Big Data Analytics

Isara Anantavrasilp

Lecture 8: HDFS Storage

HDFS Server Processes

- HDFS is implemented with two main server processes:
  - **NameNode**: A server process that holds all metadata of HDFS filesystem
  - **DataNodes**: Processes that manages the actual data blocks, distributed on many servers.
- They are configured in a master-slave setup
- All files in HDFS are split into several DataNodes.
- The NameNode tracks how to reconstruct the blocks into files.

Filesystem Information Files

- The information of how to reconstruct each file in the filesystem from blocks is stored in ***one*** file: **`fsimage`**
  - Without it, HDFS is useless
  - The file is stored in NameNode
- When the files within HDFS are changed, NameNode does not update `fsimage` immediately
- The changes are kept in another file: **`edits`**
- `edits` tracks all changes of HDFS since last `fsimage` save

NameNode Startup

- NameNode reads and stores `fsimage` in it's memory when it is started
- Next it reads the `edits` file
- Then, it applies all changes stored in `edits` file onto `fsimage`
- Finally, it is ready to receives new client's commands
- This is why NameNode requires particularly large memory

DataNode Startup

- When the DataNode is started, it catalogs the blocks that it holds
  - The blocks are stored as normal files on the node
- Then, the DataNode performs consistency checking on the blocks
- The DataNode sends the list of blocks to the NameNode
  - This is how NameNode learns which DataNodes hold which blocks
- DataNode is now registered with NameNode
- DataNode will keep sending heartbeats to NameNode

Secondary NameNode

- To speed up NameNode startup process, Hadoop also implements **Secondary NameNode**
- It is responsible for periodically reading the latest version of the `fsimage` and `edits` file and creating a new up-to-date `fsimage` with the outstanding edits applied.
- Essentially, it keeps updating a copy of `fsimage` in background while the actual `fsimage` is still unchanged

What If NameNode Fail?

- NameNode and Secondary NameNode are introduced in Hadoop 1
- This scheme leads to one large flaw: **NameNode is the single point of failure**
- You might have several copies of blocks, but if the `fsimage` is corrupted, the entire HDFS will not be usable anymore
- In Hadoop 1, `fsimage` has to be backed-up separatedly

Backup NameNode

- In Hadoop 2, Backup NameNode is introduced
- Backup NameNode keeps a local up-to-date copy of the filesystem metadata
- If original NameNode is down, the admin can switch to Backup NameNode manually
- Such manual process might still take too much time and effort

NameNode HA

- In current production clusters, NameNode High Availability (NameNode HA) is normally used
- It is also introduced in Hadoop 2
- In HA setting, two NameNodes are running simultaneously
  - One is master, one is backup
  - Both have up-to-date info of the filesystem
  - If the master is down, the backup can takeover immediately

Failover Process

- Switching NameNode from original to the back-up one is not trivial process
  - Aka, **failover**: To switch to backup one when the original one fails
- We have to make sure that
  - The two NameNodes have consistent information
  - The clients connects only to only one node at a time (and should be the new one)
- If two NameNodes are accessed at the same time, they could be out-of-sync
- **Apache ZooKeeper** service is often used to enable automatic NameNode failover

HDFS Snapshots

- Although HDFS provides redundancy, but that does not mean your data will be safe
  - You need to keep backups
- HDFS provides a mechanism to do so: Snapshots
- Snapshots keeps a copy the metadata of the filesystem at a given point in time
  - Stored snapshots can be viewed in the future
  - Blocks associated to the snapshots will be kept, but cannot be accessed

HDFS Snapshots (2)

- Snapshot example: Consider two files
  `/Text/Shakespeare.txt` (3 blocks)
  `/Text/Big.txt` (10 blocks)
- Total size 13 blocks
- If you take a snapshot of the directory `/Text` and you erase `Big.txt`
  - You will see only `Shakespeare.txt` with 3 blocks on HDFS
  - Behind the scene, the filesystem still keeps entire 13 blocks
  - The hidden blocks will be released only when the snapshot file is deleted

Allowing Snapshots

- You can create snapshots for every directory in the filesystem, or only specific directories
- Before, creating snapshots, we have to set the path to be snapshottable first:

```
sudo -u hdfs hdfs dfsadmin
-allowSnapshot /user/cloudera/Text
```

- The command specifies that the directory `/user/cloudera/Text` is allowed to take snapshots

Allowing Snapshots (2)

- Setting snapshots directories require root privilege
  - This is why you need `sudo`
- The root username of HDFS is `hdfs`
  Thus, you have to `sudo` as `hdfs`

  ```
  sudo -u hdfs
  ```
- If the command is correct, you should see :
  `Allowing snapshot on Text succeeded`
- Note that snapshots are not yet created!

Creating Snapshots

- Creating a snapshot:

```
sudo -u hdfs hdfs dfs -createSnapshot
/user/cloudera/Text snapshot1

Created snapshot /user/cloudera/Text
/.snapshot/snapshot1
```

- Snapshot files are stored in `../.snapshot/` under the snapshotted directory

Listing Files in Snapshots

- You can list the files in snapshots

```
sudo -u hdfs hdfs dfs -ls
/user/cloudera/Text/.snapshot/snapsh
ot1
```

- Result

```
Found 4 items
-rw-r--r--   1 cloudera cloudera
6488666 2018-10-16 15:10 /user/cloudera/
Text /.snapshot/snapshot1/big.txt
...
```

Let's try to delete a file

- Deleting with
  `hadoop fs -rm Text/hello2.txt`
- Now, try to list the files

- Examine the snapshots
  `sudo -u hdfs hdfs dfs -ls /user/cloudera/Text/.snapshot/snapshot1`
- Note that the file is still there

Let's view the file

- We can view a text file with `cat` command
  ```
  hadoop fs -cat Text/hello2.txt
  cat: `WordCount/hello2.txt': No such file or directory
  ```
- The file is not in the HDFS anymore

- But it is still in the snapshot, so we can do:
  ```
  hadoop fs -cat Text/.snapshot/snapshot1/hello2.txt
  ```

Erasing Snapshots

- Snapshots practically copies your files for you
  - Can be read or copied anytime
  - Each directory can hold 65,535 snapshots
- Thus, we will have to erase them to free up some space

```
sudo -u hdfs hdfs dfs -
deleteSnapshot /user/cloudera/Text
snapshot1
```

Data Serialization

- **Serialization**: Conversion of object into stream of bytes such that the objects can be stored or streamed through a communication link.
- **Deserialization**: Conversion of byte streams back to object
- Since HDFS is distributed filesystem, Hadoop has special mechanisms to serialize and deserialize data across the network

Writable Interface

- Main package that handles serialization in Hadoop is `org.apache.hadoop.io`
- It contains Writable interface to handle serialization of different kinds of objects

```
public interface Writable   {
    void write(DataOutput out)
                        throws IOException ;
    void readFields(DataInput in)
                        throws IOException ;
}
```

Basic Writable Classes

- Hadoop provides some wrappers classes in `org.apache.hadoop.io`
  - `BooleanWritable`
  - `ByteWritable`
  - `DoubleWritable`
  - `FloatWritable`
  - `IntWritable`
  - `LongWritable`
  - `Text`: For serializing `java.lang.String`

Further Writable Classes

- Collection-based wrapper classes also available
  - `ArrayWritable`
  - `TwoDArrayWritable`
- Variable-length types
  - `VIntWritable`: Variable-length integer
  - `VLongWritable`: Variable-length long