Quiz1 Solution

Cohadoop and E3

1. Describe one disadvantage of cohadoop that may occur if the handling of the colocated data is not performed carefully.

One disadvantage is “load imbalance”. If many files are assigned the same locator, then without careful design, CoHadoop will target the same data nodes for storing the files, which makes the load imbalanced across the nodes.

2 Describe the difference between partitioning and colocation Are these two concepts independent of each other or one relay on the other?

They are two independent concepts but have some relationship. We can partition without colocation, we can colocate without partitioning (e.g., colocating a file and its index), or we can do both partition and then colocate.

3. State whether each of the following is T or F:

1.A Locator is assigned to a file automatically by cohadoop at the loading time.

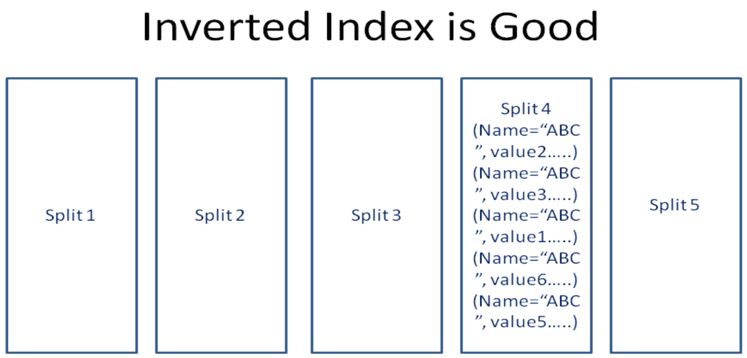
False. Locator is assigned by the application as it know the semantics of the data.

2.In cohadoop, files having the same locator are not guaranteed to be stored in the same set of nodes.

True. Colocation is best-effort approach not enforced.

4. In E3, assume we have a file F containing 5 partitions and we are searching for attribute Name="ABC" which exists 5 times in the file. Draw the best scenario(the distribution of 'ABC' across the partitions) in which an inverted index would perform the best.

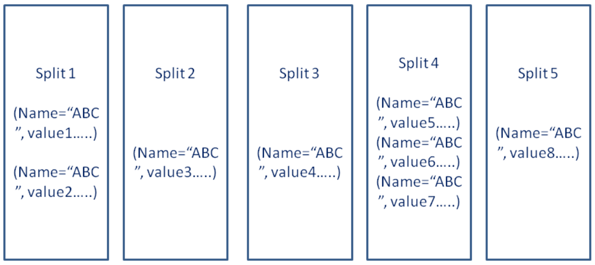
One splits contain all required records. The query guided by the inverted index only need to access one split.


Inverted Index is Good
Split 1
Split 2
Split 3
Split 4
(Name=“ABC”, value2.....)
(Name=“ABC”, value3.....)
(Name=“ABC”, value1.....)
(Name=“ABC”, value6.....)
(Name=“ABC”, value5.....)
Split 5

5.The same as in q4, but now draw the worst scenario in which the inverted index will be of no use. In this case, what other method E3 deploys(mention the name of the method and the content of the constructed file.)

The satisfied records are spread all splits. The query needs to access all splits.

Inverted Index is Bad



Split 1

(Name=“ABC”, value1.....)

(Name=“ABC”, value2.....)

Split 2

(Name=“ABC”, value3.....)

Split 3

(Name=“ABC”, value4.....)

Split 4

(Name=“ABC”, value5.....)
(Name=“ABC”, value6.....)
(Name=“ABC”, value7.....)

Split 5

(Name=“ABC”, value8.....)

Building a materialized view is a solution in which all records containing value ‘ABC’ in the Name attribute will be copied to it. Therefore, the query will be routed to access the materialized view instead of the original file.

Haloop

6 what are the types of caches that haloop propose? for each type give a brief description on how and when it can save computations.

Map-input cache:

Localize the mapper input data for later iteration. Assume map input does not change through iteration.

Reduce-input cache:

Access to loop invariant data without shuffle.

Reduce-output cache:

Distribute access to the output of previous iteration. It's used by fix point evaluation or checking convergence.

7. For the following analytical techniques k-means clustering, page rank, and naive bayes mention which types of caches offered by haloop can be used to speed up each technique?

K-means:

Mapper-input cache.

Page rank:

Reduce input cache.

Reduce output cache.

Naive Bayes:

None.

8. For each of the cache types supported by haloop, describe how cache reloading is done in the case where the node having the cached data fails.

Map Input cache fails:

Map-input cache (say on Node X) is for data that is already stored in HDFS. So if X fails, and node Y will execute the job, the cache content is read from HDFS from the original file.

Reduce Input cache fails:

We will need to read the map-output again and do the re-shuffling and sorting. If the map output is not kept, then the map functions will need to execute again over the original data to do the partitioning.

Reduce output cache fails:

If the output was written to HDFS, then read it from another node. If it is not stored in HDFS, then re-do the whole map-reduce job.