ГЛАВА 5

Оптимизация настроек сервера

В этой главе мы объясним, как создать хороший файл конфигурации для вашего сервера MySQL. Этот процесс можно сравнить с круизом, включающим осмотр большого количества достопримечательностей и периодическими отклонениями от маршрута для посещения живописных мест. Эти отклонения от прямого пути необходимы, поскольку поиск кратчайшего пути к хорошей конфигурации начинается не с изучения параметров конфигурации и выяснения того, какие из них вы должны установить или как их изменить. Он также не начинается с изучения поведения сервера и выяснения того, могут ли какие-либо параметры конфигурации улучшить его. Вначале стоит разобраться с внутренним устройством и поведением MySQL. В дальнейшем можете использовать эти знания в качестве руководства по настройке MySQL. Наконец, вы можете сравнить желаемую конфигурацию с существующей и исправить любые существенные и важные для вас различия.

Нам часто задают вопросы примерно такого плана: «Каков оптимальный конфигурационный файл для моего сервера с 32 Гбайт оперативной памяти и 12-ядерным процессором?» К сожалению, на этот вопрос нет однозначного ответа. Вы должны настроить сервер в соответствии с рабочей нагрузкой, требованиями к данным и используемым приложениям, а не только к оборудованию. В MySQL есть множество настроек, которые можно изменить, но делать этого не стоит. В основном, лучше правильно настроить базовые параметры (а в большинстве случаев важны лишь некоторые из них) и потратить больше времени на оптимизацию схемы, индексы и проектирование запросов. Если вы правильно настроили основные параметры конфигурации MySQL, потенциальная выгода от дальнейших изменений обычно невелика.

В то же время манипулирование настройками может привести к огромным проблемам. Значения MySQL по умолчанию существуют по важной причине. Их изменение без понимания последствий может привести к сбоям, постоянным зависаниям или снижению производительности. Вы никогда не должны слепо доверять тому, что кто-то сообщает об оптимальной конфигурации на популяр-

ных справочных сайтах, таких как форумы MySQL или Stack Overflow[1]. Всегда проверяйте любые изменения, прочитав соответствующий раздел руководства и тщательно протестировав.

Так что же вам следует сделать? Удостоверьтесь в том, что значения основных параметров, таких как размер буферного пула InnoDB и файла журнала, соответствуют требованиям для вашей системы. Затем следует установить несколько параметров безопасности, если хотите обеспечить хорошую работу (но обратите внимание на то, что они обычно не улучшают производительность, а лишь предотвращают проблемы). Остальные настройки оставьте в покое. Если у вас возникла проблема, начните с ее тщательной диагностики. Если она обусловлена компонентом сервера, поведение которого можно исправить с помощью параметра конфигурации, может потребоваться изменить его.

Кроме того, иногда вам может потребоваться установить определенные параметры конфигурации, которые в особых случаях способны существенно повлиять на производительность. Однако учтите, что они не должны быть частью базового файла конфигурации сервера. Вы должны устанавливать их только в том случае, если обнаружите конкретные проблемы с производительностью, которые они могут решить. Вот почему мы рекомендуем не искать специально параметры конфигурации, которые можно улучшить, и не изменять их. Если какая-то проблема требует решения, она должна сначала проявиться во времени отклика на запрос. Лучше всего заниматься улучшениями, начав с запросов и времени ответа на них, а не с параметров конфигурации. Это поможет сэкономить много времени и предотвратит множество проблем.

Еще один хороший способ сэкономить время и усилия — использовать значения по умолчанию, если только вы точно не уверены в том, что их стоит изменить. Работа с настройками по умолчанию повышает безопасность системы, поскольку многие пользователи работают с ними. В итоге эти настройки можно считать наиболее тщательно протестированными. Когда же вы без необходимости что-то меняете, могут возникнуть неожиданные ошибки.

Основы конфигурации MySQL

Мы начнем с объяснения работы механизма конфигурирования MySQL, а затем рассмотрим, что нужно настраивать в MySQL. Как правило, MySQL довольно снисходительно относится к ошибкам конфигурации, но, следуя нашим рекомендациям, вы сэкономите много времени и сил.

---

[1] Например, MySQL может работать невероятно быстро, если вы отключите настройки устойчивости, однако из-за этого во время сбоя можете потерять свои данные.

Первое, что нужно запомнить, — это то, откуда MySQL получает информацию о конфигурации: из аргументов командной строки и параметров в файле конфигурации. В Unix-подобных системах файл конфигурации обычно находится в каталоге `/etc/my.cnf` или `/etc/mysql/my.cnf`. Если вы используете скрипты запуска, входящие в состав операционной системы, то обычно только в этом файле и надо определять параметры конфигурации. Если же запускаете MySQL вручную, например, при тестовой установке, то можете указать параметры также в командной строке. Сервер фактически считывает содержимое файла конфигурации, удаляет из него все строки комментариев и символы разрыва строки, а затем обрабатывает этот файл вместе с параметрами командной строки.

ЗАМЕЧАНИЯ ПО ТЕРМИНОЛОГИИ

Поскольку многие параметры командной строки MySQL соответствуют переменным сервера, мы иногда используем термины «параметр» и «переменная» как синонимы. Большинство конфигурационных переменных имеют те же имена, что и соответствующие им параметры командной строки, но есть несколько исключений. Например, `--memlock` устанавливает переменную `locked_in_memory`.

Любые настройки, которые вы хотите использовать постоянно, должны войти в глобальный файл конфигурации, а не указываться в командной строке. В противном случае вы рискуете случайно запустить сервер без них. Рекомендуем также хранить все ваши файлы конфигурации в одном месте, чтобы вы могли легко их проверять.

Убедитесь, что знаете, где находится файл конфигурации вашего сервера! Нам случалось встречать людей, безуспешно пытавшихся настроить сервер с файлом, который этот сервер и не собирался читать, например `/etc/my.cnf` на серверах Debian, где сервер ищет свой файл конфигурации `/etc/mysql/my.cnf`. Иногда такие файлы находятся в нескольких местах, возможно, из-за того, что предыдущий системный администратор тоже запутался. Если вы не знаете, какие файлы читает ваш сервер, то лучше у него же и спросить:

```
$ which mysqld
/usr/sbin/mysqld
$ /usr/sbin/mysqld --verbose --help | grep -A 1 'Default options'
Default options are read from the following files in the given order:
/etc/mysql/my.cnf ~/.my.cnf /usr/etc/my.cnf
```

Конфигурационный файл имеет стандартный формат INI и разбит на разделы (секции), каждый из которых начинается со строки, содержащей название секции в квадратных скобках. Программа в составе дистрибутива MySQL обычно

читает раздел с тем же именем, что и эта программа, а многие клиентские программы читают также раздел `client`, в который можно поместить общие для всех клиентов параметры. Сервер обычно считывает раздел `mysqld`. Убедитесь, что вы разместили свои параметры в правильном разделе файла, иначе они не будут иметь никакого эффекта.

Синтаксис, область видимости и динамичность

Конфигурационные параметры записываются строчными буквами, слова разделяются символами подчеркивания или дефисами. Следующие варианты эквивалентны, и вы можете увидеть обе формы в командных строках или конфигурационных файлах:

```
/usr/sbin/mysqld --auto-increment-offset=5
/usr/sbin/mysqld --auto_increment_offset=5
```

Мы рекомендуем выбрать один из этих стилей и использовать его последовательно. Это упрощает поиск конкретного параметра в ваших файлах.

Конфигурационный параметр может иметь несколько областей действия. Некоторые параметры действуют на уровне всего сервера (глобальная область видимости), другие могут задаваться по-своему для каждого соединения (сеансовая область видимости), а третьи относятся к конкретным объектам. Многие параметры сеансовой области видимости имеют глобальные эквиваленты, которые можно рассматривать как значения по умолчанию. Если вы измените переменную области сеанса, это повлияет только на то соединение, из которого вы ее изменили, и изменения будут утрачены при закрытии соединения. Вот несколько примеров разнообразного поведения, о которых вам следует знать.

- Переменная `max_connections` имеет глобальную область видимости.
- Переменная `sort_buffer_size` имеет глобальное значение по умолчанию, но также вы можете установить ее для каждого сеанса.
- Переменная `join_buffer_size` имеет глобальное значение по умолчанию и может задаваться для каждого сеанса, кроме того, для каждого запроса, в котором задействует несколько таблиц, может выделять один буфер для каждой *операции соединения*, поэтому для одного запроса могут существовать несколько буферов соединения.

Помимо установки переменных в конфигурационных файлах, вы можете изменять многие из них (но не все) во время работы сервера. В MySQL они называются *динамическими* переменными конфигурации. Следующие операторы

показывают различные способы динамического изменения значений переменной `sort_buffer_size` на уровне сеанса и глобально:

```
SET sort_buffer_size = <value>;
SET GLOBAL sort_buffer_size = <value>;
SET @@sort_buffer_size := <value>;
SET @@session.sort_buffer_size := <value>;
SET @@global.sort_buffer_size := <value>;
```

Если вы устанавливаете переменные динамически, имейте в виду, что эти настройки будут утрачены при завершении работы MySQL. Если хотите сохранить измененные параметры, запишите их в конфигурационный файл.

Если вы устанавливаете значение глобальной переменной во время работы сервера, значения для текущего сеанса и любых других существующих сеансов не затрагиваются. Имейте это в виду, если ваши клиенты используют постоянные соединения с базой данных. Это связано с тем, что значения сеансовых переменных инициализируются из глобального значения при создании соединений. После каждого изменения выполняйте команду SHOW GLOBAL VARIABLES, чтобы удостовериться, что изменение произвело желаемый эффект.

Существует также специальное значение `DEFAULT`, которое вы можете присвоить переменным с помощью команды `SET`. Присвоение данного значения сеансовой переменной устанавливает ее в соответствующее значение переменной глобальной области. Это полезно для сброса переменных области сеанса обратно к значениям, которые они имели при открытии соединения. Мы рекомендуем не использовать это значение по отношению к глобальным переменным: результат может отличаться от того, чего вы хотите. То есть вы не получите значения, которые были при запуске сервера, или даже значение, указанное в файле конфигурации, — переменная будет установлена в скомпилированное значение по умолчанию.

Сохраняемые системные переменные

Если уж на то пошло, работа с областью действия переменных и с конфигурацией не была слишком сложной и вы должны были узнать, что, если MySQL будет перезапущена, она вернется к тем параметрам, которые были в вашем файле конфигурации, даже если вы использовали `SET GLOBAL` для изменения глобальной переменной. Это означало, что, администрируя конфигурационный файл и конфигурацию среды выполнения MySQL, необходимо обеспечивать их синхронизацию друг с другом. Если вы хотели увеличить для своих серверов параметр `max_connections`, требовалось выполнить команду `SET GLOBAL`

max_connections для каждого запущенного экземпляра, а затем отредактировать файл конфигурации, чтобы отразить новую конфигурацию.

В MySQL 8.0 представлена новая функция, называемая сохраняемыми системными переменными (persisted system variables), которая помогает сделать это немного проще. Новая команда SET PERSIST теперь позволяет установить значение один раз во время выполнения, и MySQL запишет этот параметр на диск, чтобы его можно было использовать при следующем перезапуске.

Побочные эффекты установки переменных

Динамическая установка переменных может иметь неожиданные побочные эффекты, такие как сброс на диск изменившихся блоков из буферов. Будьте осторожны с настройками, которые вы меняете онлайн, потому что это может привести к значительной нагрузке на сервер.

Иногда назначение переменной можно понять по ее имени. Например, max_heap_table_size делает именно то, что подразумевает ее наименование: она указывает максимальный размер, до которого могут увеличиваться в памяти временные таблицы. Однако соглашения об наименованиях не всегда последовательны, поэтому вы далеко не всегда сможете догадаться о назначении переменной по ее названию.

Рассмотрим некоторые часто используемые переменные и последствия их динамического изменения.

- table_open_cache. Установка этой переменной не дает немедленного эффекта — действие откладывается до следующей попытки потока открыть таблицу. Когда это происходит, MySQL проверяет значение данного параметра. Если значение больше, чем текущее количество таблиц в кэше, поток может поместить вновь открытую таблицу в кэш. Если значение меньше, чем количество таблиц в кэше, MySQL удалит неиспользуемые таблицы из кэша.
- thread_cache_size. Установка этой переменной не дает немедленного эффекта — действие откладывается до момента следующего закрытия соединения. В это время MySQL проверяет, есть ли в кэше место для хранения потока. Если это так, то поток кэшируется для повторного применения в будущем другим соединением. Если нет, поток уничтожается вместо кэширования. В этом случае количество потоков в кэше и, следовательно, объем памяти, который использует кэш потоков, сразу не уменьшается — это происходит только тогда, когда новое соединение удаляет поток из кэша, чтобы задействовать его. (MySQL добавляет потоки в кэш только при закрытии соединений и удаляет их из кэша только при создании новых соединений.)

- `read_buffer_size`. MySQL не выделяет память для этого буфера до тех пор, пока он не понадобится запросу. Когда же необходимость возникает, MySQL немедленно выделяет весь блок запрошенного размера.
- `read_rnd_buffer_size`. MySQL не выделяет память для этого буфера до тех пор, пока он не понадобится запросу. Когда же необходимость возникает, MySQL немедленно выделяет весь блок запрошенного размера. (Название `max_read_rnd_buffer_size` более точно описывало бы эту переменную.)

Официальная документация MySQL подробно объясняет назначение этих переменных. И это не исчерпывающий список. Наша цель — просто показать, какого поведения следует ожидать при изменении нескольких важных переменных.

Не следует глобально увеличивать параметр, относящийся к соединению, если не уверены, что это правильно. Некоторые буферы выделяются сразу одним куском, даже если они не нужны, и глобальное изменение параметра может привести к растрачиванию памяти впустую. Вместо этого вы можете увеличить значение, когда это необходимо для конкретного запроса.

Планирование изменений ваших переменных

Будьте осторожны при установке переменных. Больше — не всегда лучше, и если вы установите слишком большие значения, то можете легко вызвать проблемы — у вас может не хватить памяти или сервер начнет выгружать ее в файл подкачки.

Как говорилось в главе 2, отслеживайте свои SLO, чтобы убедиться, что сделанные вами изменения не влияют на качество обслуживания клиентов. Эталонного тестирования недостаточно, поскольку оно не характеризует реальное использование сервера. Если не измерять фактическую производительность сервера, то можно ухудшить производительность, даже не подозревая об этом. Мы видели много случаев, когда кто-то менял конфигурацию сервера и думал, что это повысило производительность, хотя на самом деле производительность сервера в целом ухудшалась, поскольку в разное время дня или в разные дни недели рабочая нагрузка варьировалась.

В идеале необходимо использовать систему контроля версий для отслеживания изменений в файлах конфигурации. Эта стратегия может быть очень эффективной при сопоставлении изменения производительности или нарушения SLO с конкретным изменением конфигурации. Просто имейте в виду, что изменение файла конфигурации ничего не делает по умолчанию — нужно также изменить настройку среды выполнения.

Прежде чем приступить к изменению конфигурационных параметров, следует оптимизировать запросы и схему, обратившись хотя бы к таким очевидным

вещам, как добавление индексов. Если вы слишком углубитесь в настройку конфигурации, а затем измените свои запросы или схему, вам, возможно, потребуется начинать настройку заново. Имейте в виду, что, если ваши оборудование, рабочая нагрузка и данные не являются абсолютно неизменными, скорее всего, придется вернуться к конфигурированию позже. На самом деле большинство серверов не имеют постоянной рабочей нагрузки в течение дня, а это означает, что идеальная конфигурация для середины утра не подходит для полудня! Очевидно, что гнаться за мифической идеальной конфигурацией совершенно нецелесообразно. Таким образом, вам не нужно выжимать из своего сервера всю производительность до последней капли. Ведь в реальности отдача от таких затрат времени, вероятно, будет очень небольшой. Мы рекомендуем сосредоточиться на оптимизации пиковой рабочей нагрузки, а затем остановиться на приемлемом результате, если у вас нет оснований полагать, что можно добиться значительного повышения производительности.

Чего делать не следует

Прежде чем приступить к настройке сервера, мы хотим посоветовать вам избегать нескольких распространенных практик, которые, по нашему мнению, являются рискованными или практически не стоящими затраченных усилий. Предупреждение: громкие слова впереди!

Возможно, от вас ожидают (или полагают, что от вас ожидают) создания набора эталонных тестов и настройки сервера путем итеративного изменения его конфигурации в поисках оптимальных параметров. Обычно мы не советуем это делать. Это требует так много работы и исследований, а потенциальная отдача в большинстве случаев настолько мала, что много времени будет затрачено впустую. Вероятно, лучше потратить его на другие действия, такие как проверка резервных копий, мониторинг изменений в планах запросов и т. д.

Вы не должны производить тонкую настройку по коэффициенту. Классический пример коэффициента тонкой настройки — это эмпирическое правило, согласно которому коэффициент попаданий в кэш ключей InnoDB должен быть выше некоторого процента и следует увеличить размер кэша, если коэффициент попаданий слишком низкий. Это совершенно неправильный совет. Независимо от того, что вам говорят, *коэффициент попаданий в кэш не имеет никакого отношения к тому, слишком большой кэш или нет*. Начнем с того, что коэффициент совпадений зависит от рабочей нагрузки — некоторые рабочие нагрузки просто не кэшируются независимо от размера кэша, к тому же попадания в кэш бессмысленны по причинам, которые мы объясним позже. Иногда бывает, что, когда кэш слишком мал, коэффициент попадания низок, а увеличение размера

кэша увеличивает коэффициент попаданий. Однако это случайная корреляция, и ничто не говорит о производительности или правильном определении размера кэша.

Проблема с корреляциями, которые иногда кажутся верными, заключается в том, что люди начинают думать, что те всегда будут верны. Администраторы баз данных Oracle отказались от настройки на основе коэффициентов много лет назад, и мы хотим, чтобы администраторы баз данных MySQL последовали их примеру[1]. Еще больше мы желаем, чтобы люди не писали «скрипты настройки», которые кодифицируют эти опасные практики и обучают им тысячи людей. Из этого вытекает следующая рекомендация: не используйте скрипты настройки! Несколько очень популярных вы можете найти в Интернете. Вероятно, лучше всего их игнорировать.

Мы также предлагаем вам избегать словосочетания «тонкая настройка», которое часто употребляли в нескольких последних разделах. Вместо него стоит применять термины «конфигурация» или «оптимизация» (если это то, что вы на самом деле делаете). Когда мы видим словосочетание «тонкая настройка», воображение рисует образ недисциплинированного новичка, который настраивает сервер и смотрит, что происходит. В предыдущем разделе мы рекомендовали оставить эту практику тем, кто исследует внутреннее устройство сервера. «Тонкая настройка» вашего сервера может оказаться пустой тратой времени.

Что касается смежной темы, то поиск в Интернете рекомендаций по разработке конфигурации тоже не всегда является хорошей идеей. Вы можете найти много плохих советов в блогах, на форумах и т. д. Хотя многие эксперты делятся своими знаниями в Интернете, не всегда легко определить, кто является квалифицированным специалистом. Конечно, мы не можем дать объективных и беспристрастных рекомендаций о том, где найти настоящих специалистов. Но можем сказать, что заслуживающие доверия и авторитетные поставщики услуг MySQL в целом являются более безопасным выбором, чем результаты простого поиска в Интернете, потому что люди, у которых есть довольные клиенты, вероятно, делают что-то правильно. Однако даже их советы могут быть опасны, если их применять без проверки и понимания, потому что они могут относиться к ситуации, отличающейся от вашей, и неясным для вас деталям.

---

[1] Если вы не уверены, что тонкая настройка по коэффициенту — это плохо, прочтите книгу «Oracle. Оптимизация производительности» Кэри Миллсэп и Джеффа Холта (*Millsap Cary, Holt Jeff*. Optimizing Oracle Performance. — O'Reilly). Они даже посвятили этой теме приложение с инструментом, который может искусственно генерировать любое соотношение попаданий в кэш, какое вы пожелаете, независимо от того, насколько плохо работает ваша система! Конечно, все это для того, чтобы проиллюстрировать бесполезность такого соотношения.

Наконец, не верьте популярной формуле потребления памяти — да, той самой, которую выводит сама MySQL при сбое. (Мы не будем повторять ее здесь.) Она сохранилась с древнейших времен и не является надежным или хотя бы полезным способом понять, сколько памяти MySQL может использовать в худшем случае. Кроме того, вы можете найти некоторые варианты этой формулы в Интернете. Они также ошибочны, даже если в них добавлены факторы, которых нет в исходной формуле. По правде говоря, вы не можете установить верхнюю границу потребления памяти MySQL. Она не регулируется жестко сервером базы данных, который управляет распределением памяти.

Создание конфигурационного файла MySQL

Как мы предупреждали в самом начале этой главы, у нас нет универсального «наилучшего конфигурационного файла», скажем, для сервера с четырьмя процессорами, 16 Гбайт оперативной памяти и 12 жесткими дисками, который годился бы на все случаи жизни. Вам действительно придется разработать собственные конфигурации, потому что даже хорошая отправная точка будет сильно различаться в зависимости от того, как вы используете конкретное оборудование.

Минимальная конфигурация

Для этой книги мы создали минимальный пример файла конфигурации, который вы можете применять в качестве хорошей отправной точки для создания собственных серверов[1]. Вы должны выбрать значения для некоторых параметров, мы объясним их позже в этой главе. Наш базовый файл, созданный для MySQL 8.0, выглядит следующим образом:

```
[mysqld]
# GENERAL
datadir                          = /var/lib/mysql
socket                           = /var/lib/mysql/mysql.sock
pid_file                         = /var/lib/mysql/mysql.pid
user                             = mysql
port                             = 3306
# INNODB
innodb_buffer_pool_size = <value>
innodb_log_file_size             = <value>
innodb_file_per_table            = 1
innodb_flush_method              = O_DIRECT
# LOGGING
```

[1] Обратите внимание на то, что в новых версиях MySQL некоторые параметры удаляются, изменяются и устаревают. Проверьте детали в документации.