BDA 4-2 DFS

1 Hello everyone, I am Haiying Che, from Institute of Data Science and knowledge Engineering

School of Computer Science, in Beijing Institute of Technology , in this session we discuss about **distributed file system** .

2 The big data computing system can be summarized into three categories:

Data storing system, Data processing system, Data application system

The data storage architecture is the foundation of big data computing.

In data storing system , there are 4 parts to accomplish different tasks,

which are Data collection and modeling, **Distributed file system, Distributed database/data warehouse and Unified Data Access Interface.**

**we learned data collection and modeling, now we focus on distributed file system.**

**Actually File system could be Centralized or distributed file system, but in the big data scenario, mostly we use distributed file system to achieve the scale and the efficiency.**

3

The distributed file system provides a physical storage architecture for data.

File system with data stored on servers. In distributed file system the data is accessed and processed **as if** it was stored on the local client machine.

It is Convenient to share information and files among users on a network in a controlled and authorized way, which is transparent to the users.

And it reaches multiplied storage of a single server.

At present, there are the two main file systems in the big data computing architecture, which are

the open source community’s architecture HDFS and Google’s GFS (Google file system) which has evolved into Colossus.

4

Let ‘s look at the HDFS

HDFS adopts a master-slave structure. And HDFS save 3 copies of each data block, which means use redundancy to achieve high availability.

An HDFS cluster includes a Name node, which is the master node, and several Data Nodes, which are slave nodes.

As the central service node, the Name node is responsible for managing the file system namespace, the mapping relationship from data files to data blocks to Data nodes,

and client scheduling of file access. And the metadata is stored in **memory for quick access.**

HDFS also has a secondary name node, which is regularly connected to the primary name node, and the instant image of the system directory is stored on the local disk.

When the primary name node is fails or crashes, the secondary name node can provide the name node rollback recovery and restart functions.

Data node Store **file data block, Realize the mapping of data blocks to the local file system of the data node and Data blocks are stored on the local disk**

In HDFS, each storage file is first divided into multiple data blocks with a fixed length of 64MB or 128MB,

these data blocks are replicated to 3 copies and stored on different Data nodes according to a certain rule.

When one data node crashes, we can still retrieve the same data blocks from other 2 copies of another 2 data nodes.

Once the data is written, it can’t be changed , so the data in HDFS is immutable. So HDFS just support batch reading and writing operation, but doesn’t support updating operation.

This means that a Data Node can store data blocks from different files.

Each data node runs a node program or process, which is responsible for processing read and write requests from the file system client.

The creation, deletion, and replication of data blocks are performed under the unified scheduling of the Name node.

The master node Name node and the slave node Data node perform their respective tasks

5

Now let’s understand the Process of writing data in HDFS , The diagram summarizes file write operation in Hadoop.

1. The client creates the file by calling create() method on DistributedFileSystem.
2. DistributedFileSystem makes an RPC call to the namenode to create a new file in the filesystem’s namespace, with no blocks associated with it.
3. The namenode performs various checks to make sure the file doesn’t already exist and the client has the right permissions to create the file. If all these checks pass, the namenode makes a record of the new file; otherwise, file creation fails and the client is thrown an IOException. TheDistributedFileSystem returns an FSDataOutputStream for the client to start writing data to datanode. FSDataOutputStream wraps a DFSOutputStream which handles communication with the datanodes and namenode.
4. As the client writes data, DFSOutputStream splits it into packets, which it writes to an internal queue, called the data queue. The data queue is consumed by the DataStreamer, which is responsible for asking the namenode to allocate new blocks by picking a list of suitable datanodes to store the replicas. The list of datanodes forms a pipeline, and default replication level is three, so there are three nodes in the pipeline. The DataStreamer streams the packets to the first datanode in the pipeline, which stores the packet and forwards it to the second datanode in the pipeline.
5. Similarly, the second datanode stores the packet and forwards it to the third (and last) datanode in the pipeline.
6. DFSOutputStream also maintains an internal queue of packets that are waiting to be acknowledged by datanodes, called the ack queue. A packet is removed from the ack queue only when it has been acknowledged by all the datanodes in the pipeline.
7. When the client has finished writing data, it calls close() on the stream.It flushes all the remaining packets to the datanode pipeline and waits for acknowledgments before contacting the namenode to signal that the file is complete The namenode already knows which blocks the file is made up of , so it only has to wait for blocks to be minimally replicated before returning successfully.

6

The Process of reading data in HDFS 2.0 is like this,

1. The client opens the file by calling open() method on DistributedFileSystem.
2. DistributedFileSystem makes an RPC call to the namenode to determine location of datanodes where files is stored in form of blocks.For each blocks,the namenode returns address of datanodes(metadata of blocks and datanodes) that have a copy of block. Datanodes are sorted according to proximity(depending of network topology information).  
   The DistributedFileSystem returns an FSDataInputStream (an input stream that supports file seeks) to the client for it to read data from. FSDataInputStream in turn wraps a DFSInputStream, which manages the datanode and namenode I/O.
3. The client then calls read() on the stream. DFSInputStream, which has stored the datanode addresses for the first few blocks in the file, then connects to the first (closest) datanode for the first block in the file.
4. Data is streamed from the datanode back to the client (in the form of packets) and read () is repeatedly called on the stream by client.
5. When the end of the block is reached, DFSInputStream will close the connection to the datanode, then find the best datanode for the next block (Step 5)
6. When the client has finished reading, it calls close() on the FSDataInputStream (step 6).

In addition, During reading, if the DFSInputStream encounters an error while communicating with a datanode, it will try the next closest one for that block.It will also remember datanodes that have failed so that it doesn’t needlessly retry them for later blocks.

The DFSInputStream also verifies checksums for the data transferred to it from the datanode. If a corrupted block is found, the DFSInputStream attempts to read a replica of the block from another datanode; it also reports the corrupted block to the namenode.

7

In this session we learned the big data distributed file system mechanism using HDFS as an example, which is physical store of big data .

We learned the architecture of HDFS, name node, data node, and their responsibilities,

We also learned the data writing and data reading process of HDFS.

8

thank you for your attention, if you have any question, feel free to contact me.