<https://segmentfault.com/a/1190000007515396>

学习网络编程时，自己动手实现一个Web Server是一个很有意思的经历。大多数Web Server都有一个特点：在单位时间内需要处理大量的请求，并且处理这些请求的时间往往还很短。《深入理解计算机系统》 (CSAPP) 在讲解网络编程时实现了一个经典的Web Server，这个Web Server不仅满足了静态请求，同时还满足了动态请求 (CGI)。虽然这个Web Server能够正常使用，但是仍存在一个明显的缺陷：它是一个迭代式的Web Server，这意味着在一个请求处理完毕前，不能同时处理另一个请求，而我们之前提到Web Server的一个重要特点就是在单位时间内可能会有大量的请求，所以如果投入工业界，这种情况自然是无法容忍的。

### 多进程 Web Server 模型

解决上面提到的Web Server只能一个接着一个处理请求的第一个方案是：当accept到一个请求时，fork一个子进程去处理这个请求，而主进程仍然在监听是否有新的连接请求。多进程模型在表面上看似乎解决了问题，但是我们都知道fork一个进程的开销是非常大的，基于以下几个事实。

* 从概念上说，可以将fork认作对父进程程序段、数据段、堆段以及栈段创建拷贝。但是如果真的只是简单的将父进程虚拟内存页拷贝到子进程，那就太浪费了。现代UNIX(Linux) 在实现fork时往往会采用两种技术来避免这种浪费。一是内核将每一进程的代码段标记为只读，从而使得父进程和子进程都无法修改代码段。这样，父进程和子进程可以共共享同一代码段。二是对于父进程数据段、堆段和栈段中的各页，内核采用写时复制(copy-on-write) 的方式，这么做的原因之一是：fork之后常常伴随着exec，这会用新程序替换进程的代码段，并重新初始化其数据段、堆段和栈段。但是无论如何，仍存在复制页表的操作，这也是为什么在UNIX(Linux) 下创建进程要比创建线程开销大的原因。
* 并发量一大，此时系统内便会有存在大量的进程，这会导致CPU花费大量的时间在进程调度上，并且进程上下文的切换开销也很大。

因此，相比于多进程模型，多线程是一个更优的模型：创建线程要快于创建进程，线程间的上下文切换消耗的时间一般也比进程要短。

### 多线程 Web Server 模型

换用多线程Web Server模型：每accept一个请求，创建一个线程，将请求交由该线程处理。换用多线程模型可以解决由fork带来的开销问题，但是调度问题依然还是存在的。因此，一个显而易见的解决办法是使用线程池，将线程的数量固定下来。基本的实现思路如下。

* 将每个请求封装为一个Job，每个Job包含线程要执行的方法、传递给线程的参数以及用于描述该Job处于Job队列的位置的参数。
* 线程池维护着一个Job队列，每个线程从Job队列中取下一个Job执行。因为该Job队列是一个共享资源，因此需要控制线程的同步。
* 初始化线程池时，马上创建一定数量的线程。此时，这些线程都是阻塞状态的，因为Job队列为空。

### 代码实现

[tinyhttpd](https://github.com/tinylcy/tinyhttpd)是我为了更有效的学习网络编程而实现的一个轻量级的Web Server，目前仍有部分问题需要解决以及优化。按照上面的思路，我实现了一个简单的线程池，并将其引入到[tinyhttpd](https://github.com/tinylcy/tinyhttpd)中。具体的代码实现请参考[threadpool.h](https://github.com/tinylcy/tinyhttpd/blob/master/threadpool.h)和[threadpool.c](https://github.com/tinylcy/tinyhttpd/blob/master/threadpool.c)。

### 剩余问题

当固定了线程池的线程数量后，仍然存在一个严重的问题：实际情况下，很多连接都是长连接，这意味着一个线程在处理一个请求时，read到的数据将会是是不连续的。当线程处理完一批数据后，如果继续read，而下一批数据还未到来时，由于默认情况下file descriptor是blocking的，因此该线程就会进入阻塞状态。所以，如果线程池中所有的线程都处于阻塞状态，此时如果有新的请求到来，那么是无法处理的。

解决方案是将file descriptor设置为non-blocking，利用事件驱动(Event-driven)来处理连接。

### 参考

* [Linux/UNIX系统编程手册](https://book.douban.com/subject/25809330/)
* [深入理解计算机系统](https://book.douban.com/subject/5333562/)
* [zaver](https://github.com/zyearn/zaver)