Hadoop Distributed File System

Read Chapter 3 of the book "Hadoop – The Definitive Guide" and answer the following questions.

1. Understand the design considerations of HDFS and also when is it not a suitable solution.

HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters on commodity hardware. In some areas, HDFS is not a good fit: 1. Low-latency data access. Because HDFS is optimized for delivering a high throughput of data, and this may be at the expense of latency. 2. Lots of small files. Since the namenode holds filesystem metadata in memory, the limit to the number of files in a filesystem is governed by the amount of memory on the namenode. 3. Multiple writers, arbitrary file modifications. Files in HDFS may be written to by a single writer. Writes are always made at the end of the file. There is no support for multiple writers, or for modifications at arbitrary offsets in the file.

HDFS Blocks

1. Why are block sizes in HDFS so large? If the total file size is 1000 MB and seek time for each block is 10 ms and transfer rate is 100 MB/s, calculate the total file transfer time assuming a block size of 128 MB? If you double the block size, what would happen to the total time?

To minimize the cost of seeks. By making a block large enough, the time to transfer the data from the disk can be made to be significantly larger than the time to seek to the start of the block.
1000MB/128MB * 10ms + 1000MB/100MB/s = 10.08s
If the block size if doubled, the total time will be decreased to be 10.04s

2. What are some of the advantages of having a block abstraction for a multi-disk distributed file system?

1. A file can be larger than any single disk in the netword. 2. Making the unit of abstraction a block rather than a file simplifies the storage subsystem. 3. Blocks fit well with replication for providing fault tolerance and availability

Namenodes and Datanodes

1. Understand the functions of namenodes and datanodes.

Namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree. The namenode also knows the datanode on which all the blocks for a given file are located. Datanodes are the work horses of the filesytem. They store and retrieve blocks when they are told to, and they report back to the namenode periodically with lists of blocks that they are storing.

2. What are the two mechanisms that Hadoop provides for making the namenode resilient to failure?

The first way to back up the files that make up the persistent state of the filesystem metadata. Hadoop can be configured so that the namenode writes its persistent state to multiple filesystems. It is also possible to run a secondary namenode, whose main role is to periodically merge the namespace image with the edit log to prevent the edit log from becoming too large. It keeps a copy of the merged namespace image, which can be used in the event of the namenode failing.

3. Understand the purpose of secondary namenode. Does it run on the same machine as the primary namenode?

Its main role is to periodically merge the namespace image with the edit log to prevent the edit log from becoming too large. No, the secondary namenode usually runs on a separate physical machine, since it requires plenty of CPU and as much as memory as the namenode to perform the merge.

Block Caching

1. If an application wanted to cache a file, it would instruct the namenode (namenode/datanode), and the file would be cached by the one (one/multiple) datanodes by default.

HDFS Federation

1. If a cluster became very large, what would be the limiting factor? How does HDFS federation solve this problem. Understand the concept of namespace volume.

The limiting factor is the memory capacity of a NameNode memory (RAM). When there are more data nodes with many files, the NameNode memory will reach its limit and it becomes the limiting factor for cluster scaling. HDFS federation can make the namespace horizontally (add more RAM) scaled. Each of these NameNode can manage a portion of the filesystem namespace. A Namespace and its block pool together are called Namespace Volume. It is a self-contained unit of management. When a Namenode/namespace is deleted, the corresponding block pool at the datanodes is delete. Each namespace volume is upgraded as a unit, during cluster upgrade.

HDFS High Availability

1. If the namenode fails, the whole cluster goes down and time to recover and restore it to the previous running state would be large. How does HDFS high availability solve this problem in Hadoop 2.x? Understand the various architectural changes needed.

Hadoop 2.x introduceds HDFS Federation (a federation of multiple Active NameNodes that statically partition/divide the filesystem namespace and each Active NameNode contain only one partition/portion), which allows a cluster to scale by adding more NameNodes horizontally, each of which manages a portion of the filesystem namespace.

It has default 3 replicas of data to get high availability.

HDFS Command Line Interface

1. Besides the commands mentioned in the text, understand common file system shell commands: (No written answer required)
https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/FileSystemShell.html

The Java Interface

1. Understand the Java interface for interacting with HDFS. Go through the examples and try to understand the various classes involved. No written answer required.