一、说明三类分布式存储系统的区别

- 块存储系统
  - 以数据块 Block 作为操纵对象；
  - 不用考虑语义，效率很高；
  - 常用于底层物理存储；
- 文件存储系统
  - 以文件作为操纵对象；
  - 数据总是以文件为单位进行存取，语义性较强，在使用时可以忽略存储的物理细节，这方便了文件的共享；
  - 实际上文件的各部分可能分布在不同的物理位置，所以存取效率比较低；
  - 常用于共享的网络文件系统；
- 对象存储系统：
  - 以对象作为操纵对象；
  - 将数据本身与元数据分离；
  - 兼顾了块存储直接访问硬件和文件存储逻辑访问数据的特点，可以实现较大规模的数据存取与共享；
  - 常用于海量数据分布式存储系统。

二、阅读论文《The Hadoop Distributed File System》并回答问题

1.客户端读取HDFS系统中指定文件指定偏移量处的数据时，工作流程是什么?

When a client opens a file to read, it fetches the list of blocks and the locations of each block replica from the NameNode. The locations of each block are ordered by their distance from the reader. When reading the content of a block, the client tries the closest replica first. If the read attempt fails, the client tries the next replica in sequence. A read may fail if the target DataNode is unavailable, the node no longer hosts a replica of the block, or the replica is found to be corrupt when checksums are tested.

（1）Client 向 NameNode 询问，获取数据块列表和每个数据块副本的位置；

（2）Client 在获取数据块列表和每个数据块副本的位置，直接与 DataNode 联系，请求传输所需要的块。每个块的位置按其与读取器的距离排序。当读取块的内容时，客户端首先尝试最接近的副本。如果读取尝试失败，客户端将按顺序尝试下一个复制副本。如果目标DataNode 不可用，节点不再托管数据块的副本，或者在测试校验和时发现副本损坏，则读取可能会失败。

2. 客户端向HDFS系统中指定文件追加写入数据的工作流程是什么？

A. File Read and Write

An application adds data to HDFS by creating a new file and writing the data to it. After the file is closed, the bytes written cannot be altered or removed except that new data can be added to the file by reopening the file for append. HDFS implements a single-writer, multiple-reader model.

The HDFS client that opens a file for writing is granted a lease for the file; no other client can write to the file. The writing client periodically renews the lease by sending a heartbeat to the NameNode. When the file is closed, the lease is revoked.

An HDFS file consists of blocks. When there is a need for a new block, the NameNode allocates a block with a unique block ID and determines a list of DataNodes to host replicas of the block. The DataNodes form a pipeline, the order of which minimizes the total network distance from the client to the last DataNode. Bytes are pushed to the pipeline as a sequence of packets. The bytes that an application writes first buffer at the client side. After a packet buffer is filled (typically 64 KB), the data are pushed to the pipeline. The next packet can be pushed to the pipeline before receiving the acknowledgement for the previous packets. The number of outstanding packets is limited by the outstanding packets window size of the client.

After data are written to an HDFS file, HDFS does not provide any guarantee that data are visible to a new reader until the file is closed. If a user application needs the visibility guarantee, it can explicitly call the *hflush* operation. Then the current packet is immediately pushed to the pipeline, and the hflush operation will wait until all DataNodes in the pipeline acknowledge the successful transmission of the packet. All data written before the hflush operation are then certain to be visible to readers.

（1）向打开文件以进行写入的HDFS客户端授予对该文件的租约；其他客户端无法写入该文件。写客户端通过向 NameNode 发送心跳来定期续订租约。当文件关闭时，租约将被撤销。

（2）客户端向 NameNode 进行询问，如果指定文件原来的最后一个 Block 未满，则选定该 Block，否则分配一个新的 Blcok，并确定用于承载这个 Block 副本的 DataNode 列表。

（3）DataNode 形成一个管道，其顺序使从客户端到最后一个DataNode的总网络距离最小化。字节作为数据包序列推送到管道。应用程序在客户端写入第一个缓冲区的字节数。数据包缓冲区填满后（通常为64KB），数据将被推送到管道。在接收到对先前分组的确认之前，可以将下一个分组推送到流水线。

注意：在将数据写入HDFS文件后，HDFS不保证数据在文件关闭之前对新读取器可见。如果用户应用程序需要可见性保证，它可以显式调用hflush操作。然后，当前数据包将立即推送到管道中，并且Hflush操作将等待，直到管道中的所有 DataNode 都确认数据包已成功传输。在HFLUSH操作之前写入的所有数据肯定对读取器可见。

3. 新增加一个数据块时，HDFS如何选择存储该数据块的物理节点?

The default HDFS block placement policy provides a tradeoff between minimizing the write cost, and maximizing data reliability, availability and aggregate read bandwidth. When a new block is created, HDFS places the first replica on the node where the writer is located, the second and the third replicas on two different nodes in a different rack, and the rest are placed on random nodes with restrictions that no more than one replica is placed at one node and no more than two replicas are placed in the same rack when the number of replicas is less than twice the number of racks. The choice to place the second and third replicas on a different rack better distributes the block replicas for a single file across the cluster. If the first two replicas were placed on the same rack, for any file, two-thirds of its block replicas would be on the same rack.

The default HDFS replica placement policy can be summarized as follows:

1. No Datanode contains more than one replica of any block.
2. No rack contains more than two replicas of the same block, provided there are sufficient racks on the cluster.

- 创建新块时，HDFS 将第一个副本放置在写入程序所在的节点上，将第二个和第三个副本放置在不同机架中的两个不同的节点上，其余放置在随机节点上，且当副本数少于机架数的两倍时，有以下限制：在同一节点上放置的副本不能超过一个，在同一机架上放置的节点不能超过两个。将第二个和第三个副本放置在不同机架上的选择可以更好地在整个集群中分配单个文件的块副本。
- 总的来说，遵循两个原则：
  - 一个 DataNode 上至多放置一个相同数据块的副本；
  - 一个 rack 上至多放置两个相同数据块的副本，假设集群中的机架数是充足的。

4.HDFS采用了哪些措施应对数据块损坏或丢失问题?

In a cluster of thousands of nodes, failures of a node (most commonly storage faults) are daily occurrences. A replica stored on a DataNode may become corrupted because of faults in memory, disk, or network. HDFS generates and stores checksums for each data block of an HDFS file. Checksums are verified by the HDFS client while reading to help detect any corruption caused either by client, DataNodes, or network. When a client creates an HDFS file, it computes the checksum sequence for each block and sends it to a DataNode along with the data. A DataNode stores checksums in a metadata file separate from the block's data file. When HDFS reads a file, each block's data and checksums are shipped to the client. The client computes the checksum for the received data and verifies that the newly computed checksums matches the checksums it received. If not, the client notifies the NameNode of the corrupt replica and then fetches a different replica of the block from another DataNode.

E. Block Scanner

Each DataNode runs a block scanner that periodically scans its block replicas and verifies that stored checksums match the block data. In each scan period, the block scanner adjusts the read bandwidth in order to complete the verification in a configurable period. If a client reads a complete block and checksum verification succeeds, it informs the DataNode. The DataNode treats it as a verification of the replica.

The verification time of each block is stored in a human readable log file. At any time there are up to two files in top-level DataNode directory, current and prev logs. New verification times are appended to current file. Correspondingly each DataNode has an in-memory scanning list ordered by the replica's verification time.

Whenever a read client or a block scanner detects a corrupt block, it notifies the NameNode. The NameNode marks the replica as corrupt, but does not schedule deletion of the replica immediately. Instead, it starts to replicate a good copy of the block. Only when the good replica count reaches the replication factor of the block the corrupt replica is scheduled to be removed. This policy aims to preserve data as long as possible. So even if all replicas of a block are corrupt, the policy allows the user to retrieve its data from the corrupt replicas.

- 采用"三副本"策略，一个 Block 会拥有多个副本，且分布在不同的节点、不同的机架上；
- 每个 DataNode 上运行一个块扫描器，该扫描器定期扫描块副本并验证其存储的校验和是否与块数据匹配。如果客户端读取了完整的块并且校验和验证成功，它将通知 DataNode，DataNode 将其视为对副本的验证。每当读取客户端或块扫描程序检测到损坏的块时，它都会通知 NameNode。NameNode 将复制副本标记为损坏，但不计划立即删除复制副本。仅当好的副本计数达到块的复制因子时，才计划删除损坏的副本。

5.HDFS采用了什么措施应对主节点失效问题?

The BackupNode accepts the journal stream of namespace transactions from the active NameNode, saves them to its own storage directories, and applies these transactions to its own namespace image in memory. The NameNode treats the BackupNode as a journal store the same as it treats journal files in its storage directories. If the NameNode fails, the BackupNode's image in memory and the checkpoint on disk is a record of the latest namespace state.

The BackupNode can create a checkpoint without downloading checkpoint and journal files from the active NameNode, since it already has an up-to-date namespace image in its memory. This makes the checkpoint process on the BackupNode more efficient as it only needs to save the namespace into its local storage directories.

The BackupNode can be viewed as a read-only NameNode. It contains all file system metadata information except for block locations. It can perform all operations of the regular NameNode that do not involve modification of the namespace or knowledge of block locations. Use of a BackupNode provides the option of running the NameNode without persistent storage, delegating responsibility for the namespace state persisting to the BackupNode.

- 当主节点失效时，从内存中读取 BackupNode 来恢复主节点状态。

6.NameNode维护的"数据块—物理节点对应表"需不需要在硬盘中备份？为什么?

A DataNode identifies block replicas in its possession to the NameNode by sending a *block report*. A block report contains the *block id*, the *generation stamp* and the length for each block replica the server hosts. The first block report is sent immediately after the DataNode registration. Subsequent block reports are sent every hour and provide the NameNode with an up-to-date view of where block replicas are located on the cluster.

During normal operation DataNodes send *heartbeats* to the NameNode to confirm that the DataNode is operating and the block replicas it hosts are available. The default heartbeat interval is three seconds. If the NameNode does not receive a heartbeat from a DataNode in ten minutes the NameNode considers the DataNode to be out of service and the block replicas hosted by that DataNode to be unavailable. The NameNode then schedules creation of new replicas of those blocks on other DataNodes.

- 不需要在硬盘中备份；
- NameNode 是基于内存的，块位置信息只存储在内存中；DataNode 与 NameNode 维持心跳，定期以 block report 的形式汇报自己持有的 block 信息。