Auto-compaction at Carbon-Data connecter

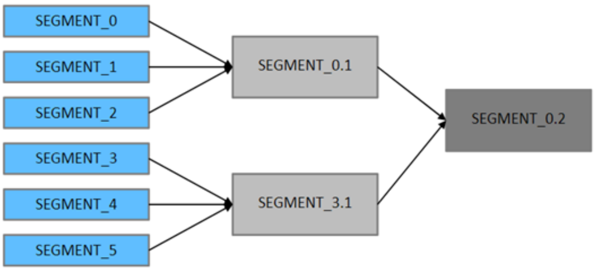
Background:

Openlookeng integration with Carbondata not supporting auto compaction of tables. This becomes the limitation of openlookeng since other processing engine like spark provides the same.

Objective:

System automatically initiates task in Crabondata connecter scans through all Db and tables and selects the carbontables for minor compaction. Gives the selected list of table to Engine task that do compaction.

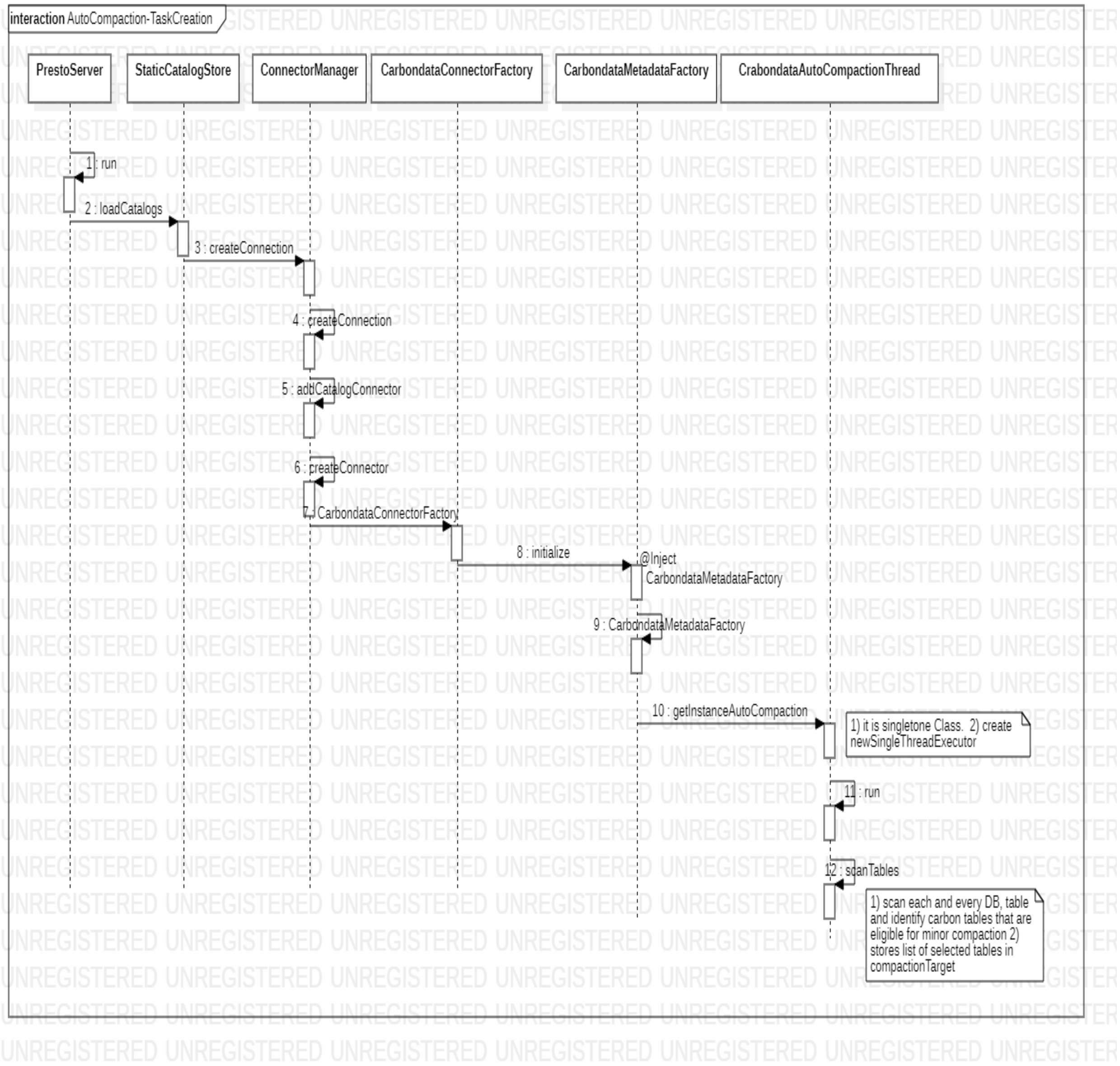
Minor compaction: Merges multiple segments to a single segment.



High Level Design

Design for Creating Auto-Compaction & scanning task at Carbondata connecter

1) At time of carbondata connecter bootstrap, it creates singletone class for CrabondataAutoCompactionThread.
2) Create newSingleThreadExecutor to scan tables
3) The created task will scan all DB and tables. Identify carbondata tables eligible for miner compaction and store them in list.

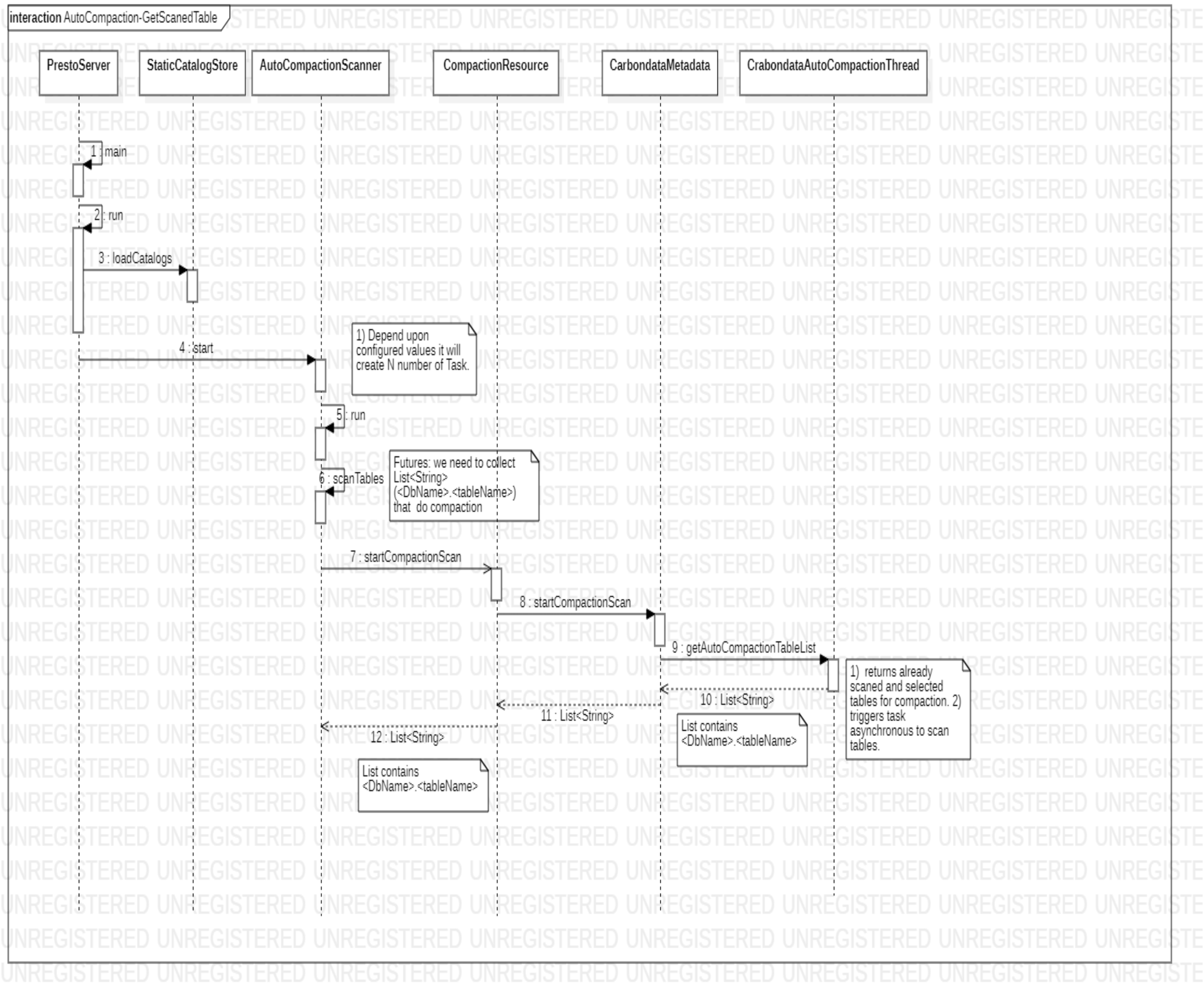

interaction AutoCompaction-TaskCreation
PrestoServer
StaticCatalogStore
ConnectorManager
CarbondataConnectorFactory
CarbondataMetadataFactory
CrabondataAutoCompactionThread
1 : run
2 : loadCatalogs
3 : createConnection
4 : createConnection
5 : addCatalogConnector
6 : createConnector
7 : CarbondataConnectorFactory
8 : initialize
@Inject CarbondataMetadataFactory
9 : CarbondataMetadataFactory
10 : getInstanceAutoCompaction
11 : run
12 : scanTables

1) it is singletone Class. 2) create newSingleThreadExecutor

1) scan each and every DB, table and identify carbon tables that are eligible for minor compaction 2) stores list of selected tables in compactionTarget

Design for Getting Engine getting table info from Carbondata connecter

1) Engine will span task depend up on auto-compaction.scan.threads,
2) Engine will uses REST to interact with catbondata connecter.
3) When carbondata connecter receives request, it will give the scanned tables information. And also asynchronously trigger task to scan tables.
4) getAutoCompactionTableList will returns list of tables in form of List<String>. Sting contains DBName.TableName. It also triggers task scanTables.


interaction AutoCompaction-GetScanedTable
PrestoServer
StaticCatalogStore
AutoCompactionScanner
CompactionResource
CarbondataMetadata
CrabondataAutoCompactionThread
1 : main
2 : run
3 : loadCatalogs
4 : start
1) Depend upon configured values it will create N number of Task.
5 : run
6 : scanTables
Futures: we need to collect List<String> (<DbName>.<tableName>) that do compaction
7 : startCompactionScan
8 : startCompactionScan
9 : getAutoCompactionTableList
1) returns already scaned and selected tables for compaction. 2) triggers task asynchronous to scan tables.
10 : List<String>
11 : List<String>
12 : List<String>
List contains <DbName>.<tableName>
List contains <DbName>.<tableName>

In Phase 1 design: Each time we will scan all DB and tables
In Phase 2 design: First time will scan all DB and Table and Next time we will scan only the table which get insert, update, delete. For that we will maintain HashMap for the tables which get Insert, update and delete. Scan though only the HashMap. With this design we increase performance and avoid visiting unwanted tables.