会议纪要

时间：2009年11月17日星期二

地点：

与会：许式伟、莫华枫、段雪涛

主题：epoll驱动的socket io基础设计

结构

核心是epoll驱动的Socket IO模块。通过类SocketIODriver实现。在一个应用程序中，此类只有一个实例，存放于IoService中，负责所有socket的通讯。

此外，SocketFileObject类包装了每一个Socket链接的fd，io操作等。SocketFileObject通过SocketIODriver访问epoll，实现SocketIO。

运行过程：

程序启动时，创建一个SocketIODriver的实例，在构造过程中，创建epoll句柄（析构中释放）。同时SocketIODriver启动一个线程，并且建立循环，执行epoll\_wait()，以及相关的read/write/accept，以及epoll event控制操作。

在需要实现Socket IO时，首先创建SocketFileObject类，然后用一个socket句柄open此对象。一旦调用SocketFileObject的read/write成员，这些函数会在read或write队列中写入一个消息。该消息包含buffer指针，期望完成字节数，实际完成的字节数（包括出错信息），以及一个fiber指针。然后构造epoll\_event：对于read操作，除了设置EPOLLIN之外，如果write队列存在消息，则同样设置EPOLLOUT；对于write亦然。然后将SocketFileObject对象自身的指针赋予event.data.ptr中。接着，将事件注册到epoll中去(如何注册，参见下文）。最后，调用yield()将控制权转交给IOCP。

SocketIODriver的epoll循环被阻塞在epoll\_wait()调用上，直到事件到达（存在一次到达多个event的可能性）。一旦epoll\_wait()返回，便可以遍历所得到的事件，执行相应的操作。如果返回的events设置了EPOLLIN，则执行read操作，如果设置了EPOLLOUT，则执行write操作。执行操作时，可以从event对象的data.ptr中获得io操作消息，从中获得buffer指针等参数，用于read/write操作。完成后，从read或write队列中删除相应的消息。随后，程序从io操作消息中提取SocketFileObject对象指针，生成新的epoll event对象，在epoll中注册。（如果SocketFileObject对象的两个队列都没有消息，则删除对应的事件）。完成后，删除对应的io消息。将io消息中的fiber post到IoService中去。IoService将继续执行read/write中后续的代码。

SocketFileObject

此类封装了一个socket句柄，及其相关操作。数据成员主要包括：

1）SocketDesc，socket句柄

2）ReadOpQueue，读消息队列

3）WriteOpQueue，写消息队列

4）OldEvents，上一次事件注册的操作变量

5）mutex，锁

主要成员函数：

1）OpenHandle(HANDLE socket)，打开句柄

2）read\_some(void\* buf, size\_t cb)，读取数据，不保证读取全部数据。

3）write\_some(void\* buf, size\_t cb)，写入数据，不保证写入全部数据。

4）read(void\* buf, size\_t cb)，读取数据，保证读取全部数据。

5）write(void\* buf, size\_t cb)，写入数据，保证写入全部数据。

read\_some()/write\_some()的基本流程：

1）构造read/write消息

2）lock

3）向ReadOpQueue/WriteOpQueue写入read消息

4）生成epoll::events，位操作。需检测WriteOpQueue/ReadOpQueue队列是否存在消息。

5）调用SocketIODriver::BindEvent()，注册epoll事件

6）unlock

7）yield()

8）返回结果整理

read()/write()流程类似，但需多次迭代

read/write消息

read/write消息包含以下成员：

1）fiber，产生消息的fiber的指针

2）buf，数据缓存

3）bufsz，数据缓存大小

4）result，操作所处理的数据量

SocketIODriver

此类封装epoll相关的操作，建立线程，执行epoll循环和io操作。数据成员主要包括：

1）epollfd，epoll句柄

主要成员函数：

1）BindEvent()，注册epoll事件

epoll循环基本流程：

1）epoll\_wait()

2）迭代所发生的event

3）提取SockFileObject对象指针，lock

4）若EPOLLIN被设置，执行recv()操作，schedule fiber，完成后删除read消息。

4）若EPOLLOUT被设置，执行send()操作，schedule fiber，完成后删除write消息。

5）检测read和write消息队列，构造并注册epoll event。

6）unlock

7）返回1）

epoll事件注册

epoll事件通过epoll\_ctl()函数注册。有三种操作：EPOLL\_CTL\_ADD, EPOLL\_CTL\_MOD, EPOLL\_CTL\_DEL。为提高效率，减少系统调用，引入一个oldEvents变量。该变量保存最近一次事件注册的epoll\_event::events的值。通过比较当前准备注册的events值和oldEvents，决定对epoll\_ctl()的调用：

1）events == oldEvents，不做调用

else

2）events == 0，以EPOLL\_CTL\_DEL调用epoll\_ctl()

else

3）oldEvents == 0 && events != 0，以EPOLL\_CTL\_ADD调用epoll\_ctl()

else

4) oldEvents !=0 && events != 0，以EPOLL\_CTL\_MOD调用epoll\_ctl()

这个算法可以使得io操作越密集，epoll\_ctl()调用越少。

注：asio中使用直白的两级调用：先MOD，如果event不存在，再做一次ADD调用。

锁的使用

由于存在read/write两个队列，以及oldevents变量，在read/write调用时，只能锁住整个SocketFileObject对象。在epoll循环中，需要在注册epoll事件和删除io消息时，锁住SocketFileObject对象。由于加锁时间段的操作简短，不会对其他线程产生影响。