Nginx与Apache学习

1. Nginx和Apache的作用

Nginx和Apache都是http server，在tomcat之前，对用户的http请求做一些操作，比如用户认真模块，域名主次映射模块，静态数据缓存，图片的防盗链等功能。

2. Nginx和Apache的区别

主要的区别还是在于两者使用的网络IO事件机制不同，Nginx使用的是基于epoll系统调用的事件机制，而Apache使用的是select/poll的IO机制，所以Nginx在处理大量并发网络连接的情况下，在性能方面更有优势；而且使用epoll的策略，能够管理的连接fd是没有上限的。

3. Linux缓存IO

用户请求的IO数据，会先存放在内核空间中，当内核空间中的数据准备完毕后，会将数据从内核空间拷贝到用户空间中，用户的系统调用返回。

4. Linux提供的五种网络IO模式的方案

A. 阻塞IO：使用recvfrom系统调用后，用户进程会等待对应的数据，在内核空间准备完毕（整个等待过程中进程是阻塞的）后，将内核空间中的数据拷贝到用户空间中。

B. 非阻塞IO：使用非阻塞IO，用户进程不会阻塞住等待内核空间数据，而是去做别的事情，每隔固定的时间，调用read看一下数据的准备情况，如果还未准备好，返回error，如果已经准备好了，则立刻开始拷贝到用户空间。（定时查看需要用户进程自己实现）

C. IO多路复用：select/poll/epoll。单个process处理多个网络连接的请求（文件描述符），整个过程和阻塞IO相似，一个进程会拥有一个阻塞的队列，通过select等待其中任何一个IO请求数据准备完毕后，用户进程再调用read获取数据。

当请求连接数很少的情况下，IO多路复用的方式性能还不如多核下的阻塞IO；但是当并发请求很多的情况下，多路复用的方法就能发挥出他性能的优势了。

D. 异步IO：用户的read会立刻返回，内核在准备完数据后，会自行将数据拷贝到用户空间，然后通过signal通知用户进程。

5. IO多路复用select/poll/epoll

Select：将文件描述符分成三类writefds、readfds、和exceptfds，调用后select会阻塞，直到有描述符就绪，select默认可以监听1024个文件描述符，select返回后，需要遍历fd队列，找到一个就绪的描述符。

Poll：struct pollfd \*fds使用指针来描述fd队列，poll返回后，也需要遍历fd队列，找到一个就绪的描述符。

因为select和poll方法，在返回后都需要遍历监听队列，获得一个就绪的描述符，所以在请求数较多的情况下，性能会变差。

Epoll：给每个fd注册了回调函数，在fd就绪后，操作系统会自行调用注册的回调函数，因此当并发请求很多的情况下，性能会比每次都去遍历fd的队列来的高。（**很复杂，不是很明白**）

理想的网络连接情况是，虽然有大量的socket连接在，但是其中大部分是死亡的，只有少部分是活跃的，在这种情况下，使用Epoll的策略，性能会大大超过使用select/poll。然而，如果大量socket请求都是活跃的，或者是只有少量的请求的情况下，这三种策略的性能都差不多，甚至epoll的性能可能会低于select/poll。