Flow Control in Galera Cluster

Flow Control is a key feature in MariaDB Galera Cluster that ensures nodes remain synchronized. In synchronous replication, no node should lag significantly in processing transactions.

> Picture the cluster as an assembly line; if one worker slows down, the whole line must adjust to prevent a breakdown.

Flow Control manages this by aligning all nodes' replication processes:

Preventing Memory Overflow

Without Flow Control, a slow node's replication queue can grow unchecked, consuming all server memory and potentially crashing the MariaDB process due to an Out-Of-Memory (OOM) error.

Maintaining Synchronization

It maintains synchronization across the cluster, ensuring all nodes have nearly identical database states at all times.

Flow Control Sequence

The Flow Control process is an automatic feedback loop triggered by the state of a node's replication queue.

1. Queue Growth: A node (the "slow node") begins receiving write-sets from its peers faster than it can apply them. This causes its local receive queue, measured by the `wsrep_local_recv_queue` variable, to grow.
2. Upper Limit Trigger: When the receive queue size exceeds the configured upper limit, defined by the `gcs.fc_limit` parameter, the slow node triggers Flow Control.
3. Pause Message: The node broadcasts a "Flow Control PAUSE" message to all other nodes in the cluster.
4. Replication Pauses: Upon receiving this message, all nodes in the cluster temporarily stop replicating *new* transactions. They continue to process any transactions already in their queues.
5. Queue Clears: The slow node now has a chance to catch up and apply the transactions from its backlog without new ones arriving.
6. Lower Limit Trigger: When the node's receive queue size drops below a lower threshold (defined as `gcs.fc_limit * gcs.fc_factor`), the node broadcasts a "Flow Control RESUME" message.
7. Replication Resumes: The entire cluster resumes normal replication.

Monitoring Flow Control

As an administrator, observing Flow Control is a key indicator of a performance bottleneck in your cluster. You can monitor it using the following global status variables:

Variable Name	Description
wsrep_flow_control_paused	Indicates the fraction of time since the last `FLUSH STATUS` that the node has been paused by Flow Control. A value near `0.0` is healthy; `0.2` or higher indicates issues.
wsrep_local_recv_queue_avg	Represents the average size of the receive queue. A high or increasing value suggests a node struggling to keep up, likely triggering Flow Control.
wsrep_flow_control_sent	Counter for the number of "PAUSE" messages a node has sent. A high value indicates the node causing the cluster to pause.
wsrep_flow_control_recv	Counter for the number of "PAUSE" messages a node has received.

Troubleshooting Flow Control Issues

If you observe frequent Flow Control pauses, it is essential to identify and address the underlying cause.

Key Configuration Parameters

These parameters in `my.cnf` control the sensitivity of Flow Control:

Parameter	Description	Default Value
gcs.fc_limit	Maximum number of write-sets allowed in the receive queue before Flow Control is triggered.	`100`
gcs.fc_factor	Decimal value used to determine the "resume" threshold. The queue must shrink to `gcs.fc_limit * gcs.fc_factor` before replication resumes.	`0.8`

Modifying these values is an advanced tuning step. In most cases, it is better to fix the underlying cause of the bottleneck rather than relaxing the Flow Control limits.

Common Causes and Solutions

Cause	Description	Solution
Single Slow Node	One node is slower due to mismatched hardware, higher network latency, or competing workloads.	Investigate and either upgrade the node's resources or move the workload.
Insufficient Applier Threads	Galera may not utilize enough parallel threads, leading to bottlenecks on multi-core servers.	Increase wsrep_slave_threads according to your server's CPU core count.
Large Transactions	Large UPDATE, DELETE, or INSERT statements can create large write-sets, slowing down application by other nodes.	Break large data modification operations into smaller batches.
Workload Contention	Long-running SELECT queries on InnoDB tables can create locks that prevent replication, causing receive queues to grow.	Optimize read queries and consider wsrep_sync_wait for consistent read-after-write checks to avoid long locks on resources needed for replication.

Previous
Performance Tuning

Next
Using Streaming Replication for Large Transactions

Last updated 5 months ago

Was this helpful? ☺ 😐 ☹