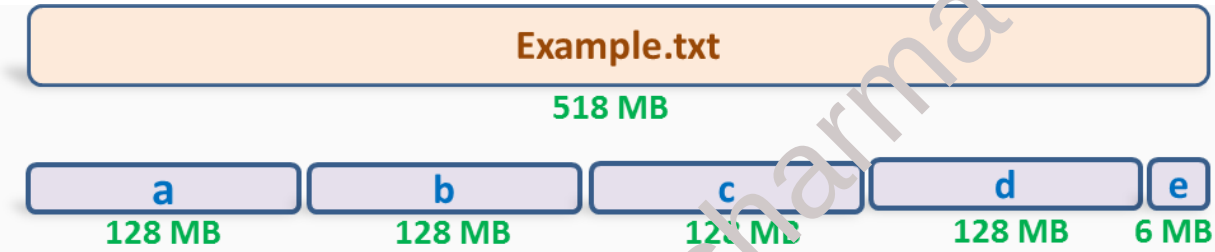
HDFS Data Block

- In Hadoop, HDFS splits huge files into small chunks known as data blocks. HDFS Data blocks are the smallest unit of data in a filesystem. We (client and admin) do not have any control over the data block like block location.
- Namenode decides all such things.
- HDFS stores each file as a data block. However, the data block size in HDFS is very large. The default size of the HDFS block is 128MB which you can configure as per your requirement.
- All blocks of the file are the same size except the last block, which can be either the same size or smaller. The files are split into 128 MB blocks and then stored into the Hadoop file system.
- The Hadoop application is responsible for distributing the data block across multiple nodes.



kratika sharma

- Now from above example where file size is 518MB suppose we are using the default configuration of data block size 128MB. Then 5 data blocks are created, the first four blocks will be of 128MB, but the last data block will be of 6 MB size only. From the above example it clear that it is not necessary that in HDFS, each file stored should be an exact multiple of the configured block size 128mb, 256mb etc., so final block for file uses only as much space as is needed.

What is Rack Awareness in Hadoop HDFS?

In a large cluster of Hadoop, in order to improve the network traffic while reading/writing HDFS file, namenode chooses the datanode which is closer to the same rack or nearby rack to Read/Write request. Namenode achieves rack information by maintaining the rack id's of each datanode. This concept that chooses closer datanodes based on the rack information is called Rack Awareness in Hadoop.

Why Rack Awareness?

In **Big data** Hadoop, rack awareness is required for below reasons:

- To improve data **high availability** and reliability.
- **Improve the performance** of the cluster.
- To improve network bandwidth.
- Avoid losing data if entire rack fails though the chance of the rack failure is far less than that of node failure.
- To keep bulk data in the rack when possible.
- An assumption that in-rack id's higher bandwidth, lower latency.

Advantages of Implementing Rack Awareness

- **Minimize the writing cost and Maximize read speed –** Rack awareness places read/write requests to replicas on the same or nearby rack. Thus minimizing writing cost and maximizing reading speed.
- **Provide maximize** network bandwidth and low latency – Rack awareness maximizes network bandwidth by blocks transfer within a rack. Especially with rack awareness, the YARN is able to optimize MapReduce job performance. It assigns tasks to nodes that are 'closer' to their data in terms of network topology. This is particularly beneficial in cases where tasks cannot be assigned to nodes where their data is stored locally.
- **Data protection against rack failure** – By default, the namenode assigns 2nd & 3rd replicas of a block to nodes in a rack different from the first replica. This provides data protection even against rack failure; however, this is possible only if Hadoop was configured with knowledge of its rack configuration.

Kratika Sharma