# Merging small file size splits

## Background:

For hive data source, driver is created per split, so more splits means more parallelism, this is working without any issue if we have good file size ,however , there is a case which the file size is far less than hive split size(hive.max-initial-split-size, hive.max-split-size), thus number of splits is determined by number of small files.

To process a large number of small splits in parallel, it definitely introduces overhead of CPU context switch, causing performance issues when doing the data shuffling and resulting in low CPU/memory usage.

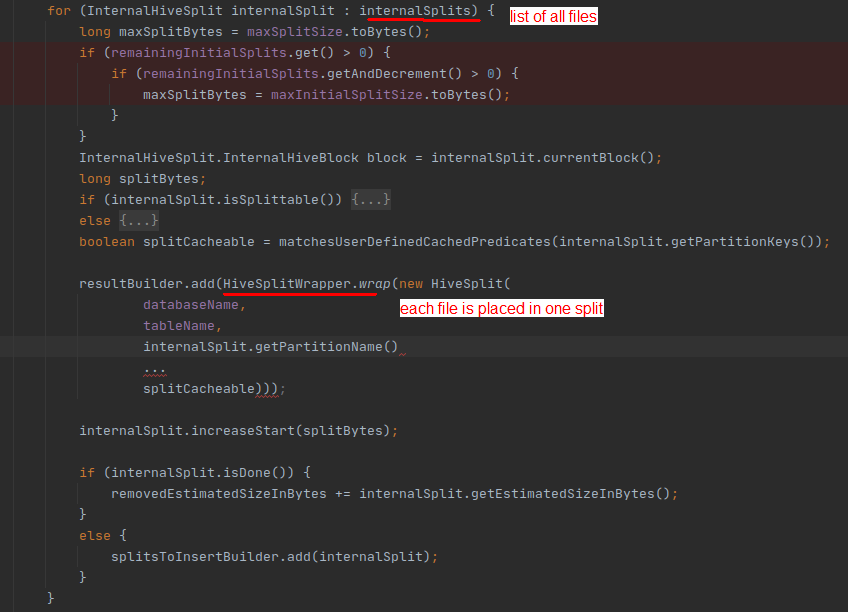
Based on our testing we are seeing huge performance impact in high concurrency environment since task slot is occupied by those small split

## Objective:

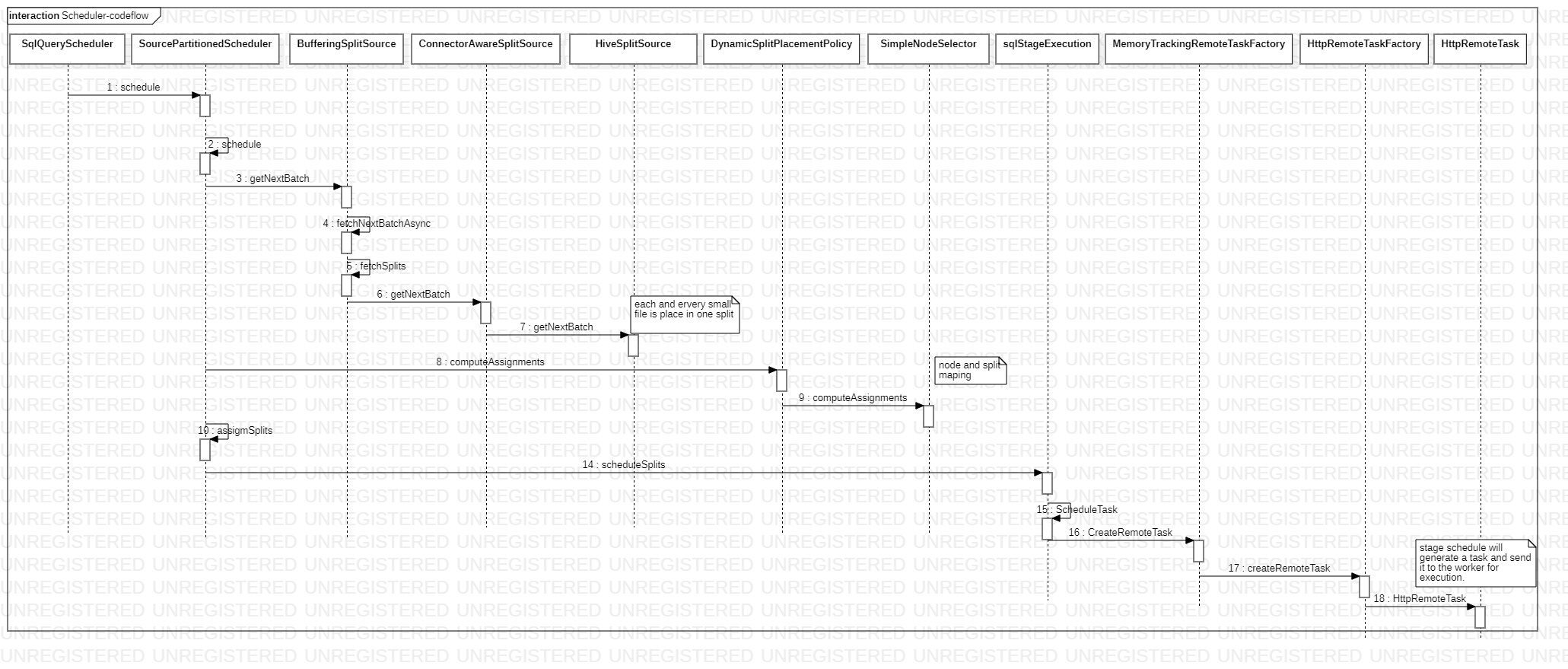
To reduce the overall CPU context switch and improve overall performance by merging small splits to bigger split without crossing the boundary(hive.max-initial-split-size / hive.max-split-size)

## High Level Current Design:

1. SourcePartitionedScheduler expects list of ConnectorSplits. i.e. It asks HiveSplitSource to generate splits to schedule.
2. For each source file, if file’s size is less than MaxSplitBytes, then a single HiveSplit is created for entire file.
3. If size is more than MaxSplitBytes, and source format allows splitting then multiple HiveSplits will be created for same file.
4. HiveSplitWrapper implements ConnectorSplit and will wrap each HiveSplit separately in this case.

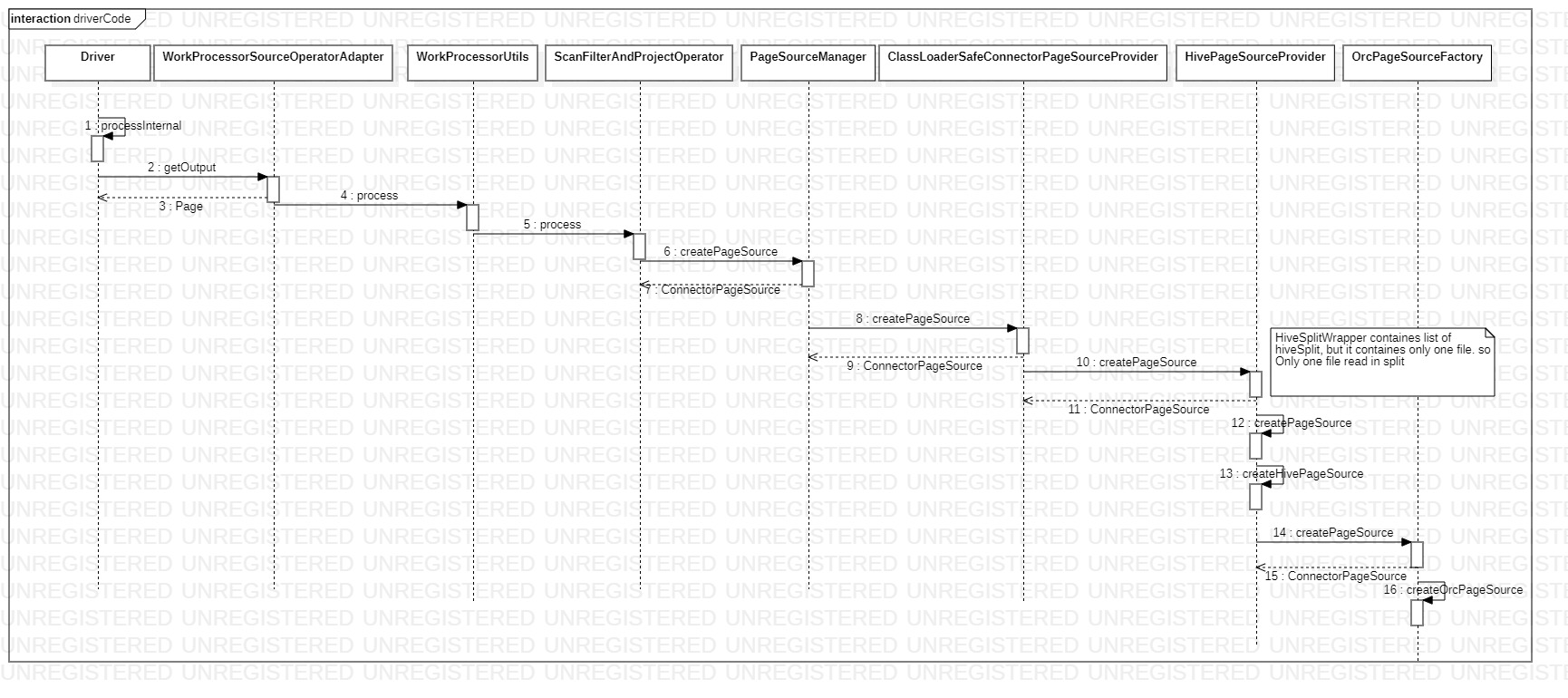


1. Splits and node mapping is done
2. Splits are assigned to task
3. Stage schedule will generate a task and send it to the worker for execution. Using http rest request.



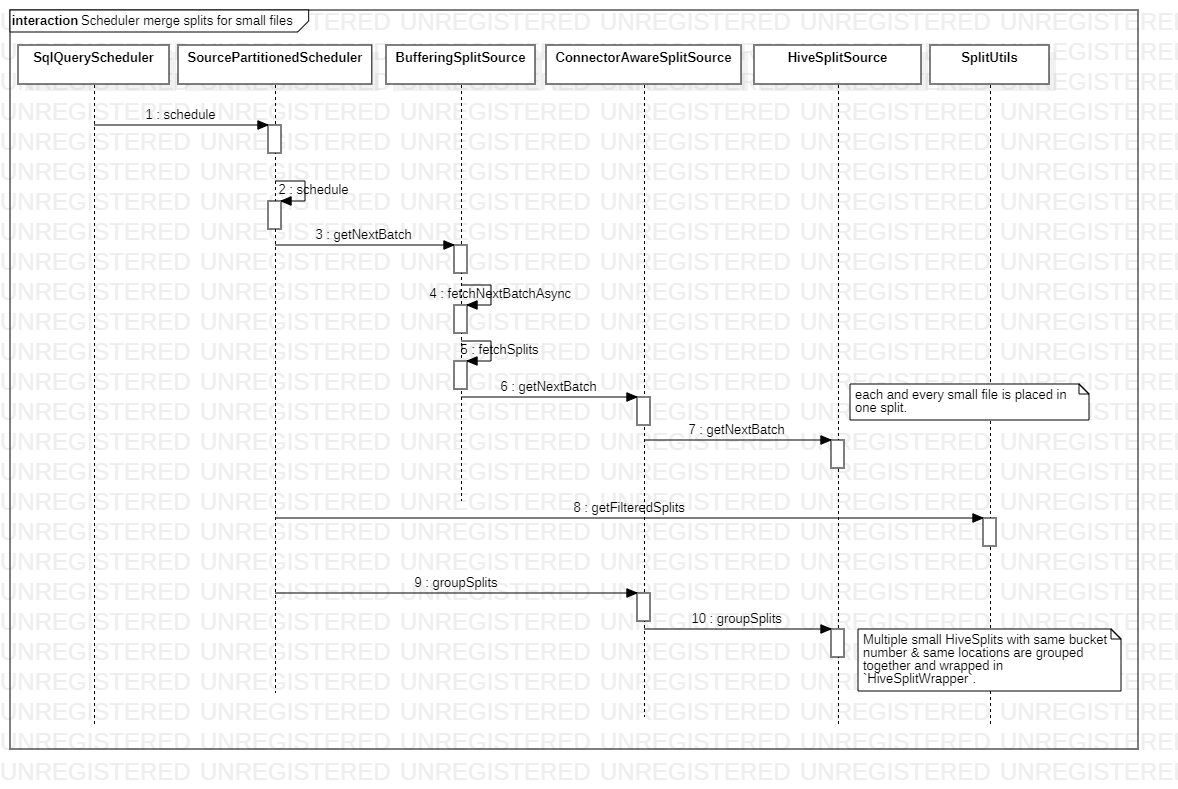
### Driver code flow

* 1. Every split is executed separately.
  2. Each Split creates a single `ConnectorPageSource` to read data from file



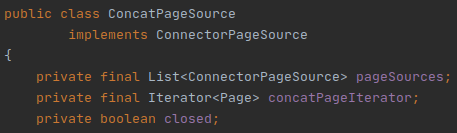
## High level design:

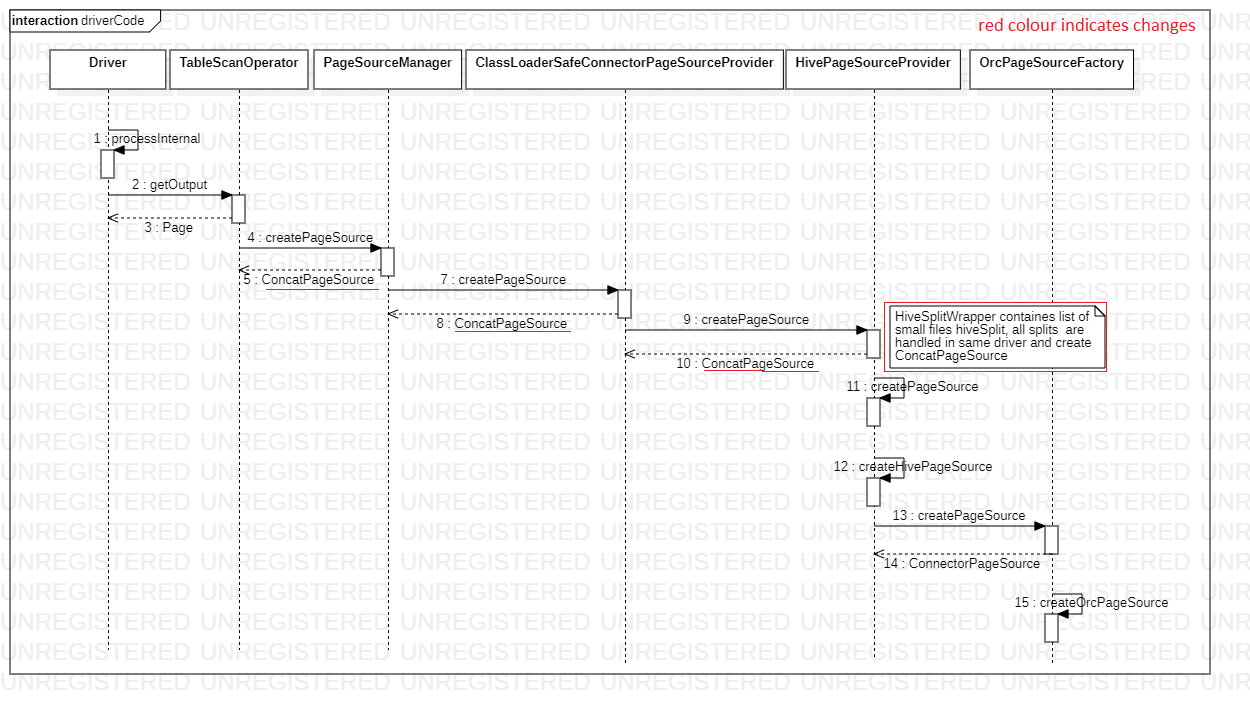
1. `SourcePartitionedScheduler` gets the list of `SplitBatch` which contains list for `Splits` from HiveSplitSource.
2. ` SplitUtils.getFilteredSplit` Based on hieuristic index of data certain splits might be avoided.
3. After filter splits, `HiveSplitSource.groupSplits` will group together Multiple small HiveSplits with same bucket number, location & non-cacheable (!HiveSplit.isCacheable) and wrapped in `HiveSplitWrapper`. Total size of group does not exceed maxSplitSize(hive.max-split-size/ hive.max-initial-split-size).
   1. Partitioned tables can be cached before start reading using SQL ddl command CACHE TABLE <table name >. For this, table's files will be cached in the workers and map of split->worker will be maintained in coordinator. For further reads splits will be scheduled in the same worker wherever the file is cached. **Caching is done based on the file name**. So after grouping we cannot ensure that all grouped files are cached in the same worker.
   2. NodePartitioningProvider expects one split to return only one bucket number to provide partitioning of data if required. So in this case, merged split also should return same bucket number for entire data. Hence small files with same bucket number only can be merged to bigger split.
4. Grouping is done based on number of files, file size and location.
   1. In grouping files are filtered based on matching first 3 location.
   2. Number of files should not cross MinValue(max-small-splits-grouped, Total number of files with location / Number of selected locations).
   3. Total size should not cross maxSplitSize.
5. This wrapped `HiveSplitWrapper` is considered as a single Split by scheduler since it already implements ConnectorSplit spi.
6. Number of small HiveSplits grouped in to one `HiveSplitWrapper` should not exceed hive.max-small-splits-grouped
7. If HiveSplit size is already bigger, then there is no grouping & kept as it is.



### Driver Code flow

* 1. For each HiveSplitWrapper a separate driver is created.
  2. In case of small files, HiveSplitWrapper now contains list of HiveSplits.
  3. HivePageSourceProvider creates separate ConnectorPageSources for each of these HiveSplits and combine (concat/mergesort) them together to represent as Single and bigger ConnectorPageSource.
  4. If table does not have any ‘sort\_by’ columns, then order of reading multiple raw files does not affect overall result. In this case, can use `ConcatPageSource` which concats list of page sources one-after-other other and iterates through all of them.
  5. If table does have ‘sort\_by’ columns, then combined page iterator should also give sorted results across small files. Since it’s expected that individual files have sorted data, can simply do merge sort on all page sources and return MergeSortedPagesource.
  6. More optimizations can be added in case of Transactional table by avoiding multiple references to delete delta files from each page-source separately (this can be taken as improvements).



Code flow change in case of table scan

### Configuration

In hive.properties below configuration need to be added

|  |  |  |
| --- | --- | --- |
| Name | Description | Default Value |
| hive.max-small-splits-grouped | Max number of small splits can be grouped to one split. | 10 |