BASIC TASKS

(1) a) **Main advantage**

Improving access speed: Users can access file replicas on the nearest server, reduce network latency, and improve file access speed and response time.

Load balancing: By distributing file requests to multiple servers, it avoids single point overload and improves the stability and efficiency of the entire system.

Enhanced availability: Even if one server fails, users can still retrieve files from other servers, improving the system's fault tolerance and availability.

b) **Potential drawbacks**

storage cost: Copying files to multiple sites will significantly increase storage costs as each site requires additional storage space.

Data consistency: If a file needs to be updated, ensuring consistency across all replicas will become complex and may require complex synchronization mechanisms.

Network bandwidth consumption: The initial replication and subsequent updates of files consume a significant amount of network bandwidth, especially when the files are large or frequently updated.

Management complexity: Managing file replicas distributed across multiple sites, including monitoring, updating, and maintenance, increases the complexity and cost of management.

(2) a) **main drawback**

Network latency and bandwidth limitations: Transferring large files can consume a significant amount of network bandwidth, potentially leading to network congestion, increased transmission latency, and reduced overall system performance.

Resource consumption: Processing large files on node A may consume a significant amount of CPU and memory resources, especially when analyzing large files in real-time, which may put pressure on the resources of node A.

Inefficiency: If only a small part of the data in the file is filtered, transferring the entire file to node A and then filtering it is inefficient, especially in poor network conditions or when the file is particularly large.

b) **a better way**

performing filtering operations at the location of the data (node B)

data locality processing: first filter the data on node B, and only transmit the filtered results to node A. This can significantly reduce data transmission volume, save network resources, and lower latency.

Distributed computing: If possible, distributed computing frameworks such as Hadoop or Spark can be used to process files in parallel on multiple nodes, distributing processing tasks to the location of data and improving efficiency.

c) **locality of data**

Data locality refers to arranging computing tasks as closely as possible to the location of data in distributed computing or data processing, in order to reduce remote transmission of data, thereby reducing network latency and bandwidth consumption. Data locality is one of the key principles for improving the performance and efficiency of distributed systems, especially when dealing with large-scale datasets. By optimizing data locality, task execution speed and overall system performance can be significantly improved.

(3) Sequential / parallel processing.

(a) Look up the following terms:

i) Sequential processing:

Sequential processing refers to the process of processing data or executing tasks in a pre-defined order or step by step, where each step or task is fully executed before starting the next step or task. In sequential processing, system resources (such as CPU and memory) can only handle one task at a time, and the result of the previous task is often the input of the next task.

ii) Parallel/Distributed Processing:

Parallel processing typically refers to the use of multiple processors within a single computing system to simultaneously execute multiple tasks or different parts of tasks. Distributed processing involves simultaneously processing data or tasks across multiple computing nodes (such as different computers or servers), exchanging data and results through network connections.

(4) **a)** Calculations (a+b) and (c/d) can be performed simultaneously without waiting for each other's results. Similarly, the calculations of (e * f) and (g/h) are also independent and can be executed in parallel.

b) Although addition, subtraction, multiplication, and division can be performed in parallel at different stages, in the end, the results of (a+b) * (c/d) and (e * f) + (g/h) must be read and used to perform the final subtraction operation. This means that in the final stage of computation, the system may require some synchronization mechanism to ensure that both results are correctly read and applied to subtraction, which may affect the smoothness of parallel execution.

c) Data transmission delay: The transfer of results from one node to another consumes time, which may become a key factor limiting the overall parallel processing speed. Synchronization and consistency: Ensure that all nodes have the same version of data and perform calculations at the appropriate time, which requires complex synchronization mechanisms and may increase additional latency. Fault recovery: A node failure in a distributed system may cause the entire computing process to be blocked, and a robust fault recovery strategy needs to be designed.

5) 1. Divide the file evenly into 50 parts, and assign each copy to a computer node. Assuming the file size is large enough, each copy of data still contains enough digital samples.

2. Each node runs the algorithm independently on its allocated data segment, finding out the maximum value of that section. This can be done by traversing its assigned data segment and recording the maximum value encountered.

3. After all nodes have completed the calculation of the local maximum, they send their results to a central node or pass them between all nodes through a protocol.

4. The central node or any node collects all 50 local maximums, and then finds the largest of these 50 values as the global maximum.

(5). Perhaps it could be divided into 25 parts and be stored in the empty nodes in the next step.

6) the distributed/parallel manner are barely the same. The file size of 4. a) is smaller

7) a) In HDFS (Hadoop Distributed File System), the roles of NameNode and DataNode are complementary. As the master node, the NameNode is responsible for managing the namespace of the file system, including maintaining the file directory tree structure, file metadata, and mappings from blocks to DataNodes. The DataNode is responsible for storing the actual data blocks and reporting their status to the NameNode on a regular basis.

b) For a Hadoop cluster consisting of 3 shelves with 3 nodes each:

i) Suppose the customer requests to store a file of 180MB size, and the NameNode response has 4 nodes that can be used to store shards, which are Node 1, Node 3, Node 5, Node 7.

ii) In HDFS, files are divided into 64MB chunks by default. Thus, a 180MB file will be divided into 3 blocks: S1, S2, S3. The size of each block is 64MB, 64MB and 52MB (the rest) respectively.

In HDFS, a minimum of 3 replicas of each block are required for fault tolerance. Replication is achieved through automatic control by NameNodes, which store copies of data blocks on other nodes in the cluster. This way, even if a node fails, data is not lost.

iii) Select a slice, such as S1, with the following configuration:

1.S1: Node 1, Node 3, Node 5

iv) If S1 is stored in Node 3, Node 5, or Node 7, but Node 5 crashes suddenly, NameNode will

take the following actions:

- NameNode will detect a fault on Node 5.
- It will copy S1 again from the remaining Node 1 and Node 7. （
- NameNode will select another node in the cluster (possibly something other than Node 1 or Node 7, if any) to store the missing S1 replica to restore the original level of redundancy.
- This new node will receive data from the existing S1 replica (either Node 3 or Node 7) to create a new replica.

8) a) JobTracker is responsible for coordinating the execution of the entire job, including assigning tasks, monitoring task status, and failing back. TaskTracker is responsible for running the actual map and reduce tasks on each node. Communication between JobTracker and TaskTracker is done through a heartbeat mechanism, which periodically sends heartbeat information to JobTracker to indicate its current status. If TaskTracker doesn't send heartbeat information for a while, JobTracker thinks the TaskTracker has failed and reassigns its tasks.

b) The functionality of the map function is to convert the input data into intermediate key-value pairs for subsequent reduce operations. The map function takes a key-value pair as input and processes the value to produce a series of key-value pairs as output. The map function is commonly used to filter, split, and preprocess data.

The functionality of the reduce function is to merge and process the intermediate key-value pairs generated by the map function to produce the final result. The reduce function takes a series of intermediate key-value pairs as input and processes their values to produce a key-value pair as output. The reduce function is typically used to aggregate, count, and calculate data.

c) In MapReduce, we can design a map function that outputs the color and number of each square as key-value pairs. The map function takes a square as input and outputs its color and number as key-value pairs. Next, we can design a reduce function to merge the number of squares of the same color. The reduce function takes a series of key-value pairs as input and adds up the number of squares of the same color to calculate the number of squares of each color.

9) In order to find the total number sum of each device (equip_name), a MapReduce algorithm can be designed. In the Map phase, the input key will contain order_id and equip_name, and the output key should be equip_name and the corresponding number of key-value pairs. The Reduce stage aggregates the quantities under the same equip_name to finally output the total quantity for each device.

The input key of the Map function:

"order_id": The unique identifier of the order

equip_name: The name of the device

The output key of the Map function:

equip_name: The name of the device

"quantity": The total quantity of the order for the device
The input key of the Reduce function:
equip_name: The name of the device
The output key of the Reduce function:
equip_name: The name of the device
total_quantity: The total quantity of all orders for this device

10) MapReduce can be used to split information such as account number, movie or TV show name, price, and more into subsets to calculate the number of streams and charges separately.
In the MAP phase, you can use the account number as the key and the movie/TV show name as the value, and send the data to multiple nodes for processing. Each node can match the account number and movie/TV show name with the corresponding price information and store the results in memory.
In the Reduce phase, you can use the account number and the movie/TV show name as keys, and merge all the corresponding price information as values. This can be achieved through hash functions in order to group data into different nodes for processing; And multiplying the number of streams and the price of each movie/TV show to get what each movie/TV show will pay. The sum of the costs is then calculated in order to pay the producer of the movie/TV show.