


文章 (/blogs) > 大CC (/blog/me115) > 文章详情

IO设计模式：Reactor和Proactor对比 (/a/1190000002715832)

 大CC (/u/me115) 271 4月28日 发布

推荐

2 推荐

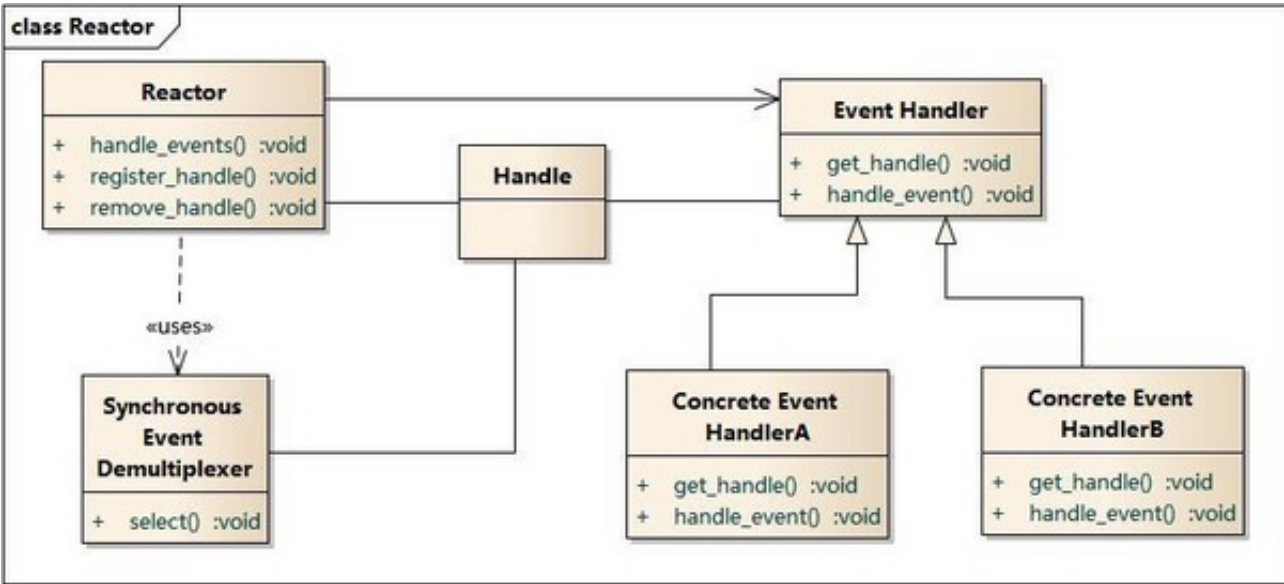
收藏

4 收藏, 361 浏览

平时接触的开源产品如Redis、ACE，事件模型都使用的Reactor模式；而同样做事件处理的Proactor，由于操作系统的原因，相关的开源产品也少；这里学习下其模型结构，重点对比下两者的异同点；

反应器Reactor

Reactor模式结构



Reactor包含如下角色：

- Handle 句柄；用来标识socket连接或是打开文件；
- Synchronous Event Demultiplexer：同步事件多路分解器：由操作系统内核实现的一个函数；用于阻塞等待发生在句柄集合上的一个或多个事件；（如select/epoll；）
- Event Handler：事件处理接口
- Concrete Event HandlerA：实现应用程序所提供的特定事件处理逻辑；

Reactor：反应器，定义一个接口，实现以下功能：

- 1) 供应应用程序注册和删除关注的事件句柄；
- 2) 运行事件循环；
- 3) 每当事件结束时，分发事件到之前注册的回调函数上处理；

文章目录



(/user/login)

业务流程及时序图

Proactor模式

“反应”即“倒置”控制逆转

业务流程及时序图

具体事件处理程序不调用反应器，而是由反应器分配一个具体事件处理程序，具体事件处理程序对某个指定的事件发生做出反应；这种控制逆转又称为“好莱坞法则”（不要调用我，让我来调用你）

对比两者的区别

主动和被动

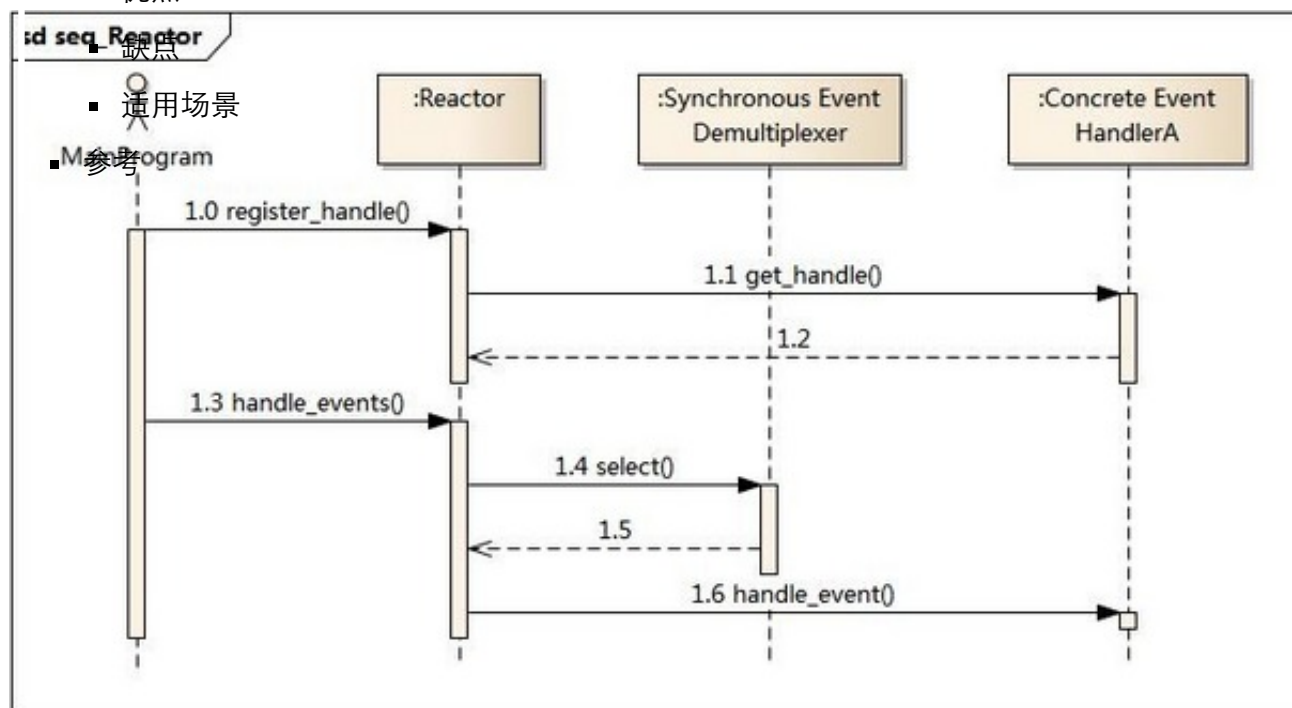
实现
业务流程及时序图

优点

缺点

适用场景

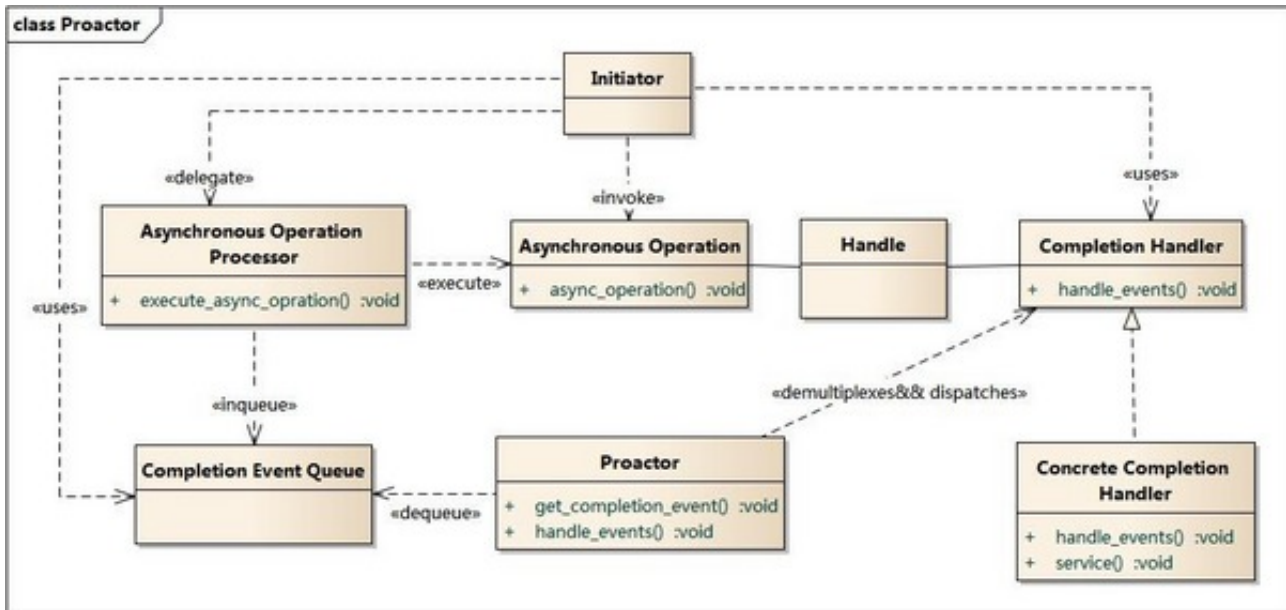
参考



1. 应用启动，将关注的事件handle注册到Reactor中；
2. 调用Reactor，进入无限事件循环，等待注册的事件到来；
3. 事件到来，select返回，Reactor将事件分发到之前注册的回调函数中处理；

Proactor模式

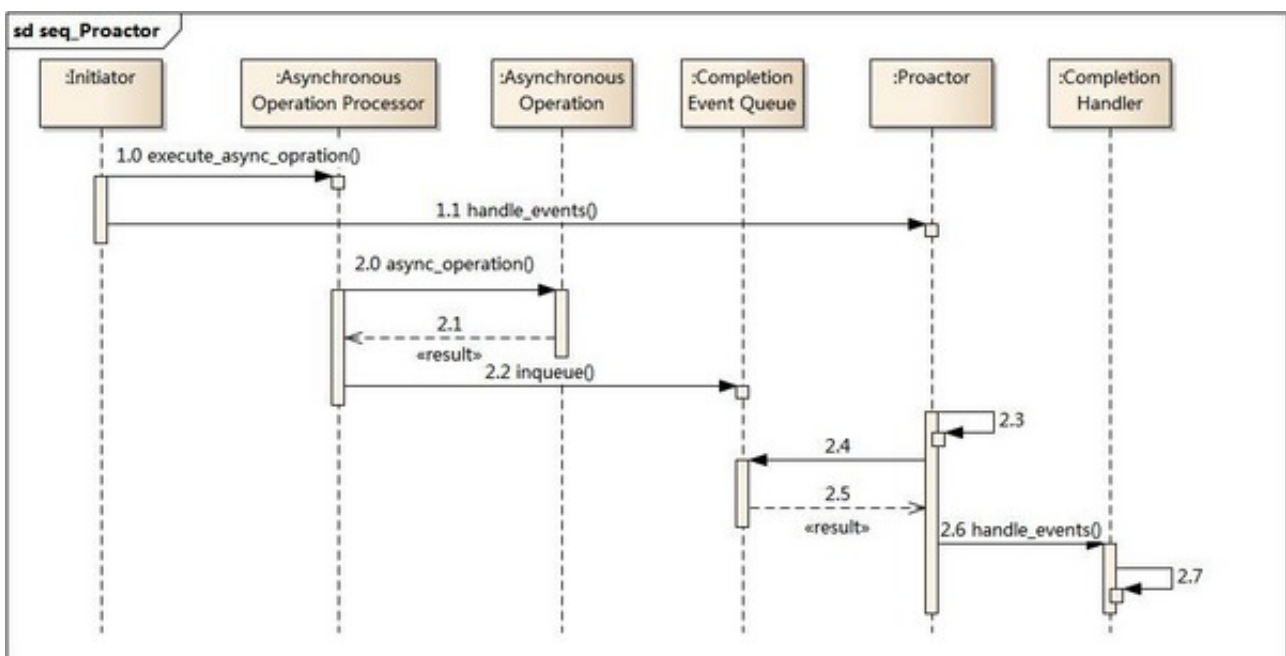
Proactor模式结构



Proactor主动器模式包含如下角色

- **Handle** 句柄；用来标识socket连接或是打开文件；
- **Asynchronous Operation Processor**：异步操作处理器；负责执行异步操作，一般由操作系统内核实现；
- **Asynchronous Operation**：异步操作
- **Completion Event Queue**：完成事件队列；异步操作完成的结果放到队列中等待后续使用
- **Proactor**：主动器；为应用程序进程提供事件循环；从完成事件队列中取出异步操作的结果，分发调用相应的后续处理逻辑；
- **Completion Handler**：完成事件接口；一般是由回调函数组成的接口；
- **Concrete Completion Handler**：完成事件处理逻辑；实现接口定义特定的应用处理逻辑；

业务流程及时序图



1. 应用程序启动，调用异步操作处理器提供的异步操作接口函数，调用之后应用程序和异步操作处理就独立运行；应用程序可以调用新的异步操作，而其它操作可以并发进行；

2. 应用程序启动Proactor主动器，进行无限的事件循环，等待完成事件到来；
3. 异步操作处理器执行异步操作，完成后将结果放入到完成事件队列；
4. 主动器从完成事件队列中取出结果，分发到相应的完成事件回调函数处理逻辑中；

对比两者的区别

主动和被动

以主动写为例：

- Reactor将handle放到select(), 等待可写就绪，然后调用write()写入数据；写完处理后续逻辑；
- Proactor调用aoi_write后立刻返回，由内核负责写操作，写完后调用相应的回调函数处理后续逻辑；

可以看出，Reactor被动的等待指示事件的到来并做出反应；它有一个等待的过程，做什么都要先放入到监听事件集合中等待handler可用时再进行操作；

Proactor直接调用异步读写操作，调用完后立刻返回；

实现

Reactor实现了一个被动的事件分离和分发模型，服务等待请求事件的到来，再通过不受间断的同步处理事件，从而做出反应；

Proactor实现了一个主动的事件分离和分发模型；这种设计允许多个任务并发的执行，从而提高吞吐量；并可执行耗时长任务（各个任务间互不影响）

优点

Reactor实现相对简单，对于耗时短的处理场景处理高效；

操作系统可以在多个事件源上等待，并且避免了多线程编程相关的性能开销和编程复杂性；

事件的串行化对应用是透明的，可以顺序的同步执行而不需要加锁；

事务分离：将与应用无关的多路分解和分配机制和与应用相关的回调函数分离开来，

Proactor性能更高，能够处理耗时长的并发场景；

缺点

Reactor处理耗时长的操作会造成事件分发的阻塞，影响到后续事件的处理；

Proactor实现逻辑复杂；依赖操作系统对异步的支持，目前实现了纯异步操作的操作系统少，实现优秀的如windows IOCP，但由于其windows系统用于服务器的局限性，目前应用范围较小；而Unix/Linux系统对纯异步的支持有限，应用事件驱动的主流还是通过select/epoll来实现；

适用场景

Reactor: 同时接收多个服务请求, 并且依次同步的处理它们的事件驱动程序;

Proactor: 异步接收和同时处理多个服务请求的事件驱动程序;

参考

《面向模式的软件体系结构 卷2》

《面向模式的软件架构 卷4》

Posted by: 大CC | 28APR,2015

博客: blog.me115.com (<http://blog.me115.com>) [订阅 (<http://feed.feedsky.com/me115>)]

微博: 新浪微博 (<http://weibo.com/bigcc115>)

设计模式 (</t/%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F/blogs>)

事件驱动 (</t/%E4%BA%8B%E4%BB%B6%E9%A9%B1%E5%8A%A8/blogs>) reactor (</t/reactor/blogs>)

链接 (</a/1190000002715832>) 更多 ▾

分享

2 推荐

收藏

本文由 大CC (</u/me115>) 创作, 采用 知识共享署名 3.0 中国大陆许可协议 (<http://creativecommons.org/licenses/by/3.0/cn>) 进行许可。

可自由转载、引用, 但需署名作者且注明文章出处。

你可能感兴趣的文章

解耦你的事件系统(基于事件驱动的设计、使用方式) (</a/1190000000653634>)

2 收藏, 579 浏览

Backbone.js学习笔记 (一) (</a/1190000002386651>)

25 收藏, 3.2k 浏览

详解JavaScript函数模式 (</a/1190000000758184>)

17 收藏, 1.4k 浏览

讨论区

请先 登录 后评论



本文隶属于专栏

大CC (/blog/me115)

关注专栏

Copyright © 2011-2015 SegmentFault. 当前呈现版本 15.05.04
浙ICP备15005796号-2 (<http://www.miibeian.gov.cn/>)