# 概述

当有大量请求并发访问时，一定伴随着资源的不断申请和释放，导致资源的利用率低，降低服务质量。线程池通过预先创建一定数量的线程，当有请求到达时，线程池分配一个线程提供服务，请求结束后，该线程又去服务其他请求，通过这种方式，避免了线程和内存对象的频繁申请和释放，降低了服务端的并发度，减少了上下文切换和资源的竞争，提高资源利用效率。

所有服务的线程池本质都是为了提高资源利用效率，并且实现方式也大致相同。

在MySQL5.6之前，MySQL处理连接的方式是One-Connection-Per-Thread，即对于每一个数据库连接，MySQL-Server都会创建一个独立的线程服务，请求结束后，销毁线程。再来一个连接请求，则再创建一个连接，结束后再进行销毁。这种方式在高并发的情况下，会导致线程的频繁创建和释放。当然，可以通过thread-cache将线程缓存起来，以供下次使用，避免频繁地创建和释放的问题，但是无法解决高连接数的问题。One-Connection-Per-Thread方式随着连接数暴增，导致需要创建同样多的服务线程，高并发线程意味着高内存消耗，更多的上下文切换（CPU cache命中率降低）以及更多的资源竞争，导致服务出现抖动。相对于One-Connection-Per-Thread方式，一个线程对应一个连接，Thread-Pool实现方式中，线程处理的最小单元不再是连接而是statement（语句），一个线程可以处理多个连接的请求。这样，在保证充分利用硬件资源情况下（合理设置线程池大小），可以避免瞬间连接数暴增导致的服务器抖动。

# 调度方式

MySQL-Server同时支持3种连接管理方式，包括No-Thread，One-Thread-Per-Connection和Pool-Threads。No-Threads表示处理连接使用主线程处理，不额外创建线程，这种方式主要用于调试；One-Thread-Per-Connection是线程池出现以前最常用的方式，为每一个连接创建一个线程服务；Pool-Thread是线程池方式。

MySQL-Server通过一组函数指针来同时支持3中连接管理方式，对于特定的方式，将函数指针设置成特定的回调函数，连接管理方式通过thread\_handing参数控制，代码如下：

if (thread\_handling <= SCHEDULER\_ONE\_THREAD\_PER\_CONNECTION)

one\_thread\_per\_connection\_scheduler(thread\_scheduler,

&max\_connections,

&connection\_count);

else if (thread\_handling == SCHEDULER\_NO\_THREADS)

one\_thread\_scheduler(thread\_scheduler);

else

pool\_of\_threads\_scheduler(thread\_scheduler, &max\_connections,&connection\_count);

# 连接管理流程

1、通过poll监听mysql端口的连接请求；

2、收到连接后，调用accept接口，创建通信socket；

3、初始化thd实例，vio对象等；

4、根据thread\_handling方式设置，初始化thd实例的scheduler函数指针；

5、调用scheduler特定的add\_connection函数新建连接。

# 线程池参数

thread\_handling:表示线程池模型。

thread\_pool\_size:表示线程池的group个数，一般设置为当前CPU核心数目。理想情况下，一个group一个活跃的工作线程，达到充分利用CPU的目的。

thread\_pool\_stall\_limit:用于timer线程定期检查group是否“停滞”，参数表示检测的间隔。

thread\_pool\_idle\_timeout:当一个worker空闲一段时间后会自动退出，保证线程池中的工作线程在满足请求的情况下，保持比较低的水平。

thread\_pool\_oversubscribe:该参数用于控制CPU核心上“超频”的线程数。这个参数设置值不含listen线程计数。

threadpool\_high\_prio\_mode:表示优先队列的模式。

# 线程池实现