C11 高并发服务器

多进程并发服务器

在多进程并发服务器中，若有用户请求到达，服务器将会调用fork()函数，创建一个子进程，之后父进程将继续调用accept()，而子进程则去处理用户请求。

说明：

（1）多进程并发服务器效率高且更加稳定，服务器中的进程不会受其它进程状态的影响；

（2）多进程并发服务器中进程数量受可打开文件描述符的限制；

（3）多进程并发服务器中进程数量受内存容量限制。

多线程并发服务器

考虑到每个进程可打开的文件描述符数量有限，且进程占用资源较多，系统中进程的数量又受到内存大小的限制，为在保证服务器效率的前提下，降低服务器的消耗，可利用多线程机制搭建并发服务器。

与多进程服务器相比：

（1）线程占用的空间资源大大减少，因此内存堆服务器的限制也被降低；

（2）多线程并发服务器稳定性较差。

因此在搭建服务器时，应从需求出发，选择更为合适的服务器。

线程池

当服务器程序启动后，预先在其中创建一定数量的线程，并将这些线程依次加入队列中。

在没有客户端请求抵达时，线程队列中的线程都处于阻塞状态，此时这些线程只占用一些内存，但不占用cpu。

若随后有用户请求到达，由线程池从线程队列中选出一个空闲线程，并将用户请求传给选出的线程，由该线程完成用户请求。

用户请求处理完毕，该线程并不退出，而是再次被加入线程队列，等待下一次任务。

此外，若线程队列中处于阻塞状态的线程较多，为节约资源，线程池会自动销毁一部分线程。

若线程队列中所有线程都有任务执行，线程池会自动创建一定数量的新线程，以提高服务器效率。

I/O多路转接服务器

为进一步提升服务器效率，人们提出了一种被称为I/O多路转接的模型。其中“多路”指代连接到服务器的多个客户端程序，而“转接”则是指在服务器主线与各分支之间设置一个“岗位”，由该岗位实现监控多路连接中数据状态的功能，若某路连接中数据就绪，就通知服务器，使主程序对该路请求作出处理。

与多进程和多线程并发服务器相比，I/O多路转接服务器实现了I/O多路复用，系统不必创建多进程或多线程，也不必维护多个进程或线程，因此大大降低了系统开销。

Linux系统中提供了select()、poll()和epoll()函数来实现I/O多路转接，下面将对这几种机制进行详细讲解与演示。