MySQL Optimisation, Performance, and Logging

==========

MySQL Optimisation

==========

MySQL Tuner Script

```
wget -O mysqltuner.pl mysqltuner.pl && perl mysqltuner.pl
```

Check MySQL default config files

```
/usr/sbin/mysqld --verbose --help | grep -A 1 "Default options"
```

MySQL Variables

innodb_buffer_pool_size	Specifies the size of the memory buffer used by InnoDB to cache data and indexes. Setting this appropriately can significantly improve read/write performance for InnoDB tables. ***Recommendation:*** *For dedicated MySQL servers, set to 70-80% of available memory.*
query_cache_size	Controls the size of the query cache, which stores results of SELECT queries to speed up subsequent identical queries. Effective use depends on query patterns and workload. ***Recommendation:*** *Enable and configure it based on the workload if beneficial; otherwise, keep it disabled.*

key_buffer_size	Sets the size of the buffer used for index blocks for MyISAM tables. Adjusting this can enhance performance for MyISAM-based applications. ***Recommendation:*** *Set according to the size of your MyISAM indexes; for mixed storage engines, allocate more memory to InnoDB buffer pool.*
max_connections	Limits the number of simultaneous client connections allowed to the MySQL server. Properly setting this prevents resource exhaustion and maintains server stability.
tmp_table_size = max_heap_table_size =	Define the maximum size of temporary tables created in memory. Optimizing these can reduce disk I/O and improve query performance.
sort_buffer_size=256K	Buffer size for sorts.
max_connections=151	Maximum number of simultaneous client connections. ***Recommendation:*** *Increase based on expected load and server capacity.*
innodb_log_file_size	Size of each log file in the log group. ***Recommendation:*** *Increase to improve performance, particularly for write-heavy applications.*
innodb_log_buffer_size	Size of the buffer for log data before it is written to disk. ***Recommendation:*** *Increase for high transaction throughput to reduce disk I/O.*

Logging

slow_query_log	Enables logging of queries that exceed a certain execution time. ***Recommendation:*** *Enable and use for identifying slow queries; configure* `long_query_time` *to set the threshold.*
log_error = /var/log/mysql.log	Path to the error log file.
general_log=/var/log/mysql.log	Enables logging of all queries. ***Recommendation:*** *Use with caution due to potential performance impact and disk consumption; useful for debugging.*

==========

MyISAM v InnoDB

InnoDB has row-level locking. MyISAM only has full table-level locking.

- **InnoDB:**
  - Default storage engine in MySQL.
  - Supports ACID-compliant transactions.
  - Provides row-level locking and foreign key constraints.
  - Uses a clustered index for primary key storage.
- **MyISAM:**
  - Older storage engine, primarily used before InnoDB became the default.
  - Provides table-level locking.
  - Faster for read-heavy operations.
  - Does not support transactions or foreign keys.

Check table engine:

```
mysql -e "SELECT TABLE_SCHEMA as DbName ,TABLE_NAME as TableName ,ENGINE as Engine FROM information_schema.TABLES WHERE ENGINE='MyISAM' AND TABLE_SCHEMA NOT IN('mysql','information_schema','performance_schema');"
```

Generate commands to swap table engine from MyISAM to InnoDB

```
mysql -e "SELECT CONCAT('ALTER TABLE \`', TABLE_SCHEMA,'\`.\`',TABLE_NAME, '\` ENGINE = InnoDB;') FROM information_schema.TABLES WHERE ENGINE='MyISAM' AND TABLE_SCHEMA NOT IN('mysql','information_schema','performance_schema');"
```

==========

Deadlocks

A deadlock is a situation where two or more transactions are unable to proceed because each one is waiting for a lock held by the other transaction. This creates a cycle of dependencies that cannot be resolved without external intervention.

Check for deadlocks:

```
SHOW ENGINE INNODB STATUS;
```

```
SHOW STATUS LIKE 'Table_locks_%';
```

- **Table_locks_immediate**: The number of times a table lock was acquired immediately.
- **Table_locks_waited**: The number of times a table lock couldn't be acquired immediately and had to wait.

Deadlock logging

This will cause MySQL to log all deadlock errors to the error log, which can then be examined for more details.

```
SET GLOBAL innodb_print_all_deadlocks = 1;
```

MySQL Monitoring software such as NewRelic can provide further information on deadlocks and general MySQL performance issues.

Resolving Deadlocks

Once identified, you can resolve deadlocks by:

- **Rewriting Queries**: Optimize queries to reduce lock contention.
- **Using Consistent Locking Order**: Ensure all transactions acquire locks in a consistent order.
- **Adding Indexes**: Improve query performance and reduce the duration of locks.
- **Reducing Transaction Size**: Break large transactions into smaller ones to reduce lock time.
- **Implementing Retries**: Add retry logic in your application to handle deadlocks gracefully.

Revision #17
Created 2023-07-30 16:07:08 UTC by Daniel
Updated 2024-07-06 02:07:47 UTC by Daniel