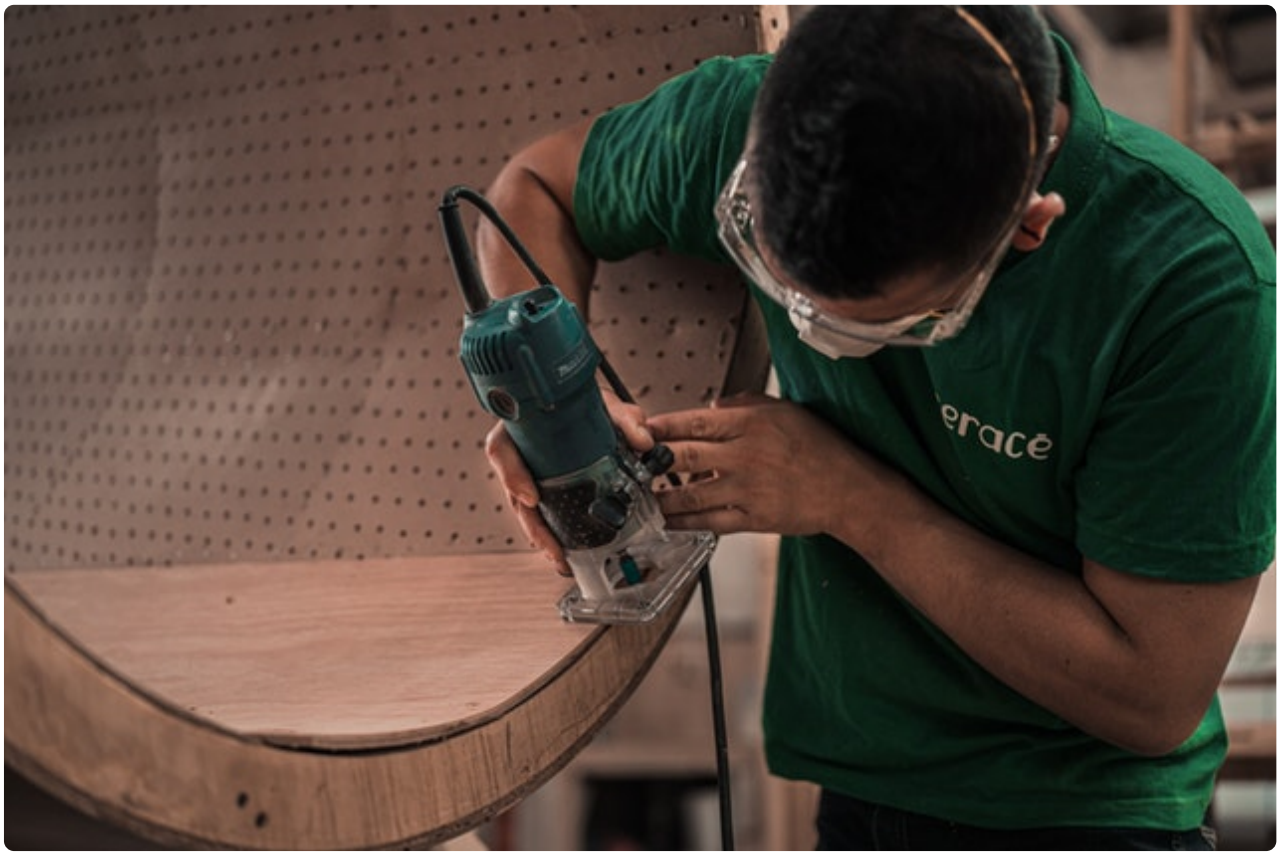
27 服务器性能调优，你知道怎么做吗？

更新时间：2020-05-11 09:27:46



> 最聪明的人是最不愿浪费时间的人。——但丁

之前所讲解的所有方案设计、调优手段都是针对于表设计或 SQL 语句的，这一节将思考的角度转移到 MySQL 服务器自身。想要调优 MySQL 服务器的性能，就必须要熟练掌握它的各项配置，即对 my.cnf 文件属性的理解。这一节我们来看一看影响 MySQL 性能的主要配置项，聊一聊怎样调整能够提高服务器性能。

1. 服务器性能调优指标

在实际的讲解调优参数之前，我们需要先搞清楚衡量 MySQL 性能的指标。在理解这些指标的基础之上，再去调整参数，才能确定我们的调整是否能给 MySQL 服务器带来性能提升。

1.1 QPS

QPS 是每秒查询次数，在软件工程中，我们通常用它来衡量应用的访问频率。这里，我们需要延伸下它的含义，即 MySQL 服务器能够处理的峰值 QPS（也是压测的一个重要指标）。影响 QPS 的因素主要有两个：服务器计算性能和磁盘的 IO 读写能力。对于计算性能，除了要考虑 SQL 查询的复杂度之外，还要考虑硬件的处理能力。而对于 IO 读写能力来说，完全取决于硬件。所以，绝大多数情况下，升级硬件就能够提升 QPS。

另外，MySQL 提供了 `STATUS` 命令可以让我们查看当前服务器的 QPS（最后的统计信息中打印了平均每秒查询次数）。如下所示：

```
mysql> STATUS
--------------
mysql  Ver 14.14 Distrib 5.7.28, for macos10.14 (x86_64) using  EditLine wrapper

Connection id:  5
Current database: imooc_mysql
Current user:  root@localhost
SSL:   Not in use
Current pager:  less
Using outfile:  ''
Using delimiter: ;
Server version:  5.7.28-log MySQL Community Server (GPL)
Protocol version: 10
Connection:  Localhost via UNIX socket
Insert id:  1
Server characterset: latin1
Db     characterset: latin1
Client characterset: utf8
Conn.  characterset: utf8
UNIX socket:  /tmp/mysql.sock
Uptime:   2 days 16 hours 36 min 10 sec

Threads: 3  Questions: 2183  Slow queries: 0  Opens: 167  Flush tables: 1  Open tables: 156  Queries per second avg: 0.009
--------------
```

1.2 TPS

TPS 是每秒处理的事务数，同样，我们需要把它的含义延伸为：MySQL 服务器能够处理的峰值 TPS。与 QPS 不同，TPS 是需要我们自己来计算的：

```
TPS = (Trans_commit + Trans_rollback) / Seconds
```

其中，Trans_commit 代表事务提交数，Trans_rollback 代表事务回滚数。作为衡量 MySQL 服务器的性能指标，它与 QPS 是类似的，值越大说明性能越高。

虽然升级硬件能够显著的提高 QPS 和 TPS，但并不是所有的企业都会这么做。如果你接触的工具或软件较多，你会知道，在相同的硬件配置条件下，通过调优配置参数也是可能给性能带来很大提升的。下面，我们就来看一看影响 MySQL 性能的常用配置参数。

2. MySQL 参数配置优化

MySQL 有着非常多的配置项，即提供的定制化功能非常丰富。但是，就像是惯例一样，常用的配置选项也不会很多。另外，根据配置项所负责的功能不同，可以把它们分类。下面，我们就来看一看各个功能点的常用配置参数以及如何对它们进行优化。

2.1 连接相关参数

连接指的是 MySQL 服务器与客户端之间的连接，相关参数的设定主要会影响客户端的行为。例如：是否能够连接上 MySQL 服务器、关闭交互连接前等待的时间等等。

2.1.1 max_connections 和 max_connect_errors

首先，所有的这些参数，如果之前没有做过改动（通过命令行或者是修改配置文件 my.cnf），则通过 `SHOW VARIABLES` 命令就可以看到其默认值。例如，我们可以看看这两个参数的默认值：

```
mysql> SHOW VARIABLES LIKE 'max_connect%';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| max_connect_errors | 100   |
| max_connections    | 151   |
+--------------------+-------+
2 rows in set (0.00 sec)
```

其中，max_connections 用于设定服务器的最大并发连接数，取值范围是 1 ~ 10 万，默认值是151。我们需要非常重视这个参数，因为它决定了 MySQL 允许多少个会话同时建立连接。几乎可以肯定，对于企业级应用来说，默认值是不够的的，我们通常把它设定为 500 ~ 1000。需要注意，建立连接除了需要占用内存之外，还要求有计算能力，所以，不要把这个参数设置的太大。

max_connect_errors 这个参数的名字非常形象，它用于指定允许连接不成功的最大尝试次数。它的取值范围是 1 ~ 2^64 之间，默认值是 100。需要注意的是，这个参数所表达的并不是 “单次连接最大的 Retry 次数”，而是尝试连接总的失败次数上限。如果尝试连接的错误数量超过了参数所设定的值，MySQL 服务器则拒绝新的连接。所以，我们通常会把这个参数设置的比较大，建议1万以上。

2.1.2 interactive_timeout 和 wait_timeout

我们大概率曾经遇到过这样的场景：打开的客户端或者程序在长时间没有被访问之后，需要重新与 MySQL 服务器建立连接。这其实是 MySQL 的一种优化保护机制，在超过了一定的时间始终没有使用之后，则认为当前的客户端已经不再使用了，断开连接回收资源。

interactive_timeout 和 wait_timeout 都与客户端会话的自动超时断开有关，其中 interactive_timeout 指的是服务器在关闭连接之前在一个交互连接上等待的秒数，默认值是 28800，即 8 个小时，建议将这个值调小。wait_timeout 则用于指定关闭非交互连接前等待的秒数，默认值仍是 8 个小时，可以根据业务场景适当的调大这个值。

可以查看这两个参数的默认值（注意，它们的单位都是秒），如下所示：

```
mysql> SHOW VARIABLES WHERE VARIABLE_NAME IN ('interactive_timeout', 'wait_timeout');
+---------------------+-------+
| Variable_name       | Value |
+---------------------+-------+
| interactive_timeout | 28800 |
| wait_timeout        | 28800 |
+---------------------+-------+
2 rows in set (0.00 sec)
```

2.1.3 back_log

我们在使用线程池的时候，如果任务加入的速度超过了线程处理的速度，那么，不断加入的新任务将会被 “丢到” 阻塞队列中去。与之类似，当短时间内有大量的连接请求，MySQL 主线程无法及时的分配时，就会将连接请求放入阻塞（等待）队列中。而这个等待队列的长度则由 back_log 参数控制。另外，它与线程池的思想还有一点类似，就是当队列满了之后，如果还有连接请求，则直接会报拒绝连接错误。

在我们当前的 5.7 版本中，这个参数的默认值是 80，最大值是 65535。我们通常不需要修改这个值，因为对于大多数 “正常” 的业务场景来说，短暂的建立连接与关闭连接会消耗很多性能，所以，很少会遇到这样的代码逻辑。

2.2 日志相关参数

MySQL 中最核心的两个日志是慢查询日志和 Binlog（二进制）日志。其中，慢查询相关的参数只有一个，即设定慢查询记录时间的阈值，这在“日志系统”和“慢查询问题”中都已经多次强调了，这里不再赘述。所以，下面，我们来看一看 Binlog 相关的参数。

2.2.1 log_bin

这个参数是标识是否打开 Binlog 的开关，8.0 之前的版本默认都是 OFF，即关闭状态。对于线上应用来说，Binlog 可以做增量数据收集、数据恢复、主从同步（主备环境）等等。所以，通常情况下，这个参数一定是设置打开的。另外，由于这是个只读参数，我们只能通过修改 MySQL 的配置文件来打开 Binlog。

2.2.2 max_binlog_size 和 expire_logs_days

不论是用什么语言、什么框架去编写应用程序，都一定会设计日志产出的方式和格式。其中，“方式”中包含两个重要的点：

- 日志怎样滚动，一般会有两种方式：按照大小、按照时间。当然，也可以结合大小和时间。例如：单个日志文件超过 1GB 就生成一个新的日志文件
- 日志清理机制，一般会设定为清理 X 天之前的日志，即只保留 X 天的日志

我们先看一看它们的默认值，如下所示：

```
mysql> SHOW VARIABLES WHERE VARIABLE_NAME IN ('max_binlog_size', 'expire_logs_days');
+------------------+------------+
| Variable_name    | Value      |
+------------------+------------+
| expire_logs_days | 0          |
| max_binlog_size  | 1073741824 |
+------------------+------------+
2 rows in set (0.00 sec)
```

expire_logs_days 的取值范围是 0 ~ 99，建议把它修改为 7 ~ 14 天；max_binlog_size 的取值范围是 4KB ~ 1GB，默认为 1GB（1024 * 1024 * 1024），可以直接使用默认值。

2.3 缓存相关参数

这里的缓存概念是显而易见的，它针对于用户的查询过程。合理的调整这些参数，有利于提高服务器的查询性能，毕竟从内存中取数据或在内存中完成计算，一定会比操作磁盘的过程更加高效。

2.3.1 have_query_cache、query_cache_type、query_cache_size 和 query_cache_limit

这几个都是与查询缓存相关的参数，其中，have_query_cache 用于标识当前的 MySQL 版本是否支持查询缓存。对于我们的 5.7 版本，一定是支持的，它的值是 YES。query_cache_type 控制着查询缓存功能的开启与关闭，它有三个可选项：

- **0（OFF）**：关闭
- **1（ON）**：打开
- **2（DEMAND）**：只有 SELECT 中明确指定 SQL_CACHE 时才缓存

这个参数需要特别注意，打开它需要去设置配置文件，之后再重启 MySQL 服务。如果直接在命令行上设定，则会返回错误。另外，如果想要关闭查询缓存，唯一的办法也是修改配置文件，在命令行中的修改也不会生效。

query_cache_size 用于指定缓存查询结果集的内存区大小，这个值需要设置为 1024 的整数倍。如果物理机内存足够，这个值可以设置的大一些。query_cache_limit 标识缓存单个查询所被允许的结果集最大值，即超出这个值之外的查询结果并不会缓存。它的默认值是 1MB，对于大多数的场景来说，基本是足够的，不需要调整。

2.3.2 sort_buffer_size

sort_buffer_size 是与排序相关的参数，指定单个会话能够使用的排序内存大小，默认值是 256KB（256 * 1024）。它是一个 connection 级别的参数，即在每一个连接第一次需要使用这个内存区的时候，MySQL 一次性分配所需的内存。关于这个值的设定问题，MySQL 文档中有这样一句话：

> On Linux, there are thresholds of 256KB and 2MB where larger values may significantly slow down memory allocation

它所表达的意思是，sort_buffer_size 并不是越大越好，当超过 2MB 时，则会使用 mmap() 而不是 malloc() 来进行内存分配，性能大约会降低 30% 左右。所以，将这个值设定为 256KB ~ 2MB 是一个不错的选择。

2.4 InnoDB 存储引擎相关参数

对于 MySQL 数据库来说，存储引擎就像是插件一样，是独立存在的。但是，由于 MySQL 与存储引擎是相辅相成的，我这里仍然把 InnoDB 相关的参数归类为 MySQL 服务器的参数（据统计，90% 的应用都会选择使用 InnoDB 存储引擎）。

2.4.1 innodb_buffer_pool_size

这是 InnoDB 的核心参数，用于指定 InnoDB 存储引擎专用的缓存区大小，默认值是 128MB（128 * 1024 * 1024），如下所示：

```
mysql> SHOW VARIABLES LIKE 'innodb_buffer_pool_size';
+-------------------------+-----------+
| Variable_name           | Value     |
+-------------------------+-----------+
| innodb_buffer_pool_size | 134217728 |
+-------------------------+-----------+
1 row in set (0.01 sec)
```

这个缓存区用来缓存表对象的数据及其索引信息，最大能够支持 (2^64 -1)B。如果我们的写（插入、更新、删除）事件比较多，那么，增大这个值会节省大量的磁盘 IO。另外，innodb_buffer_pool_size 是个全局参数，其所分配的内存将供所有的 InnoDB 表使用。所以，在条件允许的情况下，这个值越大越好（可以是物理机内存的 70% ~ 80%）。

那么，如何确定这个值我们设定的是否合理呢？所谓合理，也就是说数据从内存中读取而不是硬盘。我们可以查询系统变量 Innodb_buffer_pool_pages_free 看看当前 buffer pool 的剩余量。如下所示：

```
mysql> SHOW GLOBAL STATUS LIKE 'innodb_buffer_pool_pages_free';
+--------------------------------+-------+
| Variable_name                  | Value |
+--------------------------------+-------+
| Innodb_buffer_pool_pages_free | 7737  |
+--------------------------------+-------+
1 row in set (0.00 sec)
```

也就是说，对于我们当前的存储引擎环境，buffer pool 空间还有剩余。如果发现 Innodb_buffer_pool_pages_free 值很小或为 0，则说明 buffer pool 已经使用殆尽，需要增加 innodb_buffer_pool_size 的值。

2.4.2 innodb_max_dirty_pages_pct

这个参数用于控制脏页（在缓冲区中）的比例，取值范围是 1% ~ 100%，默认值是 75%。如果有大量写入，且写入数据并不活跃（报表型数据），可以将这个值调低，例如 30%。反之，如果写入或更新的数据是热数据，可以考虑调大这个值，例如 90%。

这个参数的另一个用途是：当 InnoDB 分配的内存过大时，会导致 Swap 占用严重，可以适当的减小这个值，使 Swap 空间释放出来。MySQL 官方对这个值的建议是不超过 90%，且不低于 15%。设置的太大，缓存中每次更新需要交换的数据页太多；设置的太小，数据页太小，更新操作缓慢。

3. 服务器调优案例

到目前为止，已经讲解了很多 MySQL 的参数配置项。那么，为了更好的理解它们，我们来看一个调优案例。需要注意的是，MySQL 自身也需要部署在物理机上，所以，参数配置同样需要去考虑物理机的配置。

3.1 参数配置调优

在调优参数配置之前，我们先来设定物理机的硬件条件。需要注意，由于 CPU 聚焦于计算，且其资源分配完全由操作系统控制，我们这里将直接忽略 CPU 相关的“内容”。

- 物理机内存有 64GB
- 峰值连接数为 500个
- 所有的表使用 InnoDB 存储引擎

首先，需要为操作系统预留 25% 的内存，即 64 * 0.25 = 16GB，余下 48GB。为了方便，我们假定这台物理机上只安装 MySQL 这一个应用，那么，余下的所有内存都可以分配给 MySQL 服务器（理想情况下）。

接下来，配置连接、日志和缓存相关的参数。如下所示（带有解释说明）：

```
# 最大连接数设定为峰值连接数的一倍
max_connections = 1000
# 调大连接不成功的最大尝试次数
max_connect_errors = 10000

# 超时断开使用默认值，即8小时
interactive_timeout = 28800
wait_timeout = 28800

# 等待队列可以使用默认值，也可以稍微调大一些
back_log = 100

# 开启 Binlog
log_bin = ON
# 单个 Binlog 文件最大为 1GB
max_binlog_size = 1024 * 1024 * 1024
# 保存近7天的 Binlog（完全看业务数据的重要程度，这里是一个常见值）
expire_logs_days = 7

# 开启查询缓存
query_cache_type = ON
# 查询缓存大小，物理机内存足够，可以多分配一些
query_cache_size = 256 * 1024 * 1024
# 单条语句结果集缓存限制，使用默认值，即1MB
query_cache_limit = 1024 * 1024

# 排序区大小，注意，这会分配给每个会话
sort_buffer_size = 2 * 1024 * 1024
```

除了当前配置的内存空间之外，每个会话在读写、保持连接上面也需要一定的内存，我们预估为 10MB。所以，当连接数达到峰值时，所有的会话占用的内存可以达到：500 * (2MB + 10MB) + 256MB = 6GB。那么，物理内存还剩下 48GB - 6GB = 42GB，可以全部分配给 InnoDB 的缓存池，相关的参数设定如下：

```
innodb_buffer_pool_size = 42 * 1024 * 1024 * 1024
```

3.2 数据预热

在生产环境中，重启 MySQL 之后，会发现查询过程将会变得比原来慢。这是因为 MySQL 经常操作的热点数据从 InnoDB Buffer Pool 中清空了，那么，从磁盘中读取数据的速度当然比不上内存。为了解决这个问题，MySQL 提出了预热的概念，即将热点数据重新加载到 InnoDB Buffer Pool 中。在 5.7 版本之前，这会是一个很麻烦的过程；而在 5.7 之后，这个功能就是默认开启的了。我们先来看看与 “数据预热机制” 相关的两个参数：

```
-- 值都是 ON，代表是开启状态
mysql> SHOW VARIABLES WHERE VARIABLE_NAME IN ('innodb_buffer_pool_dump_at_shutdown', 'innodb_buffer_pool_load_at_startup');
+---------------------------------------+-------+
| Variable_name                         | Value |
+---------------------------------------+-------+
| innodb_buffer_pool_dump_at_shutdown | ON    |
| innodb_buffer_pool_load_at_startup  | ON    |
+---------------------------------------+-------+
2 rows in set (0.01 sec)
```

其中，innodb_buffer_pool_dump_at_shutdown 所表达的意思是：关闭 MySQL 时导出 InnoDB Buffer Pool 中的数据。这份数据默认会保存在 InnoDB 的数据目录下，名字和路径由 innodb_buffer_pool_filename 参数控制，如下所示：

```
-- ib_buffer_pool 是 InnoDB Buffer Pool 中数据保存到磁盘上的文件名
mysql> SHOW VARIABLES LIKE 'innodb_buffer_pool_filename';
+-----------------------------+---------------+
| Variable_name               | Value         |
+-----------------------------+---------------+
| innodb_buffer_pool_filename | ib_buffer_pool |
+-----------------------------+---------------+
1 row in set (0.01 sec)
```

在启动 MySQL 之后，受 innodb_buffer_pool_load_at_startup 参数控制，会将 ib_buffer_pool 文件数据恢复到 InnoDB Buffer Pool 中。所以，我们在使用 MySQL 的时候，这两个参数最好使用默认值（打开状态），这也是服务器性能调优的重要一环。

4. 总结

我们在实际使用 MySQL 的时候，性能调优的时间都几乎花在了数据表和 SQL 语句上面，而服务器自身的性能调优却很容易被忽略。这确实是一个非常严重的问题，很可能就是由于参数配置的并不合理而导致了性能不足。所以，除了“表面上”的调优之外，也要考虑“根上”的调优。

5. 问题

谈一谈你对 MySQL 服务器参数的理解 ？

关于服务器性能调优，你有怎样的心得呢 ？可以一起分享下吗 ？

对于你当前正在使用的 MySQL，可以做一些调优工作吗 ？怎么做呢 ？

6. 参考资料

《高性能 MySQL（第三版）》

MySQL 官方文档：Server Status Variables

MySQL 官方文档：Server System Variables

MySQL 官方文档：Query Cache Configuration

MySQL 官方文档：Optimizing InnoDB Disk I/O

MySQL 官方文档：InnoDB Startup Configuration

MySQL 官方文档：How MySQL Uses Memory

MySQL 官方文档：InnoDB Startup Options and System Variables

MySQL 官方文档：Configuring InnoDB Buffer Pool Size

}

26 遇到慢查询问题，可以这样思考与解决

28 带你认识MySQL的系统架构