to generate sample key-value pairs to use in your own benchmarks. See libmemcached Command-Line Utilities for details.

15.20.6.4 Controlling Transactional Behavior of the InnoDB memcached Plugin

Unlike traditional memcached, the daemon_memcached plugin allows you to control durability of data values produced through calls to add, set, incr, and so on. By default, data written through the memcached interface is stored to disk, and calls to get return the most recent value from disk. Although the default behavior does not offer the best possible raw performance, it is still fast compared to the SQL interface for InnoDB tables.

As you gain experience using the daemon_memcached plugin, you can consider relaxing durability settings for non-critical classes of data, at the risk of losing some updated values in the event of an outage, or returning data that is slightly out-of-date.

Frequency of Commits

One tradeoff between durability and raw performance is how frequently new and changed data is committed. If data is critical, is should be committed immediately so that it is safe in case of an unexpected exit or outage. If data is less critical, such as counters that are reset after an unexpected exit or logging data that you can afford to lose, you might prefer higher raw throughput that is available with less frequent commits.

When a memcached operation inserts, updates, or deletes data in the underlying InnoDB table, the change might be committed to the InnoDB table instantly (if daemon_memcached_w_batch_size=1) or some time later (if the daemon_memcached_w_batch_size value is greater than 1). In either case, the change cannot be rolled back. If you increase the value of daemon_memcached_w_batch_size to avoid high I/O overhead during busy times, commits could become infrequent when the workload decreases. As a safety measure, a background thread automatically commits changes made through the memcached API at regular intervals. The interval is controlled by the innodb_api_bk_commit_interval configuration option, which has a default setting of 5 seconds.

When a memcached operation inserts or updates data in the underlying InnoDB table, the changed data is immediately visible to other memcached requests because the new value remains in the memory cache, even if it is not yet committed on the MySQL side.

Transaction Isolation

When a memcached operation such as get or incr causes a query or DML operation on the underlying InnoDB table, you can control whether the operation sees the very latest data written to the table, only data that has been committed, or other variations of transaction isolation level. Use the innodb_api_trx_level configuration option to control this feature. The numeric values specified for this option correspond to isolation levels such as REPEATABLE READ. See the description of the innodb_api_trx_level option for information about other settings.

A strict isolation level ensures that data you retrieve is not rolled back or changed suddenly causing subsequent queries to return different values. However, strict isolation levels require greater locking overhead, which can cause waits. For a NoSQL-style application that does not use long-running transactions, you can typically use the default isolation level or switch to a less strict isolation level.

Disabling Row Locks for memcached DML Operations

The innodb_api_disable_rowlock option can be used to disable row locks when memcached requests through the daemon_memcached plugin cause DML operations. By default, innodb_api_disable_rowlock is set to OFF which means that memcached requests row locks for get and set operations. When innodb_api_disable_rowlock is set to ON, memcached requests a table lock instead of row locks.