Practical Hadoop with Pig

HDFS

HDFS has 3 main actors

The Name Node

The Name Node is “The Conductor”.

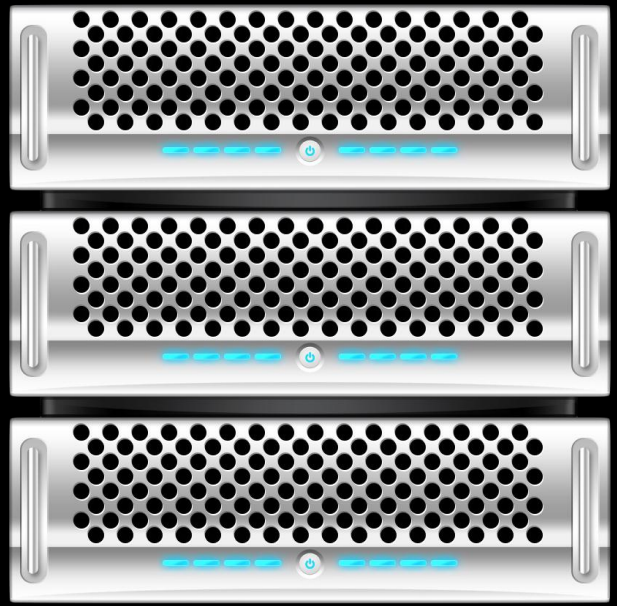
It directs the performance of the cluster.

The Data Nodes:

A Data Node stores blocks of data.

Clusters can contain thousands of Data Nodes.

*Yahoo has a 40,000 node cluster.



The Client

The client is a window to the cluster.

The Name Node

The heart of the System.

Maintains a virtual File Directory.

Tracks all the nodes.

Listens for “heartbeats” and “Block Reports”

If the NameNode is down, the cluster is offline.

Storing Data

The Data Nodes

Add a Data Node:

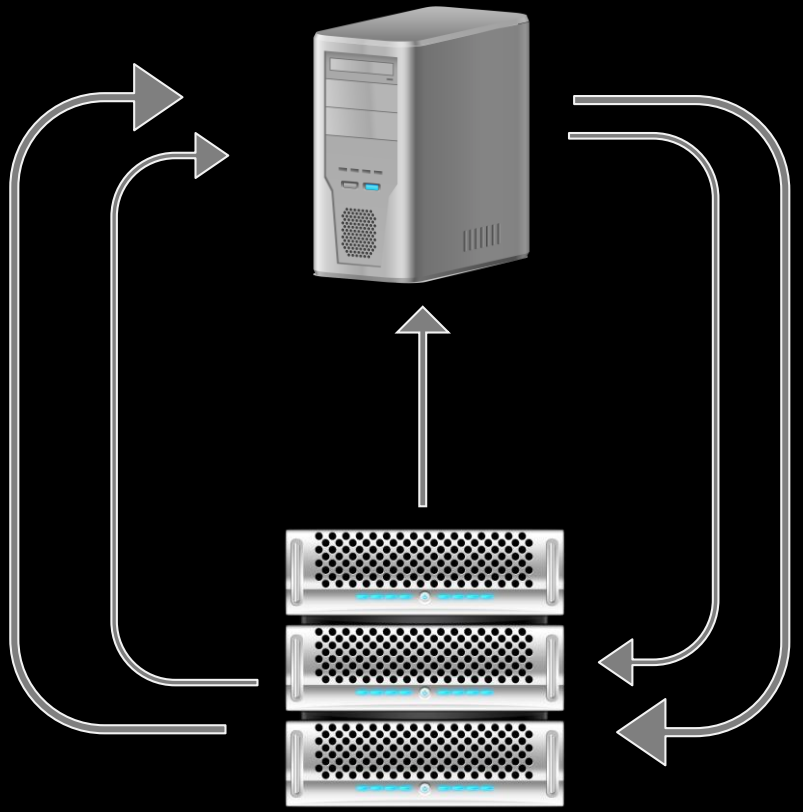
The Data Node says “Hello” to the Name Node.

The Name Node offers the Data Node a handshake with version requirements.

The Data Node replies back to the Name Node, “Okay”

The Name Node hands the Data Node a *NodeId* that it remembers.

The Data Node is now part of cluster and it checks in with the Name Node *every 3 seconds*.



Data Node Heartbeat:

The “check-in” is a simple HTTP Request/Response.

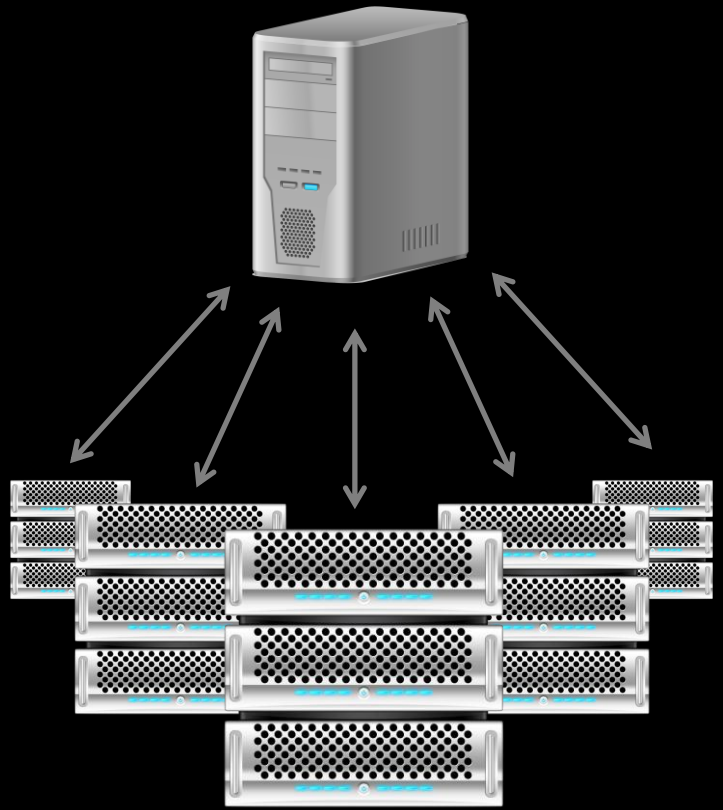
This "check-in" is very important communication protocol that guarantees the health of the cluster.

Block Reports – what data I have and is it okay.

Name Node controls the Data Nodes by issuing orders when they return and report their status.

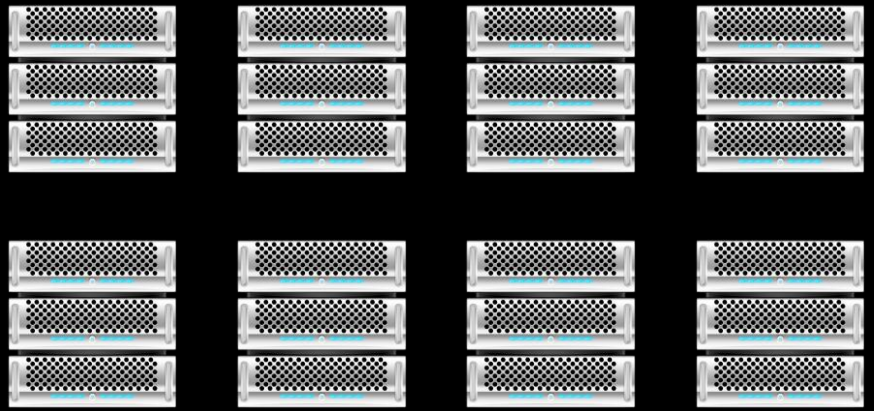
Replicate Data, *Delete* Data, *Verify* Data

Same process for all nodes within a cluster.



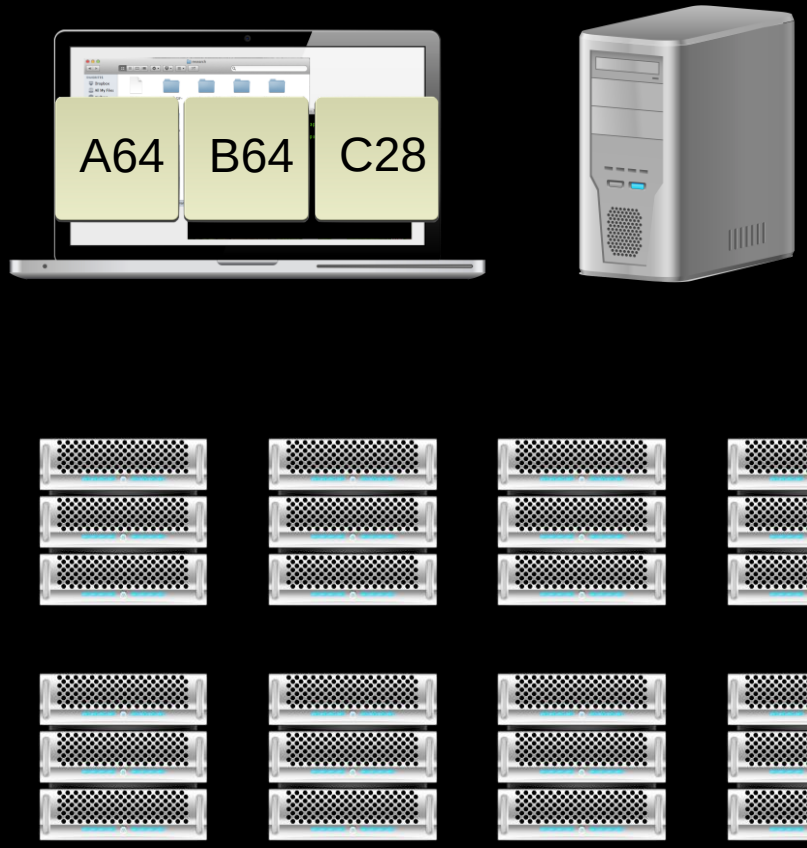
Writing Data

The client “tells” the NameNode the virtual directory location for the file.



The client “tells” the NameNode the virtual directory location for the file.

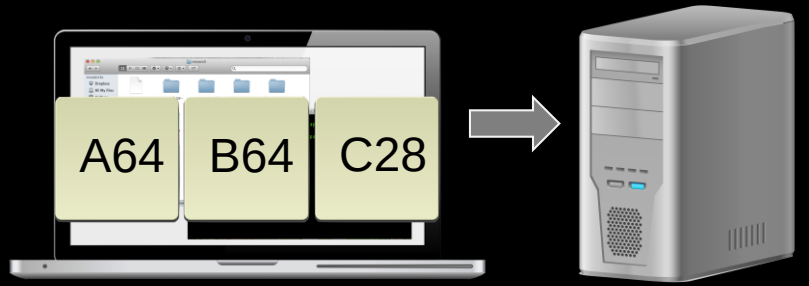
The Client breaks the file into 64MB “blocks”

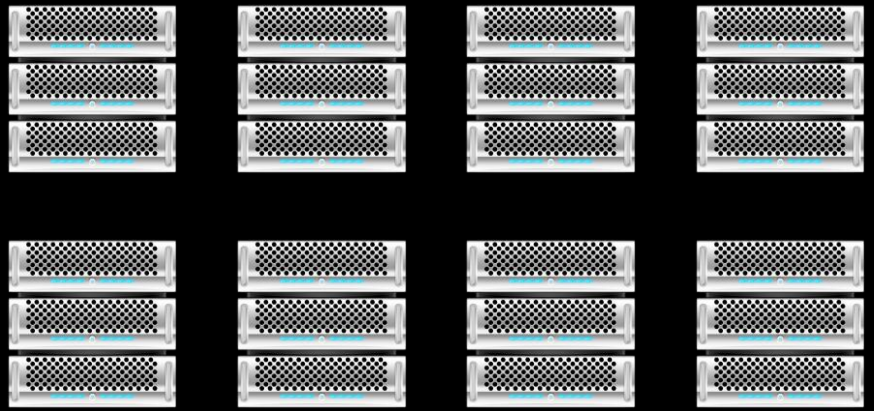


The client “tells” the NameNode the virtual directory location for the file.

The Client breaks the file into 64MB “blocks”

The client “ask” the NameNode where the blocks go.



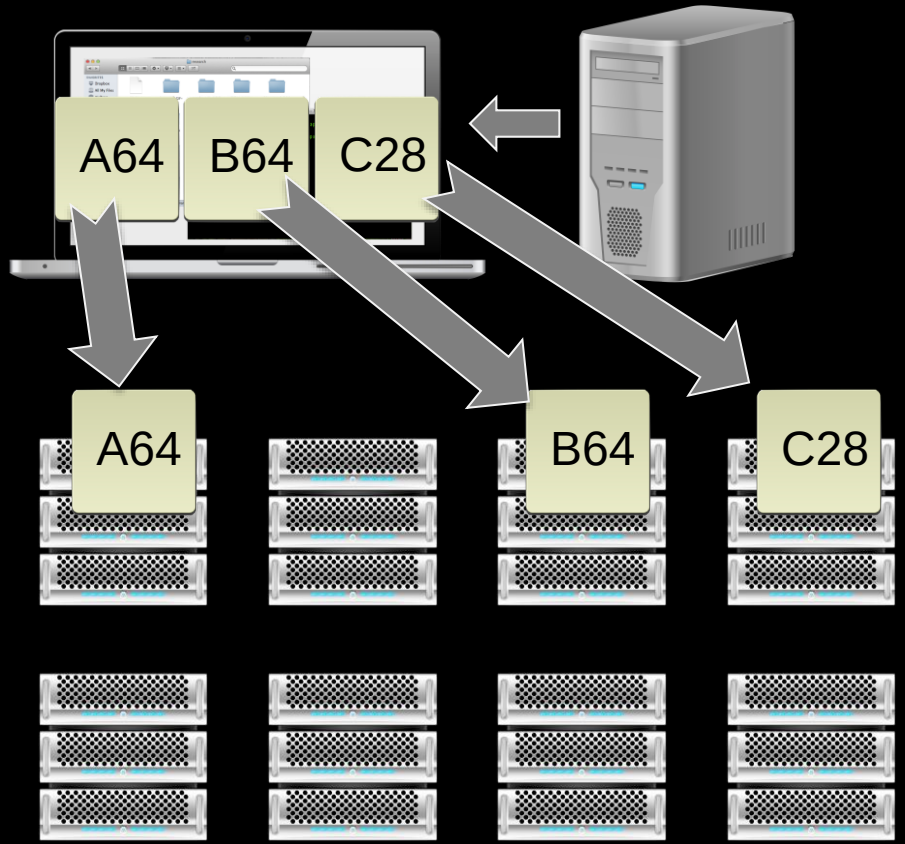


The client “tells” the NameNode the virtual directory location for the file.

The Client breaks the file into 64MB “blocks”

The client “ask” the NameNode where the blocks go.

The Client “stream” the blocks, in parallel, to the DataNodes.



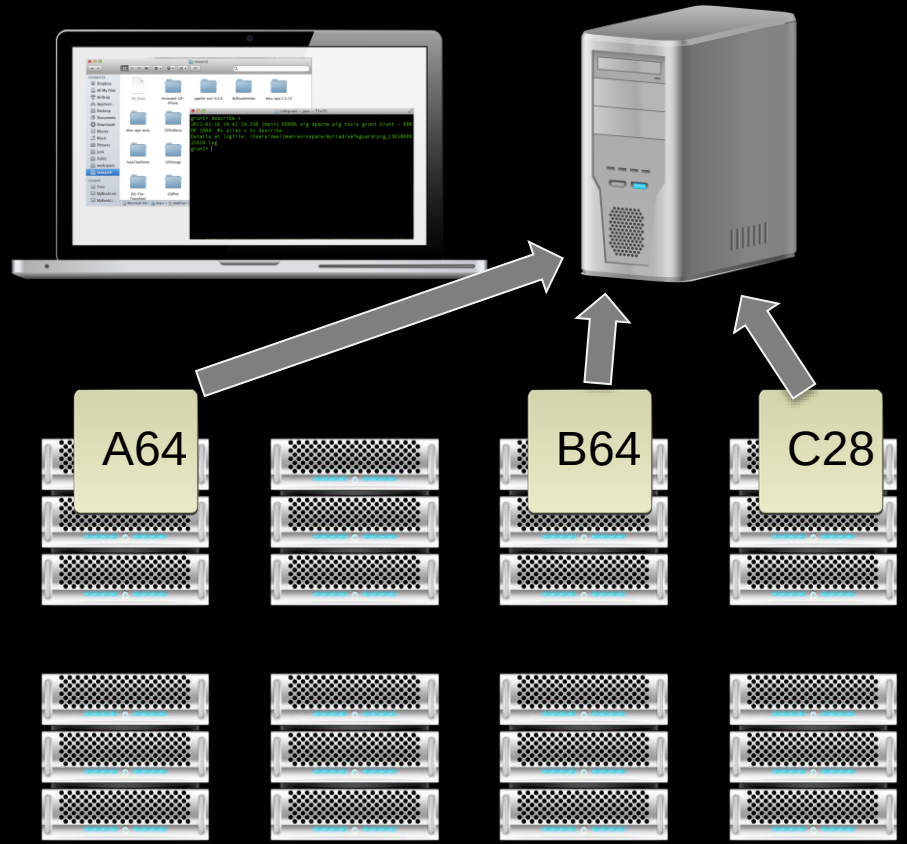
The client “tells” the NameNode the virtual directory location for the file.

The Client breaks the file into 64MB “blocks”

The client “ask” the NameNode where the blocks go.

The Client “stream” the blocks, in parallel, to the DataNodes.

DataNode(s) tells the NameNode they have the data via the block report



The client “tells” the NameNode the virtual directory location for the file.

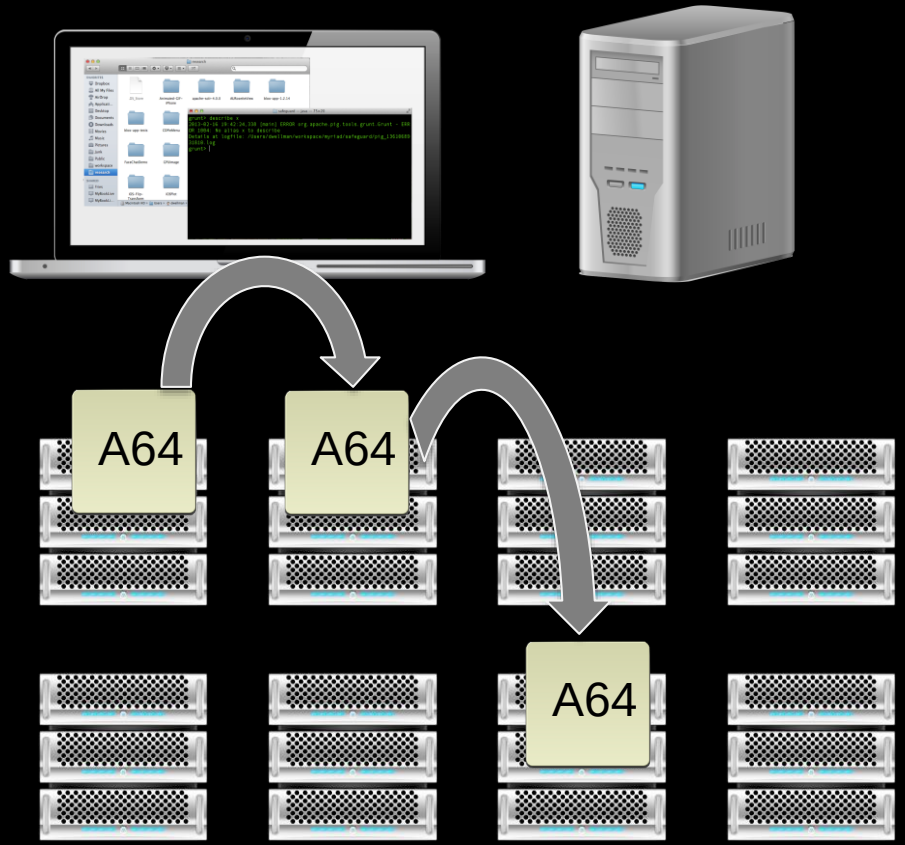
The Client breaks the file into 64MB “blocks”

The client “ask” the NameNode where the blocks go.

The Client “stream” the blocks, in parallel, to the DataNodes.

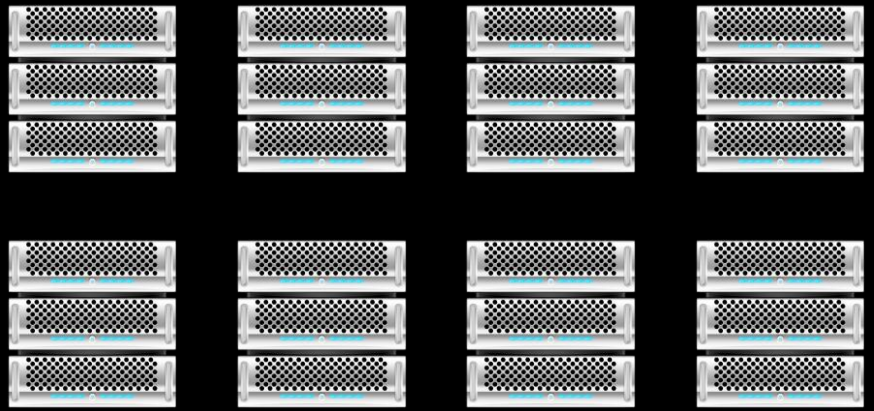
DataNode(s) tells the NameNode they have the data via the block report

The NameNode tells the DataNode where to replicate the block.



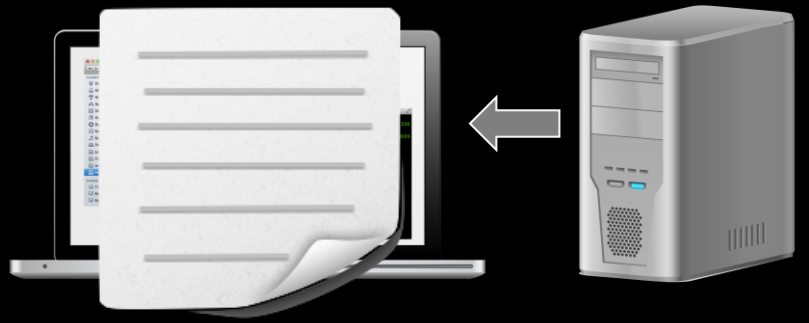
Reading Data

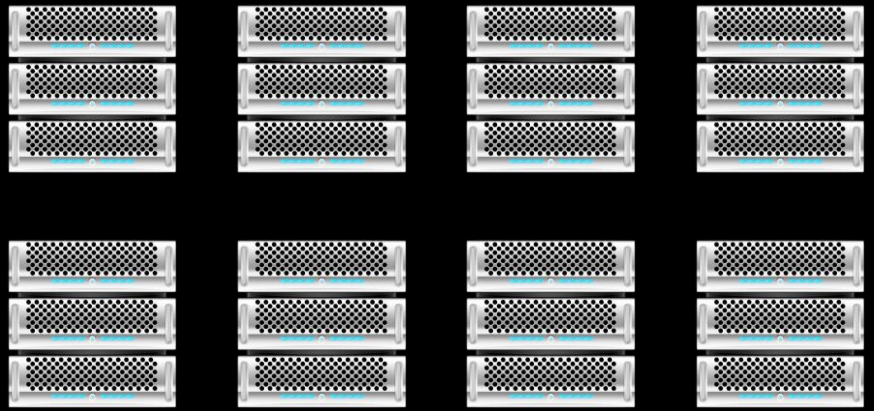
The client tells the NameNode it would like to read a file.



The client tells the NameNode it would like to read a file.

The NameNode reply's with the list of blocks and the nodes the blocks are on.

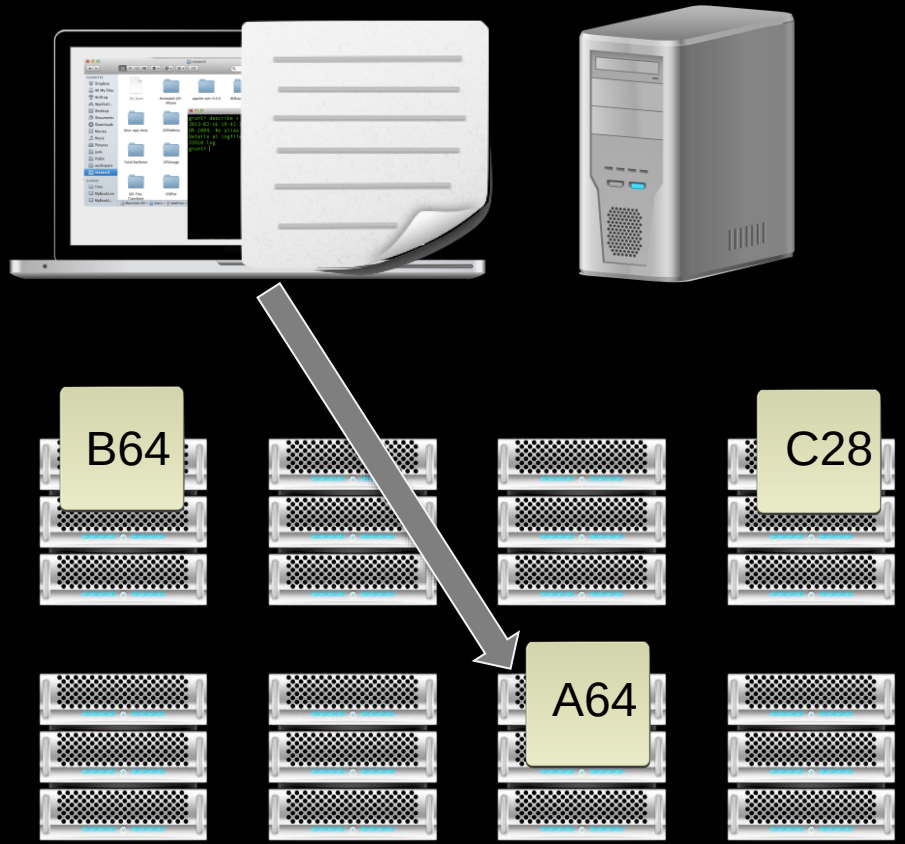




The client tells the NameNode it would like to read a file.

The NameNode reply's with the list of blocks and the nodes the blocks are on.

The client request the first block from a DataNode

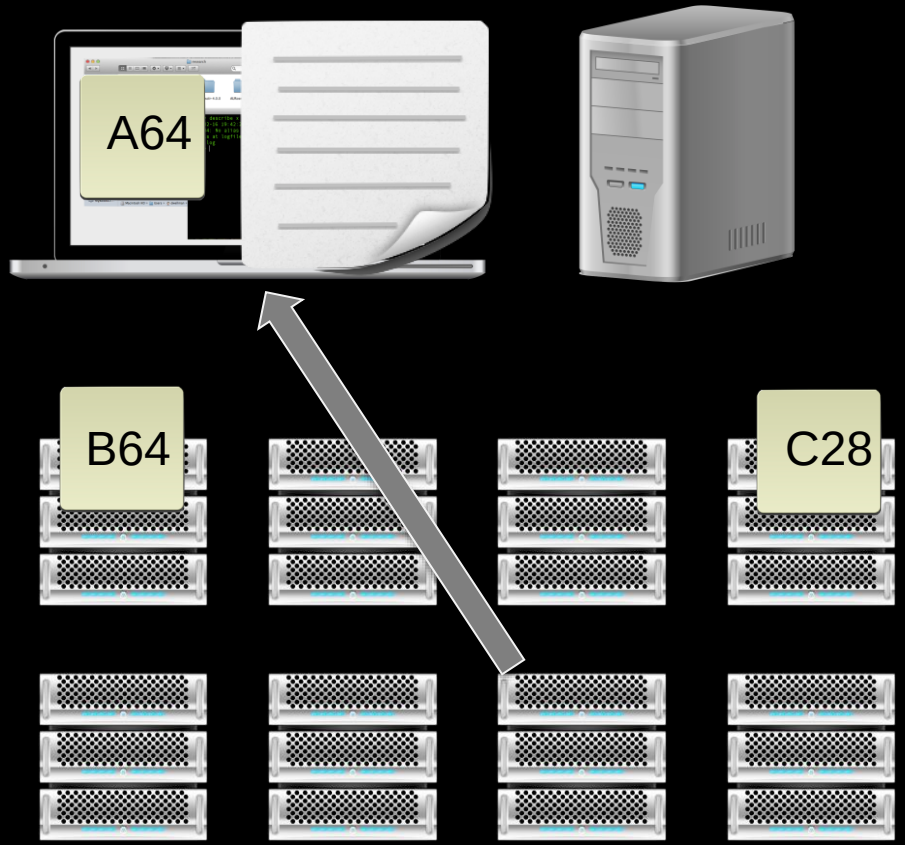


The client tells the NameNode it would like to read a file.

The NameNode reply's with the list of blocks and the nodes the blocks are on.

The client request the first block from a DataNode

The client compares the checksum of the block against the manifest from the NameNode.



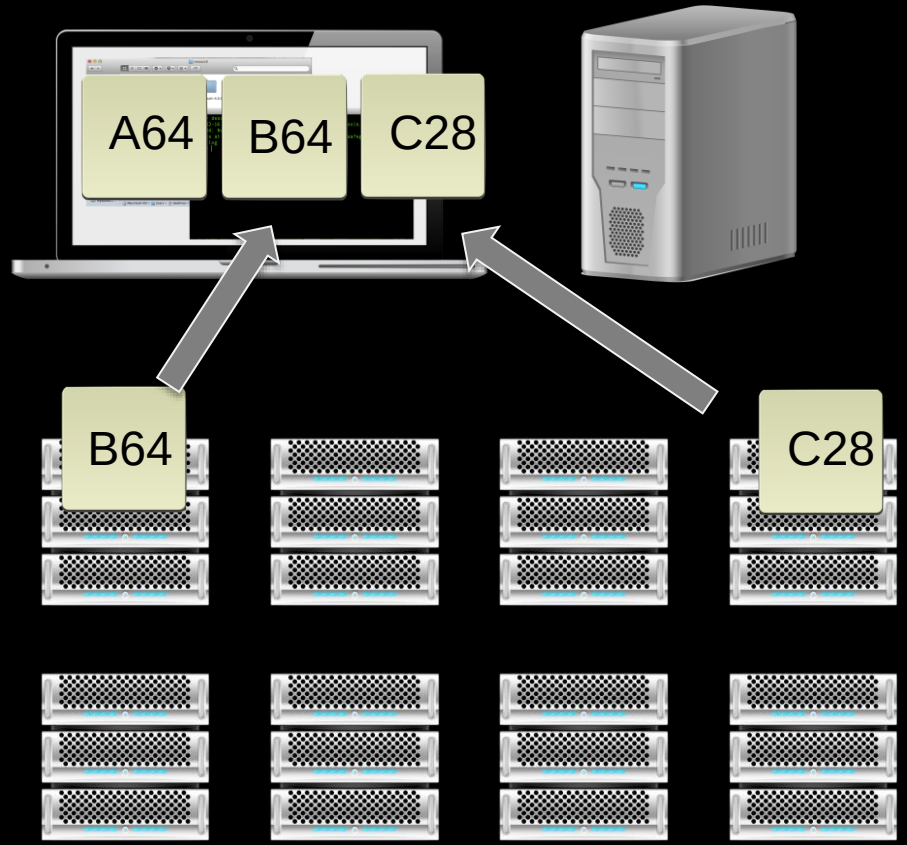
The client tells the NameNode it would like to read a file.

The NameNode reply's with the list of blocks and the nodes the blocks are on.

The client request the first block from a DataNode

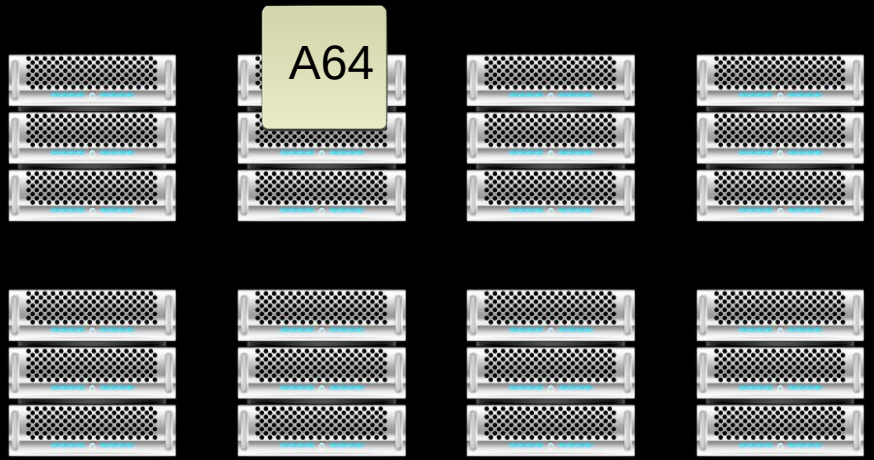
The client compares the checksum of the block against the manifest from the NameNode.

The client moves on to the next block in the sequence until the file has been read.



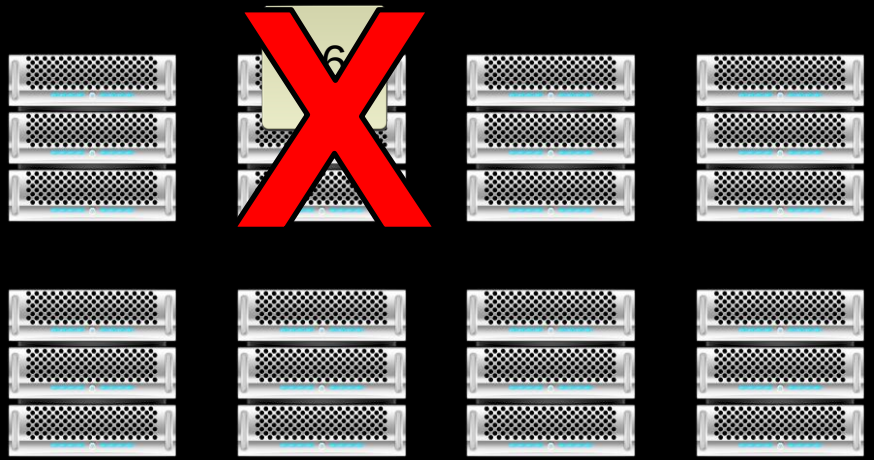
Failure Recovery

A Data Node Fails to “check-in”



A Data Node Fails to “check-in”

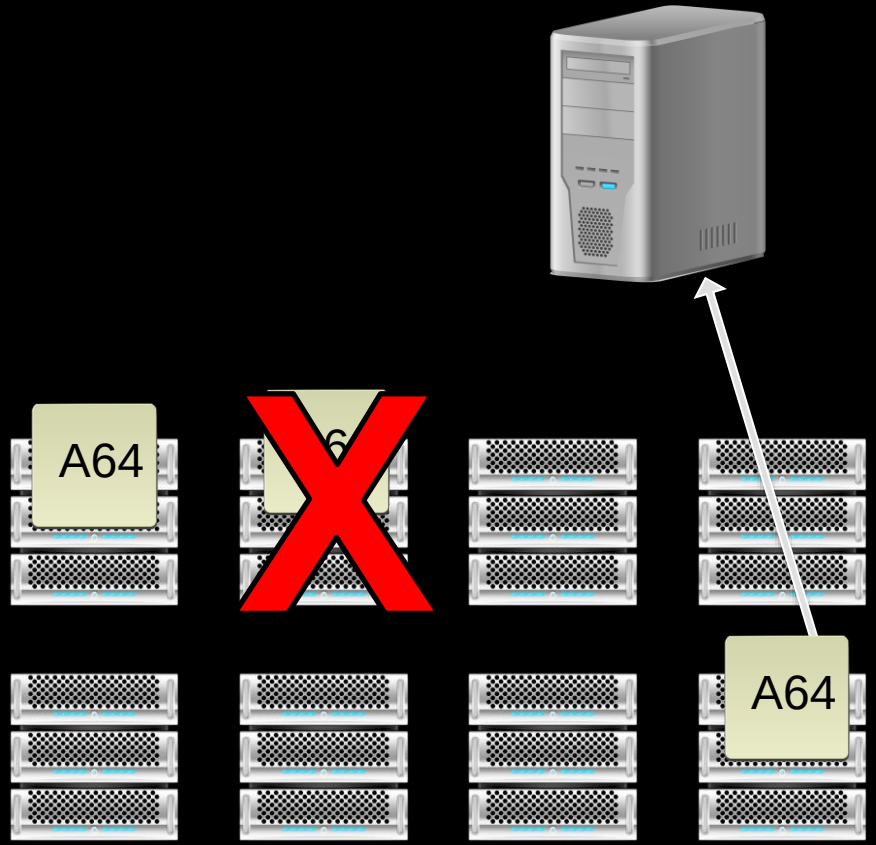
After 10 minutes the Name Node gives up on that Data Node.



A Data Node Fails to “check-in”

After 10 minutes the Name Node gives up on that Data Node.

When another node that has blocks originally assigned to the lost node checks-in, the name node sends a block replication command.

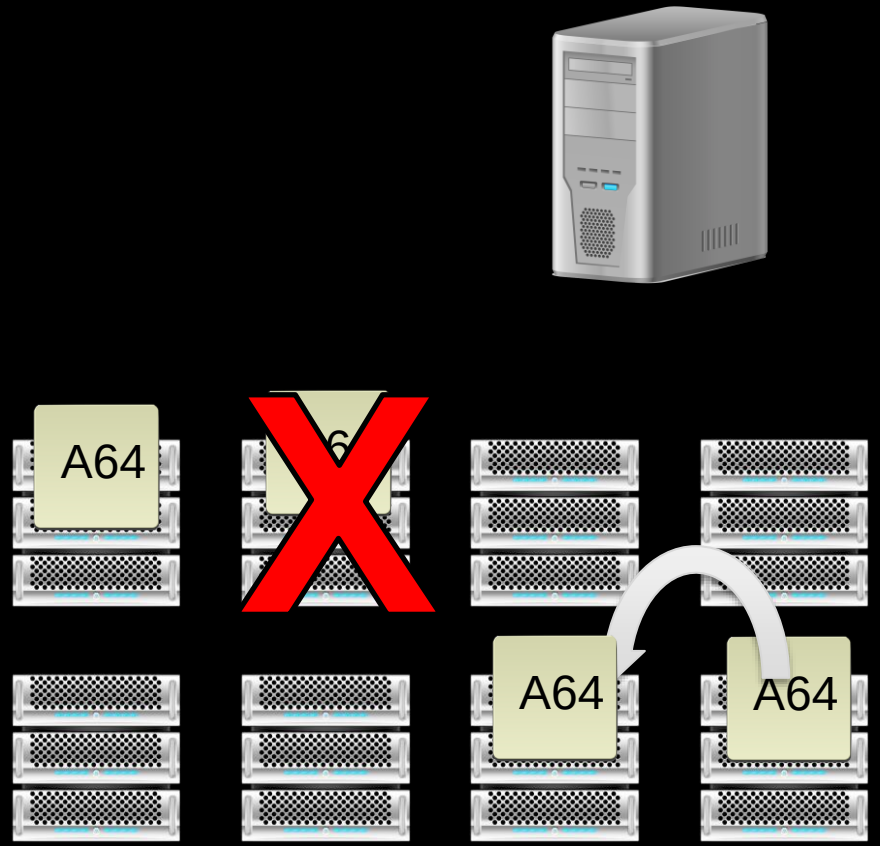


A Data Node Fails to “check-in”

After 10 minutes the Name Node gives up on that Data Node.

When another node that has blocks originally assigned to the lost node checks-in, the name node sends a block replication command.

The Data Node replicates that block of data. (Just like a write)



Interacting with Hadoop

HDFS Shell Commands

HDFS Shell Commands.

```
> Hadoop fs –ls <args>
```

Same as unix or osx ls command.

```
/user/hadoop/file1
/user/hadoop/file2
...
```

HDFS Shell Commands.

```
> Hadoop fs –mkdir <path>
```

Creates directories in HDFS using path.

HDFS Shell Commands.

```
> hadoop fs -copyFromLocal <localsrc>
URI
```

Copy a file from your client to HDFS.

Similar to `put` command, except that the source is restricted to a local file reference.

HDFS Shell Commands.

```
> hadoop fs -cat <path>
```

Copies source paths to stdout.

HDFS Shell Commands.

```
> hadoop fs -copyToLocal URI
<localdst>
```

Copy a file from HDFS to your client.

Similar to `get` command, except that the destination is restricted to a local file reference.

HDFS Shell Commands.

```
cat
chgrp
chmod
chown
copyFromLocal
copyToLocal
cp
du
dus
expunge
get
getmerge
ls
lsr
mkdir
movefromLocal
mv
put
rm
rmr
setrep
stat
tail
test
text
touchz
```