# Pool based compaction queues.

This document outlines the requirements, concept and implications of the new Pool based compaction queues.

## Problem statement:

The single compaction queue with the workers picking up the compaction requests in the sequence of their arrival just doesn't fit into a multi-tenant environment. Like compaction requests from a use-case into a yarn queue which is shared and/or has limited resources blocking compaction requests where workers would find plenty of resources in other dedicated queues. Compaction requests should be able to be put into different *compaction pools* where each of these pools has a set of workers assigned to. As a result high priority/urgent requests can use a dedicated compaction pool to avoid being blocked by other (long running) compaction requests.

## Concept:

#### Automatic pool assignment

Tables, partitions could be assigned to compaction pools through the hive.compactor.worker.pool={poolname} property.

* Database level: If the property is set on Database level, it applies to all tables and partitions
* Table/partition level: The pool also can be defined on a table/partition level, in this case it overrides the DB level value (if set).

If any of the above is set, it is used by the initiator during the creation of the compaction requests.

#### Manual pool assignment

The compaction request also can be assigned to a pool by using the ALTER TABLE COMPACT command (E.g. manual compaction). If provided, this value overrides the hive.compactor.worker.pool value on any level.

#### Implicit pool assignment

Tables, partitions and manual compaction requests without specified pool name are implicitly assigned to the default pool.

#### Pool timeout

If a compaction request is not processed by any of the labeled pools within a predefined period, it falls back to the default pool. The timeout can be set through the

hive.compactor.worker.pool.timeout configuration property.

This approach is to cover the following use cases:

1. The request is accidentally assigned to a non-existent pool. (E.g.: Typo in the pool name when issuing an ALTER TABLE COMPACT command.
2. Typo in the DB or table property used by the initiator to create compaction requests.
3. A HS2 instance is stopped due to a scaledown or schedule, and its pending compaction requests still should be processed.

The timeout can be disabled by setting the configuration property to 0.

The default pool worker should log a warning message when processing timed out compaction requests.

#### Labeled worker pools

The labeled worker pools can be defined through the

hive.compactor.worker.{poolname}.threads={thread\_count}

configuration setting.

#### Default pool

The default pool is responsible for processing the non-labeled and fallen back compaction requests. On a cluster-wide level, at least 1 worker thread on at least one node should be assigned to the default pool, otherwise non-labeled compaction requests will never be processed.

#### Worker allocation

The already existing hive.compactor.worker.threads holds the maximum number of worker threads. The worker allocation happens as follows:

1. Labeled pools are initialized in a sequential manner with random ordering.
2. Each pool decreases the number of available workers by its own worker count.
3. If the number of assignable workers is less than the configured one, the pool size will be adjusted, and a warning message will be logged.
4. If the number of assignable workers is 0, the pool won’t be initialized, and a warning message will be displayed/logged.
5. All remaining workers not used up by the labeled pools, are assigned to the default pool.
6. If no workers remain for the default pool, a warning message will be logged.

While the possibility to set the default pool to 0 has some risk (See the [default pool](#_xvbvylhnk2m2) section), there is a good reason behind allowing it: Imagine that in a Data HUB environment some HS2 instances are configured to process only their own labeled compaction requests, and a dedicated HS2 instance is set up to take care of the default (non-labeled) pool requests. If the default pool cannot be fully disabled, this separation cannot be done, some of the default pool items will be processed by the HS2 instances dedicated for other compaction requests.

## Implications

#### Compaction request (DB schema) changes

A new optional field must be added to the CompactionInfo object, and a new nullable column must be added to the COMPACTION\_QUEUE and COMPLETED\_COMPACTIONS tables to store the assigned pool’s name.

#### Initiator changes

The Initiator must respect the value of the hive.compactor.worker.pool DB and/or table property and set it in the compaction request.

#### Cleaner changes

Cleaner should be changed in a way that a single cleaner loop picks up one (or zero if there are no elements for a particular pool) request for each labeled pool, and for the default pool. This will guarantee that eventually every request will be picked up by the cleaner, regardless of the pool in which it exists.

#### ALTER TABLE COMPACT changes

The ALTER TABLE COMPACT command must accept the poolname, and set it to the compaction request if provided.

#### SHOW COMPACTIONS changes

Show compactions should show the assigned pool for each compaction request, and should be able to limit the results to a particular pool.

#### Worker pool initialization

A new initialization process must take place, which is responsible for collecting all the defined pools in the configuration, and creating these pools respectively. After the initialization, the entire pool configuration should be logged.

Since the pool defining configuration keys are dynamic (remember: hive.compactor.worker.{poolname}.threads), the configuration values must be collected by matching all the entries against a regular expression. Since this functionality does not exist yet, it must be implemented.

#### Observability

A new metric should be created with the number of pending compaction requests per pool. This could be used to detect misconfigured/boken pools where the entries are accumulating over time.

## Open points

#### Initiator and/or cleaner pool?

In some cases a dedicated compaction queue may provide a partial solution only. Since the initiator and the cleaner are not using pools, prioritization is only partial. After some discussion with [Janos Kovacs](mailto:jkovacs@cloudera.com) we decided that while the pooling concept in the initiator could be useful, it should have its own task and design doc. It can be implemented independently from the worker pooling. Regarding the global min txn id limitation in cleaner, there is a separate task as well: <https://jira.cloudera.com/browse/CDPD-24047>.