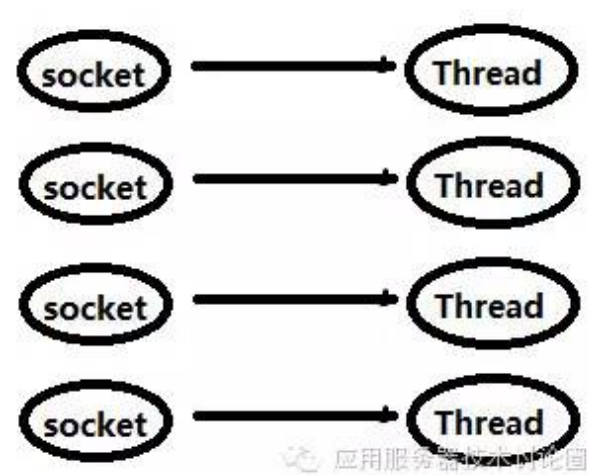


# 浅谈应用服务器前端框架常见的Reactor架构模式

2016-02-03 17:27 feiying 0 阅读 146

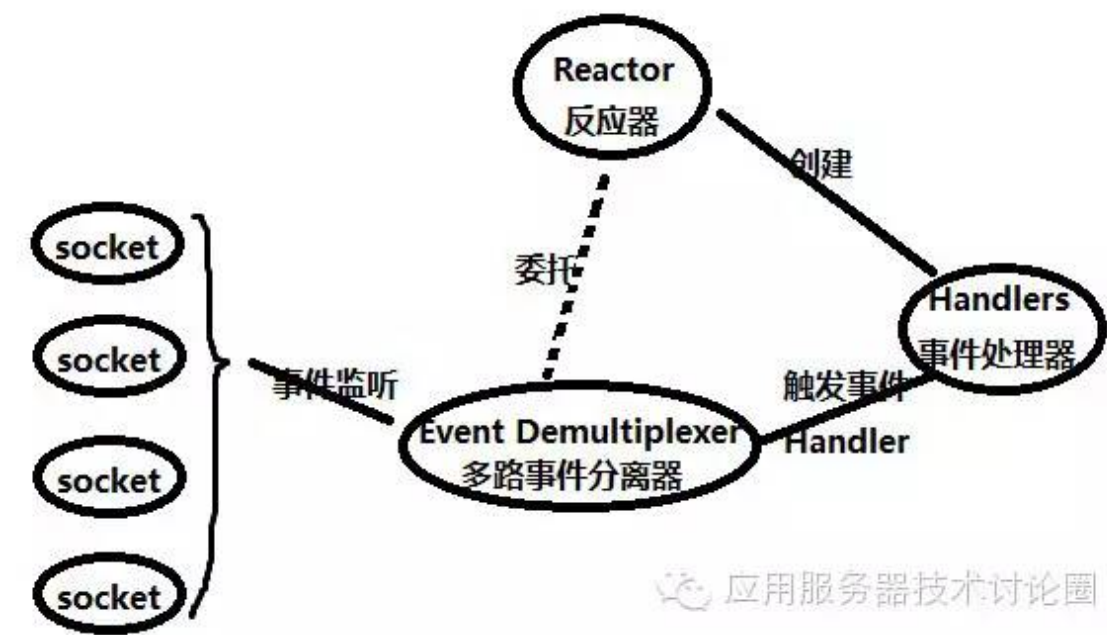
Reactor模式是一个架构模式，它主要解决的问题是高并发场景下的服务器前端的性能问题。

举个例子，原来的服务器与客户端的链接是一对一的，也就是说一个客户端socket接到后，对应一个线程去接收和处理：



这种模式的好处，是思路很清晰，一个线程处理一个socket请求，但是这种太消耗线程资源，因为毕竟socket不是实时都有数据接入的，

网卡属于是典型的慢速设备，因此，如何能高效的利用一些模式，可以改变这种局面，这个时候Reactor模式就出来了。



上述的图就是基于socket的Reactor模式的模拟。 [下载《开发者大全》](#) [下载dwm17.dev.apk](#)

socket作为文件fd，被多路事件分离器监听，这个多路事件分离器说大白话就是Select，poll这种IO多路复用的机制，select将事件分离出来后，

委托给反应器去寻找创建对应的事件处理器，当创建完成后，以后有socket事件进来，直接由select分离出来，交由对应的Handler去处理。

Reactor模式其实最初其实来源于 华盛顿大学 Douglas C.Schmidt的一篇论文：

## Reactor

An Object Behavioral Pattern for  
Demultiplexing and Dispatching Handles for Synchronous Events

Douglas C. Schmidt

[schmidt@cs.wustl.edu](mailto:schmidt@cs.wustl.edu)

Department of Computer Science

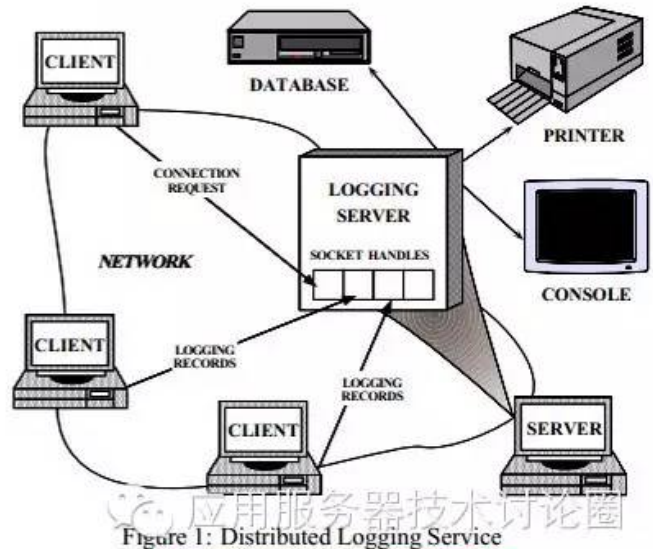
Washington University, St. Louis, MO<sup>1</sup>

An earlier version of this paper appeared as a chapter in the book "Pattern Languages of Program Design" ISBN 0-201-6073-4, edited by Jim Coplien and Douglas C. Schmidt and published by Addison-Wesley, 1995.

### 1 Intent

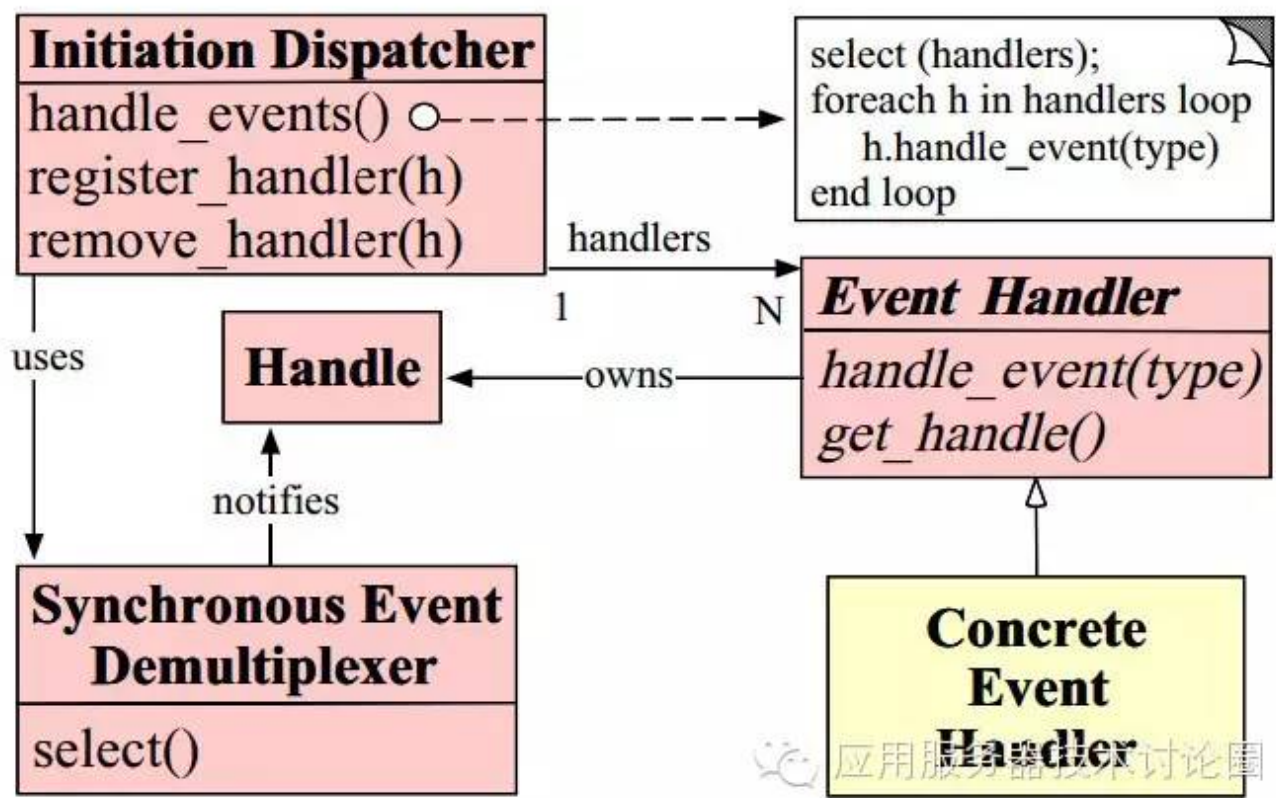
The Reactor design pattern handles service requests that are delivered concurrently to an application by one or more clients. Each service in an application may consist of several methods and is represented by a separate event handler that is responsible for dispatching service-specific requests. Dispatching of event handlers is performed by an initiation dispatcher, which manages the registered event handlers. Demultiplexing of service requests is performed by a synchronous event demultiplexer.

### 2 Also Known As



最早的解决问题是分布式日志系统，问题和前面的问题类似，不能每一个线程针对于慢速的设备，如上图所示，慢速设备有数据库，还有更慢的打印机。

这篇论文中的类图比较精确：



标准的Reactor模式图一共有5个角色：

**描述符（handle）：**

由操作系统提供，用于识别每一个事件，如Socket描述符、文件描述符等。在Linux中，它用一个整数来表示。

事件可以来自外部，如来自客户端的连接请求、数据等。事件也可以来自内部，如定时器事件。

====》实际就是一个文件描述符fd，socket描述符。

**同步事件分离器（demultiplexer）：**

是一个函数，用来等待一个或多个事件的发生。调用者会被阻塞，直到分离器分离的描述符集上有事件发生。

Linux的select函数是一个经常被使用的分离器。

====》实际就是select这种IO多路复用机制。

**事件处理器接口（event handler）：**

是由一个或多个模板函数组成的接口。这些模板函数描述了和应用程序相关的对某个事件的操作。

===》业务需要实现的接口

**具体的事件处理器：**

下载《开发者大全》

下载 (/download/dev.apk)

是事件处理器接口的实现。它实现了应用程序提供的某个服务。

每个具体的事件处理器总和一个描述符相关。它使用描述符来识别事件、识别应用程序提供的服务。

===》业务handler的类

**Reactor 管理器 ( reactor ) :**

定义了一些接口，用于应用程序控制事件调度，以及应用程序注册、删除事件处理器和相关的描述符。它是事件处理器的调度核心。

Reactor管理器使用同步事件分离器来等待事件的发生。

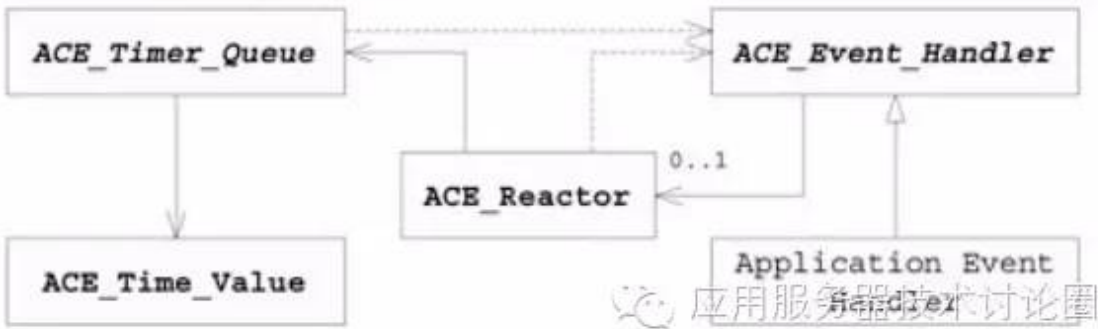
一旦事件发生，Reactor管理器先是分离每个事件，然后调度事件处理器，最后调用相关的模板函数来处理这个事件。

===》管控整个Reactor的具体的类。

对于类图的描述已经比较清晰了，其实对Reactor实现的最好实现在C++的ACE框架上，ACE框架用于替代传统socket这种编程方式的，提供了很多封装的模式的框架，其中有一个模式，就叫Reactor框架。

上述的几个角色，在Reactor框架都有非常明确的对应关系，但是ACE的整体类图和Reactor还略有区别，加了一些基于时间的事件调度的内容。

ACE Reactor 框架 类关系图



Reactor对应的就是ACE的Reactor，负责handler注册，和整个Reactor模式的管控；

具体的事件源，也就是handle，并没有在这张图中出现，其实也就是对应的fd，和socket的文件描述符等事件产生源；

多路复用事件分离器，其实就是操作系统中的select，poll等机制，因为ACE框架是对操作系统层面的系统调用的封装，因此对应的操作系统的机制并没有在ACE类图中。

Event Handler接口就是对应的事件接口，ACE将这个Reactor的接口标准化了，供应用自己去实现自己的Event Handler；



Application Event Handler是对应的应用的事件Handler，继承自Event Handler接口；

上图中其实已经可以满足Reactor模式了，但是对于ACE Reactor框架，还提供了诸如定时任务队列等的支持，见上面类图的左侧部分，功能可谓非常强大。

在Java中，其实大家都非常比较清楚了，NIO其实也比较类似Reactor模式。

Selector可以理解为操作系统底层映射到java API层级的一个多路复用事件的分离器。

SelectionKey其实就是对应的事件，如socket read，socket listen，accept等等。


当对应的事件触发之后，NIO需要你自己针对于SelectionKey中的事件进行if else的判断，而这部分就没有用到Handler，

当然，如ServerSocketChannel，FileChannel对应的事件就那么几个，完全自己可以封装一个完美的Reactor框架。

**总结一下，Reactor模式产生的年头较早，它最初解决的问题是一客户端连接一线程模型种，针对于慢速客户端如打印机等的线程时间片的浪费问题，**

**后期逐渐演变为服务器前端架构模式，ACE，JAVA的NIO都是基于Reactor的实现案例。**

分享：

阅读 146  0

应用服务器技术讨论圈 更多文章

东方通加码大数据业务 拟募资8亿收购微智信业 (/html/308/201504/206211355/1.html)

玩转Netty – 从Netty3升级到Netty4 (/html/308/201504/206233287/1.html)

金蝶中间件2015招聘来吧！Come on！ (/html/308/201505/206307460/1.html)

GlassFish 4.1 发布，J2EE 应用服务器 (/html/308/201505/206323120/1.html)

Tomcat对keep-alive的实现逻辑 (/html/308/201505/206357679/1.html)

猜您喜欢

我眼中的移动网络测试 (/html/85/201509/207781073/1.html)

下载《开发者大全》

下载 (/download/dev.apk)

谷歌的人力资源部门是如何运转的？ (/html/32/201401/100009472/1.html)
福利、福利、福利！重要的事情要说三遍。 (/html/346/201604/403486904/1.html)
项目敏捷成熟度模型解读 (/html/168/201604/2649652396/1.html)
有没有主宰世界的主算法? (/html/315/201512/401652290/1.html)