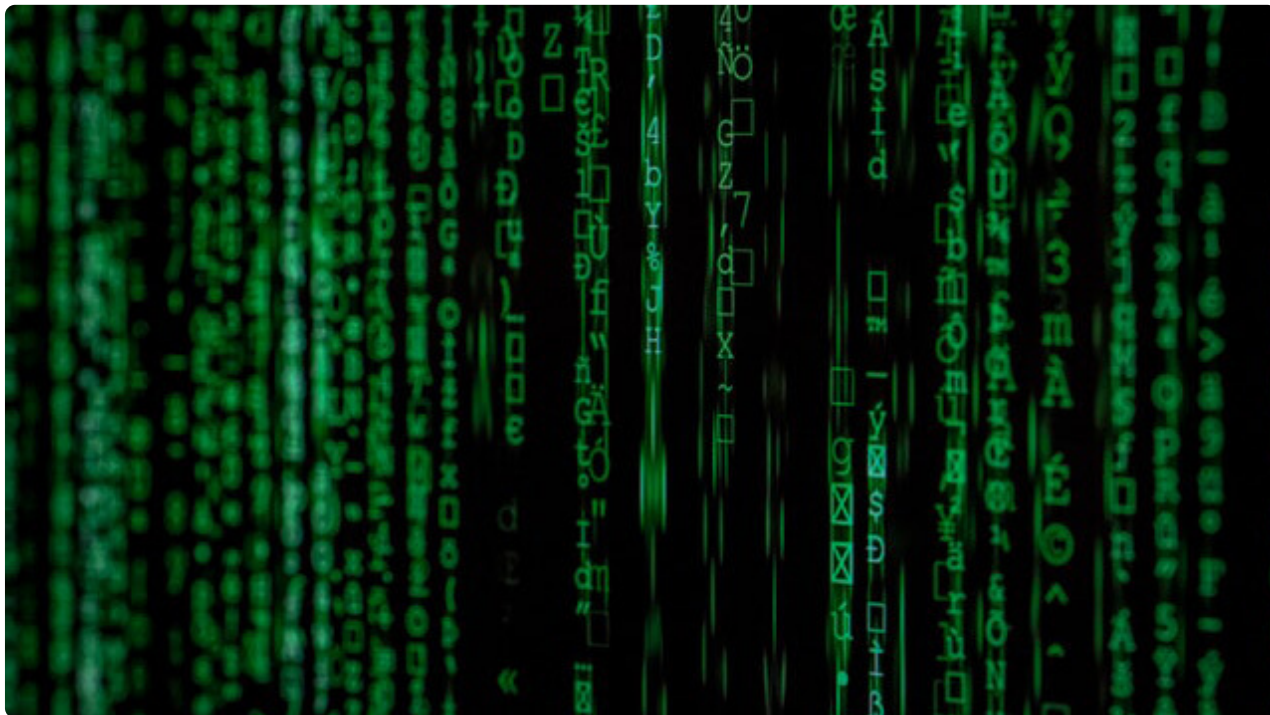


## 30 使用MySQL时，应用层可以这么优化

更新时间：2019-10-24 12:03:51



时间像海绵里的水，只要你愿意挤，总还是有的。

——鲁迅

对于数据库的使用，我们不仅仅要考虑 MySQL 本身的优化，还需要考虑应用层的优化。因为有些线上问题，就是由于应用层设置不合理导致的。

本节就来聊聊几种应用层的优化方法。

### 1 使用连接池

MySQL 如果频繁创建和断开连接，那 MySQL 的开销会比较大，可能会占用过多的服务器内存资源，甚至导致响应时间变慢。此时就可以考虑使用连接池来改进性能。

连接池可以理解为：创建一些持久连接的“池”，新的请求可以使用这些连接池，减少创建和断开连接的次数。

其大致原理是：

1. 当进程启动时，创建相应的数据库连接池对象；
2. 如果程序需要请求数据库，则直接从连接池获取到一个连接；
3. 数据库请求完成后，释放数据库连接池。

那么连接池会不会导致服务器连接过多呢？

通常情况下，连接池不会导致服务器连接过多，因为它们会在进程间排队和共享连接。

在《高性能 MySQL》第 14 章：应用层优化中就提到：当遇到连接池完全占满时，应该将连接请求进行排队，而不是扩展连接池。这样可以避免将压力都转到 MySQL 上而导致 MySQL 连接数过多。

## 2 减少对 MySQL 的访问

避免对同一行数据做重复检索，比如查询某个用户信息。

首先查出这个用户的联系方式：

```
select phone from user_info where user_id=111;
```

然后再查出这个用户的姓名：

```
select name from user_info where user_id=111;
```

显然上面的方式并不是最优的，可以将两条 SQL 合并成一条：

```
select phone,name from user_info where user_id=111;
```

然后返回给客户端。这样跟数据库建立连接的次数从 2 次降低到 1 次，从而节省了部分建立连接所花费的内存和时间。

## 3 增加 Redis 缓存层

在很多业务场景，Redis 充当着不可或缺的角色。这里介绍几种通过 Redis 环境 MySQL 压力的场景：

### 3.1 计数器

在专栏第 9 节，就提到了使用 Redis 做计数器的场景。当统计数增加，则在 Redis 中执行下面的命令让计数器加 1：

```
INCR t1_count
```

通过这种方式缓解在 MySQL 中执行 update 的压力。

### 3.2 K-V 数据缓存

在 MySQL 中，如果某个字段会被频繁查询，而该字段内容变化的概率又不是很大，就可以考虑使用 Redis 缓存。比如电商业务，查看上个季度某类型商品的销量排行，如果这一个功能放在主页，点击量可能会非常高，因此可以考虑放在 Redis 中。

### 3.3 消息队列

Redis 中可以非常方便的使用消息队列。

生产者通过 lpush 将消息放在 list 中，消费者通过 rpop 取出该消息。

我曾经工作的一家公司就使用 **Redis** 实现短信消息队列。如果用户在 **APP** 上点击注册用户名密码，需要填下手机号验证，程序会将该用户的手机号放在 **Redis** 的 **list** 中，然后另外一个程序一直去消费 **list** 中的手机号，取出手机后，则调用第三方短信接口，发送手机短信息给到用户。

## 4 单表过大及时归档

比如单张表过大，可能有下面这些影响：

1. 在修改表结构时导致长时间主从延迟；
2. 备份时间过久；
3. 查询速度可能也会变慢。

因此，可以考虑对历史数据归档（比如日志数据），控制单表的数据量。

## 5 代码层读写分离

在配置了 **MySQL** 主从环境的情况下，可以考虑使用读写分离，通过程序配置的这种方式，在专栏第 27 节有提到过。更新走主库，查询走从库。

当然，主从同步可能因为大事务或者网络等原因导致同步延迟，在使用读写分离是也需要考虑到延迟这一点。

## 6 表的索引提前规划

在专栏第 10 节中，我们讲到了为什么添加索引能提高查询速度？因此，条件字段有索引显得格外重要。

当开发或者 **DBA** 在创建新表时，就应该考虑在表的条件字段添加合适索引。这样可以避免业务上线后，数据量一上来就出现大量慢查询而导致 **MySQL** 服务器高负载。

## 7 总结

本节讲解了使用 **MySQL** 时，应用层的一些优化。

提到了以下几点优化方案：

- 1、使用连接池；
- 2、减少对 **MySQL** 的访问；
- 3、增加 **Redis** 缓存层；
- 4、单表过大及时归档；
- 5、代码层读写分离；
- 6、表的索引提前规划；

.....

实际工作中也是，不单单 **DBA** 要去优化 **MySQL**，开发也应该考虑在应用层去做一些优化，以保证业务稳定高效。

## 8 问题

你觉得连接 MySQL 的应用层还有哪些情况是可以去优化改善的？欢迎在留言区讨论。

## 9 参考资料

《深入浅出 MySQL》第 23 章：应用优化

《高性能 MySQL》第 14 章：应用层优化

}

← 29 如何安全高效地删除大量无用数据？

31 MySQL整体优化思路 →

