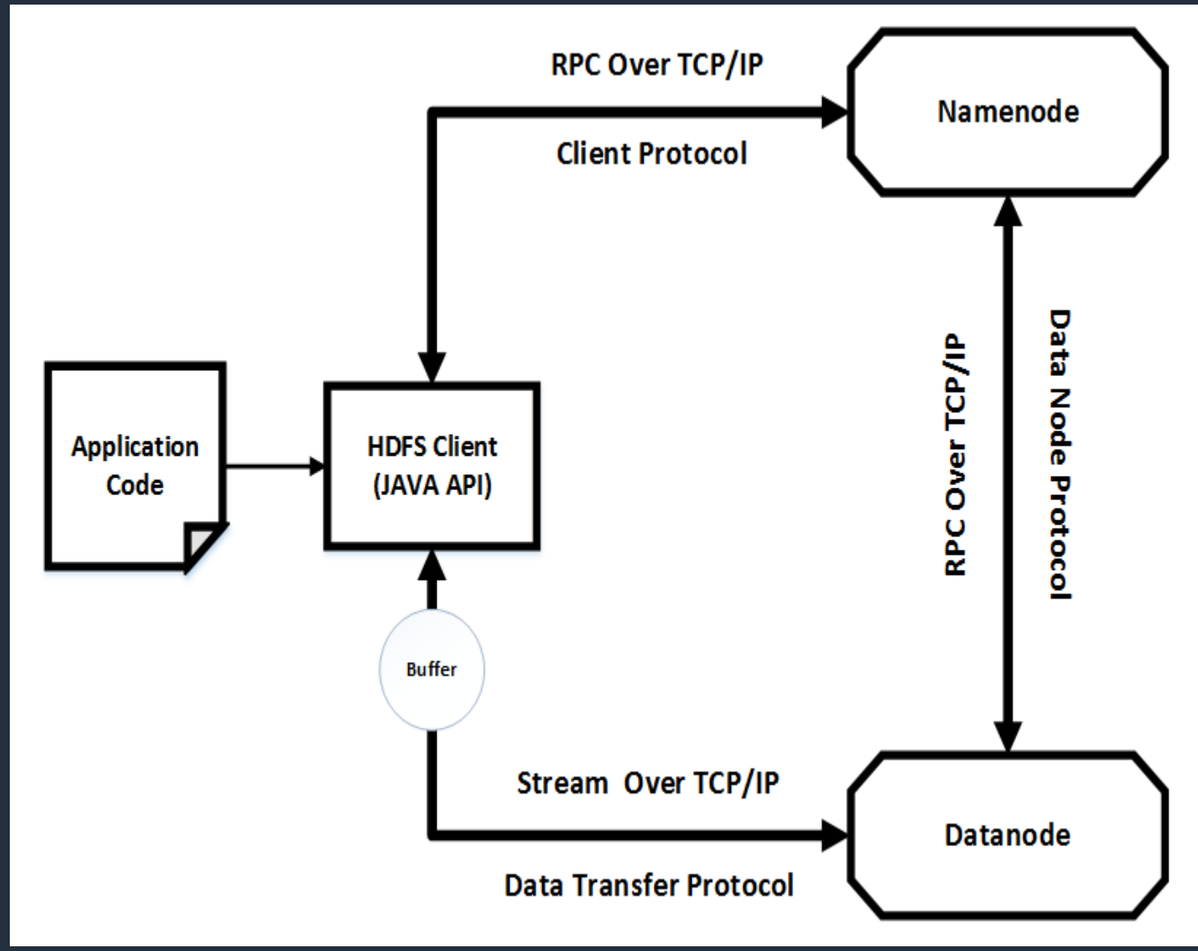
1. NameNode enters Safe Mode.
2. NameNode reads the FsImage and EditLog from disk and applies all transactions from the EditLog to the in-memory representation of the FsImage.
3. The in-memory representation of the FsImage is flushed into a new FsImage on disk.
4. The old EditLog is truncated as all transactions have been applies.
5. NameNode exits Safe Mode.

- This process ensures that HDFS maintains a consistent view of the files system metadata.



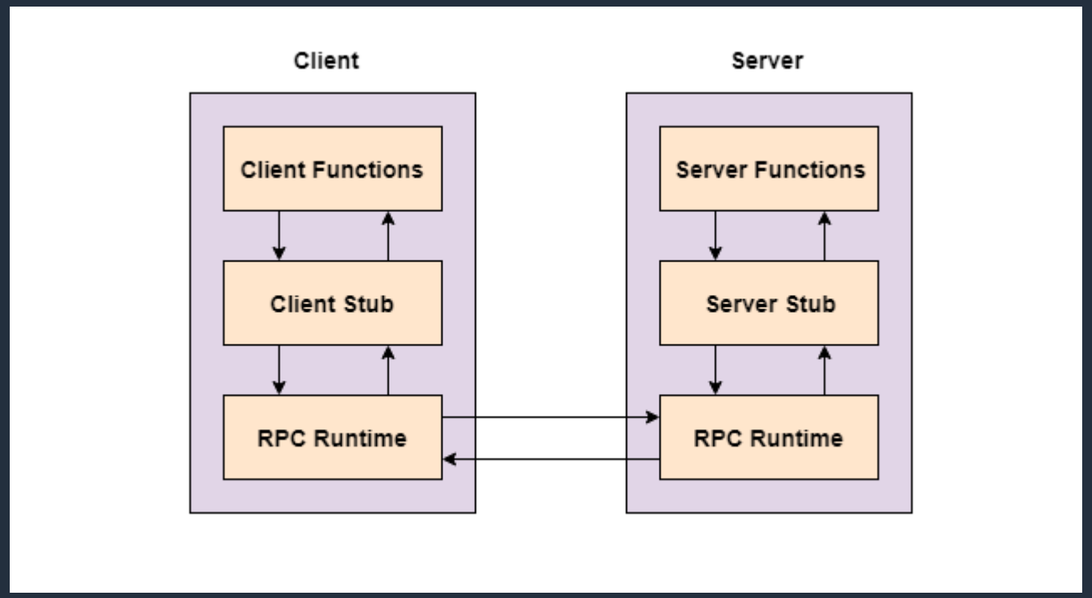
HDFS Communication Protocols

- TCP/IP
- Client Protocol
- Data Node Protocol
- Remote Procedure Call (RPC) wraps both.
- The NameNode never initiates, it only responds to DataNodes or clients.



Remote Procedure Calls

- Inter-process communication technique.
- Abstracts message passing system away from user.
- Stalls execution on Client until a response is received.

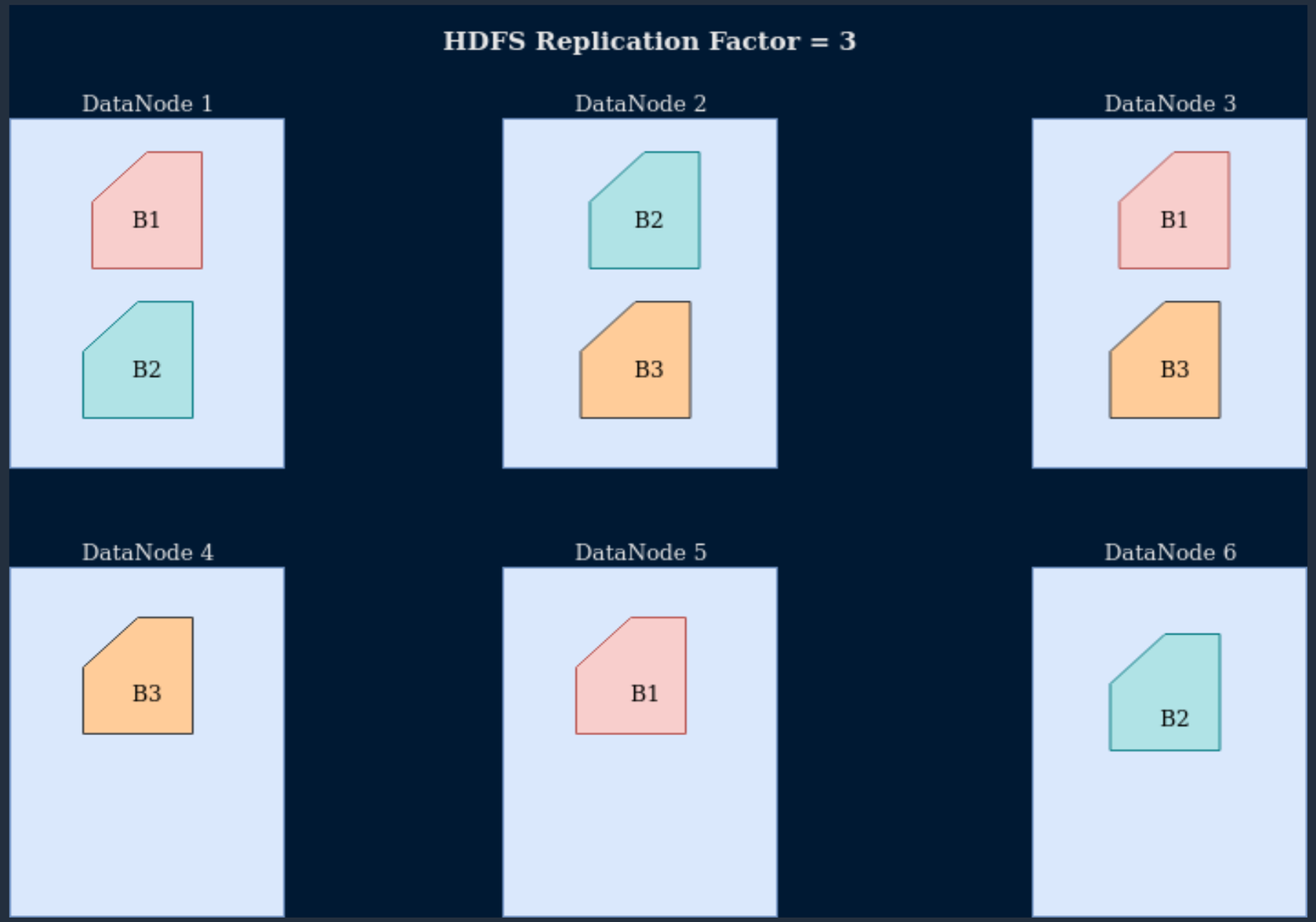


Poll – What happens when DataNode fails?

1. Blocks are replicated back to the set factor if possible
2. Nothing. The blocks have a copy stored on the NameNode
3. The blocks are loaded into the FsImage
4. None of the above

HDFS Replication Factor

- To ensure that HDFS is fault tolerant, data is replicated to a set factor.
- 3 blocks in HDFS with a Replication factor of 3 equals a total of 9 blocks.
- Replication increases overall storage needs. Storing 1TB with a replication factor of 3 needs 3TB storage.

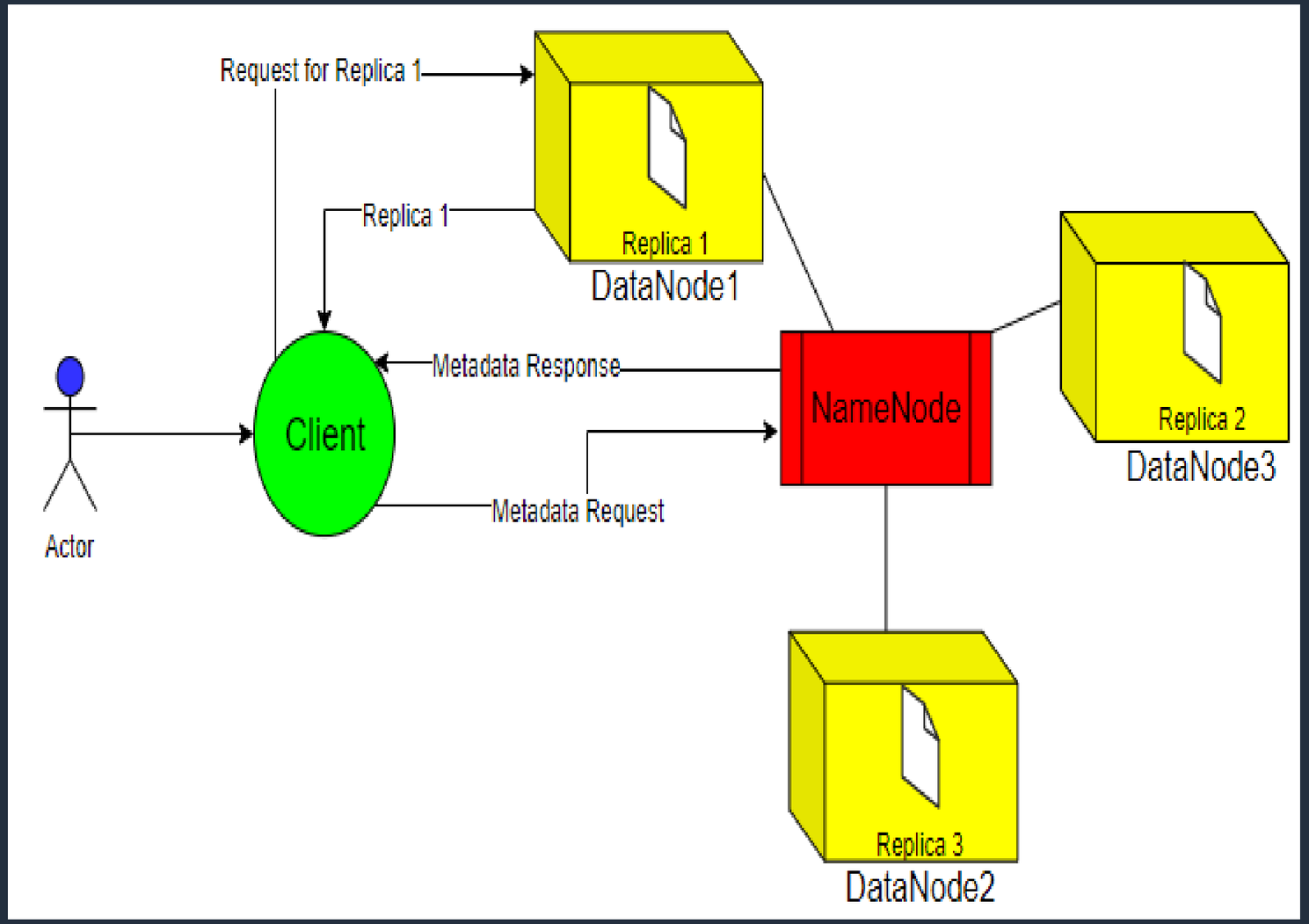


Rack Awareness

- HDFS block placement avoids placing replicas on the same rack to provide better data resiliency and availability.
- Rack ID found using either a specified external script or via a configured Java class.
- Interface expects a one-to-one correspondence to be maintained and topology information to be in the format ‘/rack/host’ where ‘/’ is the topology delimiter, ‘rack’ is the rack identifier and ‘host’ is the individual host.

Replica Selection on Read

- HDFS will try satisfy a read with the replica that is closest to the reader.
- Local replicas are favoured over remote replicas.
- Reduces global bandwidth consumption and read latency.



Data Node Disk Usage and HDFS

- Files stored on HDFS are distributed amongst all Data Nodes.
- Overutilization of HDFS can cause high disk usage on Data Node instances and cause issues with frameworks such as YARN.
- Check total HDFS size and usage using ‘hdfs dfsadmin –report’
- Check Data Node instance disk utilisation using ‘df –h’

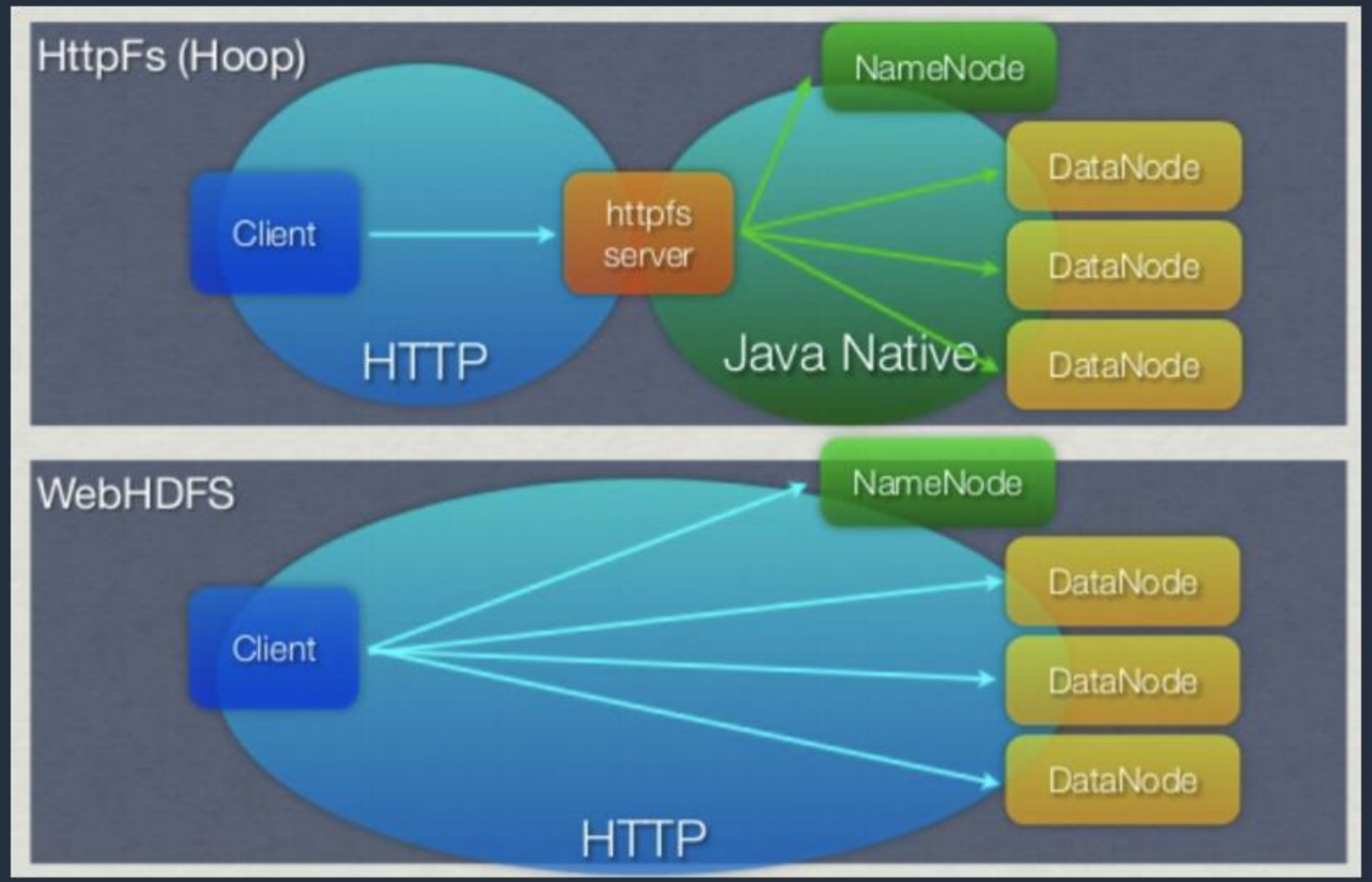
HttpFS vs WebHDFS

HttpFS

- A single node acts as a ‘gateway’
- Potential chokepoint during file transfer.
- Help minimize footprint required to access HDFS.

WebHDFS

- Needs access to all nodes.
- Data is transmitted from node directly when read



Using HttpFS - Setup

1. Ensure that the Security Group on the Master allows traffic on port 14000
2. If the cluster is in a private subnet:
    1. Configure a Bastion host in a public subnet of the same VPC
    2. Ensure the Bastion host Security Group allows traffic from your IP on port 22
    3. Ensure Master Security Group allows traffic from the Bastion Security Group
    4. SSH into the Bastion to access HttpFS

Poll – What Command Line tool can be used to make HTTP requests?

1. ping
2. netstat
3. traceroute
4. curl

Using HttpFS

- Create a directory inside the tmp directory called mydir

```
curl -i -X PUT "http://<master-ip>:14000/webhdfs/v1/tmp/mydir?op=MKDIRS&user.name=hadoop"
```

- Push a local files inside mydir

```
curl -i -X PUT -L --header "Content-Type: application/octet-stream"
"http://<masterip>:14000/webhdfs/v1/tmp/mydir/file1.txt?op=CREATE&user.name=hadoop" -T file1.txt
```

- List all files inside the mydir directory

```
curl -i "http://<master-node-ip>:14000/webhdfs/v1/tmp/mydir?op=LISTSTATUS&user.name=hadoop"
```

- Get the file content of a file

```
curl -i -X GET -L "http://<master-nodeip>:14000/webhdfs/v1/tmp/mydir/file1.txt?op=OPEN&user.name=hadoop"
```

- Delete a file.

```
curl -i -X DELETE "http://<masterip>:14000/webhdfs/v1/tmp/mydir/file1.txt?op=DELETE&user.name=hadoop"
```

- Delete the mydir folder

```
curl -i -X DELETE "<master-ip>:14000/webhdfs/v1/tmp/mydir/?op=DELETE&user.name=hadoop&recursive=true"
```

HDFS and User Home Directories

- Only users that exist on the NameNode exist to HDFS.
- Requests for HDFS files all have to go through the NameNode.
- User and groups data is retrieved from the NameNode requests.
- If a user or group exists on a DataNode instance, but not on the NameNode instance, it does not exist in HDFS.

aws

The Small Files Problem

- Each file is stored in its own block.
- The NameNode needs to store information about each block.
- MapReduce processes each block one at a time.
- Lots of small files means potentially hitting the OS open file limits.

- 10 000 1MB files vs 10 1GB files?
- ➢ 10 000 1MB files = 10 000 blocks in HDFS
- ➢ 10 1GB files = 80 ((1GB/128) * 10) blocks in HDFS

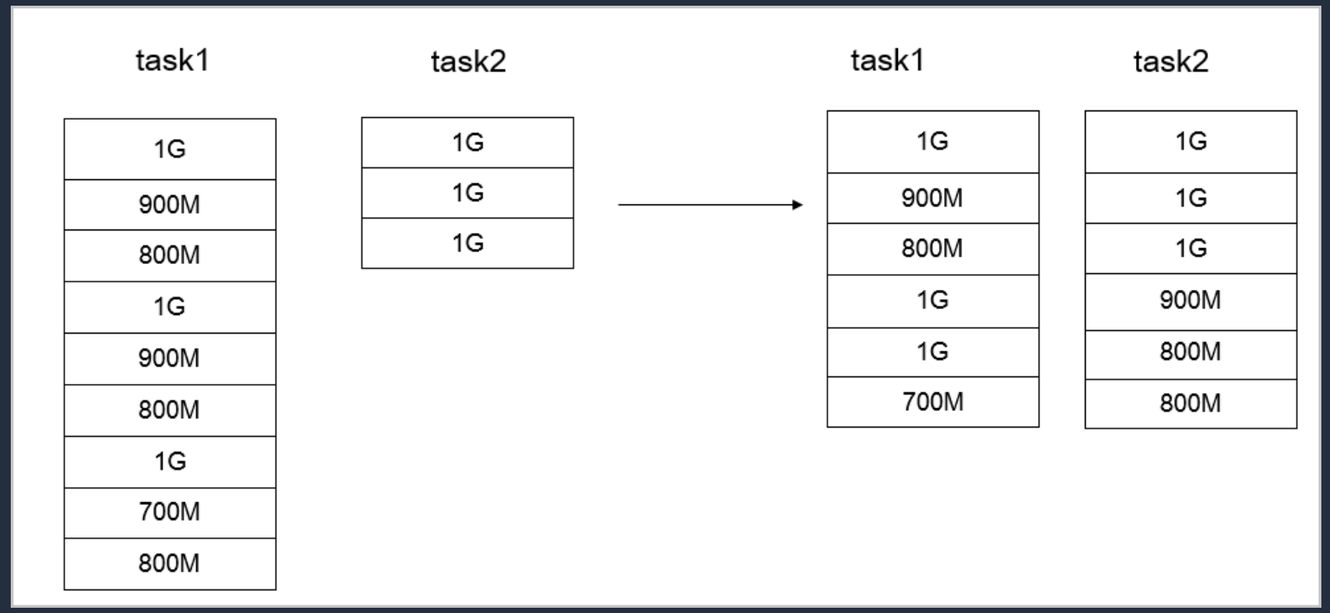
The Small Files Problem - Solutions

- Group files together into larger files.

This method effectively avoid the small files problem, but does not work for all situations (e.g. thousands of image files)

- Use HDFS's sync() method.

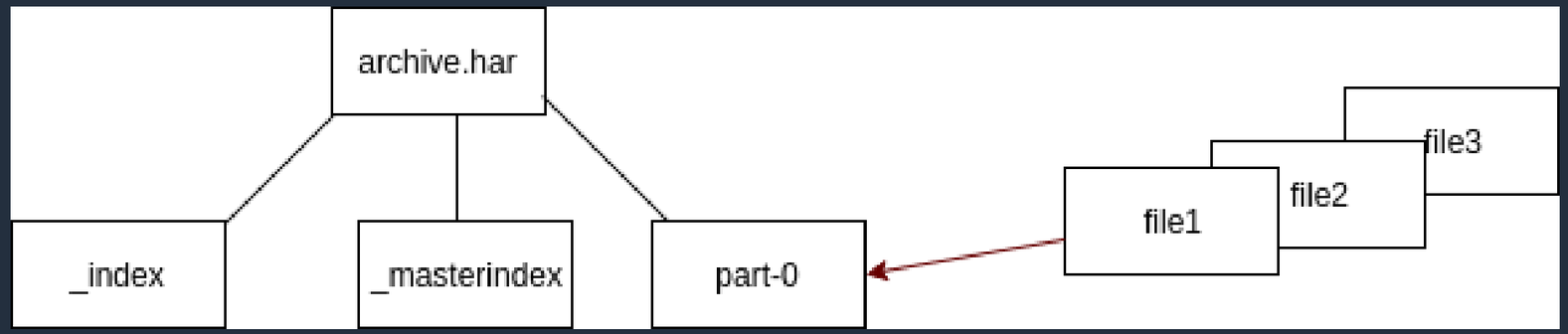
This is an HDFS method of writing large files, but also does not work for all situations.



The Small Files Problem - Solutions

- Use HAR (Hadoop Archives) to build a layered file system on top of HDFS.

HAR files aim to reduce the number of files by grouping them in archives. However, this comes at the cost of file access being slowed due to two index file reads on top of the data file read.



- Use Sequence Files to allow for streaming of the small files.

Sequence files work on the concept of using the filename as a key and the contents as a value. These files can then be processed in a streaming fashion and split into chunks for better parallelism.

Break

15 minute break before the lab

Hands-on Lab for HDFS

- Log in to the EMR Master Node
- Run HDFS DFS and Admin commands
- Increase the replication factor
- Use Safe Mode

- Find the lab at: https://github.com/aws-support-bigdata-cpt-vls/2021/tree/main/Day%201/HDFS%20Lab

aws