Тонкая настройка MySQL (часть 1).

Эта заметка, в основном, является переводом соответствующих разделов руководства по MySQL. Здесь собраны возможные способы настройки и использования MySQL-сервера для достижения максимальной производительности. Все нижесказанное относится к версии 3.22 для Linux. Для других операционных систем некоторые из утверждений могут не выполняться.

Содержание:

Компилирование исходников.

Для достижения наибольшей производительности сервера нужно учитывать такие факты:

1. При компиляции pgcc с опцией –O6 mysqld работает на 11% быстрее, чем если компилировать обычным gcc.
2. Если использовать динамическую линковку, то результат будет на 13% медленней, чем при статической.
3. Если использовать TCP/IP соединения, то результат на 7.5% хуже, чем при использовании UNIX – сокетов.

В связи с этим рекомендуется поставить компилятор pgcc. Этот совет, конечно, имеет смысл, если у вас Pentium процессор. Pgcc будет полезен не только для компилирования MySQL, разработчики этого компилятора утверждают, что откомпилированные им программы минимум на 5% работают быстрее, чем откомпилированные с помощью gcc.
Запуск конфигуратора может иметь такой вид (принимая во внимание вышеизложенные факты):

```
CFLAGS="-O6 -fomit-frame-pointer" \
CXX=gcc \
CXXFLAGS="-O6 -fomit-frame-pointer \
            -felide-constructors -fno-exceptions -fno-rtti" \
./configure \
    --enable-assembler \
    --disable-shared \
    --with-mysqld-ldflags="-all-static" \
    --with-client-ldflags="-all-static" \
    --with-unix-socket-path=/tmp/mysql.sock \
    --prefix=/usr
```

Настройка переменных.

Если запустить mysqladmin –variables, то можно увидеть примерно такую картину.

```
+---------------------------+---------------------------------+
| Variable_name             | Value                           |
+---------------------------+---------------------------------+
| back_log                  | 5                               |
| connect_timeout           | 5                               |
| basedir                   | /usr/local/                     |
| datadir                   | /home/www/data/                 |
| delayed_insert_limit      | 100                             |
| delayed_insert_timeout    | 300                             |
| delayed_queue_size        | 1000                            |
| join_buffer               | 131072                          |
| flush_time                | 0                               |
```

```
| key_buffer                 | 8388600                          |
| language                   | /usr/local/share/mysql/english/  |
| log                        | OFF                              |
| log_update                 | OFF                              |
| long_query_time            | 10                               |
| low_priority_updates       | OFF                              |
| max_allowed_packet         | 1048576                          |
| max_connections            | 100                              |
| max_connect_errors         | 10                               |
| max_delayed_insert_threads | 20                               |
| max_join_size              | 4294967295                       |
| max_sort_length            | 1024                             |
| net_buffer_length          | 16384                            |
| pid_file                   | /usr/local/var/mysqld.pid        |
| port                       | 3306                             |
| protocol_version           | 10                               |
| record_buffer              | 131072                           |
| skip_locking               | ON                               |
| skip_networking            | OFF                              |
| socket                     | /tmp/mysql.sock                  |
| sort_buffer                | 2097144                          |
| table_cache                | 64                               |
| thread_stack               | 65536                            |
| tmp_table_size             | 1048576                          |
| tmpdir                     | /tmp/                            |
| version                    | 3.22.27                          |
| wait_timeout               | 28800                            |
+----------------------------+----------------------------------+
```

От настройки этих переменных сильно зависит производительность сервера. Надо сказать, что не существует наилучших значений для всех случаев. Для каждого конкретного сервера и поставленных задач нужно выбирать свои значения.
Рассмотрим подробнее эти переменные (красным выделены наиболее важные из них).

back_log – этот параметр показывает, сколько одновременно может быть невыполненных запросов на соединение (connection requests). Параметр имеет большое значение в тех случаях, когда к MySQL поступает ОЧЕНЬ много запросов на соединение в малый промежуток времени. Когда к MySQL поступает connect-запрос, производятся следующие действия – проверяется, разрешен ли доступ к серверу, и если разрешен, то порождается новый процесс. Все это занимает достаточно мало времени. Однако если за это время поступит еще один connect-запрос, то он заносится в очередь. Параметр **back_log** определяет длину этой очереди. Если количество запросов превысит данное значение, то все непомещающиеся запросы будут игнорироваться. По умолчанию значение **back_log** равно 5, что вполне достаточно для большинства серверов. Максимально значение **back_log** ограничено операционной системой.

connect_timeout – количество секунд, которое сервер ждет connect-пакета, по истечении этого времени будет выдан пакет “Плохое соединение”. Для более детальных разъяснений см. описание TCP/IP протокола.

delayed_insert_timeout – как долго поток INSERT DELAYED будет ожидать данных для INSERT. Более подробно значение слова DELAYED расписано в описании INSERT запроса.

delayed_insert_limit – INSERT DELAYED вставив количество записей, равное **delayed_insert_limit,** проверяет, есть ли SELECT-запрос к этой же таблице. Если есть, то выполняется SELECT, и только после этого продолжается INSERT.

delayed_queue_size – Для выполнения INSERT DELAYED будет выделятся очередь длиной в **delayed_queue_size** строк. Когда очередь заполниться все остальные конкурирующие INSERT DELAYED запросы будут ждать, пока не освободиться место в этой очереди.

flush_time – Если значение больше нуля, то каждые **flush_time** секунд все таблицы будут закрываться. Это позволит освобождать неиспользуемые ресурсы и синхронизировать данные на диске.

join_buffer – Величина буфера, который используется для полных JOIN запросов (т.е. для полного объединения двух таблиц без использования индексов). Память под такой буфер выделяется один раз для каждого запроса. Увеличение параметра ускорит выполнение таких запросов. Более естественный путь ускорить полные JOIN запросы – использовать индексы.

key_buffer – Величина буфера (в байтах), который используется для индексов. Этот буфер общий для всех потоков. Если используется много DELETE или INSERT запросов к таблицам с большим кол-вом индексов, то увеличение значения повысит скорость выполнения таких запросов. Для достижения еще большей скорости нужно использовать LOCK TABLES.

long_query_time – Если время выполнения запроса превысит данное значение (в сек.), то внутренний счетчик `slow_queries` будет увеличен на 1. Посмотреть значение счетчика можно командой mysql>status.

max_allowed_packet – Максимальный размер пакета для передачи данных. Данные между клиентом и сервером передаются пакетами. В начале создается пакет длиной `net_buffer_length` затем, если размер данных больше, то размер пакета увеличивается до необходимого значения, при этом его длина не может превысить значение **max_allowed_packet**. Если используются поля BLOB большого размера, то рекомендуется увеличить значение этого параметра. В идеале нужно присвоить этой переменной значение размера самого большого BLOB поля.

max_connections – Максимальное количество открытых соединений. Определяет, сколько клиентов одновременно могут работать с сервером. Увеличение параметра увеличивает количество используемых дескрипторов файла.

max_connect_errors – Если в процессе общения с клиентом произошел обрыв соединения (interrupt connection), то счетчик ошибок для хоста клиента увеличивается на 1. Когда значение этого счетчика достигнет **max_connect_errors,** то все последующие соединения с данного хоста будут игнорироваться. Для обнуления счетчиков использовать команду FLUSH HOSTS.

max_delayed_threads – Максимальное количество потоков, которые выполняют INSERT DELAYED. Если будет вызван запрос INSERT DELAYED, а при этом достигнуто значение **max_delayed_threads**, то такой запрос будет выполнен как обычный INSERT (без опции DELAYED).

max_join_size – Максимальное количество записей, которое может быть возвращено полным JOIN запросом. Если в JOIN запросе кол-во записей превысит это значение, то будет возвращена ошибка. Увеличение значения этого параметра позволит выполнять большие запросы, но при этом следует учитывать, то такие запросы съедают много процессорного времени и могут содержать миллионы записей.

max_sort_length – При сортировке BLOB или TEXT полей из каждого поля берутся только первые **max_sort_length** байт, а остальные отбрасываются и при сортировке не учитываются.

max_tmp_tables – Максимальное количество временных таблиц, которые клиент может сохранять открытыми одновременно. На самом деле в версии 3.22 это поле ни на что не влияет.

net_buffer_length – Размер пакета для передачи данных (см. **max_allowed_packet**) Обычно этот параметр не нужно изменять, но если у вас очень мало памяти, то его можно уменьшить до ожидаемого размера результата запроса.

record_buffer – Каждый поток, который осуществляет последовательное сканирование таблиц (обычно это происходит в SELECT запросах), для каждой таблицы, участвующей в сканировании выделяет память размером **record_buffer.** Если будет много запросов, требующих последовательно сканировать таблицы, то значение этого параметра рекомендуется увеличить.

sort_buffer – Каждый поток, который осуществляет сортировку данных (ORDER BY или GROUP BY), выделяет память размером **sort_buffer.** Для повышения быстродействия запросов с ORDER BY или GROUP BY это значение необходимо увеличить.

table_cache – Количество открытых таблиц для ВСЕХ потоков. Увеличение значения приведет к увеличению количества используемых дескрипторов файла. MySQL необходимо 2 дескриптора для каждой открытой таблицы.

tmp_table_size – Максимальный размер временных таблиц. При превышении этого размера возвращается ошибка **table tbl_name is full**. При использовании сложных GROUP BY запросов значение нужно увеличить.

thread_stack - Размер стека для каждого потока. Обычно значение по умолчанию является достаточным.

wait_timeout – Время, которое поток ждет повторного обращения. Если за это время к потоку не было ни одного обращения, то поток убивается.

Параметры **table_cache**, **max_connections** и **max_tmp_tables** определяют, как много файлов сервер будет держать открытыми. Максимально количество открытых файлов для каждого процесса ограничивается операционной системой. На многих ОП это количество можно увеличить. Для более детальной информации см. руководство по вашей ОП.

table_cache зависит от **max_connections**. Например, если у вас 200 открытых соединений, то вам может понадобиться до 200*n открытых таблиц, где n – количество таблиц, участвующих в запросах.

MySQL при работе с таблицами использует хорошо масштабируемые алгоритмы, так что MySQL может работать даже при малых объемах памяти. Естественно для лучшей производительности нужно больше оперативной памяти.

Изменить значение настроек можно при запуске сервера опцией –O, например,
>safe_mysqld -O key_buffer=16M
(в значениях, которые измеряются в байтах, для сокращения, можно использовать буквы K и M).

Если у вас много памяти и много таблиц, то для увеличения производительности, при запуске сервера можно использовать такие значения.

```
>safe_mysqld -O key_buffer=16M -O table_cache=128 \
               -O sort_buffer=4M -O record_buffer=1M &


Если у вас мало памяти и ожидается мало соединений, то лучше сервер запускать с
такими опциями
>safe_mysqld -O key_buffer=512k -O sort_buffer=100k \
           -O record_buffer=100k &

или даже
> safe_mysqld -O key_buffer=512k -O sort_buffer=16k \
           -O table_cache=32 -O record_buffer=8k -O net_buffer=1K &
```

Необходимо принимать во внимание, что, если сервер сконфигурировать для использования малого объема памяти, то в случае большого числа соединений может возникнуть “проблема своппинга”, которая сильно затормозит сервер.

Как MySQL работает с памятью.

Здесь приведено описание того, как MySQL сервер работает с оперативной памятью. В скобочках жирными буквами указаны переменные, которые влияют на то или иное значение.

1. Key_buffer является общим для всех потоков. Все остальные буфера выделяются по мере необходимости.
2. Каждое соединение использует некоторое количество памяти. Это память для стека (**`thread_stack`**), буфер соединения (**`net_buffer_length`**) и буфер результата (**`net_buffer_length`**)
3. Для каждого запроса, требующего последовательно сканировать таблицу, выделяется буфер чтения (**`record_buffer`**).
4. Все JOIN запросы выполняются в один проход и для большинства таких запросов нет необходимости применять вспомогательные таблицы. В основном вспомогательные таблицы формируются в памяти, но если такая таблица имеет слишком большой размер записи или используется тип BLOB, то она сохраняется на диске. Если размер вспомогательной таблицы, которая хранится в памяти, превысит **`tmp_table_size`**, то будет возвращена ошибка table_name is full. Для избежания такой ошибки нужно или увеличить значение tmp_table_size, или включить опцию SQL_BIG_TABLES (это можно сделать либо запросом SET SQL_BIG_TABLES=1, либо запускать mysqld с опцией –big-tables). При включенной опции SQL_BIG_TABLES **все** вспомогательные таблицы формируются не в памяти, а на диске.
5. Запросы с сортировкой выделяют в памяти буфер для сортировки (**sort_buffer**) и используют один или два временных файла.
6. Таблицы с данными открываются каждым конкурирующим потоком. Индексные файлы открываются всего один раз, не зависимо от количества потоков, использующих эти файлы.
7. Для таблиц с BLOB-полями буфер автоматически увеличивается до размера самого большого BLOB-поля.
8. Дескрипторы всех открытых таблиц хранятся в кэше, который работает по принципу FIFO. Размер кэша определяется переменной **table_cache**. Если несколько потоков открывают одну и туже таблицу, то для каждого потока выделяется свой дескриптор таблицы.
9. Команда mysqladmin flash-tables закрывает все таблицы, которые не используются в данный момент, а все используемые таблицы помечает для закрытия. Такая операция позволяет освободить неиспользуемую память.

Как работать с таблицами для достижения большей производительности.

1. По возможности все поля декларировать как NOT NULL. Это сделает работу с таблицами более быстрой и сохранит 1 бит на каждое такое поле.
2. Применять значения по умолчанию (DEFAULT). При вызове запроса INSERT в таблицу будут записываться только те поля, значения которых отличаются от DEFAULT.
3. Используйте настолько малые типы INT, насколько это возможно. Например, применять MEDIUMINT намного лучше, чем обычный INT.
4. Если у вас нет записей с переменной длиной (нет ни одного поля с типом VARCHAR, BLOB или TEXT), то таблица сохраняется в формате «с постоянной длиной записи». Это несколько расходует память, но намного повышает скорость работы.
5. При использовании нескольких последовательных INSERT запросов, лучше все данные указать в одном INSERT, чем делать несколько INSERT.

6. При загрузке данных в таблицу лучше использовать LOAD DATA INFILE, чем INSERT, такой метод в 20 раз быстрее.
7. Для увеличения скорости LOAD DATA INFILE и INSERT нужно увеличить значение переменной **key_buffer.**
8. Если ожидается много запросов INSERT или UPDATE, работающих одновременно, то для большей скорости рекомендуется приметь LOCK TABLES.
9. Время от времени нужно дефрагметировать таблицы. Это делается утилитой `isamchk с опциями -evi.`

Форматы таблиц в MySQL

MySQL для хранения данных использует три типа таблиц: с фиксированной длиной строки, с динамической длиной строки и сжатые таблицы.

Таблицы с фиксированной длиной строки.

Этот формат применяется по умолчанию, если в таблице нет полей с типом VARCHAR, BLOB или TEXT.

Все поля типа CHAR, NUMERIC и DECIMAL дополняются в конце пробелами.

Высокая скорость работы.

Легко кэшируются.

Легко восстановить после краха, так как все строки имеют постоянную длину.

Не требуют реорганизации (с помощью isamchk), до тех пор, пока не будет удалено очень много записей, и вы захотите освободить место на диске.

Обычно такие таблицы занимают больше места, чем таблицы с динамической длиной строки.

Таблицы с динамической длиной строки.

Этот формат применяется, если в таблице есть поля с типом VARCHAR, BLOB или TEXT.

Все строки динамические (CHAR хранятся как VARCHAR, кроме тех у которых длина меньше 4).

Каждое поле имеет дополнительный бит, который устанавливается, если строковое поле равно «» (пустая строка), или если числовое поле равно 0 (это не то же самое, когда поле может иметь значение NULL).

Непустые строки хранятся в виде {ДЛИНА_СТРОКИ} {СОДЕРЖАНИЕ_СТРОКИ}

Обычно такие таблицы занимают намного меньше места, чем таблицы с фиксированной длиной.

Ожидаемая длина строки вычисляется по формуле:

3+(количество полей + 7)/8+(количество полей типа CHAR)+(размер числовых типов в бинарном виде)+(длина всех строк)+(количество NULL-полей + 7)/8.

Сжатые таблицы.

Таблицы «только-для-чтения», их можно получить с помощью утилиты pack_isam. Эту утилиту получают все покупатели, которые приобрели расширенную поддержку MySQL. Основная характеристика – занимают мало места.

Использование индексов.

Все индексы (PRIMARY, UNIQUE и INDEX) хранятся в B-дереве. В строковых типах автоматически происходит сжатие начальных и конечных пробелов.

Индексы используются для:

Быстрого поиска записей по условию WHERE;

Для объединения таблиц с посредством JOIN;

Поиска MAX() и MIN() значений для ключевых полей;

Для сортировки и группировки таблиц (директивы ORDER BY и GROUP BY);

Для извлечения данных не из таблицы с данными, а из индексного файла. Это возможно только в некоторых случаях, например, когда все извлекаемые поля проиндексированы.

Рассмотрим следующий запрос SELECT:

SELECT * FROM tbl_name WHERE col1=val1 AND col2=val2;

Если таблица имеет множественный индекс (col1,col2), то соответствующие записи будут выбраны напрямую. Если существуют только одиночные индексы для col1 и col2, то оптимизатор сначала решит, при использовании какого индекса, количество возвращаемых записей будет меньше, а затем из этих записей будет произведена выборка по другому условию.

Если таблица имеет множественный индекс, то любой «левый префикс» этого индекса может использоваться для оптимизации запроса. Например, если есть индекс (col1, col2, col3), то можно считать, что существуют индексы (col1); (col1,col2); (col1,col2,col3).

Любая другая часть индекса не может быть использована для оптимизации. Рассмотрим для примера такие запросы:

mysql> SELECT * FROM tbl_name WHERE col1=val1;
mysql> SELECT * FROM tbl_name WHERE col2=val2;
mysql> SELECT * FROM tbl_name WHERE col2=val2 AND col3=val3;

Если есть индекс (col1,col2,col3), то только в первом запросе будет использоваться индекс. Хотя второй и третий запросы содержат столбцы, которые присутствуют в индексе, но (col2) и (col2,col3) не являются левыми частями множественного индекса, и поэтому при выполнении этих запросов индекс применятся не будет.

MySQL также использует индексы для LIKE операций, если аргумент LIKE является строковой константой и при этом не начинается с символа шаблона (% или _). Например, следующие SELECT запросы используют индекс для key_col:

mysql> select * from tbl_name where key_col LIKE "Patrick%";
mysql> select * from tbl_name where key_col LIKE "Pat%_ck%";

А следующие два запроса выполняются без использования индекса:

mysql> select * from tbl_name where key_col LIKE "%Patrick%";
mysql> select * from tbl_name where key_col LIKE other_col;

В первом из этих запросов аргумент после LIKE начинается с символа шаблона, а во втором аргумент не является константой.

Общие рекомендации по повышению производительности.

1. Запускать mysqld с правильно подобранными опциями (см. настройка переменных).
2. Для ускорения SELECT запросов построить индексы для тех полей, которые участвуют в условии WHERE.
3. Оптимизировать типы полей. По возможности использовать NOT NULL. (см. работу с таблицами).
4. В MySQL применяется два способа блокировки таблиц (lock table) – внутренняя и внешняя блокировки. Внутренняя блокировка позволяет делать операции по изменению/извлечению данных атомарными (не конфликтующими с другими пользователями). Внешняя блокировка применяется для одновременного доступа нескольких MySQL серверов к одним и тем же базам данных, а также внешняя блокировка позволяет запускать isamchk без остановки MySQL. Чтобы запретить использование внешней блокировки нужно запускать mysqld с опцией –skip-locking. Запрет внешней блокировки существенно **повысит** скорость работы, но при этом перед запуском isamchk нужно предварительно сбросить все данные на диск командой **`mysqladmin flush-tables`**. Также при запрете внешней блокировки нельзя будет использовать несколько серверов для работы с теми же базами данных.
5. Задание прав доступа на конкретную таблицу или поле снижает производительность.

Во второй части будет описана работа с блокировкой таблиц при изменении и добавлении данных и как это влияет на скорость работы.

Все замечания и пожелания принимаются по адресу vvtk@stealthcomp.com