

InnoDB Performance Worksheet

Source: Drupal Performance Agency

Pages: 4

Last Modified: 10:49 PM Jan 12, 2008

License: Creative Commons Attribution-ShareAlike2.0

Description: *InnoDB is a transaction-safe, ACID compliant MySQL storage engine. It has commit, rollback, and crash recovery capabilities, and offers row level locking. The engine's overview page explains, "InnoDB has been designed for maximum performance when processing large data volumes. Its CPU efficiency is probably not matched by any other disk-based relational database engine."*

This checklist was designed by the Drupal Performance Agency. For more information, email perf@tag1consulting.com.

INSERT

I. Increase the log buffer

Tunable: `innodb_log_buffer_size`

As the InnoDB storage engine flushes its buffer a minimum of once ever second, there's no reason to set this buffer very large unless you are inserting very large data into the table (such as large BLOB fields). For most systems doing lots of INSERTS, increasing this tunable to anywhere from 2M to 8M should be sufficient.

II. Increase the InnoDB log file size

Tunable: `innodb_log_file_size`

When data is added into an InnoDB table, it is first stored in an InnoDB log file. If you're inserting large quantities of data, it can greatly boost performance to increase the size of these log files. If you do a lot of inserts, you should boost your log file size to at least 25% the size of your buffer pool. For best performance, you may want to increase your total log file size up to the size of your buffer pool (up to a current limit of 4GB). However, note that allocating large InnoDB log files mean that recovery time is very slow. If MySQL crashes, InnoDB will have to parse the entire log files at startup time, a very time consuming process.

Note: It's not enough to just edit my.cnf to change the size of your log files. Instead, you must do all of the following steps:

1. *Edit my.cnf, setting a new log file size.*
2. *Gracefully shut down the MySQL server.*

Drupal Performance and Scalability Checklist

2 of 4

3. *Remove (or archive) the existing InnoDB log files*
4. *Start the MySQL server, allowing it to create new log files.*
5. *Verify that the new log files are the size you set in step #1.*

III. Only flush logs once per second

Tunable: `innodb_flush_logs_at_trx_commit`

By default, the InnoDB storage engine is ACID compliant, meaning that it flushes each transaction to the file system when it is committed. You can set the above tunable to 0 to disable this, telling InnoDB to flush to disk only once per second. Alternatively, you can set the above tunable to 2, letting the file system handle the flushing of data to the disk.

IV. Test alternative flush methods

Tunable: `innodb_flush_method`

By default, InnoDB uses the `fsync()` system call. On some systems it can be faster to flush using `O_DSYNC`. It is important to benchmark this change, as which flush method will perform best is dependent on your system.

V. Disable AUTOCOMMIT

InnoDB treats all statements as transactions, adding overhead when inserting lots of data. If you need to INSERT lots of data, first call “`SET AUTOCOMMIT = 0;`”. Then, execute a group of INSERT statements followed by a manual “`COMMIT;`”. It is best to run some benchmarks to determine the optimal size for your transactions. If you make your transactions too big, they will become disk-bound, reducing INSERT performance. You can increase the supported size of your commits by increasing the size of your Buffer Pool.

VI. Disable UNIQUE_CHECKS

If you have UNIQUE constraints on any of your secondary keys, you can greatly boost insert performance into large tables by running “`SET UNIQUE_CHECKS=0`” before you INSERT the data, and then “`SET UNIQUE_CHECKS=1`” when you are done. However, be certain that you are not inserting any duplicate data before you do this.

VII. Presort your data by the Primary Key

InnoDB physically stores data sorted by the primary key. Thus, if you presort your data before you INSERT it into the database the InnoDB storage engine can handle it more efficiently. If you INSERT data in a random order, it can also cause your tables to become

Drupal Performance and Scalability Checklist

3 of 4

fragmented. If there's no way to INSERT data in order, you may want to consider making your primary key an auto_increment field.

General

I. Search with your Primary Key

InnoDB offers optimal performance for searching by PRIMARY KEY compared to any other index. This is because data is stored on disk and in memory sorted by the primary key.

II. Keep your Primary Key short

If your Primary Keys are long, this will result in very large, slow indexes. If your existing Primary Key is long, you may want to convert it to a Unique Key, then add a new auto_increment field as the primary key. If, however, you do your lookups using only the Primary Key, then leave it as is even if the Primary Key column is long.

III. Only create necessary indexes

InnoDB stores an uncompressed copy of your Primary Key with each Secondary Key. Thus, if you have lots of indexes and you have a large Primary Key, your indexes are going to use a lot of disk space.

IV. Optimizing SELECT COUNT(*)

The design of the InnoDB storage engine prevents it from storing the actual row count of each table, thus it actually has to count rows to return SELECT COUNT(). However, if you add a WHERE clause to the COUNT(*), InnoDB will offer the same performance as MyISAM. Alternatively, you can parse SHOW TABLE STATUS LIKE "NAME" to get quick access to an estimate of the number of rows in table NAME. (For static tables, this will be accurate. For quickly changing tables, it will just be close.)*

V. Don't empty a table with DELETE FROM or TRUNCATE

Emptying a large table using DELETE FROM or TRUNCATE is slow on InnoDB. This is because for both operations InnoDB has to process each row of the table, and due to its transactional design it first writes each delete action to the transaction log then applies it to the actual table. For better performance if not limited by foreign keys, use DROP

Drupal Performance and Scalability Checklist

4 of 4

TABLE followed by CREATE TABLE to empty a large table.