PLAN MY TRIP

Large Scale Joins Using HBase and MapReduce

Team: Sparklings

By Shubham, Ritika, Abhishek

GOAL

Compare and contrast the performance of joins using RS Join versus HBase for computation of the best 3 hop itinerary for a traveler between all possible sources and destinations using an Airline Dataset.

OUTCOME

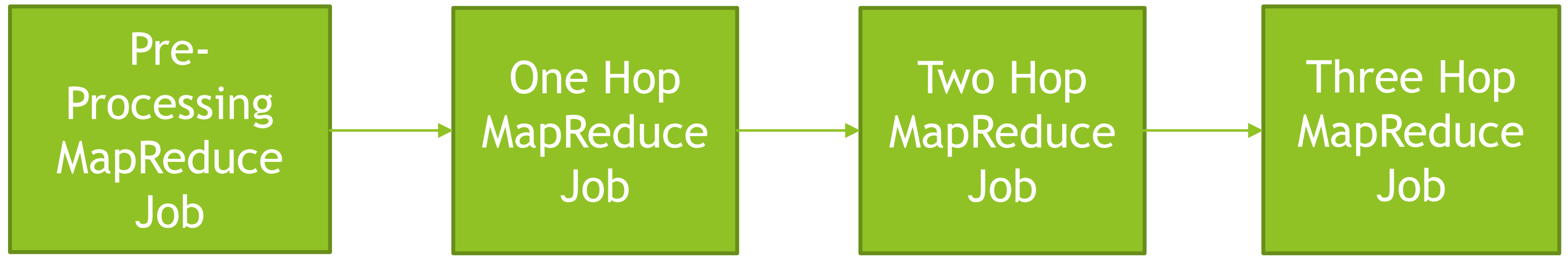
- Our analysis from this could help travelers (like explorers, bloggers etc.) who want to visit 3 different locations between a source and destination, plan out the best itinerary possible with the least possible trip duration. We will include constraints while calculating results so that the traveler can spend minimum 10 hours and max of 72 hours in each city.

Outcome Contd..

- Running queries on the data stored on HBASE for getting insightful information: Find the busiest airport in terms of number of flights, number of flights delayed per airline and average delay per month per airport etc. **Our analysis from this could be used to point out which cities would need an additional airport to divert some traffic.**

Join implementation using Reduce Side Join

- Flow :



Join Conditions

- A -> B -> C -> D -> A
- Join on the Airport Id.
- Departure airport Id of B == Arrival airport Id for C [one hop]
- Departure airport Id of C == Arrival airport Id for D [two hop]
- Departure airport Id of D == Arrival airport Id for A [third hop back home]

Join Condition Contd..

- Constraints -1 : Time spent at each city [B, C, and D] should be greater than 10 hours and less than 72 hours.
- Constraint -2 : Unique cities visited in a single itinerary i.e. B, C and D are all unique.
- Note : The intermediary data of each hop was maintained to provide a better plan of the trip.
- The calculation of total trip time was done using the start date-time and end date-time.

RS Join Performance

Machine	Job	Time	Data Shuffled in Bytes
M4-large 5 workers and 1 master	One - Hop	8 mins	2531844052
	Two – Hop	12 mins	32430960720
	Three – Hop	32 mins	390077396856
M4-large 10 workers and 1 master	One - Hop	7 mins	4216476904
	Two – Hop	10 mins	104678775100
	Three – Hop	26 mins	7835373927301

RS Join Performance

- Speed Up = 0.8

Data	5 m4.large Cluster	10 m4.large Cluster
7 years data	56 mins	46 mins

Join implementation using HBase

- HBase helps in overcoming some of the limitations of Reduce side join by providing random access to the data.
- For airline data, A two hop path is calculated by identifying destination for each source. Each mapper emits a (source, destination) pair.
- In reducer, for each source, we get a list of destinations. For every destination d in the list, we use them as a source and find all the nodes reachable from each of the destination.
- If the arrival time and departure time satisfies the constraints of being greater than 10 hours but greater than 72, the result is then added to TwoHop table.
- Further, Two hop table is used, to compute three Hop Path.
- The three hop table is then used to find the four hop path and looks for cities

  which are similar to starting city

Choice of Row Key

The are four tables have the following row keys :

Airlines	:> Source + Destination + TimeStamp
AirLinesTwoHops	:> Source + Destination1 + Destination2 + TimeStamp
AirLinesThreeHops	:> Source + Destination1 + Destination2 + Destination3 +Timestamp
AirLinesFourHops	:> Source + Destination1 + Destination2 + Destination3 + Destination4 + Timestamp

Comparison and analysis of results HBase vs RS Join

- One of the biggest disadvantages of reduce side join on huge data sets is its tendency to duplicate data while transferring it from mapper to reducer. This becomes a huge bottleneck in terms of memory.

RS Join Vs HBase contd..

- **Why data duplication/transfer is needed in RSJoin?**

It is needed so that it is possible to apply join logic to the entire data sets containing common attributes. If only a sample of data is transferred to avoid memory costs, then it will not lead to efficient join computations and join results will have missing tuples.

RS Join Vs HBase contd..

- **How HBase solves the problem of duplication ?**

Because of its ability to access random data stored on hdfs in the form of regions, HBase acts as a hash index on the data store. Hence, whenever a join needs to perform on a common attribute, HBase requests for that row key from Zookeeper (performs coordination among multiple region servers and contains meta data to handle incoming requests). Zookeeper directs the request to the region server from which it can fetch all the matching keys as per the join attribute.

RS Join Vs HBase contd..

- HBase does not require shuffling for implementing the join condition, whereas reduce side join depends on mappers to emit the join attribute as the key which then has to be shuffled to reach the right reduce tasks.
- HBase provides a way to query data randomly. Also, if key queried in subsequent fetches belongs to the same partition as a previous fetch, then HBase efficiently remembers the partition and fetches from it.
- HBase seems to be a good alternative for rep join as it provides the benefits of a rep join but does not necessitate one table of the join to be significantly smaller than the other like in rep join.

RS Join vs HBase

	RS JOIN	HBASE
Time	56 mins	45 mins
Data Shuffled	390077396856	104678775100

THANK YOU!

And Happy Holidays!

Questions?