IntelliPaat

Hadoop Administration

Agenda

In this session we will cover :

- Configuration overview and important configuration files
  - Configuration parameters and values
  - HDFS parameters and Yarn parameters
  - Hadoop environment setup - ‘Include’ and ‘Exclude’ nodes
  - MapReduce Performance Tuning
- Administration and Maintenance
  - Namenode/Datanode directory structures and files
  - File system image and Edit log

Cluster Configuration

Configuration of Hadoop

- Following files occupy prominent place in hadoop Configuration located at /etc/hadoop/conf

  hadoop-env.sh
  core-site.xml
  mapred-site.xml
  hdfs-site.xml

hadoop-env.sh

This file is used to set Hadoop Specific environment Variables.

- HADOOP_MAPRED_HOME - This is the location where all the jars (Libraries) related to MapReduce framework can be located.
- HADOOP_HEAPSIZE - Maximum amount of Heap to use in MB. Default value is 1000.If your server is crashing with OutOfMemory issues then you have to set HADOOP_HEAPSIZE to a lower value
- HADOOP_NAMENODE_INIT_HEAPSIZE - For Namenode
- HADOOP_OPTS - JAVA Runtime options
- HADOOP_CLIENT_OPTS -Java Runtime Options

hdfs-site.xml

This file contains HDFS daemons configuration settings like NameNode, DataNode and Secondarynamenode.

- dfs.replication - Default block replication. The actual number of replications can be specified when the file is created. The default is used if replication is not specified in create time.

Change this in xml and restart only namenode. Now copy any file to hadoop and see replication factor using hadoop fs -ls ‘path’

hdfs-site.xml

Safe Mode - it's a maintenance state of NameNode. No writes are allowed - read only.

- In Safe mode, blocks are not replicated
- As hdfs user, admin can put namenode into safe mode and can leave from safe mode

hdfs dfsadmin -safemode get
hdfs dfsadmin -safemode enter
hdfs dfsadmin -safemode leave

*Cannot do write operations when namenode is in Safe Mode

hdfs-site.xml

- dfs.namenode.safemode.extension - Determines extension of safe mode in milliseconds after the threshold level is reached.

- dfs.namenode.safemode.min.datanodes - Specifies the number of datanodes that must be considered alive before the namenode exits safemode. Values less than or equal to 0 mean not to take the number of live datanodes into account when deciding whether to remain in safe mode during startup. Values greater than the number of datanodes in the cluster will make safe mode permanent.

hdfs-site.xml

- hadoop.tmp.dir - Directory in which Hadoop can write files locally
- dfs.namenode.name.dir - Determines where on the local filesystem the DFS namenode should store the name table(fsimage). If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy.
- dfs.namenode.checkpoint.dir - Determines where on the local filesystem the DFS secondary namenode should store the temporary images to merge. If this is a comma-delimited list of directories then the image is replicated in all of the directories for redundancy.

hdfs-site.xml

- dfs.datanode.data.dir - Determines where on the local filesystem an DFS data node should store its blocks. If this is a comma-delimited list of directories, then data will be stored in all named directories, typically on different devices. Directories that do not exist are ignored.

core-site.xml

This file points Hadoop daemon about where NameNode is running on the cluster.

- fs.defaultFS - Name Node URL

mapred-site.xml

This file is used to set HDFS Specific configurations.

- mapred.job.tracker - The host and port that the MapReduce job tracker runs at. If "local", then jobs are run in-process as a single map and reduce task.
- mapreduce.framework.name - Specifies whether it has to use MapReduce v1 or Yarn
- mapreduce.jobhistory.address - URL for job history server
- mapreduce.jobhistory.webapp.address - URL which is exposed for accessing through HTTP

More Details

http://hadoop.apache.org/docs/r1.1.2/core-default.html
http://hadoop.apache.org/docs/r1.1.2/mapred-default.html
http://hadoop.apache.org/docs/r1.1.2/hdfs-default.html

Include and exclude nodes

This file is used to set HDFS Specific configurations.

- dfs.hosts.exclude in hdfs-site.xml - Names a file that contains a list of hosts that are not permitted to connect to the namenode. The full pathname of the file must be specified. If the value is empty, no hosts are excluded.
- mapred.hosts.exclude in mapred-site.xml - Names a file that contains the list of hosts that should be excluded by the jobtracker. If the value is empty, no hosts are excluded.

Perfomance Tuning for MapReduce

- Adjust memory setting for a task using mapred.child.java.opts in mapred-site.xml file
  mapred.child.java.opts -Xms1024M -Xmx2048M
- Mount partition given for HDFS using no atime option defaults,noatime,nodiratime
- ‘mapreduce.local.dir’ in mapred-site.xml and ‘dfs.data.dir’ in hdfs-site.xml are set to one directory on each of the disks. This will utilize overall IO capacity.

Perfomance Tuning for MapReduce

- Use Compression for Mapper output
- In mapred-site.xml mapred.compress.map.output and mapred.map.output.compression.codec as true
- Or in Driver code of MapReduce program

```
conf.set("mapred.compress.map.output", "true")
conf.set("mapred.map.output.compression.codec",
"org.apache.hadoop.io.compress.LzoCodec");
```

Perfomance Tuning for MapReduce

- Reduce data on network using Combiner

- Set ‘mapreduce.map.tasks.speculative.execution’ and ‘mapreduce.reduce.tasks.speculative.execution’ to true
- Speculative execution will reduce job execution time if the task progress is slowed down due to memory unavailability by backing up slow tasks to other nodes - ‘mapreduce.map.speculative’ and ‘mapreduce.reduce.speculative’

- Implement hash function in Partitioner class such that keys are really distributed. This will reduce unbalanced reducer tasks

Perfomance Tuning for MapReduce

- We can specify number of reducer tasks but number of mapper tasks is set implicitly.

- To allow mappers to run in parallel hadoop split file into smaller chunks. But new mapper initialization also takes few seconds. To reduce this overhead you can use reuse jvm task **mapred.job.reuse.jvm.num.tasks.** -1 means reuse unlimited number of times

- If the average mapper running time is less than a minute, you can increase the **mapred.min.split.size**, to reduce mappers per slot. This will reduce mapper initializing overhead.

Agenda

In this session we will cover :

- Administration and Maintenance
  - Namenode/Datanode directory structures and files
  - File system image and Edit log
  - Namenode failure and recovery procedure
  - Metadata and Data backup
  - Potential problems and solutions / what to look for
  - Adding and removing nodes
  - Lab: MapReduce File system Recovery

Agenda

- Monitoring and Troubleshooting
  - Best practices of monitoring a cluster
  - Using logs and stack traces for monitoring and troubleshooting
  - Using open-source tools to monitor the cluster

- Job Scheduler: Map reduce job submission flow
  - How to schedule Jobs on the cluster
  - FIFO Schedule, Fair Scheduler and its configuration

Administration and Maintenance

File system image and Edit log

- fsimage – fsimage file contains the complete state of the file system at a point in time. Every file system modification is assigned a unique, monotonically increasing transaction ID. An fsimage file represents the file system state after all modifications up to a specific transaction ID.

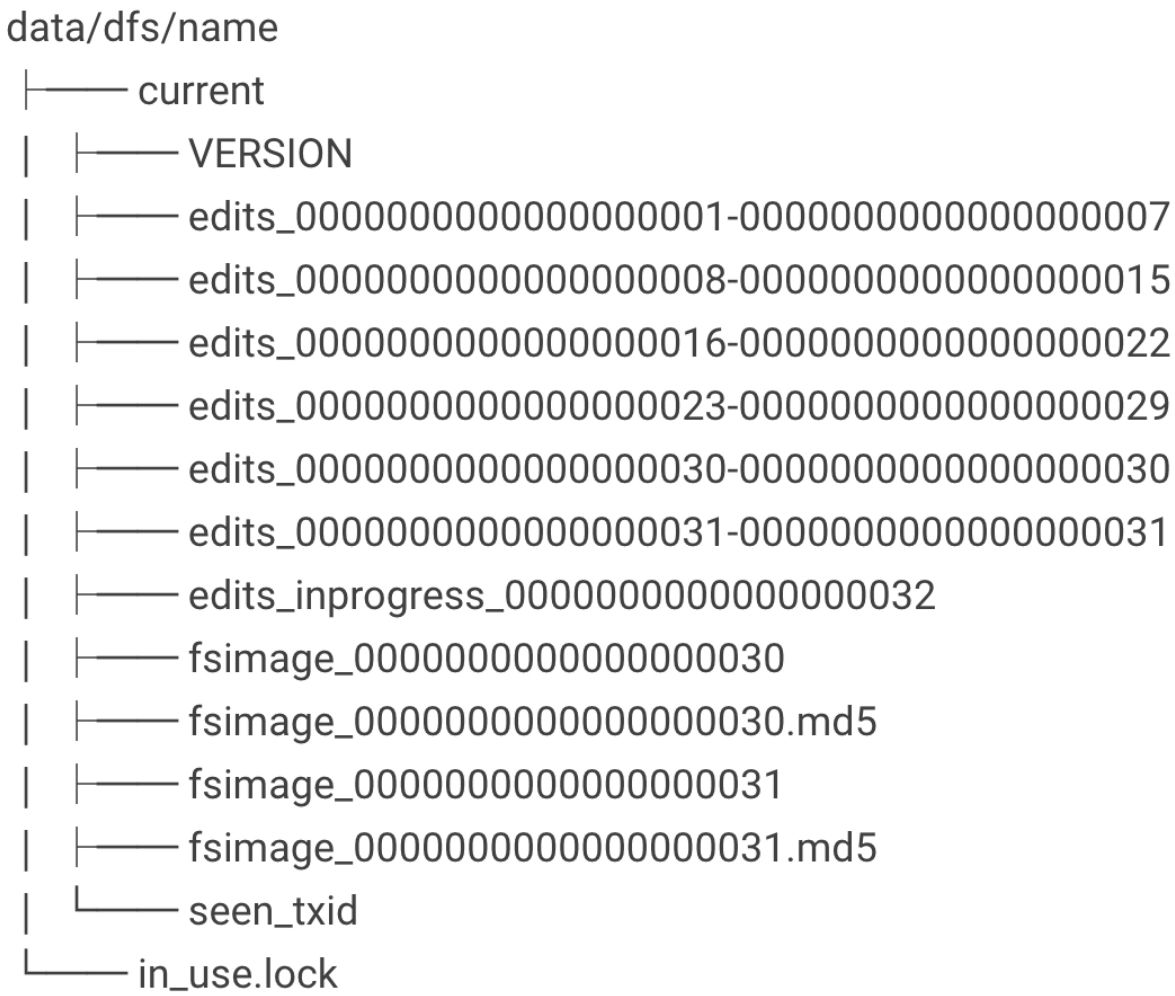
- edits – edits file is a log that lists each file system change (file creation, deletion or modification) that was made after the most recent fsimage.

- Checkpointing is the process of merging the content of the most recent fsimage with all edits applied after that fsimage is merged in order to create a new fsimage. Checkpointing is triggered automatically by configuration policies or manually by HDFS administration commands.

Namenode directories and files

```
[training@hadoop intellipaat]$ ls /var/lib/hadoop-hdfs/cache/hdfs/dfs/name/
current  in_use.lock
[training@hadoop intellipaat]$ ls /var/lib/hadoop-hdfs/cache/hdfs/dfs/name/current/VE
RSION
/var/lib/hadoop-hdfs/cache/hdfs/dfs/name/current/VERSION
[training@hadoop intellipaat]$ ls /var/lib/hadoop-hdfs/cache/hdfs/dfs/name/current/ed
its_^C
[training@hadoop intellipaat]$ ls /var/lib/hadoop-hdfs/cache/hdfs/dfs/name/current/fs
image_0000000000000000137^C
[training@hadoop intellipaat]$ ls /var/lib/hadoop-hdfs/cache/hdfs/dfs/name/current/se
en_txid
/var/lib/hadoop-hdfs/cache/hdfs/dfs/name/current/seen_txid
```

Namenode directories and files

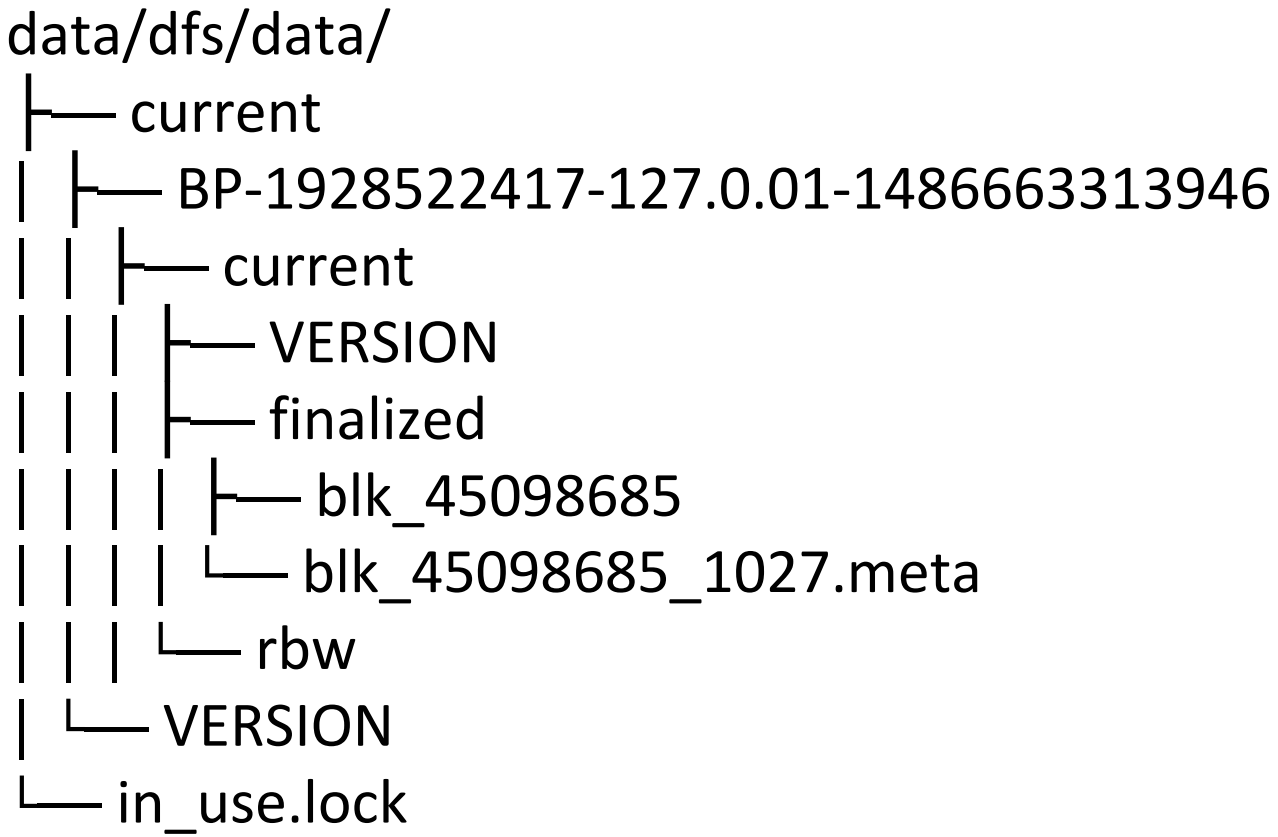


Namenode directories and files

- VERSION File

```
#Thu Feb 09 13:01:53 EST 2017
namespaceID=1453867033
clusterID=CID-640b2bb2-f746-4cc9-a7de-ee4d74786331
cTime=0
storageType=NAME_NODE
blockpoolID=BP-1928522417-127.0.0.1-1486663313946
layoutVersion=-40
```

Datanode directories and files



Datanode directories and files

- VERSION

```
#Fri Feb 10 15:04:39 EST 2017
storageID=DS-1522345539-127.0.0.1-50010-1486663688108
clusterID=CID-640b2bb2-f746-4cc9-a7de-ee4d74786331
cTime=0
storageType=DATA_NODE
layoutVersion=-40
```

File system image and Edit log

- fsimage – fsimage file contains the complete state of the file system at a point in time. Every file system modification is assigned a unique, monotonically increasing transaction ID. An fsimage file represents the file system state after all modifications up to a specific transaction ID.

- edits – edits file is a log that lists each file system change (file creation, deletion or modification) that was made after the most recent fsimage.

- Checkpointing is the process of merging the content of the most recent fsimage with all edits applied after that fsimage is merged in order to create a new fsimage. Checkpointing is triggered automatically by configuration policies or manually by HDFS administration commands.

Thank You