# Name Node Meta Data

There are two types of Meta data

**File metadata**

File1.txt -> owner, permission, size last modified time, creation time, replication, [block1, block2, block3]

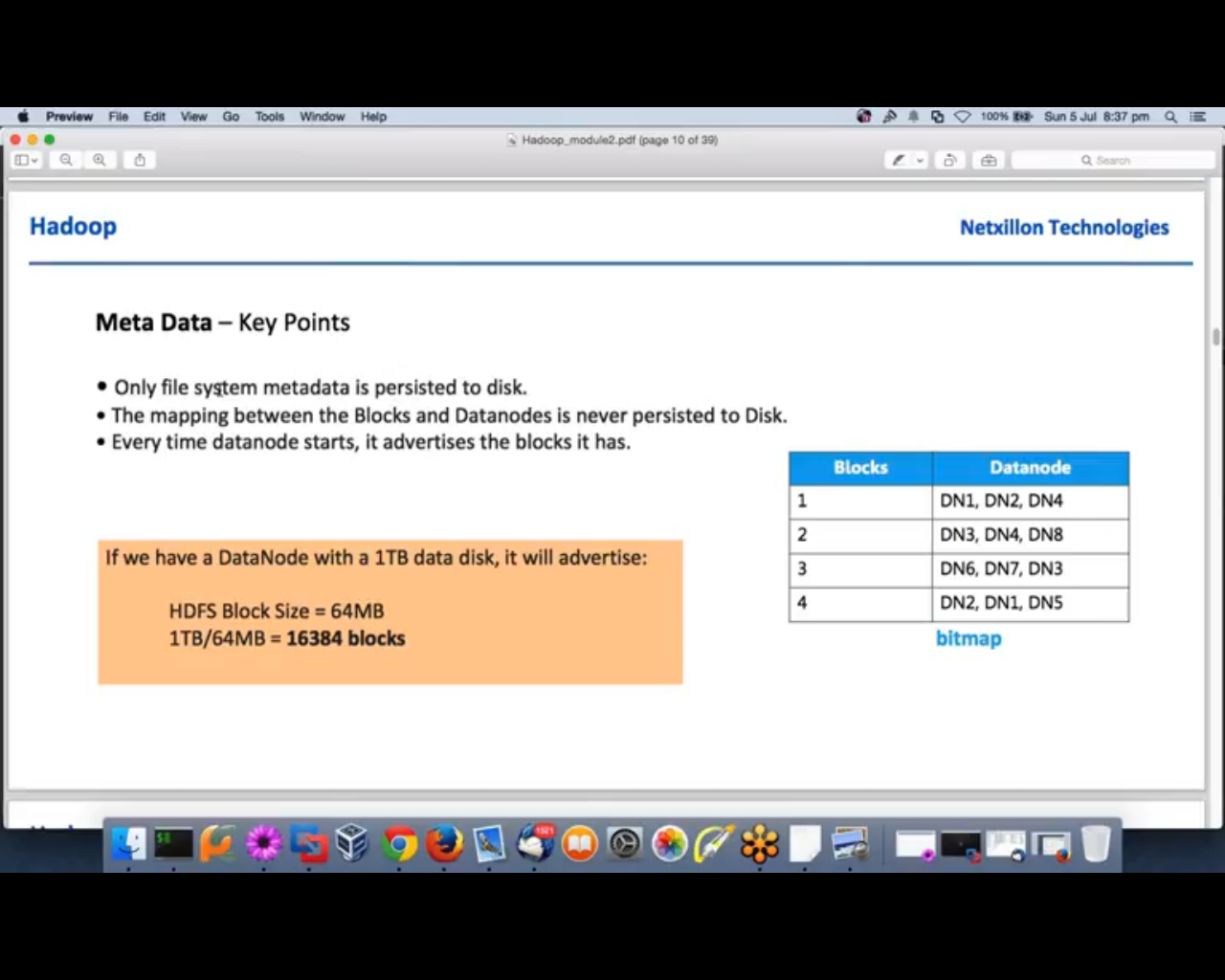
This is called File to block mapping

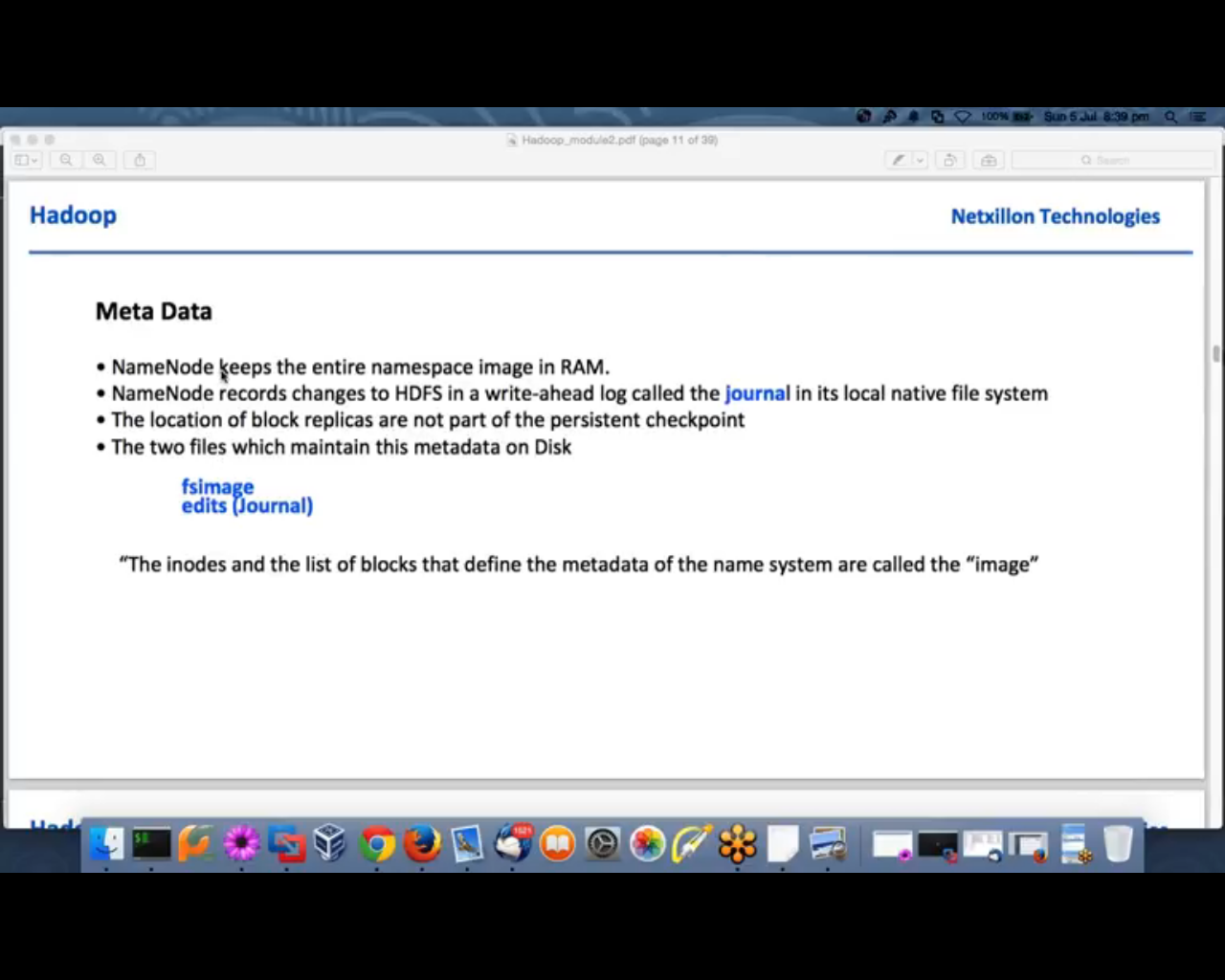
**Bitmap metadata**

Bitmap meteadata contains block to data node mapping

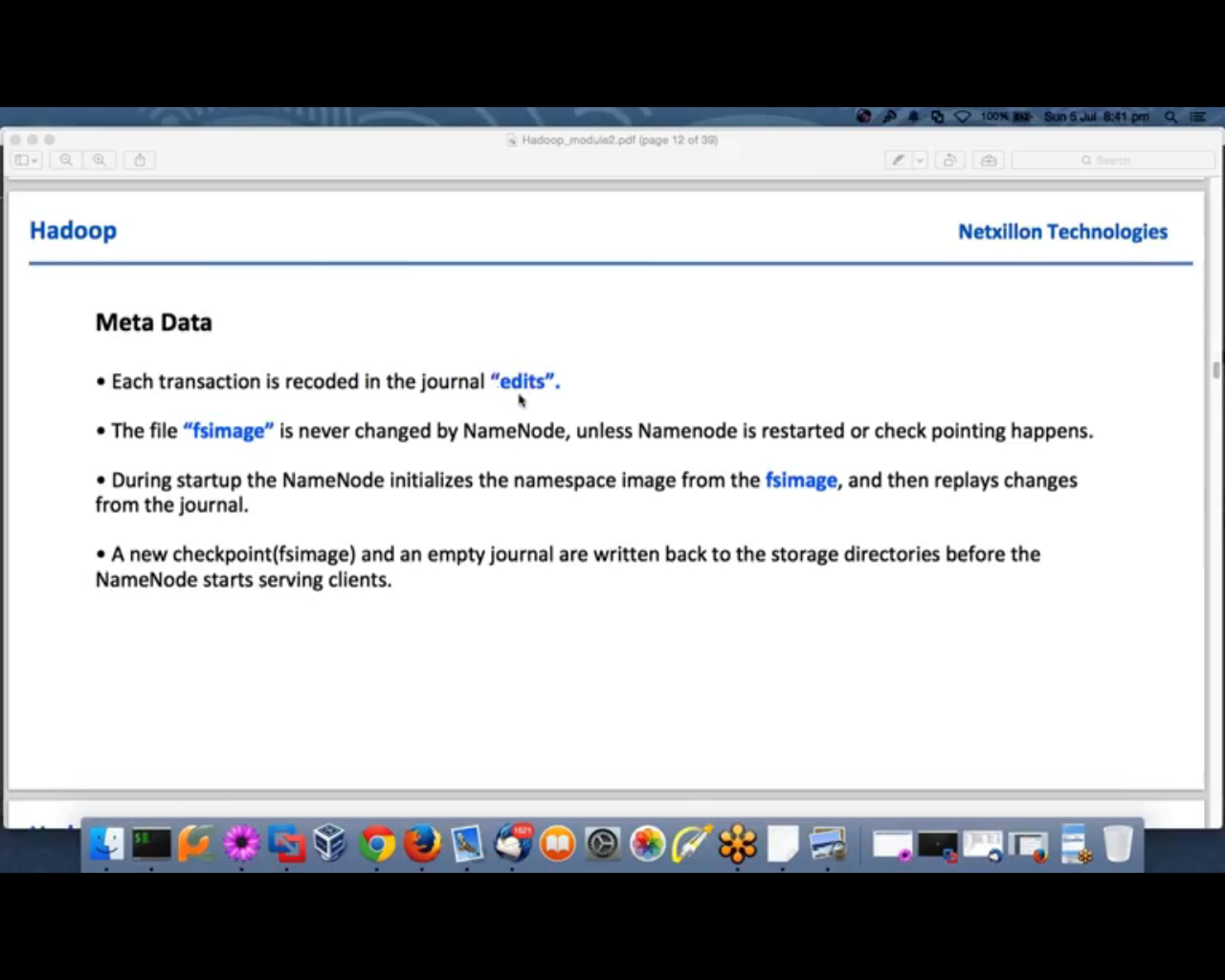
|  |
| --- |
| Block1 - > datanode2, datanode4, datanode5  Block2 - > datanode1, datanode3, datanode4  Block3 - > datanode2, datanode3, datanode4 |

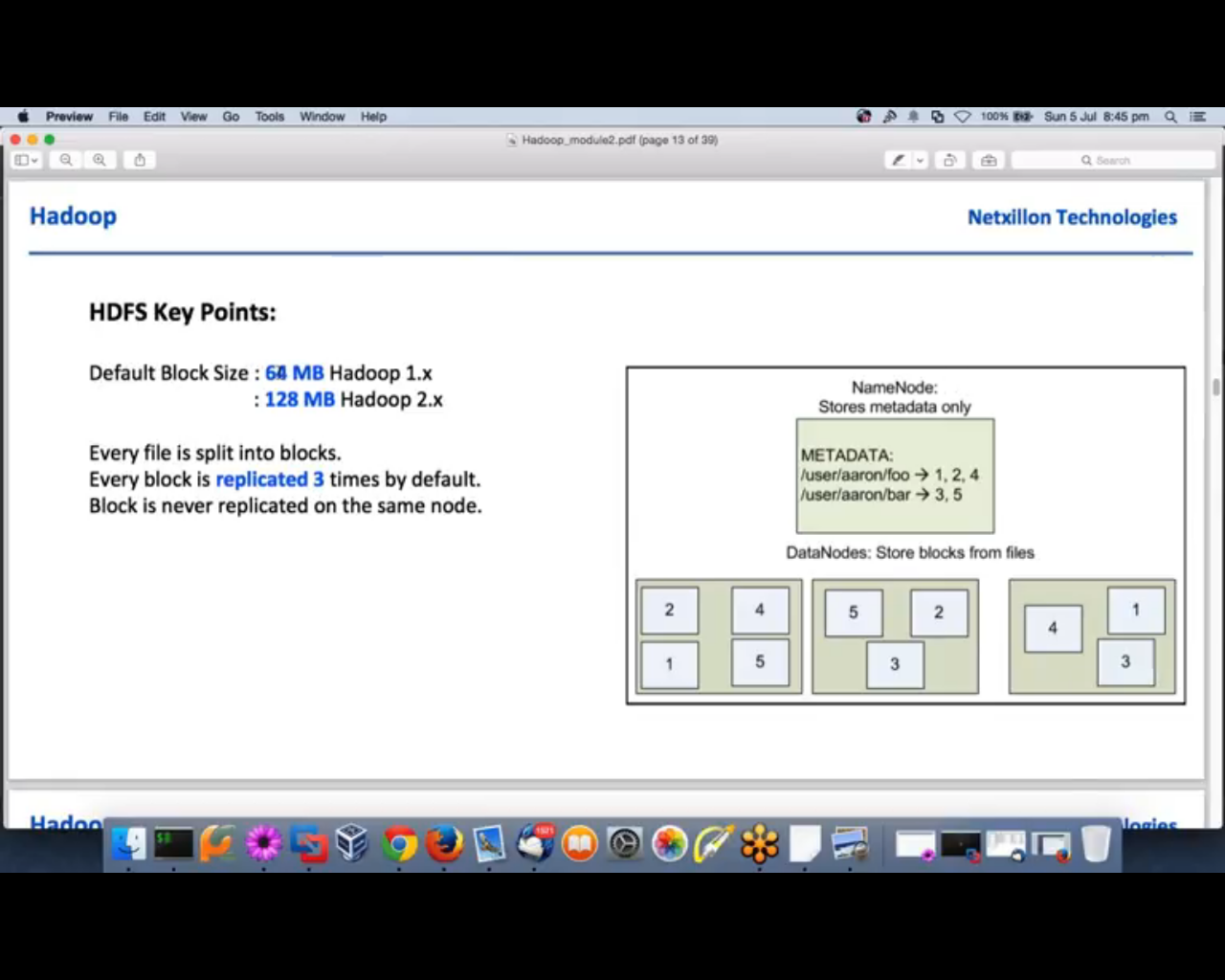
Bitmap metadata never store on the disk (fsimage) it stores on the namenode memory only

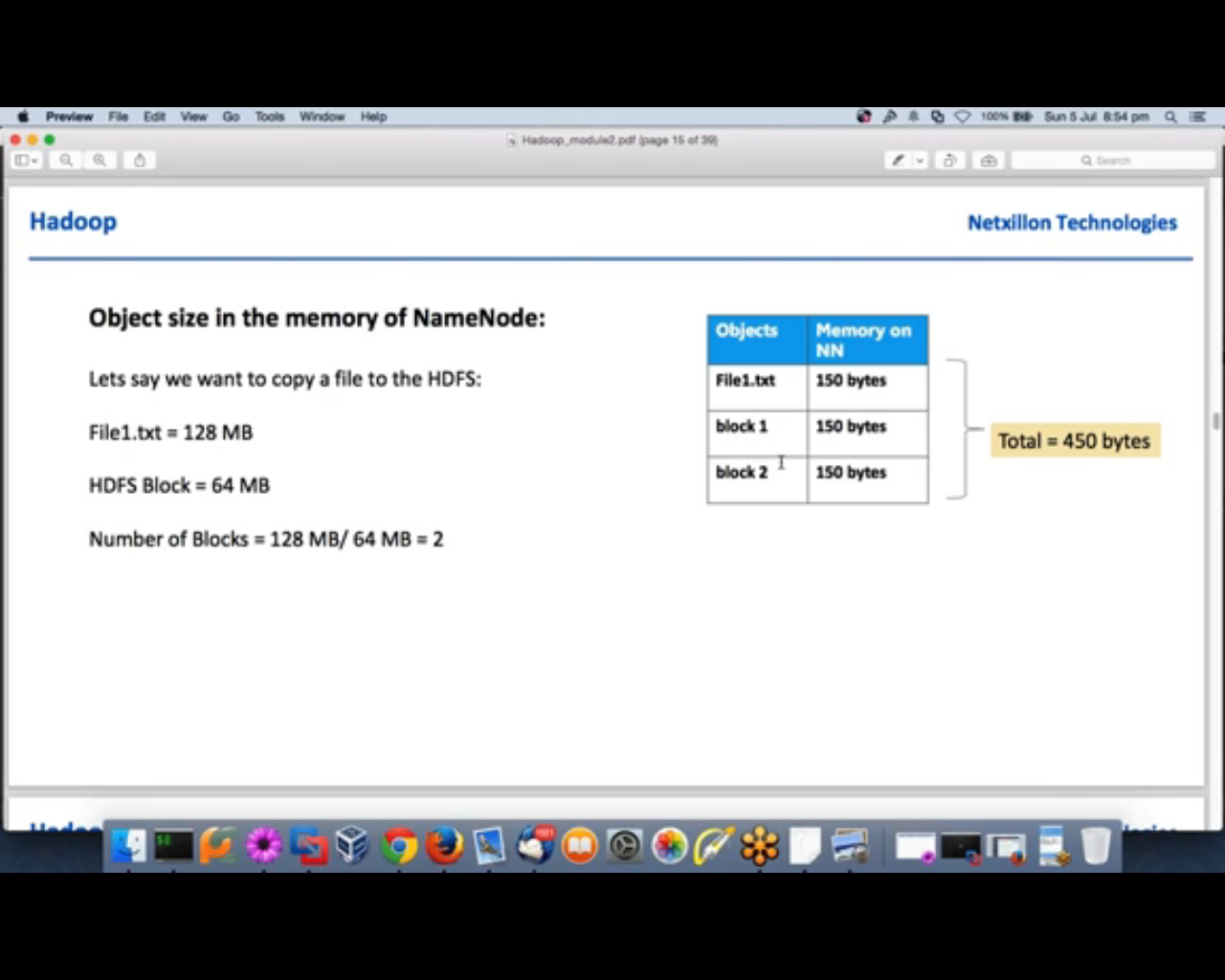




Namespace is simply a file structure







In Hadoop everything is an object as it is written in java

Every file, directory, blocks is an object is object occupy 150 bytes to represent in metadata

This number is only for one replica, so we will have +600 bytes if we maintaining total 3 replicas

# Checkpoint, Edit logs and FsImage

When we format the name node first time it creates fsimage file fsimage\_0000000000000000000 that is the initial file. Here **0000000000000000000** is transaction id

For the sake of simplicity we consider it as fsimage\_0

Now let’s take one example that we are putting sample.txt file from local file system to hdfs using below command

#hdfs dfs-put /root/sample.txt /user/input/sample.txt

This command will first appended in edit logs file and it will have one transaction id associate with it that is an incremental number

Let’s say current transaction id is 1, so our current edit log file is edit\_inprogress\_1 is having information about that all the transaction starting from transaction id 1

This file will store all the changes that client will do starting from transaction id 1 and the same will updated in main memory of name node

# Checkpoint Process

Check pointing is triggered by one of two conditions: if enough time has elapsed since the last checkpoint (dfs.namenode.checkpoint.period), or if enough new edit log transactions have accumulated (dfs.namenode.checkpoint.txns). The checkpointing node (secondary name node/Standby name node) periodically checks if either of these conditions is met and if so, kicks off the checkpointing process.

Default value is 1 hour and 1 million transactions

During checkpoint process, the namesystem also needs to restrict concurrent access from other users, and client cannot write anything on datanode during this period, so this process is done by checkpoint node (secondary name node/stand by name node) so that user can do the write operation all the time

During checkpoint process , first checkpoint node get the latest files from namenode if it doesn’t have the latest file ( usually checkpoint node has latest files ) then Checkpoint node merge latest edit logs into new fsimage file (under checkpoint directory) and later transferred it to active namenode

Now let’s suppose within 1 hours we have 20 transaction written in edit log file so the check point process create a new file namely **edit\_inprogress\_21** and create **edit\_1-20** file that will have all the transactions from id 1 to 20 and it will merge all the change in fsimage file till transaction 20 and store it in file **fsimage\_20** and move this file to namenode direcotry of acitve namenode so now active namenode directory contains following files

* edit\_1-20
* edit\_inprogress\_21
* fsimage\_0
* fsimage\_20

Now let’s say when new checkpoint process triggers again in next 1 hour there are total 40 transactions happened (another 20 transcations)

So now we will have **edit\_inprogress\_41**, **fsimage\_40** and **edit\_21-40**

So at this time we will have following files under namenode direcotry

* edit\_1-20
* edit\_21-40
* edit\_inprogress\_41
* fsimage\_20
* fsimage\_40

Now let’s suppose during next half an hour another 10 transactions executed so now the current transaction id is 50 and this is updated in edit logs under edit\_inprogress\_41 file and also updated in main memory

if now namenode crashes and we manually start the namenode , first it creates new fsimage file (fsimage\_50 ) by merging contents of edit\_inprogress\_41 into fsimage\_40 and load this fsimage into main memory and delete old fsimage\_20

So now we will have following files under namenode

* edit\_1-20
* edit\_21-40
* edit\_41-50
* edit\_inprogress\_51
* fsimage\_40
* fsimage\_50

At most we have just two fsimage file only, old fsimage files are being deleted

We can manually trigger checkpoint process and create new fsimange by running following commands (the same thing happen when we restart name node)

Manual steps to checkpoint are as below:

1. On primary NameNode, we will try to save latest metadata to the fsimage as:

|  |
| --- |
| 1. **Enter in safe mode**  hdfs dfsadmin -safemode enter  2. **Save image**  hdfs dfsadmin -saveNamespace  3. **Leave safe mode**  hdfs dfsadmin -safemode leave |

When safemode is on, namenode would not be available for writing the data but you can read the data

|  |
| --- |
| [root@mac127 ~]# hdfs dfs -put fsimage.xml /user/test/input  put: Cannot create file/user/test/input/fsimage.xml.\_COPYING\_. Name node is in safe mode.  [root@mac127 ~]# hdfs dfs -cat /user/test/input/sample.txt  Paragraphs and Topic Sentences  A paragraph is a series of sentences that are organized and coherent, and are all related to a single topic. Almost every piece of writing you do that is longer than a few sentences should be organized into paragraphs. This is because paragraphs show a reader where the subdivisions of an essay begin and end, and thus help the reader see the organization of the essay and grasp its main points.  [root@mac127 ~]# |

2. On Secondary NameNode, we will now run manual checkpointing: First, stop Secondary NameNode service.

Now execute following command on the Secondary NameNode:

|  |
| --- |
| sudo -u hdfs hadoop secondarynamenode -checkpoint force |

Note: fsimage do not contain block location it just contains other information like block id, file name permission, owner last modified time, creation time etc. size of fsimage is very low as it does not contain block location

So once namenode starts from failure all data node sends block reports to namenode and it update it in its main memory

**Refer** <http://hadooptutorial.info/checkpoint-node-in-hadoop/>

Currently we have following files under namenode **/dfs/nn/current** directory

|  |
| --- |
| **-rw-r--r-- 1 hdfs hdfs 3604 Jan 19 20:07 edits\_0000000000000273028-0000000000000273053**  **-rw-r--r-- 1 hdfs hdfs 3603 Jan 19 20:08 edits\_0000000000000273054-0000000000000273079**  **-rw-r--r-- 1 hdfs hdfs 3028 Jan 19 20:09 edits\_0000000000000273080-0000000000000273098**  **-rw-r--r-- 1 hdfs hdfs 1048576 Jan 19 20:09 edits\_inprogress\_0000000000000273099**  **-rw-r--r-- 1 hdfs hdfs 204247 Jan 13 20:41 fsimage\_0000000000000026207**  **-rw-r--r-- 1 hdfs hdfs 62 Jan 13 20:41 fsimage\_0000000000000026207.md5**  **-rw-r--r-- 1 hdfs hdfs 273435 Jan 19 20:09 fsimage\_0000000000000273098**  **-rw-r--r-- 1 hdfs hdfs 62 Jan 19 20:09 fsimage\_0000000000000273098.md5** |

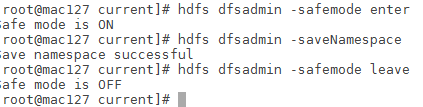
So the current fsimage is **fsimage\_0000000000000273098**

And latest editlog is **edits\_inprogress\_0000000000000273099**

So if we manually start checkpoint process now it will create new editlog file starting from 273099, there could be multiple file generated as file size is limited for each edit log

* **edits\_0000000000000273099-<transaction id K>**
* **edits\_ <transaction id K+1> -<transaction id M>**
* **edits\_ <transaction id M+1> -<transaction id N>**
* **edits\_inprogress\_< transaction id N +1>**
* **fsimage\_<transaction id N>**

Now run the checkpoint process



Again check **/dfs/nn/current** directory

|  |
| --- |
| **-rw-r--r-- 1 hdfs hdfs 2653 Jan 19 20:09 edits\_0000000000000273099-0000000000000273118**  **-rw-r--r-- 1 hdfs hdfs 3603 Jan 19 20:10 edits\_0000000000000273119-0000000000000273144**  **-rw-r--r-- 1 hdfs hdfs 994 Jan 19 20:11 edits\_0000000000000273145-0000000000000273152**  **-rw-r--r-- 1 hdfs hdfs 1048576 Jan 19 20:11 edits\_inprogress\_0000000000000273153**  **-rw-r--r-- 1 hdfs hdfs 273435 Jan 19 20:09 fsimage\_0000000000000273098**  **-rw-r--r-- 1 hdfs hdfs 62 Jan 19 20:09 fsimage\_0000000000000273098.md5**  **-rw-r--r-- 1 hdfs hdfs 273435 Jan 19 20:11 fsimage\_0000000000000273152**  **-rw-r--r-- 1 hdfs hdfs 62 Jan 19 20:11 fsimage\_0000000000000273152.md5** |

Now you can see here three new edits file created and one new edit inprogress and fsimage created also old fsimage file is deleted

# Offline Image Viewer (OIV)

fsimage files are saved in compressed binary format you can view these file from using **Offline Image Viewer (OIV)** tool that comes with hadoop 2.x version

|  |
| --- |
| hdfs oiv -i fsimage\_0000000000000431705 -o /root/fsimage.xml -p XML |

This command will create fsimage.xml file from input file fsimage\_0000000000000431705



Sample information for file sample.txt which is located under hdfs location /user/test/sample.txt

|  |
| --- |
| ….  ….  ….  [<inode>](file:///C:\Users\shalajs\Desktop\1.xml)  <id>22224</id>  <type>DIRECTORY</type>  <name>test</name>  <mtime>1484584375457</mtime>  <permission>root:supergroup:0755</permission>  <nsquota>-1</nsquota>  <dsquota>1</dsquota>  </inode>  [<inode>](file:///C:\Users\shalajs\Desktop\1.xml)  <id>22225</id>  <type>DIRECTORY</type>  <name>input</name>  <mtime>1484320932719</mtime>  <permission>root:supergroup:0755</permission>  <nsquota>-1</nsquota>  <dsquota>-1</dsquota>  </inode>  [<inode>](file:///C:\Users\shalajs\Desktop\1.xml)  <id>22274</id> -- unique id incremental number  <type>FILE</type>  <name>sample.txt</name>  <replication>3</replication> --replication factor  <mtime>1484320932695</mtime> -- modification time  <atime>1484320932463</atime> -- creation time  <preferredBlockSize>134217728</preferredBlockSize> -- block size 128 MB  <permission>root:supergroup:0644</permission> -- permission  [<blocks>](file:///C:\Users\shalajs\Desktop\1.xml)  [<block>](file:///C:\Users\shalajs\Desktop\1.xml) --this file has just one block if there are multiple blocks this element repeated  <id>1073744150</id>  <genstamp>3326</genstamp>  <numBytes>8690</numBytes> -- number of bytes  </block>  </blocks>  <storagePolicyId>0</storagePolicyId>  </inode> |

Here we can see fsimage just contains block id not the block location (which datanode contains that block), block reports send by datanode to active name node and active name node store it in memory, you can find this block under datanode directory

|  |
| --- |
| [root@mac127 ~]# cd /dfs/dn  [root@mac127 dn]# find . -name "\*1073744150\*"  ./current/BP-1877832991-172.27.155.127-1484221390087/current/finalized/subdir0/subdir9/blk\_1073744150\_3326.meta  ./current/BP-1877832991-172.27.155.127-1484221390087/current/finalized/subdir0/subdir9/blk\_1073744150  [root@mac127 dn]# |

# Offline Edits Viewer:

edit logs are saved in compressed binary format you can view these file from using **Offline Edit Viewer (OEV)** tool that comes with hadoop 2.x version

|  |
| --- |
| hdfs oev -i edits\_inprogress\_0000000000000433598 -o /root/edits.xml -p XML |

This command will create edits.xml file from input file edits\_inprogress\_0000000000000433598



Refer

<https://acadgild.com/blog/view-fsimage-edit-logs-files-hadoop/>

You can refer following videos

<https://www.youtube.com/watch?v=CsYdlAwwGtE>

<https://www.youtube.com/watch?v=HmgxVRjoPYU>

<https://www.youtube.com/watch?v=xazVhZ3wIN8>

Following command give the information about blocks

hdfs fsck / -files –blocks

