Big Data Computing
Lab - VI

September 7, 2021

I. Reduce Side Join using MapReduce

Given two datasets with more than 10K+ records each including: 1) ***customers*** storing the details of different users, and 2) ***sales*** concerning their respective shopping patterns, we will utilize Reduce-side join operations via MapReduce to output **the total sales by state**. The schema for the the ***customers*** dataset includes:

Column	Datatype	Description
CustomerID	Integer	Unique ID of the customer
CustomerName	String	Name of the customer
Street	String	Street name of the customer's address
City	String	City of residence of customer
State	String	State of residence
ZipCode	Integer	Zip code of the customer's residence

Where a sample cross-section of the data is represented below:

```
5646,Melanie Ramsey,Michael Lane,New Lisa,Indiana,98159
3536,Cheryl Miller,Elizabeth Shaw,East Kyle,West Virginia,53126
4333,John Freeman,April Walter,North John,Louisiana,06280
9634,Dennis Brewer,Andrew Williams,Paulaport,Virginia,07631
```

While attributes for the sales dataset include:

Column	Datatype	Description
Timestamp	String	Date on which the order was the customer in YYYY-MM-DDTHH:MM:SS format
CustomerID	Integer	Unique ID of the customer
SalesPrice	Double	Total amount spent by the customer

With a cross-section view of the data as follows:

```
2002-04-04T10:36:32,4333,720.0343123
1975-01-14T08:33:37,7514,2421.677841
1972-10-26T12:23:28,8510,3102.024857
2008-03-12T18:02:16,9634,8605.747956
```

A general workflow for realizing reduce side join operations including MapReduce include the following steps:

1. Mapper reads input data which are to be combined based on common column or join key.
2. The mapper processes the input and adds a tag to the input to distinguish the input belonging from different sources or data sets or databases.
3. The mapper outputs intermediate key-value pair where the key is nothing but the join key.
4. After the sorting and shuffling phase, a key and the list of values is generated for the reducer.
5. Finally, the reducer joins the values present in the list with the key to give the final aggregated output.

Taking into consideration the above workflow, you are provided with the instructions execution file and individual source code(s) that are responsible for:

1. **CustomerMapper.java:** Mapper file to parse the customer dataset for fetching the CustomerID and state information. A specific tag of "customer~" is added to the value part for each tuple to identify the records from the same.
2. **SalesMapper.java:** Mapper file for reading the tuples from "sales" dataset for fetching CustomerID and SalesPrice information. A specific tag of "sales~" is added to the value part for each tuple to identify the records from the same.
3. **CalculateTotalSales.java:** MapReduce Reduce-Side join Driver code for finding the total sales by states.
4. **CustomerSalesJoinReducer.java:** Reducer file for joining the customers and sales datasets based upon CustomerID, to output the State and SalesPrice without aggregation. This helps in obtaining the sale aggregation based on the CustomerID attribute.
5. **StateSalesMapper.java:** Mapper file to fetch the data from the intermediate directory (obtained after running ***CustomerSalesJoinReducer.java***) in hadoop.
6. **SalesAggregatorReducer.java:** Final reducer file to calculate aggregate sales price for each state. This helps in aggregating sales by State.