**MySQL Configuration Parameter Information**

**server-id = 1** is used in replication to enable master and slave servers to identify themselves uniquely.

**log-bin= wind3-mysql-bin** is used to enable the binary log on server.

**max\_binlog\_size =512M** this parameter represent the max log size of each binary log file.

**sync\_binlog =1**

When sync\_binlog=1, all transactions are synchronized to the binary log before they are committed. Therefore, even in the event of an unexpected restart, any transactions that are missing from the binary log are only in prepared state. This causes the server's automatic recovery routine to roll back those transactions. This guarantees that no transaction is lost from the binary log, and is the safest option.

**expire\_logs\_days=5** The number of days for automatic binary log file removal. If you don’t want it to be automatically set the same to 0

**max\_allowed\_packet=500M**

You must increase this value if you are using large BLOB columns or long strings. It should be as big as the largest BLOB you want to use. The protocol limit for max\_allowed\_packet is 1GB. The value should be a multiple of 1024; nonmultiples are rounded down to the nearest multiple.

**datadir=/var/lib/mysql** data directory of My-SQL from where mysql read/write file.

**socket=/var/lib/mysql/mysql.sock** On Unix, this option specifies the Unix socket file to use when listening for local connections.

**symbolic-links=0**

On Unix, enabling symbolic links means that you can link a MyISAM index file or data file to another directory with the INDEX DIRECTORY or DATA DIRECTORY options of the CREATE TABLE statement. If you delete or rename the table, the files that its symbolic links point to also are deleted or renamed

**sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES**

**tmp\_table\_size=100M**

The maximum size of internal in-memory temporary tables. This variable does not apply to user-created MEMORY tables.

If an in-memory temporary table exceeds the limit, MySQL automatically converts it to an on-disk temporary table. Increase the value of tmp\_table\_size (and max\_heap\_table\_size if necessary) if you do many advanced GROUP BY queries and you have lots of memory.

**open\_files\_limit=9072**

The number of files that the operating system permits mysqld to open. The formula behind it as follows.

1) 10 + max\_connections + (table\_open\_cache \* 2)

2) max\_connections \* 5

3) open\_files\_limit value specified at startup, 5000 if none

**max\_connections=1200**

The maximum permitted number of simultaneous client connections.

**join\_buffer\_size=500M**

The minimum size of the buffer that is used for plain index scans, range index scans, and joins that do not use indexes and thus perform full table scans. Normally, the best way to get fast joins is to add indexes. Increase the value of join\_buffer\_size to get a faster full join when adding indexes is not possible. One join buffer is allocated for each full join between two tables. For a complex join between several tables for which indexes are not used, multiple join buffers might be necessary.

**sort\_buffer\_size=2M**

If you see many Sort\_merge\_passes per second in SHOW GLOBAL STATUS output, you can consider increasing the sort\_buffer\_size value to speed up ORDER BY or GROUP BY operations that cannot be improved with query optimization or improved indexing.

**read\_rnd\_buffer\_size=25M**

When reading rows from a MyISAM table in sorted order following a key-sorting operation, the rows are read through this buffer to avoid disk seeks. Setting the variable to a large value can improve ORDER BY performance by a lot. However, this is a buffer allocated for each client, so you should not set the global variable to a large value. Instead, change the session variable only from within those clients that need to run large queries.

**read\_buffer\_size=150M**

This option is also used in the following context for all storage engines:

 For caching the indexes in a temporary file (not a temporary table), when sorting rows for ORDER BY.

 For bulk insert into partitions.

 For caching results of nested queries.

and in one other storage engine-specific way: to determine the memory block size for MEMORY tables.

**wait\_timeout=3600**

The number of seconds the server waits for activity on a noninteractive connection before closing it.

# MyISAM #

**key\_buffer\_size=400M**

You can increase the value to get better index handling for all reads and multiple writes; on a system whose primary function is to run MySQL using the MyISAM storage engine, 25% of the machine's total memory is an acceptable value for this variable. However, you should be aware that, if you make the value too large (for example, more than 50% of the machine's total memory), your system might start to page and become extremely slow. This is because MySQL relies on the operating system to perform file system caching for data reads, so you must leave some room for the file system cache. You should also consider the memory requirements of any other storage engines that you may be using in addition to MyISAM.

# INNODB #

**innodb\_file\_per\_table=1**

When innodb\_file\_per\_table is enabled (the default), InnoDB stores the data and indexes for each newly created table in a separate .ibd file instead of the system tablespace. The storage for these tables is reclaimed when the tables are dropped or truncated.

**innodb\_flush\_method=O\_DIRECT**

O\_DIRECT: InnoDB uses O\_DIRECT (or directio() on Solaris) to open the data files, and uses fsync() to flush both the data and log files. This option is available on some GNU/Linux versions, FreeBSD, and Solaris.

**innodb\_flush\_log\_at\_trx\_commit=1**

Controls the balance between strict ACID compliance for commit operations and higher performance that is possible when commit-related I/O operations are rearranged and done in batches. You can achieve better performance by changing the default value but then you can lose up to a second of transactions in a crash.

1:-The default value of 1 is required for full ACID compliance. With this value, the contents of the InnoDB log buffer are written out to the log file at each transaction commit and the log file is flushed to disk.

2:-With a value of 0, the contents of the InnoDB log buffer are written to the log file approximately once per second and the log file is flushed to disk. No writes from the log buffer to the log file are performed at transaction commit. Once-per-second flushing is not guaranteed to happen every second due to process scheduling issues. Because the flush to disk operation only occurs approximately once per second, you can lose up to a second of transactions with any mysqld process crash.

3:-With a value of 2, the contents of the InnoDB log buffer are written to the log file after each transaction commit and the log file is flushed to disk approximately once per second. Once-per-second flushing is not 100% guaranteed to happen every second, due to process scheduling issues. Because the flush to disk operation only occurs approximately once per second, you can lose up to a second of transactions in an operating system crash or a power outage.

4:-InnoDB log flushing frequency is controlled by innodb\_flush\_log\_at\_timeout, which allows you to set log flushing frequency to N seconds (where N is 1 ... 2700, with a default value of 1). However, any mysqld process crash can erase up to N seconds of transactions.

5:-DDL changes and other internal InnoDB activities flush the InnoDB log independent of the innodb\_flush\_log\_at\_trx\_commit setting.

6:-InnoDB crash recovery works regardless of the innodb\_flush\_log\_at\_trx\_commit setting. Transactions are either applied entirely or erased entirely.

For durability and consistency in a replication setup that uses InnoDB with transactions:

1:-If binary logging is enabled, set sync\_binlog=1.

2:-Always set innodb\_flush\_log\_at\_trx\_commit=1.

**innodb\_buffer\_pool\_size=5500M**

The size in bytes of the buffer pool, the memory area where InnoDB caches table and index data.

**innodb\_log\_buffer\_size =900M**

A large log buffer enables large transactions to run without the need to write the log to disk before the transactions commit. Thus, if you have transactions that update, insert, or delete many rows, making the log buffer larger saves disk I/O.

**log\_bin\_trust\_function\_creators=1**

#Logging#

**slow\_query\_log\_file=/var/lib/mysql/mysql-slow.log**

This parametes tell us the location of slow log files.

**long\_query\_time =60**

Through this parameters mysql know that query which is <=configured seconds required to be write in the slow logs.

**slow\_query\_log = 1**

To enable the slow query logs set this value to 1.

**[mysqld\_safe]**

**log-error=/var/log/mysqld.log**

Location where mysql error log file to be located.

**pid-file=/var/run/mysqld/mysqld.pid**

Location where mysql PID log file to be located.

**Partition Information of hawa\_new database**

|  |  |  |
| --- | --- | --- |
| Schema | Tables | End Date |
| hawa\_new | actual\_generation | 2018-01-11 |
| hawa\_new | forecaster\_ui\_charges | 2018-01-11 |
| hawa\_new | grid\_availability | 2018-01-11 |
| hawa\_new | inter\_extra\_polation\_fifteen\_mint | 2018-01-11 |
| hawa\_new | power\_forecast\_block | 2018-01-30 |
| hawa\_new | power\_predicate\_with\_regressed | 2018-01-11 |
| hawa\_new | power\_predication\_for\_each\_cluster | 2017-12-31 |
| hawa\_new | scada\_data | 2018-01-11 |
| hawa\_new | scada\_data\_actual\_generation | 2018-01-11 |
| hawa\_new | scada\_one\_mint\_data | 2017-06-08 |
| hawa\_new | schedule | 2018-01-11 |
| hawa\_new | turbine\_availability | 2018-01-11 |
| hawa\_new | weighted\_avarage\_for\_wind | 2018-01-11 |
| hawa\_new | wind\_forecast\_data | 2018-01-11 |
| hawa\_new | wind\_speed\_direction | 2017-12-31 |