Incremental Dataload to Datamap and MV datamap enhancement

Author: Ravindra Pesala
Version: 1.0
Date: 13/02/2019

Background

There are two types of datamap in Carbondata. 1. Index datamap and 2. OLAP datamap. Index datamap is used to store additional indexes of data for better pruning. And OLAP datamap is used to store aggregated data to give faster query performance on OLAP queries.
The current system cannot support incremental loading on OLAP datamaps. It means after creating on OLAP datamap it needs to reload the whole datamap for any newly added segments. This will slow down the loading.
It is expected to load only the new data to OLAP datamaps instead of loading whole data again.

Incremental Loading on Datamaps

On each data load, a new LoadMetaDataEntry will be added in TableStatus file. For MV incremental data loading, each LoadMetaDataEntry of datamap TableStatus file, will maintain a new entry, “ExtraInfo: which holds the information about main tables and its list of segment id’s loaded for corresponding new datamap load”.

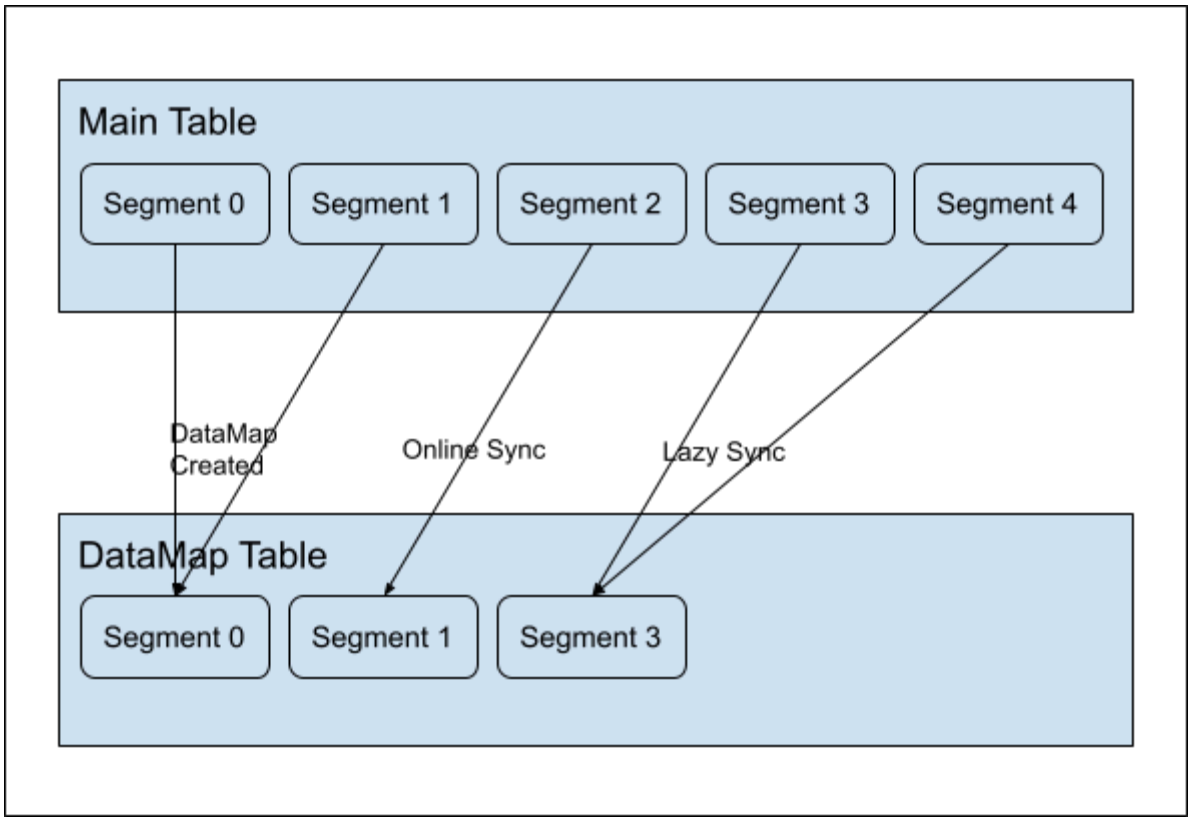
New Entry in LoadMetadataEntry	Name of the datamap
ExtraInfo	Holds the information about main tables and its list of segment id’s loaded for corresponding new datamap load

Example:

[{"timestamp":"1556451136878","loadStatus":"Success","loadName":"0","partitionCount":"0","isDeleted":"FALSE","dataSize":"643","indexSize":"419","updateDeltaEndTimestamp":"","updateDeltaStartTimestamp":"","updateStatusFileName":"","loadStartTime":"1556451136264","visibility":"true","fileFormat":"COLUMNAR_V3","segmentFile":"0_1556451136264.segment",**"extraInfo":"{\"main_table\":[\"0\",\"1\"]}"**}]

ExtraInfo says, segment0 of datamap is made from segment 0 and 1 of its parent table.

Datamap with Single table synchronization



Example for datamap with single table relation

Main Table

```
Create table main(id int, name string, dept string, age int, salary double) stored by ‘carbondata’
```

Datamap

```
Create datamap single_relation using ‘mv’ as select dept, sum(salary) from main
```

Olap datamaps created with single table relation will be synchronized with the main table in the following way using the above picture.

Create datamap

Datamap is created on the main table and it already has 2 loaded segments 0 and 1. So during datamap creation, it synchronizes both the segments with datamap table segment 0. This

synchronization happens at the time of create datamap only if it is non-lazy datamap. If it is lazy datamap then synchronization happens only if the user explicitly calls rebuild.

After the load is successful to datamap table, Tablestatus file of datamap table will maintain segment mapping with the following information

[{"timestamp":"1556451136878","loadStatus":"Success","loadName":"0","partitionCount":"0","isDeleted":"FALSE","dataSize":"643","indexSize":"419","updateDeltaEndTimestamp":"","updateDeltaStartTimestamp":"","updateStatusFileName":"","loadStartTime":"1556451136264","visibility":"true","fileFormat":"COLUMNAR_V3","segmentFile":"0_1556451136264.segment","**extraInfo":"{\"main_table\":[\"0\",\"1\"]}"**}]

The above information tells that the current datamap is synchronized till segment 1 with the main table. And says that the datamap segment0 is made from main table's segment 0 and 1.

Loading data to Main table

There is a new load happens on the main table called segment 2. After load finishes on the main table datamap trigger load on the datamap. Now datamap reads the each loadMetadata success entry from table status file for segment mapping and finds out that there is a synchronization mismatch between the main and datamap table and starts loading of main table segment 2 to the datamap table segment 1.

After the load is successful to datamap table it updates the tablestatus file with the following information

[{"timestamp":"1556451136878","loadStatus":"Success",**"loadName":"0"**,"partitionCount":"0","isDeleted":"FALSE","dataSize":"643","indexSize":"419","updateDeltaEndTimestamp":"","updateDeltaStartTimestamp":"","updateStatusFileName":"","loadStartTime":"1556451136264","visibility":"true","fileFormat":"COLUMNAR_V3","segmentFile":"0_1556451136264.segment","**extraInfo":"{\"main_table\":[\"0\",\"1\"]}"**},{"timestamp":"1556451526077","loadStatus":"Success",**"loadName":"1"**,"partitionCount":"0","isDeleted":"FALSE","dataSize":"637","indexSize":"419","updateDeltaEndTimestamp":"","updateDeltaStartTimestamp":"","updateStatusFileName":"","loadStartTime":"1556451525516","visibility":"true","fileFormat":"COLUMNAR_V3","segmentFile":"1_1556451525516.segment","**extraInfo":"{\"main_table\":[\"2\"]}"**}]

The above information tells that the current datamap is synchronized till segment 2 with the main table and segemnt1 of datamap is made from segment2 of main table.

Loading fail on datamap table or lazy loading on datamap table

There is another load happens on main table and creates the segment 3. But during the datamap synchronization with main table it got failed. So instead of failing the main table load we can just disable the datamap so that queries never uses the disabled datamap for fetching

results.

After main table load success but datamap load fails for some reason then it updates the datamapstatus file with the following information

datamap1	DISABLED

And the TableStatus file will be as below

[{"timestamp":"1556451136878","loadStatus":"Success",**"loadName":"0"**,"partitionCount":"0","isDeleted":"FALSE","dataSize":"643","indexSize":"419","updateDeltaEndTimestamp":"","updateDeltaStartTimestamp":"","updateStatusFileName":"","loadStartTime":"1556451136264","visibility":"true","fileFormat":"COLUMNAR_V3","segmentFile":"0_1556451136264.segment",**"extraInfo":"{\"main_table\":[\"0\",\"1\"]}"**},{"timestamp":"1556451526077","loadStatus":"Success",**"loadName":"1"**,"partitionCount":"0","isDeleted":"FALSE","dataSize":"637","indexSize":"419","updateDeltaEndTimestamp":"","updateDeltaStartTimestamp":"","updateStatusFileName":"","loadStartTime":"1556451525516","visibility":"true","fileFormat":"COLUMNAR_V3","segmentFile":"1_1556451525516.segment",**"extraInfo":"{\"main_table\":[\"2\"]}"**}]

The above information tells that the current datamap is synchronized only till segment 2 with the main table and it is disabled.

During the next load on main table or user explicit call of rebuild datamap can trigger the datamap load and then starts synchronizing both main table segment 3 and 4 will be loaded to datamap table segment 3. Then datamap status will be as below

datamap1	ENABLED

And tableStatus file will be as below

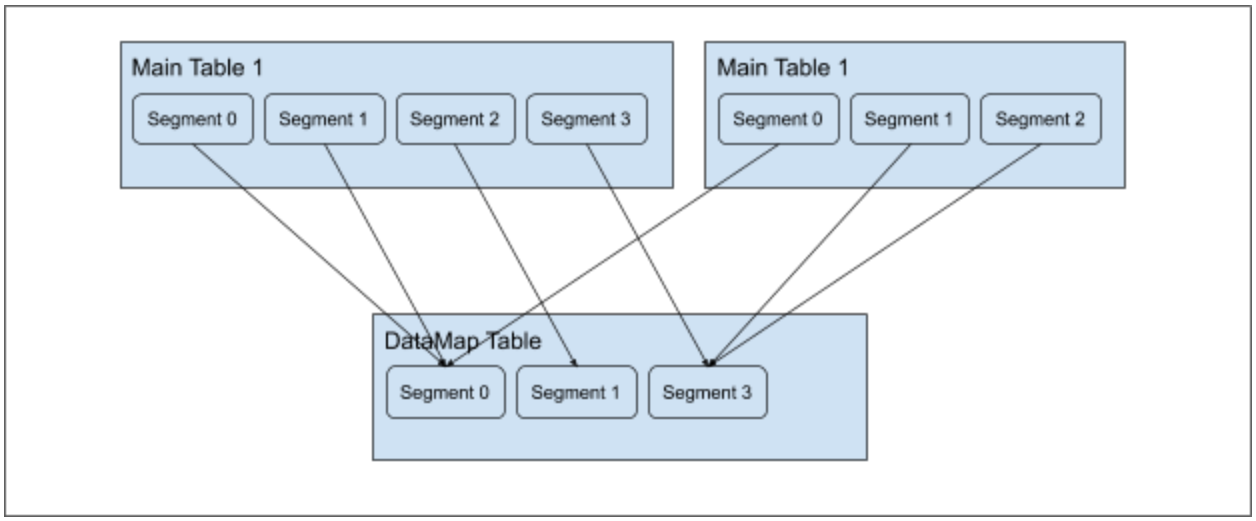
[{"timestamp":"1556451136878","loadStatus":"Success",**"loadName":"0"**,"partitionCount":"0","isDeleted":"FALSE","dataSize":"643","indexSize":"419","updateDeltaEndTimestamp":"","updateDeltaStartTimestamp":"","updateStatusFileName":"","loadStartTime":"1556451136264","visibility":"true","fileFormat":"COLUMNAR_V3","segmentFile":"0_1556451136264.segment",**"extraInfo":"{\"main_table\":[\"0\",\"1\"]}"**},{"timestamp":"1556451526077","loadStatus":"Success",**"loadName":"1"**,"partitionCount":"0","isDeleted":"FALSE","dataSize":"637","indexSize":"419","updateDeltaEndTimestamp":"","updateDeltaStartTimestamp":"","updateStatusFileName":"","loadStartTime":"1556451525516","visibility":"true","fileFormat":"COLUMNAR_V3","segmentFile":"1_1556451525516.segment",**"extraInfo":"{\"main_table\":[\"2\"]}"**}],
[{"timestamp":"1556451136858","loadStatus":"Success",**"loadName":"2"**,"partitionCount":"0","isDeleted":"FALSE","dataSize":"643","indexSize":"419","updateDeltaEndTimestamp":"","updateDeltaStartTimestamp":"","updateStatusFileName":"","loadStartTime":"1556451136269","visibi

lity":"true","fileFormat":"COLUMNAR_V3","segmentFile":"2_1556451136264.segment","extrainfo":"{\"main_table\":[\"3\",\"4\"]}"}

The above information tells that the current datamap is synchronized only till segment 4 with the main table and extraInfo says segment 2 of datamap table is made from segment 3 and 4 of main table.

Note: Lazy datamaps will not be synchronized during the main table load, it only be synchronized with explicit command rebuild datamap.

Datamap with Multi-table synchronization



It would be similar like the single table but it maintains the information of more tables. Here I will not explain more about how datamapstatus and TableStatus is updated as more information is already captured for single table datamap in the above section.

Here I will explain more about join cases how datamap table should be synched with the main table. These are general star schema OLAP scenarios so I will explain with one example for better understanding.

DataMap with inner Join

MV Datamap for the inner join query

Create datamap innerjoin using mv as Select p.product, p.amount, s.quantity from products p, sales s where p.product=s.product

For the above datamap parent tables are products and sales tables.

Products

product	Amount
Mobile	2000
Laptop	3000
Kettle	70
Washing machine	1000

Sales

product	Quantity
Mobile	1
Laptop	10
Chocolates	200
Biscuits	800

In the above case when we apply inner join on column product we only get the common items between two tables. So During MV load the datamap will be loaded as follows.

MV DataMap after loading.

product	Amount	Quantity
Mobile	2000	1
Laptop	3000	10

Incremental Update of MV inner join DataMap

If new data is added to the parent tables For example as below

Products

product	Amount
Mobile	2000
Laptop	3000
Kettle	70

Washing machine	1000
Biscuits	10
Cheese	100

In the above products table, last 2 rows are newly added so we need to refresh the MV datamap with new data. So only we just need to join the newly added segment data to the whole right table Sales.

The new incremental DataMap output as follows.

product	Amount	Quantity
Mobile	2000	1
Laptop	3000	10
Biscuits	10	800

In the same way, if the sales table is updated with new data then only the delta data of the sales table would be joined to the whole table of products.

DataMap with Left Outer Join

MV Datamap for the left outer join query

```
Create datamap leftouterjoin using mv as Select p.product, p.amount, s.quantity from products p left join sales s on p.product=s.product
```

In the same tables which are mentioned above inner join section if we apply left join on column product, we get the common items between two tables and along with non common data of products table. So During MV load the datamap will be loaded as follows.

MV DataMap after loading.

product	Amount	Quantity
Mobile	2000	1
Laptop	3000	10
Kettle	70	null
Washing	1000	null

machine		

Incremental Update of MV inner join DataMap

If new data is added to the parent tables For example as below

Products

product	Amount
Mobile	2000
Laptop	3000
Kettle	70
Washing machine	1000
Biscuits	10
Cheese	100

In the above products table, last 2 rows are newly added so we need to refresh the MV datamap with new data. So only need to join the newly added segment data to the whole right table Sales.
The new incremental DataMap output as follows.

product	Amount	Quantity
Mobile	2000	1
Laptop	3000	10
Kettle	70	null
Washing machine	1000	null
Biscuits	10	800
Cheese	100	null

In the same way if the sales table is updated with new data

Sales

product	Quantity
Mobile	1
Laptop	10
Chocolates	200

Biscuits	800
Kettle	4
Butter	80

In the above sales last 2 rows are added so we should refresh the datamap .

```
UPDATE table mvdatamap set (quantity) = (select quantity from sales<delta> join mvdatamap where mvdatamap.product=sales.product ) where quantity = null
```

product	Amount	Quantity
Mobile	2000	1
Laptop	3000	10
Kettle	70	4
Washing machine	1000	null
Biscuits	10	800
Cheese	100	null

In the above MV Datamap Kettle value is updated with new value.

Improvement Steps:

1. Bucketing on join keys of parent tables and mvdatamap table as well. It will improve the loading performance because of shuffling avoided.
2. New partition column can be added on mvdatamap and while loading we just provide the value as isNull(some random right table column). So that all null values after loading will be loaded to the null partition and query during update on the mvdatamap will be faster.

Compaction on Datamap

Compaction request on main tables will also request compaction on corresponfing datamaps.
It is better to have a seperate DDL for compacting datamaps to decouple from main tables. The command could be as follows for manual compaction.

```
Alter datamap datamapname compact ‘minor’
```

Auto-compaction can also be enabled on datamaps just like how it is enabled for main tables.

Limitations

1. If there is any group by conditions on left outer join we cannot do delta loading to it, so we should do a full refresh.
2. If the join column does not present in datamap table we cannot do delta loading to it, so we should do a full refresh.
3. Alter operations are not supported.
4. Delete segments are not supported.
5. Partition on MV and Drop partitions are not supported.