Big Data Experiment 4 Questions Answered

First, the basic tasks

(1) File copying decisions

1. **Pros**

- Improved data availability: Replicas on 20 different hosts greatly increase the chances of retrieving files in the event of a problem with some hosts or networks, allowing the system to continue to provide service.

- Improved read performance: Users can choose to read from a host that is close to each other or has a light load to reduce data transfer time, especially in high-concurrent read scenarios, which can effectively spread the read load and improve the overall read speed.

2. **Cons**

- Storage resource consumption: A large number of replicas occupy a large amount of storage space, increasing the cost and management of storage devices.

- Complex data consistency maintenance: When files are updated, ensure that all replicas are updated in a timely manner, otherwise data inconsistencies will occur, requiring a complex consistency management mechanism and increasing system overhead.

(2) Remote file processing

1. **Cons**

    - High network transmission overhead: When transferring large files from remote node B to node A for processing, the transmission process will take a lot of time when the network bandwidth is limited or the files are large, which seriously affects the overall system performance.

2. **Improvement Methods**

    - Distributed computing: Directly filters files on remote node B and transmits the processing results back to node A. This significantly reduces the amount of data transmitted over the network and increases processing efficiency.

3. **Data Locality Concept**

    - Data locality means that computing tasks should be executed on nodes where data is located or close to it. The goal is to reduce the network overhead and performance loss caused by data movement, make full use of local resources, and improve the overall performance of the system. For example, in distributed computing, if the data is on a node and the computing task can be executed on that node or its nearby nodes, the data can be avoided from being transmitted over long distances in the network, thus speeding up the computation.

(3) Matrix multiplication parallel processing

1. For multi-core systems, matrices A and B can be split into multiple submatrices in rows or columns, with each core responsible for calculating the multiplication results of a submatrix, and finally merging the results of these submatrices to obtain the final matrix C. For example, for a matrix multiplication of $n\times n$, if there are m cores, the matrices A and B can be split into m submatrices of about $\frac{n}{m}\times n$ in size, each core computes the multiplication of a pair of submatrices, and then merges the results.
2. In a distributed system, matrix data distribution can be stored on multiple nodes, and nodes can be co-computed through network communication. A common approach is to distribute the rows of matrix A and the columns of matrix B on different nodes, and each node calculates a partial result, and then calculates the final result through communication and aggregation between nodes. For example, if there are p nodes, you can store the $\frac{n}{p}$ rows of matrix A on one node, and the $\frac{n}{p}$ columns of matrix B on another node, and calculate the partial product and summarize them through messaging.

(4) Mathematical formulas perform analysis in parallel

1. **Parallel Executable Sections**

- The four subexpressions of $(a + b)$, (c/d), $(e * f)$, and (g/h) in the formula are independent of each other, have no data dependencies, and can be executed in parallel on different computer nodes.

2. **Parallel Execution Limit**

- Data dependency: Subsequent calculations$(((a + b) * (c/d)) - ((e * f) + (g/h))$ depend on the results of the previous subexpressions, and you need to wait for all parallel computations to complete and synchronize data before you can continue to execute. If the data synchronization mechanism is imperfect or inefficient, it can affect the overall performance.

- Resource contention: If nodes that execute in parallel have limited resources (such as CPU and memory), multiple subexpressions competing for resources at the same time may cause performance degradation and prevent the advantages of parallelism from being fully utilized. For example, if a node has a limited number of CPU cores, multiple computing tasks compete for CPU time slices at the same time, which will slow down the execution of each task.

(5) Distributed method for determining the maximum number

1. Divide the large file into 50 data blocks, and each computer node is responsible for processing one data block to find out the maximum

number of data blocks in this node. Each node then returns the maximum number it finds to a central node, which in turn finds the maximum value from the 50 maximum numbers, which is the maximum number in the entire file. For example, suppose the number in the file is \(1, 5, 3, 7, 9, 2, 4, 6, 8\cdots\), divide it into 50 data blocks, node 1 processes data block 1 to get the maximum number of \(7\), node 2 processes data block 2 to get the maximum number of \(9\), and finally the central node finds the maximum value from \(7, 9\cdots\).

(6) Differences from parallel processing with mathematical formulas

1. Different data characteristics: This example deals with a file composed of a large number of simple numbers, and there is no complex logical relationship between the data. Parallel processing of mathematical formulas, on the other hand, involves expression computation, with clear computational logic and data dependencies.

2. Different parallelisms: In this example, the maximum number is found in parallel at multiple nodes through data segmentation. The mathematical formula is to calculate independent subexpressions in parallel at different nodes, and then combine the results and perform subsequent calculations.

2. Medium tasks

(1) The relationship between the NameNode and the DataNode in HDFS

1. NameNode is the management node of HDFS and is responsible for managing metadata information such as namespaces, directory structures, and file permissions of the file system. It records the mapping of files to data blocks, but does not store the actual data. For example, when a user requests access to a file, NameNode determines which DataNodes the file's data blocks are stored on based on metadata information.
2. The DataNode is the data node that actually stores data, is responsible for storing and managing data blocks, and regularly reports the block information stored by the NameNode, including the status and storage location of the blocks, so that the NameNode can grasp the data storage status of the entire cluster. For example, the DataNode tells the NameNode which blocks of data it stores, whether those blocks are healthy, and so on.

(2) HDFS file storage and processing analysis

1. **File Splitting**

- The file size is 180MB, the block size is 64MB, \(180\div64 = 2\) (52MB remaining), so the file is divided into 3 chunks, the size is 64MB, 64MB, and 52MB, which are denoted as S1, S2, and S3.

2. **Number of replicas and replication method**

- HDFS 默认每个块有 3 个副本。 当客户端向 NameNode 请求存储文件时,NameNode 根据节点负载、存储空间等因素选择合适的 DataNode 存储副本。 例如,S1 可能存储在 Node1、Node3 和 Node5 上,S2 存储在 Node3、Node7 和 Node1 上(具体分配根据 NameNode 算法确定),S3 存储在 Node5、Node7 和 Node3 上,确保数据的可靠性和高可用性。

3. **Node Fault Handling**

- If Node5 crashes, NameNode detects the node fault through a heartbeat. It selects other available nodes (such as Node2 and Node6) to replicate replicas of S1 and S3 based on the storage policy to maintain the number of replicas of each block and ensure data integrity and availability. For example, NameNode might select Node2 and Node6 to replicate the replica of S1, and Node2 and Node8 to replicate the replica of S3.

3. Advanced tasks

(一)MapReduceponJob Exerciseer 与 TaskBcer 关系

1. JobTracker is the manager of MapReduce jobs, which is responsible for receiving jobs submitted by users, decomposing jobs into multiple tasks (Map tasks and Reduce tasks), and assigning them to TaskTracker for execution. It monitors task execution status, coordinates task execution sequence, and is responsible for reassigning tasks if they fail. For example, when a user submits a job that counts word frequency, JobTracker will decompose it into multiple Map tasks and Reduce tasks, and assign them to different TaskTrackers for execution, and track the progress of the task at the same time.
2. TaskTracker runs on nodes in the cluster and is responsible for executing tasks assigned by JobTracker. It periodically reports the task execution progress and status information to the JobTracker, and obtains the data blocks required for the task from the DataNode for calculation. For example, TaskTracker tells JobTracker how much of the task it is executing has been completed, if it has encountered errors, etc., and reads data from the local DataNode for calculations.

（二）MapReduce 函数功能解释

1. **Map function function**

    - Preprocess the input data and split the data into key-value pairs. For example, for text data, you can read by row and use each line

content as a value, a line number, or a specific field as a key. The Map function can filter and transform the data, and output intermediate results (also in the form of key-value pairs), which will be distributed to different Reduce tasks for subsequent processing according to the key values. For example, when counting word frequency, the Map function can use the words in each line of text as keys, and set the value to 1 to indicate one occurrence.

2. **Reduce 函数功能**

- Merge and process intermediate results with the same key. It receives a list of values of the same key from multiple Map tasks, performs operations such as summarization and calculation, and finally outputs the processing result. For example, in a task that counts the number of word occurrences, the Reduce function adds up the counts of the same word to get the final word frequency statistics.

（三）MapReduce 应用示例

1. **Color Count Example**

- **Map Phase**: Use color as the key and set the value to 1 to indicate that it occurs once. For example, for the input data "R G G G", the Map function outputs \((R, 1)\), \((G, 1)\), \((G, 1)\), \((G, 1)\).

- Reduce phase: Merge key-value pairs of the same color and add the values to get the number of each color. For example, for the color R, the Reduce function receives \((R, 1)\) and \((R, 1)\) (assuming that there is R elsewhere) and adds the values to give \((R, 2)\), i.e. there are 2 red squares. The same can be done to calculate the number of green and other colored squares.

2. **Equipment Order Quantity Counting Example**

- InputKey: The name of the device (equip_name), as the number is counted by device name.

- OutputKey: The name of the device (equip_name), and the output result is categorized by device name.

- **Map 阶段**：读取订单数据，将设备名称作为键，订单数量（qty）作为值输出。 例如，对于订单“L_id:order_id(2237),cust_id:"168",cust_name:"Coventry Building",cust_addr:"Binley Estate,CV3 2W",date:"14/2/2018",equipment:[equip_name:"Butterfly Valve",qty:"4",unit_price:"20"}”，Map 函数输出\(("Butterfly Valve", 4)\)。

- Reduce Stage: Add the order quantities of the same device name to get the total order quantity for each device. For example, for the device "pipe", the Reduce function adds multiple \(("pipe", n)\)(n

is the quantity in different orders) to get \(("pipe", total_quantity)\), which is the total order quantity of the "pipe" device.

3. **Streaming Service Billing Example**

- **Map Stage**: Use the name of the movie/TV show as the key and set the value to 1 to indicate that there is a viewing record. For example, for the input data "| 121212123 | Aladdin | 0.05 |", the Map function outputs \((Aladdin, 1)\).

- Reduce Stage: Count the watch history of the same movie/TV show to get the number of times each show has been watched. The payable amount is then calculated based on the price, and the output is the program name and the payable amount. For example, for the movie "Aladdin", the Reduce function calculates the number of views \(n\), the amount payable is \(n * 0.05\), and the output \((Aladdin, payment_amount)\).