基于ACE反应式框架的服务器模型设计

Design of Server Model Base on ACE Reactor Framwork

谭汉松 董翔宇 陈林书
Tan Hansong Dong Xiangyu Chen Linshu
(中南大学信息科学与工程学院，长沙 410083)
(School of Information Science and Engineering, Central South University,Changsha 410083)

摘 要：本文应用ACE反应式框架的事件多路分离技术实现一个简单服务器模型的设计，着重探讨ACE反应式框架的内部结构，以及如何利用该框架去处理并发连接，对不同的事件类型进行多路分离，没有涉及具体业务层面的内容。

关键词：ACE; 反应器; 服务器模型; 事件多路分离
中图分类号：TP3 文献标识码：A 文章编号：1671-4792-(2006)7-0066-03

Abstract: Applying ACE Reactor framwork's event demultiplexe technology to implement a simple server model. In this paper,we will emphasizely discuss ACE Reactor framework's architecture,and using the framework to deal with subsequent connection and event demultiplexe.To these problems such as content about idiographic operation, we will ignore it.

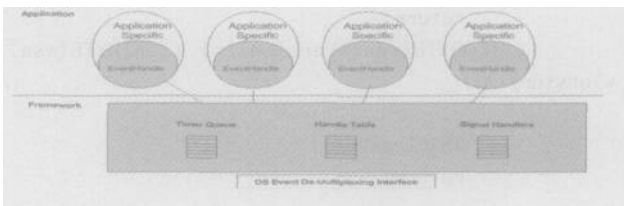
Keywords: ACE; Reactor; Server Model; Event Demultiplexe

0 引言

在传统的基于网络的应用中，为了处理多个I/O源，比如多个网络连接，一般的方法是创建新的进程（每个连接一个进程模型）或者线程（每个连接一个线程模型）。如果服务器需要处理多个并发的网络连接，这种做法特别流行，在大多数情况下这些模型都能很好的工作。但是，进程或线程创建及维护的开销可能让人无法接受，并且在许多应用中，对线程或进程的管理与控制会增加代码的复杂度。ACE(Adaptive Communication Environment)的反应式框架则很好的解决了上述问题，该框架为高效的事件多路分离和分派提供了可扩展的面向对象构架，其本质上提供了一组更高级的编程抽象，简化了事件驱动的分布式应用的设计和实现。除此之外，ACE的反应式框架还将若干不同类型的事件多路分离集成到了易于使用的API中，有应用处理此事件的特定代码。

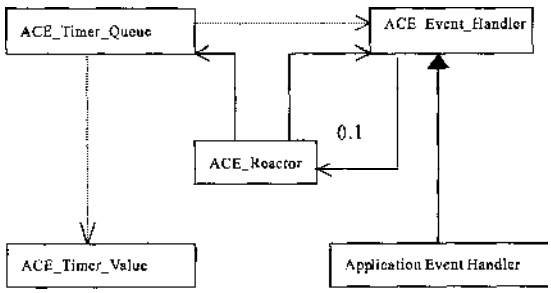
1 反应式框架总览

如图一所示，ACE中的反应堆与若干内部和外部组件协同工作。其基本概念是反应堆构架检测事件的发生（通过在OS事件多路分离接口上进行侦听），并发出对预登记事件处理器(event handler)对象中的方法的“回调”(callback)。该方法由应用开发者实现，其中含要记录哪一个事件处理器将被回调，它需要知道所有事件处理器对象的类型。这是通过替换模式的帮助来实现的（即，通过“是……类型”(is a type of)变种继承)。该构架提供名为ACE_Event_Handler的抽象接口类，所有应用特有的事件处理器都必须由此派生（这使得应用特有的处理器都具有相同的类型，即ACE_Event_Handler，所以它们可以相互替换)。其中的事件处理器的椭圆形包括灰色的event_Handler部分，对应于ACE_Event_Handler；以及白色的部分，它对应于应用特有的部分。



图一 反应堆中的内部组件和外部组件的协作

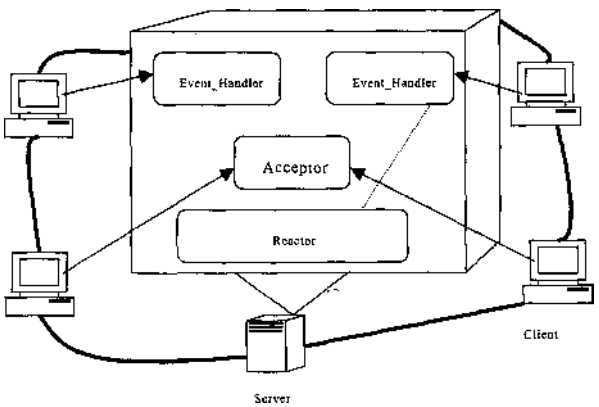
该框架所涉及的主要类关系如图二所示：



图二　框架的类关系

2　服务器模型的总体设计

服务器模型的体系结构如图三所示:



图三　服务器模型体系结构

总体的设计步骤如下:

①创建接受连接事件的处理器,以处理所连接到达的事件;

②创建连接服务事件的处理器,以处理所连接到达后I/O句柄上的读写事件或特殊信号事件;

③在反应器上登记,当有相应的事件到达时,通知对应的事件处理器,同时传递它想要的用以处理该事件的事件处理器指针给反应器;

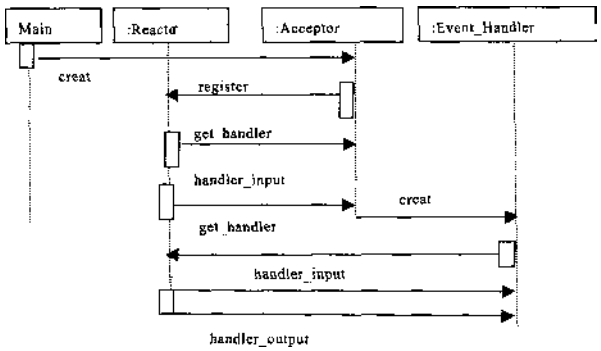
④反应器构架将自动地在内部维护一些表,将不同的事件类型与事件处理器关联起来,在用户已经登记的某个事件发生时,反应器发出对处理器中相应的方法回调。

服务器内部时序图如图四所示,具体的实现将在后面做详细介绍。

3　服务器模型的详细设计

3.1 Acceptor类

该类用于侦听连接,并对连接到达事件进行处理和分派,继承自ACE_Event_Handler:



图四　服务器内部时序

```
Class Acceptor : public ACE_Event_Handler
{ Public :
   Virtual ~Acceptor ();
   Int open( const ACE_INET_Addr &listen_addr);
   Virtual ACE_HANDLE get_handle (void) const;
    Virtual int handle_input (ACE_HANDLE fd = ACE_INVALID_HANDLE);
     Virtual int handle_close (ACE_HANDLE handle, ACE_Reactor_Mask close_mask);
  Protected:
    ACE_SOCK_Acceptor acceptor_;
};
```

类主要接口描述:

Open()方法用于在指定的地址进行侦听,并调用ACE_Reactor::register_handler()向反应器注册,让反应器监视接受器句柄上的ACCEPT事件。

handler_input():当有ACCEPT事件到达时,反应器会自动回调该挂钩方法,在具体实现这个方法时,我们实例化一个新的Event_Handler对象,调用Event_Handler类的open()方法,准备对客户的下一个动作进行分派相应的处理方法。

handle_close():一种隐式拆除处理器的方法,当服务(handle_input方法)完成或者有异常发生时,该挂钩方法被调用,将事件处理器从内存中删除,系统将回收句柄等资源。

3.2 Event_Handler 类

该类用于在连接建立后对连接的请求进行处理,处理来自Acceptor的委托,同样,该类也是继承自ACE_Event_Handler:

```
Class Event_Handler : public ACE_Event_Handler
```

```
{ Public :
    Int open(void);
    Virtual ACE_HANDLE get_handle (void) const;
     Virtual int handle_input (ACE_HANDLE fd = ACE_INVALID_HANDLE);
Virtual int handle_output (ACE_HANDLE fd = ACE_INVALID_HANDLE);
    Virtual int handle_close (ACE_HANDLE handle, ACE_Reactor_Mask close_mask);
  Protected:
     ACE_SOCK_Stream sock_;
    ACE_Message_Queue<ACE_NULL_SYNCH> output_queue_;
};
```

类接口描述：该类虽然和Acceptor类的成员很类似，但其功能却相差很大

open()方法：向反应器注册事件类型，初始化事件处理器挂钩方法，等待注册的事件类型的到达，本文所描述的服务器模型事件类型为“可读”(READ_MASK)和“可写”(WRITE_MASK)。

handle_input()：处理输入，当socket可读时，该挂钩方法被调用，用户（程序编写者）可根据需求对该方法进行重载，以满足实际的需要，例如对受到的数据进行处理等。返回值若为小于0时，事件处理器会调用handle_close()方法对处理器进行隐式拆除。

handle_output()：处理输出，当socket可写时，该挂钩方法被调用，用户（程序编写者）可根据需求对该方法进行重载。返回值若为小于0 时，事件处理器会调用handle_close()方法对处理器进行隐式拆除。

3.3 main程序

在主程序中，首先设定侦听端口，然后声明接受器对象，向反应器登记，最后运行反应器的事件循环。

```
int main(int argc,char *argv[])
{
ACE_INET_Addr listen_port(8080);
Acceptor my_acceptor;
my_acceptor.reactor (ACE_Reactor::instance());
if(my_acceptor.open(listen_port) = = -1)
  return 1;
ACE_Reactor: : instance()->run_reactor_event_loop();
return 0;
}
```

4 结束语

ACE反应式框架是一个非常强大和灵活的系统，它能够处理来自多个来源的事件，提高了应用系统的并发处理能力，并且不会带来多线程的开销。本文着重介绍了如何灵活的使用框架的各挂钩回调方法以及事件多路分离技术进行服务器模型的设计，通过测试，该模型能很好的处理并发连接，并且能够对连接到达后的读写行为作出及时的处理。在实际应用中，服务器设计要复杂的多，但ACE提供的反应式框架依然能很好的满足设计者的需求。

参考文献

[1]王继刚，顾国昌.构架与模式在通信系统软件中的应用研究[J].计算机应用，2003，11:43-45.

[2]Douglas C. Schimidt, Stephen D. Huston. C++ Network Programming，Volume 1:Mastering Complexity with ACE and Patterns[M].北京:清华大学出版社，2003，12.

[3]Douglas C. Schimidt, Stephen D. Huston. C++ Network Programming，Volume 2:Systematic Reuse with ACE and Frameworks[M].北京:清华大学出版社，2004，2.

[4]Stephen D. Huston，等著.马维达译.ACE程序员指南[M].北京:中国电力出版社，2004，11.

作者简介

董翔宇(1979—)，男，河南周口人，硕士研究生，主要研究方向：计算机应用、计算机网络与通信；

陈林书（1981—），男，湖南岳阳人，硕士研究生，主要研究方向：计算机网络通信与安全；

谭汉松(1948—)，男，湖南湘乡人，教授，主要研究方向：计算机应用、网络体系结构。