基于 IOCP 的通用异步 Windows Socket TCP 高性能服务端组件的设计与实现

一、 线程结构

与组件相关的线程有 3 种：使用者线程、Accept 线程和工作线程，其中后 2 种由组件实现。

1、 **使用者线程**：通过调用 Start/Stop/Send 等组件方法操作组件的一个或多个线程，通常是程序的主线程或其它业务逻辑线程。

2、 **Accept 线程**：使用 AcceptEx() 接收客户端连接请求并创建 Client Socket 的线程，将其独立出来，实现为单独的线程将使组件的模块划分更清晰，更重要的是避免与业务逻辑和通信处理的相互影响。

3、 **工作线程**：使用 GetQueuedCompletionStatus() 监听网络事件并处理网络交互的多个线程，工作线程处理完网络事件后会向上层应用发送 OnAccept/OnSend/OnReceive 等组件通知。工作线程的数量可以根据实际情况之行设置（通常建议为：CPU Core Number * 2 + 2）。

注意：如果上层应用在接收到 OnAccept/OnSend/OnReceive 这些组件通知时直接进行业务逻辑处理并在其中操作组件，则工作线程也成为了使用者线程。另外，如果要处理的业务逻辑比较耗时，上层应用应该在接收到组件通知后交由其他线程处理。

二、 性能

组件采用 Windows 平台效率最高的 IOCP Socket 通信模型，因此在通信接口的性能方面是有保证的，这里就不多说了。现在从组件的设计与实现的角度来来阐述性能的优化。组件在代码级别做了很多优化，一些看似多余或繁琐的代码其实都是为了性能服务；组件在设计方面主要采用了 2 中优化策略：缓存池和私有堆。

1、 **缓存池**：在通信的过程中，通常需要频繁的申请和释放内存缓冲区（TBufferObj）和 Socket 相关的结构体（TSocketObj），这会大大影响组件的性能，因此，组件为 TBufferObj 和 TSocketObj 建立了动态缓存池， 只有当缓存池中没有可用对象时才创建新对象，而当缓存对象过多时则会压缩缓存池。

2、 **私有堆（Private Heap）**：在操作系统中，new / malloc 等操作是串行化的，虽然一般的应用程序不用太在乎这个问题，但是在一个高并发的服务器中则是个不可忽略的问题，另外 TBufferObj 和 TSocketObj 均为大小固定的结构体，因此非常适合在私有堆中分配内存，避免与 new / malloc 竞争同时又减少内存空洞。（关于私有堆的使用方法请参考这里 ^_^）

三、 通用性与可用性

与《通用异步 Windows Socket TCP 客户端组件的设计与实现》描述的客户端接口一样，服务端组件也提供了两组接口：ISocketServer 接口提供组件操作方法，由上层应用直接调用；IServerSocketListener 接口提供组件通知方法，由上层应用实现，这两个接口设计得非常简单，主要方法均不超过 5 个。由于组件自身功能完备（不需要附带其它库或代码）并且职责单一（只管通信，不参与业务逻辑），因此可以十分方便第整合到任何类型的应用程序中。

四、 伸缩性

可以根据实际的使用环境要求设置工作线程的数量、 TBufferObj 和 TSocketObj 缓存池的大小、TBufferObj 缓冲区的大小、Socket 监听队列的大小、AccepEx 派发的

数目以及心跳检查的间隔等。

五、连接标识

组件完全封装了所有的底层 Socket 通信，上层应用看不到任何通信细节，不必也不能干预任何通信操作。另外，组件在 IServerSocketListener 通知接口的所有方法中都有一个 Connection ID 参数，该参数作为连接标识提供给上层应用识别不同的连接。