

Apache Hive workload management commands

Date of Publish: 2019-12-17



Contents

| | |
|---|-----------|
| Workload management command summary..... | 4 |
| ALTER MAPPING..... | 4 |
| ALTER POOL..... | 5 |
| ALTER RESOURCE PLAN..... | 6 |
| ALTER TRIGGER..... | 7 |
| CREATE MAPPING..... | 7 |
| CREATE POOL..... | 9 |
| CREATE RESOURCE PLAN..... | 10 |
| CREATE TRIGGER..... | 10 |
| DISABLE WORKLOAD MANAGEMENT..... | 12 |
| DROP MAPPING..... | 12 |
| DROP POOL..... | 13 |
| DROP RESOURCE PLAN..... | 13 |
| DROP TRIGGER..... | 13 |
| REPLACE RESOURCE PLAN WITH..... | 14 |
| REPLACE ACTIVE RESOURCE PLAN WITH..... | 14 |

| | |
|---------------------------------------|-----------|
| SHOW RESOURCE PLAN..... | 14 |
| SHOW RESOURCE PLANS..... | 15 |
| Workload trigger counters..... | 15 |

Workload management command summary

To manage your workload, you can execute Hive commands that resemble the familiar ALTER, CREATE, DROP, and SHOW statements.

Table 1: Command Summary

| Name | Brief Description | Simple Example |
|-----------------------------|--|---|
| ALTER MAPPING | Changes the routing of queries to a resource pool. | ALTER APPLICATION MAPPING 'myapp' IN myplan UNMANAGED; |
| ALTER POOL | Modifies query pool properties, adds triggers, and removes triggers. | ALTER POOL myplan.mypool SET QUERY_PARALLELISM=10; |
| ALTER RESOURCE PLAN | Enables, disables, activates, validates, or changes a plan. | ALTER RESOURCE PLAN myplan RENAME TO yourplan; |
| ALTER TRIGGER | Adds a trigger to or removes a trigger from a resource pool. | ALTER TRIGGER myplan.mytrigger DROP FROM POOL mypool; |
| CREATE MAPPING | Routes queries to a resource pool. | CREATE APPLICATION MAPPING 'myapp' IN myplan TO mypool WITH ORDER 2; |
| CREATE POOL | Creates and adds a query pool for a resource plan. | CREATE POOL myplan.mypool WITH ALLOC_FRACTION=75 AND QUERY_PARALLELISM=5; |
| CREATE RESOURCE PLAN | Creates a resource plan | CREATE RESOURCE PLAN myplan; |
| CREATE TRIGGER | Establishes and adds a trigger to a resource plan. | CREATE TRIGGER myplan.mytrigger WHEN ELAPSED_TIME > 30min DO KILL; |
| DISABLE WORKLOAD MANAGEMENT | Deactivates the existing resource plan. | DISABLE WORKLOAD MANAGEMENT; |
| DROP MAPPING | Removes a mapping from a resource plan. | DROP APPLICATION MAPPING 'myapp' IN myplan; |
| DROP POOL | Removes a query pool from a resource plan. | DROP POOL myplan.mypool; |
| DROP RESOURCE PLAN | Deletes a resource plan. | DROP RESOURCE PLAN myplan; |
| DROP TRIGGER | Deletes a trigger from a resource plan. | DROP myplan.mytrigger; |
| REPLACE RESOURCE PLAN WITH | Replaces the contents of one resource plan with the contents of another. | REPLACE ACTIVE RESOURCE PLAN WITH myplan; |
| SHOW RESOURCE PLAN | Lists plan contents. | SHOW RESOURCE PLAN myplan; |
| SHOW RESOURCE PLANS | Lists all resource plans. | SHOW RESOURCE PLANS; |

ALTER MAPPING

You reroute queries to a query pool using this command.

Syntax

```
ALTER { USER | GROUP | APPLICATION } MAPPING 'entity_name' IN plan_name { TO
  pool_path | UNMANAGED } [ WITH ORDER num ]
```

- entity_name
 - The name of the user specified in the connection string

- The name of the group of users specified in the connection string
- The name of the application specified in the connection string or through the SetClientInfo JDBC API with the ApplicationName key
- `plan_name`
The name of a resource plan
- `pool_path`
The name of a query pool, which can be a hierarchy of pools in dot notation
- `num`
An integer that establishes the priorities for the routing rules. The order of precedence is lower to higher. If you set multiple rules (a user and a group rule, or rules for multiple groups) the lowest ordered rule takes precedence. If ordering is not specified or is the same for routing rules, user rules take precedence over application rules, which take precedence over group rules. The order of group rules with the same priority is undefined.

Example

```
ALTER GROUP MAPPING 'marketing' IN rp1 IN myplan TO pool1 WITH ORDER 1;
```

The marketing group in the rp1 pool of myplan is mapped to pool with top routing priority.

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

ALTER POOL

You can use ALTER POOL to modify query pool properties, add triggers, and remove triggers.

Syntax

```
ALTER POOL plan_name.path [ SET {property=value, ... } | UNSET
  { property, ... } ]
ALTER POOL plan_name.path [ ADD | DROP ] TRIGGER name
```

| | |
|-----------------------|---|
| plan_name.path | The resource plan name and hierarchy of query pool names, in dot notation |
| plan_name.path | The resource plan name and hierarchy of query pool names in dot notation |
| property | The name of a query pool property to remove or to assign a value |
| value | The property setting |
| name | The name of a trigger |

Query pool properties

The ALTER POOL command supports the following properties:

- `ALLOC_FRACTION`
The proportion of the Hive LLAP cluster allocated to a pool. A decimal value.

- **QUERY_PARALLELISM**
The maximum parallel queries allowed in a pool. An integer value.
- **SCHEDULING_POLICY**
Determines the resource allocation of queries in a pool. Valid scheduling policy values are: 'default'; 'fair', which allocates an equal share of resources on the cluster to each query; or 'fcfs', which allocates all resources to the query started earliest.

Example

```
ALTER POOL rp1.pool1 ADD TRIGGER slow_query;
```

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

ALTER RESOURCE PLAN

You alter the resource plan status, validate the plan, or change plan properties using this command.

ALTER RESOURCE PLAN syntax

```
ALTER RESOURCE PLAN name [ DISABLE | ENABLE | ACTIVATE [WITH REPLACE]
ALTER RESOURCE PLAN name [VALIDATE]
ALTER RESOURCE PLAN name [ RENAME TO name2 | SET {property=value, ... } |
UNSET {property, ... } ]
```

- **name**
The name of the resource plan
- **property**
The name of the property to remove from the plan or to assign a value
- **value**
The property setting
- **name2**
The new name of a plan

ALTER RESOURCE PLAN examples

```
ALTER RESOURCE PLAN high_concurr_rp RENAME TO name2
ALTER RESOURCE PLAN high_concurr_rp SET DEFAULT POOL=etl_pool,
QUERY_PARALLELISM=3
ALTER RESOURCE PLAN high_concurr_rp UNSET QUERY_PARALLELISM=3, DEFAULT POOL
= etl_pool
```

ALTER RESOURCE PLAN description

After you activate a resource plan you altered, Hive minimizes the impact on existing queries. Hive alters the plan properties, triggers, and so on without stopping the cluster. Queued queries might be re-queued.

Hive puts pools having the same path in the old and new plan into the same pool when making the transition. If you delete some pools that were in the old plan from the new plan or if you reduce query parallelism, the queries that are currently running in the affected pools might be killed. If you add new pools, these pools immediately apply to corresponding queries and can result in queries being killed if they exceed the new limits.

Related Information[Managing an Apache Hive workload](#)[Setting up and using a resource plan](#)

ALTER TRIGGER

You add a trigger to or remove a trigger from a query pool in a plan using this command.

ALTER TRIGGER syntax

```
ALTER TRIGGER plan_name.name { ADD TO | DROP FROM } { POOL path | UNMANAGED }
```

- `plan_name.name`
The dot-delimited names of the resource plan and trigger
- `path`
The name of the query pool

ALTER TRIGGER example

```
ALTER TRIGGER rpl.slow_query DROP FROM POOL etl_pool
```

ALTER TRIGGER description

Adds or removes the trigger specified by `plan_name.name` to or from the pool identified by the `path`. The trigger applies only to queries in identified by the `path` and not to its children. Enables or disables the trigger specified by `plan_name.name` for unmanaged queries. The latter functionality applies to queries that are not running in low-latency analytic processing (LLAP) mode with workload management: for example, Apache Tez container queries.

Related Information[Managing an Apache Hive workload](#)[Setting up and using a resource plan](#)[Workload trigger counters](#)

CREATE MAPPING

You create a mapping that automatically routes queries to a specific pool using this command.

CREATE MAPPING syntax

```
CREATE { USER | GROUP | APPLICATION } MAPPING 'entity_name' IN plan_name { TO pool_path | UNMANAGED } [ WITH ORDER num ]
```

- `entity_name`
A JDBC connection user, group of JDBC connection users, or JDBC application name
- `plan_name`
The resource plan name

- `pool_path`
The hierarchy of query pool names, in dot notation
- `num`
An integer

CREATE MAPPING examples

```
CREATE GROUP MAPPING 'students' IN rp1 TO pool1 WITH ORDER 1
CREATE GROUP MAPPING 'teachers' IN rp1 TO pool2 WITH ORDER 3
CREATE APPLICATION MAPPING 'app1' IN rp1 TO pool3 WITH ORDER 2
```

Assuming the default pool is pool4, the following mapping occurs:

- Queries from users in group students go to pool1.
- Users using app1 go to pool2, unless they are in group students.
- Users in group teacher go to pool3, unless they are also in group students or are using app1.
- Users that are not in students or teachers and that are not using application app1 go to pool4.

CREATE MAPPING description

You can create a mapping based on a user, a group, or an application identified by the entity name. Mapping user, group, or application entities to a pool routes queries from those entities to the pool. The queries in the pool are under workload management.

User or group mapping

As administrator, you can configure valid non-application mapping for a particular user or group. The pool must exist. Alternatively, you can set `hive.server2.wm.allow.any.pool.via.jdbc` to true. For example, users in two different groups, each mapped to a separate pool, can explicitly submit queries to either of the pools regardless of the priority of the corresponding mappings that apply.

Workload management bases authorization to fetch a group or user on the HDFS group configuration on the cluster. By default, these are Apache Hadoop users and groups, but you can configure Lightweight Directory Protocol (LDAP) and other mechanisms.

Application mapping

To create a mapping based on an application, you use one of the following designations:

- The `applicationName` property in the JDBC connection string
- using the `SetClientInfo` JDBC API with the `ApplicationName` key.

To configure explicit querying of any pool, use multiple mappings through the `wmPool` property in the JDBC connection string.

Unmanaged queries

Queries from users, groups, or applications not mapped to a pool are unmanaged. The queries are not under workload management.

Ordered rules

The optional `WITH ORDER` clause establishes the priorities for the rules. If multiple rules apply (for example, a user rule and a group rule, or multiple groups), the lowest ordered rule takes precedence. If ordering is not specified (or is the same), user rules take precedence over application rules, which take precedence over group rules. The order of group rules with the same priority is undefined.

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

[HDFS ACL Permissions Model](#)

CREATE POOL

You create a query pool in a resource plan that allocates resources for the pool.

CREATE POOL syntax

```
CREATE POOL plan_name.path WITH ALLOC_FRACTION = decimal, QUERY_PARALLELISM
= num, [ SCHEDULING_POLICY = scheduling_value ]
```

- plan_name.path

The resource plan name and hierarchy of query pool names, in dot notation

- decimal

The proportion, in decimal notation, of the Hive low-latency analytical processing (LLAP) cluster allocated to a pool.

- num

The maximum parallel queries allowed in a pool.

- scheduling_value

One of 'default'; 'fair', which allocates an equal share of resources on the cluster to each query; or 'fcs', which allocates all resources to the earliest query. GLOBAL"?>

CREATE POOL example

```
CREATE POOL myplan.parent WITH ALLOC_FRACTION = 0.5, QUERY_PARALLELISM = 1
CREATE POOL myplan.parent.child1 WITH ALLOC_FRACTION = 0.75,
  QUERY_PARALLELISM = 2
CREATE POOL myplan.parent.child2 WITH ALLOC_FRACTION = 0.2,
  QUERY_PARALLELISM = 2
```

These commands illustrate how allocation fractions of child pools divide the cluster resources and reduce available cluster resources for the overall parent pool:

- ALLOC_FRACTION of myplan.parent takes 0.5 of the whole cluster.
- ALLOC_FRACTION of the child1 pool takes 0.75 of the parent pool, which is equivalent to 0.375 of the cluster (0.75 X 0.5 of the cluster, as set in the parent pool allocation fraction).
- ALLOC_FRACTION of the child2 pool takes 0.2 of the child1 pool, which is equivalent to 0.1 of the cluster (0.2 X 0.5 of the cluster as set in the parent pool allocation fraction).

Consequently, the proportion of cluster resources available to queries in the pool that are not regulated by any nested child pool is 0.025. (0.05 of all queries are not in child pools, multiplied by 0.5 of the entire cluster, as set for the parent pool, equals 0.025.)

CREATE POOL description

Query pools are resource plan rulesets that divide the cluster into different groups of queries. Each pool is allocated a fraction of the LLAP cluster. Optionally, a maximum number of concurrent queries and parallelism in each pool can be set.

You enter query parallelism values as whole numbers. You enter values to allocate resources in query pools as decimal fractions, because Hive LLAP resource cluster capacity tends to be granular and can frequently shift after small configuration changes. Query parallelism relies on the fixed number of query coordinators, which does not typically change as frequently.

You can nest query pools to divide the rules further into child pools.

In a nested pool, the path part of `plan_name.path` must include the hierarchy of parent pools, in dot notation.

The `CREATE POOL` command supports the following properties that represent the rulesets you can apply to control the pool:

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

CREATE RESOURCE PLAN

Using this command, you create a resource plan in the disabled state.

CREATE RESOURCE PLAN syntax

```
CREATE RESOURCE PLAN plan_name [ WITH QUERY PARALLELISM=number | LIKE name ]
```

- `plan_name`
The name of a new resource plan
- `number`
An integer
- `name`
The name of a resource plan you want to copy to create the new plan

CREATE RESOURCE PLAN example

```
CREATE RESOURCE PLAN plan2 LIKE plan1
```

This command makes a copy of `plan1` named `plan2`.

CREATE RESOURCE PLAN description

You can use query parallelism settings to validate that the resource plan executes queries using parallel processing. When the resource plan is applied to the cluster, Hive validates the sum of all query parallelism values specified for all the pools included in the plan. As administrator, you can use the fixed, total query parallelism figure, as determined by physical cluster size or other limitations, without manually verifying the sum of the pools. If the `hive.metastore.wm.default.pool.size` configuration value is greater than 0 (the default is 4), a pool named `default` is created in the new plan. All queries go to the default pool. The pool is set to take 100% of the cluster, and query parallelism is set to plan parallelism (if specified) or to the configuration value.

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

CREATE TRIGGER

You conditionally create a trigger in a resource plan.

CREATE TRIGGER syntax

```
CREATE TRIGGER plan_name.name WHEN condition DO action
```

- `plan_name.name`
The dot-delimited resource plan and trigger names
- `condition`
A supported condition
- `action`
A supported action

CREATE TRIGGER examples

```
CREATE TRIGGER plan1.trigger1 WHEN HDFS_BYTES_READ > 10Gb DO MOVE TO  
large_query_pool;  
CREATE TRIGGER plan1.trigger2 WHEN ELAPSED_TIME > 30min DO KILL;
```

CREATE TRIGGER description

The trigger you create is not applied to any queries. You must add the trigger you create to one or more a query pools or unmanaged queries using `ALTER TRIGGER`. The trigger fires if the specified condition is met and the action executes for any query in the applicable query pool or unmanaged query.

`CREATE TRIGGER` supports the following actions and conditions:

- Actions
 - `KILL`
Kills queries.
 - `MOVE TO pool_path`
Moves workload-managed queries in a query pool to the designated pool. Do not use on unmanaged queries. If the destination pool is at capacity with regard to query parallelism, the current query is killed.
- Conditions
 - Apache Tez counter is greater than a particular value, and that value equals a time unit suffix or size suffix that represents a timespan or number of bytes

There is no restriction on the type of Tez counter that can generate a trigger. Counter values are generated at query runtime. You can view all the the Tez counters for your specific version of Hive in the task counters of Tez View in Apache Ambari.

You typically use the following query-level counters with triggers:

- `ELAPSED_TIME`
- `EXECUTION_TIME`
- `TOTAL_TASKS`
- `HDFS_BYTES_READ`, `HDFS_BYTES_WRITTEN`, and similar counters for `FILE` and other file systems (see example below)
- `CREATED_FILES`
- `CREATED_DYNAMIC_PARTITIONS`

Many other counters are aggregated by query by default, but can also be aggregated by vertex to match the value for any single vertex. In the Tez View, there are many counters having a suffix of a vertex name. For example, for `INPUT_FILES_Map_1`, you can set a condition on `INPUT_FILES` based on the combined value for the entire query. The following counters are vertex-level or query-level counters:

- `RAW_INPUT_SPLITS`

- GROUPED_INPUT_SPLITS
- INPUT_FILES
- INPUT_DIRECTORIES

Formatting a WHEN HDFS_BYTES_WRITTEN clause

Single quotation marks are required around non-numeric arguments in a WHEN HDFS_BYTES_WRITTEN clause, as shown in the following example:

```
CREATE TRIGGER multiple_pools_pla ^Mn.too_large_write_trigger WHEN
HDFS_BYTES_WRITTEN > '10kb' DO MOVE TO pool1;
```

You can omit quotation marks in numeric arguments:

```
CREATE TRIGGER multiple_pools_pla ^Mn.too_large_write_trigger WHEN
HDFS_BYTES_WRITTEN > 503993049 DO MOVE TO pool1;
```

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

[Workload trigger counters](#)

DISABLE WORKLOAD MANAGEMENT

You can deactivate the current resource plan and disable workload management using this command.

DISABLE WORKLOAD MANAGEMENT syntax

```
DISABLE WORKLOAD MANAGEMENT
```

After you execute this command, no resource plan is active and all workload management features are disabled. This command deletes all query pools and kills running queries.

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

DROP MAPPING

You can remove a mapping from a resource plan by using this command.

DROP MAPPING syntax

```
DROP { USER | GROUP | APPLICATION } MAPPING 'entity_name' IN plan_name
```

- entity_name
 - The name of the user specified in the connection string
 - The name of the group of users specified in the connection string
 - The name of the application specified in the connection string or through the SetClientInfo JDBC API with ApplicationName key
- plan_name

The resource plan name

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

DROP POOL

You remove a query pool from a resource plan by using this command.

DROP POOL syntax

```
DROP POOL plan_name.path
```

- plan_name.path

The resource plan name and hierarchy of query pool names, in dot notation

You cannot remove a default query pool or one that mappings point to.

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

DROP RESOURCE PLAN

You can delete a resource plan by using this command. You must disable the resource plan before attempting to drop it.

DROP RESOURCE PLAN syntax

```
DROP RESOURCE PLAN name
```

- name

The resource plan name

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

DROP TRIGGER

You can delete a trigger by using this command.

DROP TRIGGER syntax

```
DROP TRIGGER plan_name.name
```

- plan_name.name

The dot-delimited names of the resource plan and trigger

REPLACE RESOURCE PLAN WITH

You can replace one resource plan with another by using this command.

Syntax

```
REPLACE RESOURCE PLAN name1 WITH name2
```

- name1
The name of the resource plan you want to replace
- name2
The name of the replacement resource plan

After replacement, only the plan2 remains, with the same status as plan1.

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

REPLACE ACTIVE RESOURCE PLAN WITH

You can replace one resource plan with another, enable the replacement, and activate it using this command.

Syntax

```
REPLACE ACTIVE RESOURCE PLAN name1 WITH name2
```

- name1
The name of the resource plan you want to replace
- name2
The name of the replacement resource plan

After executing the command, only the plan2 remains, active and enabled.

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

SHOW RESOURCE PLAN

You can view an entire resource plan by using this command. You see a summary of mappings if the plan exceeds the output limit.

SHOW RESOURCE PLAN syntax

```
SHOW RESOURCE PLAN name
```

- name

The name of the resource plan you want to see

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

SHOW RESOURCE PLANS

You can view all resource plans by using this command.

SHOW RESOURCE PLANS syntax

```
SHOW RESOURCE PLANS
```

Related Information

[Managing an Apache Hive workload](#)

[Setting up and using a resource plan](#)

Workload trigger counters

A workload management trigger executes an action when the condition defined in the trigger expression is met. You can use all counters exposed by Apache Tez, the file system, and Hive with workload management. The following types of counters are available: Directed acyclic graph (DAG), file system, input split, Hive, job, shuffle error, and task counters.

Table 2: DAG counters

| Counter Name | Aggregation Level |
|---------------------------|-------------------|
| AM_CPU_MILLISECONDS | AM |
| AM_GC_TIME_MILLIS | AM |
| DATA_LOCAL_TASKS | DAG |
| FALLOW_SLOTS_MILLIS_TASKS | DAG |
| NUM_FAILED_TASKS | DAG |
| NUM_FAILED_UBERTASKS | DAG |
| NUM_KILLED_TASKS | DAG |
| NUM_SUCCEEDED_TASKS | DAG |
| NUM_UBER_SUBTASKS | DAG |
| OTHER_LOCAL_TASKS | DAG |
| RACK_LOCAL_TASKS | DAG |
| SLOTS_MILLIS_TASKS | DAG |
| TOTAL_LAUNCHED_TASKS | DAG |
| TOTAL_LAUNCHED_UBERTASKS | DAG |

Table 3: File system counters

| Counter Name | Aggregation Level | Comment |
|--------------------|-------------------|--|
| BYTES_READ | DAG | Supports all file system schemes. If S3A then counter must be prefixed with "S3A_". Example: S3A_FILE_BYTES_READ |
| BYTES_WRITTEN | DAG | |
| FILE_BYTES_READ | DAG | Local file system |
| FILE_BYTES_WRITTEN | DAG | Local file system |
| HDFS_BYTES_READ | DAG | |
| HDFS_BYTES_WRITTEN | DAG | |
| LARGE_READ_OPS | DAG | |
| READ_OPS | DAG | |
| WRITE_OPS | DAG | |

Table 4: Input split counters

| Counter Name | Aggregation Level |
|-----------------------------|-------------------|
| DAG_GROUPED_INPUT_SPLITS | DAG |
| DAG_INPUT_DIRECTORIES | DAG |
| DAG_INPUT_FILES | DAG |
| DAG_RAW_INPUT_SPLITS | DAG |
| GROUPED_INPUT_SPLITS | DAG |
| INPUT_DIRECTORIES | DAG |
| INPUT_FILES | DAG |
| RAW_INPUT_SPLITS | DAG |
| VERTEX_GROUPED_INPUT_SPLITS | VERTEX |
| VERTEX_INPUT_DIRECTORIES | VERTEX |
| VERTEX_INPUT_FILES | VERTEX |
| VERTEX_RAW_INPUT_SPLITS | VERTEX |

Table 5: Hive counters

| Counter Name | Aggregation Level | Comment |
|----------------------------|-------------------|---|
| CREATED_DYNAMIC_PARTITIONS | DAG | |
| CREATED_FILES | DAG | |
| DAG_TOTAL_TASKS | DAG | Sum of all tasks launched for the DAG |
| DESERIALIZE_ERRORS | DAG | |
| ELAPSED_TIME_MS | QUERY | Time from start of query submission to end of query execution |
| EXECUTION_TIME_MS | DAG | Time from start of DAG to end of DAG |
| RECORDS_IN | DAG | |
| RECORDS_OUT | DAG | |
| RECORDS_OUT_INTERMEDIATE | DAG | |

| Counter Name | Aggregation Level | Comment |
|--------------------|-------------------|---|
| VERTEX_TOTAL_TASKS | VERTEX | Equivalent to task parallelism of particular vertex |

Table 6: Job counters

| Counter Name | Aggregation Level |
|----------------------------|-------------------|
| DATA_LOCAL_MAPS | DAG |
| FALLOW_SLOTS_MILLIS_MAPS | DAG |
| FALLOW_SLOTS_MILLIS_REDUCE | DAG |
| NUM_FAILED_MAPS | DAG |
| NUM_FAILED_REDUCE | DAG |
| NUM_FAILED_UBERTASKS | DAG |
| NUM_KILLED_MAPS | DAG |
| NUM_KILLED_REDUCE | DAG |
| NUM_UBER_SUBMAPS | DAG |
| NUM_UBER_SUBREDUCE | DAG |
| OTHER_LOCAL_MAPS | DAG |
| RACK_LOCAL_MAPS | DAG |
| SLOTS_MILLIS_MAPS | DAG |
| SLOTS_MILLIS_REDUCE | DAG |
| TOTAL_LAUNCHED_MAPS | DAG |
| TOTAL_LAUNCHED_REDUCE | DAG |
| TOTAL_LAUNCHED_UBERTASKS | DAG |

Table 7: Shuffle error counters

| Counter Name | Aggregation Level |
|--------------|-------------------|
| BAD_ID | DAG |
| CONNECTION | DAG |
| IO_ERROR | DAG |
| WRONG_LENGTH | DAG |
| WRONG_MAP | DAG |
| WRONG_REDUCE | DAG |

Table 8: Task counters

| Counter Name | Aggregation Level |
|---------------------------------|-------------------|
| ADDITIONAL_SPILL_COUNT | DAG |
| ADDITIONAL_SPILLS_BYTES_READ | DAG |
| ADDITIONAL_SPILLS_BYTES_WRITTEN | DAG |
| COMBINE_INPUT_RECORDS | DAG |
| COMBINE_OUTPUT_RECORDS | DAG |
| COMMITTED_HEAP_BYTES | DAG |

| Counter Name | Aggregation Level |
|----------------------------|-------------------|
| CPU_MILLISECONDS | DAG |
| FIRST_EVENT_RECEIVED | DAG |
| GC_TIME_MILLIS | DAG |
| INPUT_GROUPS | DAG |
| INPUT_RECORDS_PROCESSED | DAG |
| INPUT_SPLIT_LENGTH_BYTES | DAG |
| LAST_EVENT_RECEIVED | DAG |
| MERGE_PHASE_TIME | DAG |
| MERGED_MAP_OUTPUTS | DAG |
| NUM_DISK_TO_DISK_MERGES | DAG |
| NUM_FAILED_SHUFFLE_INPUTS | DAG |
| NUM_MEM_TO_DISK_MERGES | DAG |
| NUM_SHUFFLED_INPUTS | DAG |
| NUM_SKIPPED_INPUTS | DAG |
| NUM_SPECULATIONS | DAG |
| OUTPUT_BYTES | DAG |
| OUTPUT_BYTES_PHYSICAL | DAG |
| OUTPUT_BYTES_WITH_OVERHEAD | DAG |
| OUTPUT_LARGE_RECORDS | DAG |
| OUTPUT_RECORDS | DAG |
| PHYSICAL_MEMORY_BYTES | DAG |
| REDUCE_INPUT_GROUPS | DAG |
| REDUCE_INPUT_RECORDS | DAG |
| REDUCE_OUTPUT_RECORDS | DAG |
| REDUCE_SKIPPED_GROUPS | DAG |
| REDUCE_SKIPPED_RECORDS | DAG |
| SHUFFLE_BYTES | DAG |
| SHUFFLE_BYTES_DECOMPRESSED | DAG |
| SHUFFLE_BYTES_DISK_DIRECT | DAG |
| SHUFFLE_BYTES_TO_DISK | DAG |
| SHUFFLE_BYTES_TO_MEM | DAG |
| SHUFFLE_CHUNK_COUNT | DAG |
| SHUFFLE_PHASE_TIME | DAG |
| SKIPPED_RECORDS | DAG |
| SPILLED_RECORDS | DAG |
| SPLIT_RAW_BYTES | DAG |
| VIRTUAL_MEMORY_BYTES | DAG |

Related Information

[Create trigger](#)

[Alter trigger](#)