HDFS erasure codes in Hadoop-3.0.0

HDFS EC uses *striping*, in which unit of striping distribution is termed a striping cell (or cell). From erasure code terminology, we know that during encoding, for each stripe of original data cells, a certain number of parity cells are calculated and stored. The error on any striping cell can be recovered through decoding calculation based on surviving data and parity cells.

1. Striped files are formed of block groups, each of which contains a certain number of internal blocks.
2. NameNode uses a hierarchical block naming protocol, which enables the ID of a block group to be inferred from the ID of any of its internal blocks.
3. Client issues read/write requests from/to and EC file that works parallelly on multiple internal blocks in a block group.
4. DataNode runs an ErasureCodingWorker (ECWorker) task for background recovery of failed erasure coded blocks. Failed EC blocks are detected by the NameNode, which then chooses a surviving DataNode to do the recovery work.
5. Hadoop uses EC policies to accommodate heterogeneous workloads (permitting files and directories to have different EC policies)
6. An EC policy is defined using the following data:
    a. *The EC schema:* This includes the numbers of data and parity blocks in an EC group (e.g., 6+3), as well as the codec algorithm (e.g., Reed-Solomon, XOR).
    Note: This differs from the (n,k) notation used commonly in coding theory parlance.
    b. *The size of a striping cell.* This determines the granularity of striped reads and writes, including buffer sizes and encoding work.
    c. As of now, six built-in policies are supported by HDFS EC module:
        i. RS-3-2-1024k
        ii. RS-6-3-1024k
        iii. RS-10-4-1024k
        iv. RS-LEGACY-6-3-1024k
        v. XOR-2-1-1024k and
        vi. REPLICATION.
7. Recovery is performed as three tasks:
    a. *Read the data from source nodes:* Input data is read in parallel from source nodes using a dedicated thread pool. Based on the EC policy, it schedules the read requests to all source targets and reads only the minimum number of input blocks for reconstruction.
    b. *Decode the data and generate the output data:* New data and parity blocks are decoded from the input data. All missing data and parity blocks are decoded together.

    c. *Transfer the generated data blocks to target nodes:* Once decoding is finished; the recovered blocks are transferred to target DataNodes.

8. EC policies are set on a directory. When a file is created, it inherits the EC policy of its nearest ancestor directory.
9. Once a file has been created, its erasure coding policy can be queried but not changed.
10. If an erasure coded file is renamed to a directory with a different EC policy, the file retains its existing EC policy. Converting a file to a different EC policy requires rewriting its data.
11. Erasure coded files are spread across racks for rack fault-tolerance. For rack fault-tolerance, it is important to have at least as many racks as the configured EC stripe width.
    Note: For clusters with fewer racks than the stripe width, HDFS cannot maintain rack fault-tolerance, but will still attempt to spread a striped file across multiple nodes to preserve node-level fault-tolerance.

Erasure code commands in Hadoop

1. The *ec* command can be used to work with erasure codes in HDFS. Its options can be seen in the figure on the last page.
2. Details of the options can be found on the web page: https://hadoop.apache.org/docs/r3.0.0/hadoop-project-dist/hadoop-hdfs/HDFSErasureCoding.html
3. To work with Hadoop file system, we also need to know about file systems commands of HDFS.
4. To create a directory, we use

```
        # Create a new directory (-p option to be used if parent directories do not exist)
        hduser@hadoopmaster:~$ hadoop fs -mkdir -p /home/hduser/testDir
```

5. For testing EC, we need big files. To create a big file, with random contents, use the following command from terminal:
   base64 /dev/urandom | head -c *number of bits* > *filename*.txt
6. We can set EC policy on our directory.

```
        # Enable EC policy first
        hduser@hadoopmaster:~$ hdfs ec -enablePolicy  -policy RS-3-2-1024k
        # Set EC policy to the test directory
        hduser@hadoopmaster:~$ hdfs ec -setPolicy -path /home/hduser/testDir
-policy RS-3-2-1024k
```

7. We can copy a local file to HDFS using the command (where gbfile.txt is the local file I have)

```
        # Copy a local file to the HDFS directory created
        hduser@hadoopmaster:~$ hadoop fs -put /home/hduser/gbfile.txt
/home/hduser/testDir
```

8. We can check the contents of our directory and its EC policy

```
        # Check contents of the HDFS directory created
        hduser@hadoopmaster:~$ hadoop fs -ls /home/hduser/testDir
        # See the file contents on terminal
        hduser@hadoopmaster:~$ hadoop fs -cat /home/hduser/testDir/ gbfile.txt
        # Check the EC policy on the file
        hduser@hadoopmaster:~$ hdfs ec -getPolicy -path /home/hduser/testDir/
gbfile.txt
```

9. Node failure simulation can be done by manually shutting down data nodes

```
        # Shut down a data node
        hduser@hadoopslave:~$ hdfs --daemon stop datanode
```

10. HDFS takes 10.5 minutes by default to mark a node as ‘down’. We can change this node failure detection time by altering the property *dfs.namenode.heartbeat.recheck-interval* in the *hdfs-site.xml* file on the master node.
    (since it is convenient for us to have this time shorter for our testing purposes)

```
hduser@hadoopmaster:~$ hdfs ec
Usage: bin/hdfs ec [COMMAND]
        [-listPolicies]
        [-addPolicies -policyFile <file>]
        [-getPolicy -path <path>]
        [-removePolicy -policy <policy>]
        [-setPolicy -path <path> [-policy <policy>] [-replicate]]
        [-unsetPolicy -path <path>]
        [-listCodecs]
        [-enablePolicy -policy <policy>]
        [-disablePolicy -policy <policy>]
        [-help <command-name>]

Generic options supported are:
-conf <configuration file>       specify an application configuration file
-D <property=value>              define a value for a given property
-fs <file:///|hdfs://namenode:port> specify default filesystem URL to use, overrides
'fs.defaultFS' property from configurations.
-jt <local|resourcemanager:port>  specify a ResourceManager
-files <file1,...> specify a comma-separated list of files to be copied to the map
reduce cluster
-libjars <jar1,...> specify a comma-separated list of jar files to be included in the
classpath
-archives <archive1,...>  specify a comma-separated list of archives to be unarchived
on the compute machines
The general command line syntax is:
command [genericOptions] [commandOptions]
```

The contents here is heavily based on the information available at Apache’s official page HDFS EC documentation https://hadoop.apache.org/docs/r3.0.0/hadoop-project-dist/hadoop-hdfs/HDFSErasureCoding.html