多路复用

底层原理：

IO多路复用使用两个系统调用select/poll/epoll和recvfrom，bio只调用了recvfromselect/poll/epoll/核心是可以同时处理多个connection。而不是更快，所以如果连接数不高不一定比多线程加上阻塞io的效率更高，每个socket，设置为non-blocking，阻塞是被select阻塞而不是socket本身。

1. select机制

客户端操作服务器是会产生三种文件描述符简称（fd），分别为：writefds（写）、readfds（读）、exceptfds（异常）。select会阻塞并监视这三类fd，等有数据可读、可写或者异常、超时（timeout）时就会返回，返回后便利fdset整个数组来找到就绪的文件描述符（fd），然后进行对应的IO操作。

优点：

几乎可以在全平台上支持，平台扩展性强。

缺点：

由于是采用轮询的方式全盘扫描，所以会随着文件描述符的增加而性能下降。

每次调用select，需要把fd集合从用户态拷贝到内核态或者从内核态拷贝到用户态，并进行遍历，占用cpu资源，降低性能。

默认单线程打开的文件描述符的数量是1024个，虽然可以通过宏定义，但是效率依然很低。

1. poll机制

基本原理与select一致，只是没有最大文件描述符的限制，而是采用了链表存储fd。

1. epoll机制

epoll高性能的三个函数

1. epoll\_create()：系统启动时，在linux内核里申请一个b+树结构的文件系统，返回一个epoll对象，这个对象也是fd。

2）epoll\_ctl每新建一个连接，都通过该函数操作epoll对象，在这个对象里面修改添加删除对应的链接fd, 绑定一个callback函数。  
3）epoll\_wait() 轮训所有的callback集合，并完成对应的IO操作

****优点：****  
　　没fd这个限制，所支持的FD上限是操作系统的最大文件句柄数，1G内存大概支持10万个句柄  
　　效率提高，使用回调通知而不是轮询的方式，不会随着FD数目的增加效率下降  
　　内核和用户空间mmap同一块内存实现

ET、LT：

epoll水平触发（level trigger，LT，LT为epoll的默认工作模式）与边缘触发（edge trigger，ET）两种工作模式。

LT：只要内核缓冲区有数据就一直通知，只要socket处于可读状态或可写状态，就会一直返回sockfd；

ET：只有状态发生变化才通知，只有当socket由不可写到可写或由不可读到可读，才会返回其sockfd；

水平触发

1. 对于读操作

只要缓冲内容不为空，LT模式返回读就绪。

2. 对于写操作

只要缓冲区还不满，LT模式会返回写就绪。

边缘触发

1. 对于读操作

（1）当缓冲区由不可读变为可读的时候，即缓冲区由空变为不空的时候。

（2）当有新数据到达时，即缓冲区中的待读数据变多的时候。

（3）当缓冲区有数据可读，且应用进程对相应的描述符进行EPOLL\_CTL\_MOD 修改EPOLLIN事件时。

2. 对于写操作

（1）当缓冲区由不可写变为可写时。

（2）当有旧数据被发送走，即缓冲区中的内容变少的时候。

（3）当缓冲区有空间可写，且应用进程对相应的描述符进行EPOLL\_CTL\_MOD 修改EPOLLOUT事件时。

究竟触发了多少次呢？可以这样想：

（1）当数据量SIZE小于recv函数的接收长度len且小于系统缓冲SO\_RCVBUF的时候：ET与LT都触发一次，recv执行一次。

（2）当数据量SIZE大于recv接收长度len但小于系统缓冲SO\_RCVBUF的时候：LT触发SIZE/len次，ET触发一次。两者的recv函数执行SIZE/len。（一次ET触发可执行多次recv）

（3）当数据量SIZE大于recv接收长度len且大于系统缓冲SO\_RCVBUF的时候：LT触发SIZE/len次，ET触发SIZE/SO\_RCVBUF次。两者的recv函数都执行SIZE/len次。（同样一次ET触发执行了多次recv）

显然：

1、ET模式可以显著降低同一个Epoll事件的触发次数。

2、ET模式下同一文件描述符的同类事件有可能多次触发。如：系统读/写缓冲区溢出会分别导致同一个描述符的EPOLLIN/EPOLLOUT事件多次触发。

另外几点要知道：

（1）send和recv函数的len参数是每一次接受或发送的数据量的上限。

（2）ET模式下确实每次都终止于EAGAIN。recv的指定的接收长度较小的时候，ET模式下的一次触发会执行多次recv。直到某次recv操作把errno变成了EAGAIN为止。此时errno==EAGAIN条件成立。

（3）这种触发是系统自动完成的。某些程序在errno==EAGAIN的时候，又做了写epoll\_ctl配置有点误导人。

（4）两种模式下的recv函数执行的次数都是一样的。

（5）具体执行次数还要考虑缓冲区门限之类的，并非这么简单。