Managing the Rebalancer

The Clustrix Rebalancer is designed to run automatically as a background process to rebalance data across the cluster. The following section describes how you can configure and monitor the rebalancer, but the majority of deployments should not require user intervention.

The Rebalancer is managed primarily through a set of global variables, and can be monitored through several system tables (or vrels). As described in the Rebalancer section, the rebalancer applies a number of actions such as copying replicas, moving replicas, and splitting slices in order to maintain an optimal distribution of data on the cluster. It is designed to perform these operations in a manner that minimizes impact to user queries, and requires little administrative action. However, there may be circumstances where you wish to either increase or decrease the aggressiveness of the rebalancer, such as quickly rebalancing the cluster after node addition or eliminating any possible interference with user queries during periods of heavy load.

The sections below will discuss monitoring of rebalancer behavior, and specific use cases of rebalancer tuning.

Monitoring the Rebalancer

The table rebalancer_activity_log maintains a record of current and past rebalancer work. To see recent activity, order by started, as shown below. You can also filter for currently executing rebalancer actions with WHERE finished IS NULL.

Check recent Rebalancer activity

```
sql> select * from system.rebalancer_activity_log order by started desc limit 10;
+--------------------+------------+----------------------------+----------+--------------
+----------------------------+------------+---------------------+---------------------+-------+
| id                 | op         | reason                     | database | relation      |
representation               | bytes      | started             | finished            | error |
+--------------------+------------+----------------------------+----------+--------------
+----------------------------+------------+---------------------+---------------------+-------+
| 5832803107035702273 | rerank      | distribution read imbalance | statd    | statd_history |
__idx_statd_history__PRIMARY |  236879872 | 2017-01-13 05:35:01 | 2017-01-13 05:35:01 | NULL  |
| 5832802677131749377 | rerank      | distribution read imbalance | statd    | statd_history |
__idx_statd_history__PRIMARY |  478674944 | 2017-01-13 05:33:21 | 2017-01-13 05:33:21 | NULL  |
| 5832802504311179267 | slice split | slice too big               | statd    | statd_history |
__idx_statd_history__PRIMARY |  473628672 | 2017-01-13 05:32:41 | 2017-01-13 05:34:08 | NULL  |
| 5832791312486337538 | rerank      | distribution read imbalance | statd    | statd_history |
__idx_statd_history__PRIMARY |  475987968 | 2017-01-13 04:49:15 | 2017-01-13 04:49:15 | NULL  |
| 5832791036763671553 | slice split | slice too big               | statd    | statd_history |
__idx_statd_history__PRIMARY | 1195999232 | 2017-01-13 04:48:11 | 2017-01-13 04:49:15 | NULL  |
| 5832788503671368706 | rerank      | distribution read imbalance | statd    | statd_history |
__idx_statd_history__PRIMARY |  754778112 | 2017-01-13 04:38:21 | 2017-01-13 04:38:21 | NULL  |
| 5832788202047166465 | slice split | slice too big               | statd    | statd_history |
__idx_statd_history__PRIMARY |  471269376 | 2017-01-13 04:37:11 | 2017-01-13 04:38:29 | NULL  |
| 5832674257801927682 | rerank      | distribution read imbalance | statd    | statd_history |
__idx_statd_history__PRIMARY |  754778112 | 2017-01-12 21:15:01 | 2017-01-12 21:15:01 | NULL  |
| 5832673827981474818 | rerank      | distribution read imbalance | statd    | statd_history |
__idx_statd_history__PRIMARY |  471400448 | 2017-01-12 21:13:21 | 2017-01-12 21:13:21 | NULL  |
| 5832673526398766083 | slice split | slice too big               | statd    | statd_history |
__idx_statd_history__PRIMARY |  755400704 | 2017-01-12 21:12:11 | 2017-01-12 21:13:43 | NULL  |
+--------------------+------------+----------------------------+----------+--------------
+----------------------------+------------+---------------------+---------------------+-------+
10 rows in set (0.32 sec)
```

For details such as target/destination for in-progress rebalancer actions, JOIN (using id) to rebalancer_activity_targets, rebalancer_copy_activity, or rebalancer_splits. These are vrels (virtual relations, as opposed to actual tables), and so are only populated for the duration of the activity.

Configuring the Rebalancer

The aggressiveness of the rebalancer is controlled by several global variables.

- rebalancer_global_task_limit: specifies the number of concurrent rebalancer actions, applicable to all rebalancer actions.
- task_rebalancer_%_interval_ms: defines the interval time of when a particular rebalancer task is initiated.
- rebalancer_rebalance_task_limit: controls the number of concurrent rebalancing tasks.
- rebalancer_vdev_task_limit: limits the number of concurrent rebalancer actions that touch a single device.

The frequency of the tasks determine how often operations, such as rebalancer moves, may be enqueued. When there are many small containers, the copies and moves take only a few seconds. As such, a default frequency of 30 seconds may mean that the rebalancer queues operations less frequently than it could. Most rebalancer tasks enqueue a limited number of operations at a time, as the required operations to achieve ideal balance change over time. The notable exception is SOFTFAIL, which enqueues all work to be performed once a node or disk has been softfailed.

For operations other than reprotect, the rebalancer pauses for 5 seconds (default) after starting the transaction, before commencing the actual copy from source to target replica. This is done to reduce the chances of an outstanding user transaction conflicting with the rebalancer operation, in which case the user transaction will be canceled, with this error:

> *MVCC serializable scheduler conflict*

Note that reprotect has a higher priority and does not apply this delay.

The following are some common use cases for tuning the rebalancer settings. Please consult with Clustrix Support to change parameters not discussed below.

Increasing Rebalance Aggressiveness

By design (as described in Rebalancer) the rebalancer takes a somewhat leisurely approach to rebalancing data across the cluster. Since data imbalances between nodes typically take some time to manifest and generally do not cause significant performance issues, this is generally acceptable. However, in some situations, it is desirable to rebalance much more quickly:

- After expanding a cluster to more nodes, particularly where load is very low off-peak (or in an evaluation situation)
- After replacing a failed node, where balanced workload is critical to meeting performance requirements

Following are recommended changes to increase rebalancer aggressiveness:

Increasing Rebalance Aggressiveness

```
sql> set global rebalancer_rebalance_task_limit = 8;
sql> set global rebalancer_vdev_task_limit = 4;
sql> set global task_rebalancer_rebalance_distribution_interval_ms = 5000;
sql> set global task_rebalancer_rebalance_interval_ms = 5000;
```

If these settings cause too great a load, reduce the rebalancer_rebalance_task_limit or rebalancer_vdev_task_limit.

Once the rebalancer has finished, reset these globals back to ClustrixDB's default:

```
sql> SET GLOBAL
variable_name =
DEFAULT;
```

Increasing SOFTFAIL Aggressiveness

As described in Administering Failure and Recovery, SOFTFAIL is a means of moving all data from a node (or disk) in preparation for decommissioning or replacing a node. With proper use of SOFTFAIL, the system maintains full protection of all data; if a node is removed without SOFTFAIL, there is a window (until reprotect completes) where a failure could lead to data loss.

SOFTFAIL is treated as a high priority by the rebalancer. It differs from rebalancing, in that the per-task limit and task intervals do not apply. Changing these two globals can increase SOFTFAIL aggressiveness:

Increasing SOFTFAIL Aggressiveness

```
sql> set global rebalancer_global_task_limit = 32;
sql> set global rebalancer_vdev_task_limit = 16;
```

If these settings cause too great a load, reduce the rebalancer_global_task_limit or rebalancer_vdev_task_limit.

Once the rebalancer has finished, reset these globals back to ClustrixDB's default:

```
sql> SET GLOBAL
variable_name =
DEFAULT;
```

Disabling the Rebalancer

To disable the Rebalancer:

```
sql> set global rebalancer_optional_tasks_enabled = false;
```

This disables the rerank, split, redistribute and rebalance tasks. The value for rebalancer_optional_tasks_enabled supersedes the values in the global variables used to configure the individual rebalancer tasks.

Do not leave the Rebalancer disabled for long periods of time, as the Rebalancer plays a crucial role in maintaining optimal database performance.

The Rebalancer tasks for reprotecting data (task_rebalancer_reprotect_interval_ms) should never be disabled.

Global Variables

The following global variables impact Rebalancer activity. Note that these variables do not apply to an individual sessions.

Name	Description	Default Value
rebalancer_global_task_limit	Maximum number of simultaneous rebalancer operations.	16
rebalancer_rebalance_task_limit	Maximum number of operations that rebalancer_imbalanced and rebalancer_rebalance_distribution will each schedule at once.	2
rebalancer_rebalance_threshold	Minimum coefficient of overall write load variation that will trigger rebalance activity.	0.05
rebalancer_reprotect_queue_interval_s	Queued replicas count as healthy for this many seconds, to give missing nodes the chance to come back online before rebalancer_reprotect starts copying.	600
rebalancer_split_threshold_kb	Size at which the rebalancer splits slices.	1048576
rebalancer_vdev_task_limit	Maximum number of simultaneous rebalancer operations targeting one device.	1
task_rebalancer_rebalance_distribution_interval_ms	Milliseconds between runs of periodic task "rebalancer_rebalance_distribution". Specify 0 to disable periodic task.	30000
task_rebalancer_rebalance_interval_ms	Milliseconds between runs of periodic task "rebalancer_rebalance". Specify 0 to disable periodic task.	30000
task_rebalancer_reprotect_interval_ms	Milliseconds between runs of periodic task "rebalancer_reprotect". Specify 0 to disable periodic task.	15000
task_rebalancer_split_interval_ms	Milliseconds between runs of periodic task "rebalancer_split". Specify 0 to disable periodic task.	30000
task_rebalancer_zone_balance_interval_ms	Milliseconds between runs of periodic task "rebalancer_zone_balance". Specify 0 to disable periodic task.	60000
task_rebalancer_zone_missing_interval_ms	Milliseconds between runs of periodic task "rebalancer_zone_missing". Specify 0 to disable periodic task.	60000