`max_allowed_packet` sets an upper limit on the size of any single message between the MySQL server and clients, including replicas. If you are replicating large column values (such as might be found in `TEXT` or `BLOB` columns) and `max_allowed_packet` is too small on the source, the source fails with an error, and the replica shuts down the replication I/O thread. If `max_allowed_packet` is too small on the replica, this also causes the replica to stop the I/O thread.

Row-based replication currently sends all columns and column values for updated rows from the source to the replica, including values of columns that were not actually changed by the update. This means that, when you are replicating large column values using row-based replication, you must take care to set `max_allowed_packet` large enough to accommodate the largest row in any table to be replicated, even if you are replicating updates only, or you are inserting only relatively small values.

On a multi-threaded replica (with `slave_parallel_workers > 0`), ensure that the `slave_pending_jobs_size_max` system variable is set to a value equal to or greater than the setting for the `max_allowed_packet` system variable on the source. The default setting for `slave_pending_jobs_size_max`, 128M, is twice the default setting for `max_allowed_packet`, which is 64M. `max_allowed_packet` limits the packet size that the source can send, but the addition of an event header can produce a binary log event exceeding this size. Also, in row-based replication, a single event can be significantly larger than the `max_allowed_packet` size, because the value of `max_allowed_packet` only limits each column of the table.

The replica actually accepts packets up to the limit set by its `slave_max_allowed_packet` setting, which defaults to the maximum setting of 1GB, to prevent a replication failure due to a large packet. However, the value of `slave_pending_jobs_size_max` controls the memory that is made available on the replica to hold incoming packets. The specified memory is shared among all the replica worker queues.

The value of `slave_pending_jobs_size_max` is a soft limit, and if an unusually large event (consisting of one or multiple packets) exceeds this size, the transaction is held until all the replica workers have empty queues, and then processed. All subsequent transactions are held until the large transaction has been completed. So although unusual events larger than `slave_pending_jobs_size_max` can be processed, the delay to clear the queues of all the replica workers and the wait to queue subsequent transactions can cause lag on the replica and decreased concurrency of the replica workers. `slave_pending_jobs_size_max` should therefore be set high enough to accommodate most expected event sizes.

17.5.1.21 Replication and MEMORY Tables

When a replication source server shuts down and restarts, its `MEMORY` tables become empty. To replicate this effect to replicas, the first time that the source uses a given `MEMORY` table after startup, it logs an event that notifies replicas that the table must be emptied by writing a `DELETE` or (from MySQL 8.0.22) `TRUNCATE TABLE` statement for that table to the binary log. This generated event is identifiable by a comment in the binary log, and if GTIDs are in use on the server, it has a GTID assigned. The statement is always logged in statement format, even if the binary logging format is set to `ROW`, and it is written even if `read_only` or `super_read_only` mode is set on the server. Note that the replica still has outdated data in a `MEMORY` table during the interval between the source's restart and its first use of the table. To avoid this interval when a direct query to the replica could return stale data, you can set the `init_file` system variable to name a file containing statements that populate the `MEMORY` table on the source at startup.

When a replica server shuts down and restarts, its `MEMORY` tables become empty. This causes the replica to be out of synchrony with the source and may lead to other failures or cause the replica to stop:

- Row-format updates and deletes received from the source may fail with `Can't find record in 'memory_table'`.

- Statements such as `INSERT INTO ... SELECT FROM memory_table` may insert a different set of rows on the source and replica.