35. 基礎篇：C10K與C1000K回顧

9/10　　Freddy Fan

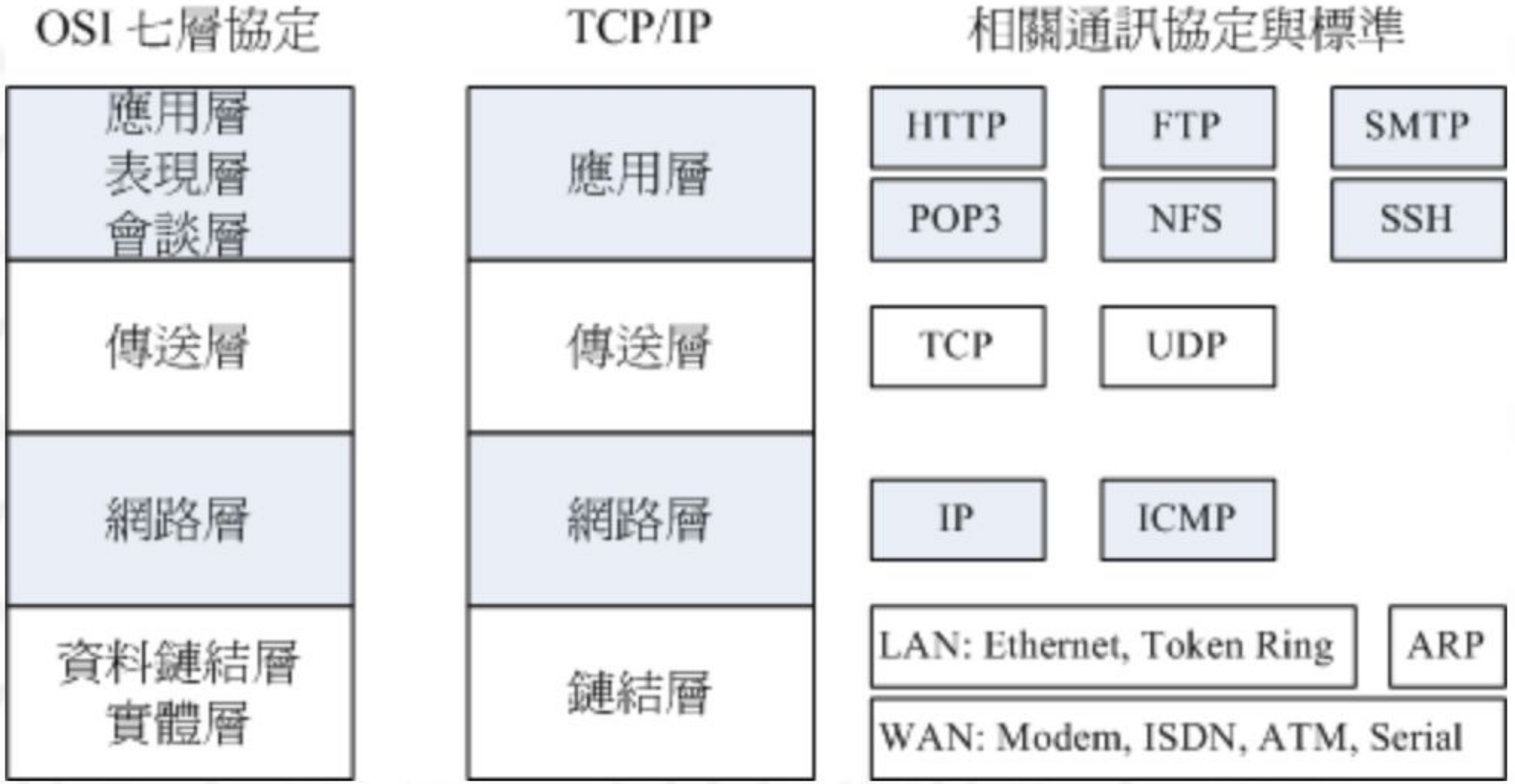


圖 2.1-4、OSI 與 TCP/IP 協定之相關性

出處：鳥哥 http://linux.vbird.org/linux_server/0110network_basic.php

1999年Dan Kegel提出 C10K問題

每秒處理 10,000 connections

以現行硬體設備是足以應付

其餘是軟體問題，特別是網路IO模型

面臨兩個問題:

1. 怎樣在一個線程內處理多個請求，是不是可以用非阻塞IO或異步IO來處裡多個網路請求？

2. 怎樣節省資源處理更多請求，是不是可以用原來100個或更少的線程來處理10000個請求？

IO模型的優化

apache

第一种，使用非阻塞 I/O 和水平触发通知，比如使用 select 或者 poll

第二种，使用非阻塞 I/O 和边缘触发通知，比如 epoll。

nginx

第三种，使用异步 I/O（Asynchronous I/O，简称为 AIO）

IO通知事件的方式：

水平触发：只要文件描述符可以非阻塞地执行 I/O，就会触发通知。也就是说，应用程序可以随时检查文件描述符的状态，然后再根据状态，进行 I/O 操作。

边缘触发：只有在文件描述符的状态发生改变（也就是 I/O 请求达到）时，才发送一次通知。这时候，应用程序需要尽可能多地执行 I/O，直到无法继续读写，才可以停止。如果 I/O 没执行完，或者因为某种原因没来得及处理，那么这次通知也就丢失了。

PS一下：IO模型

阻塞IO模型：阻塞socket、Java BIO

非阻塞IO模型：socket是非阻塞的方式

IO復用模型：select、poll、epoll三種方案，nginx都可以選擇使用這三個方案;Java NIO;

信號驅動IO模型：

異步IO模型：JAVA7 AIO、高性能伺服器應用

進程發起IO系統調用後，如果內核緩衝區沒有數據，需要到IO設備中讀取，進程返回一個錯誤而不會被阻塞；進程發起IO系統調用後，如果內核緩衝區有數據，內核就會把數據返回進程。

對於上面的阻塞IO模型來說，內核數據沒準備好需要進程阻塞的時候，就返回一個錯誤，以使得進程不被阻塞。

非阻塞IO模型

图 6.2 非阻塞 I/O 模型

多個的進程的IO可以註冊到一個復用器(select)上，然後用一個進程調用該select，select會監聽所有註冊進來的IO；

如果select沒有監聽的IO在內核緩衝區都沒有可讀數據，select調用進程會被阻塞；而當任一IO在內核緩衝區中有可數據時，select調用就會返回；

而後select調用進程可以自己或通知另外的進程(註冊進程)來再次發起讀取IO，讀取內核中準備好的數據。

可以看到，多個進程註冊IO後，

只有另一個select調用進程被阻塞。

IO復用模型

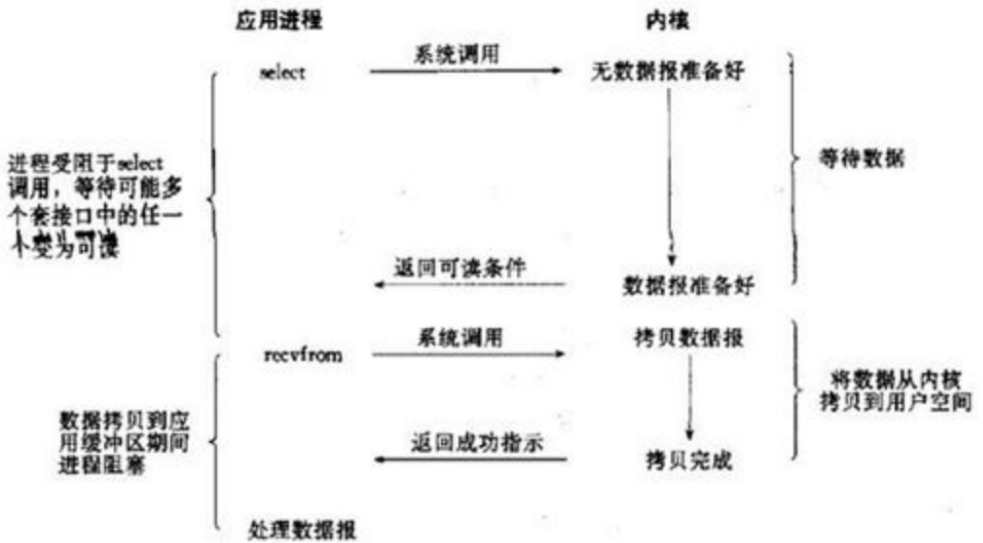


图 6.3 I/O 复用模型

图 6.6　五个 I/O 模型的比较

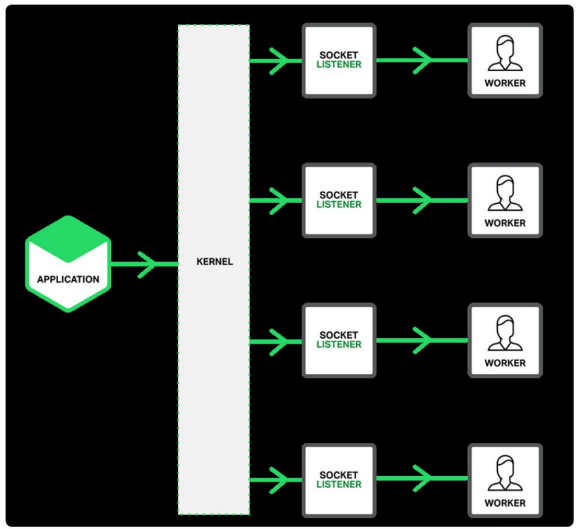
工作模型優化

第一种，主进程 + 多个 worker 子进程，这也是最常用的一种模型

- 主进程执行 bind() + listen() 后，创建多个子进程；
- 然后，在每个子进程中，都通过 accept() 或 epoll_wait()，来处理相同的套接字

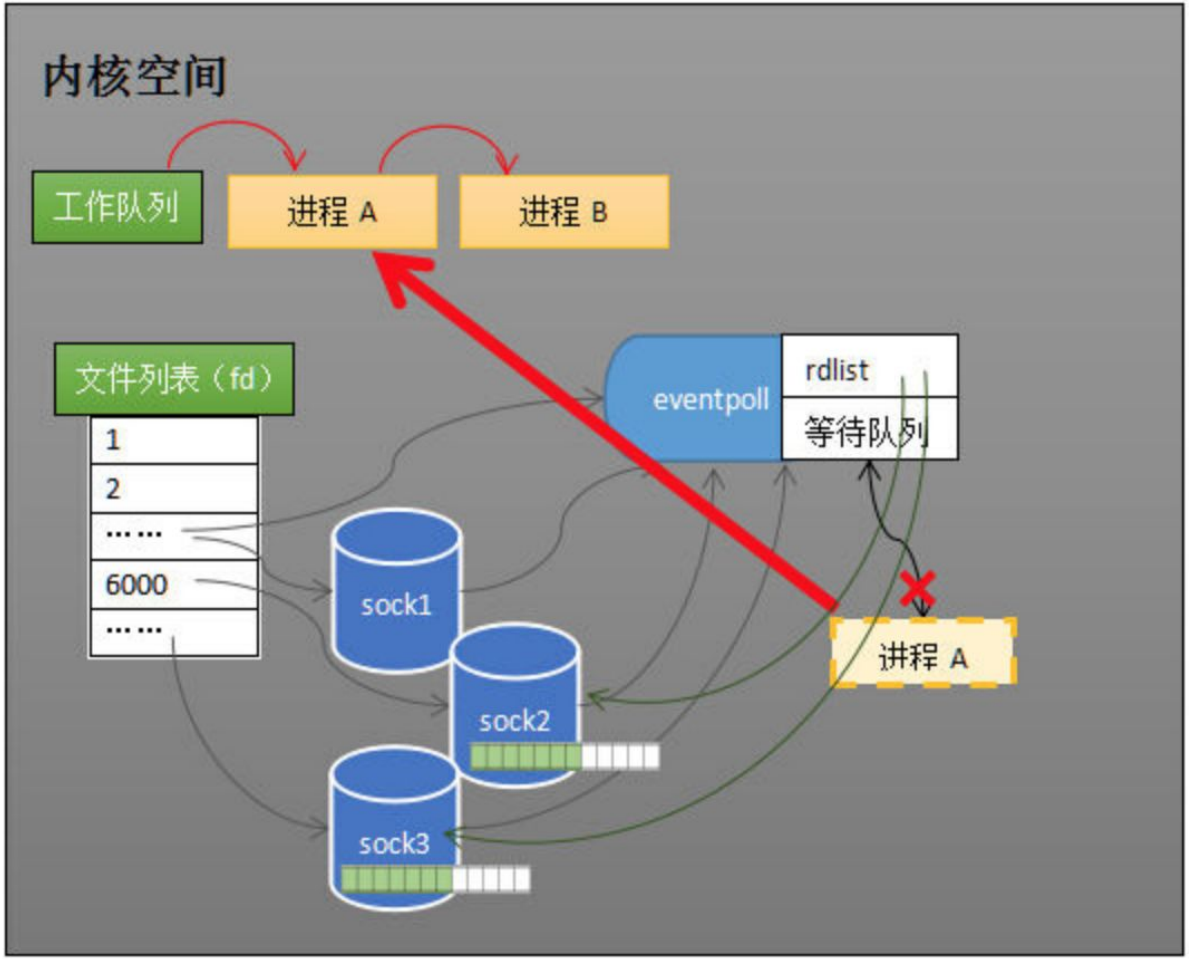
第二种，监听到相同端口的多进程模型。

SO_REUSEPOET

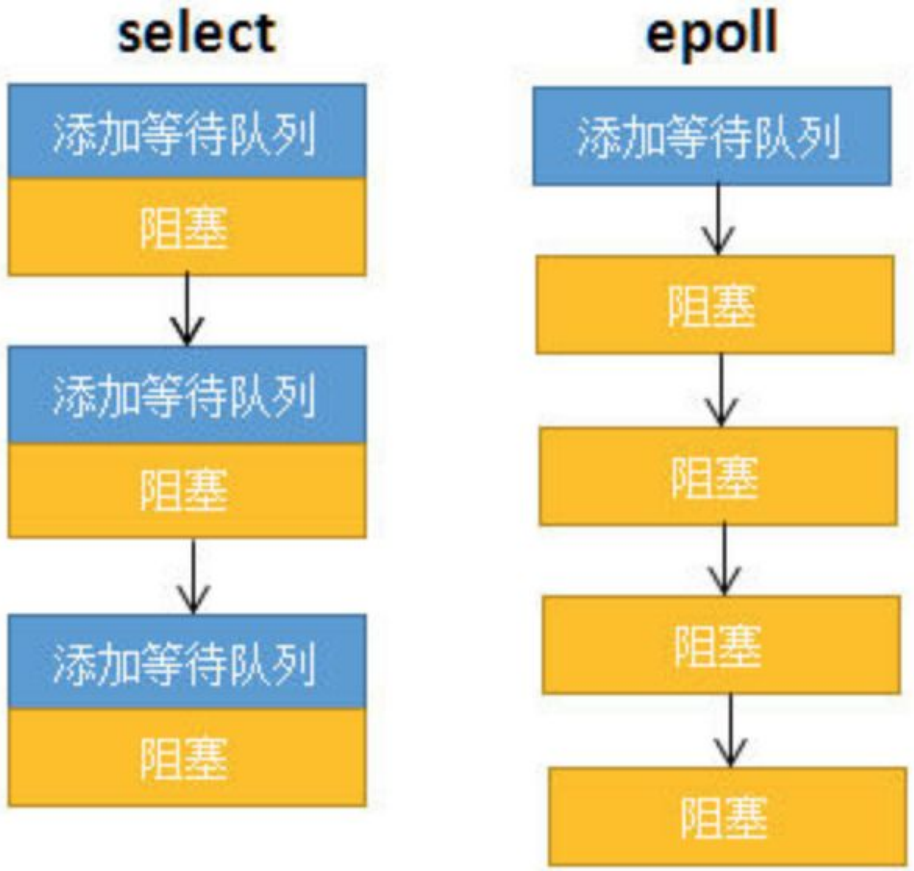


epoll原理

- 支持一个进程打开大数目的socket描述符(FD)
- IO效率不随FD数目增加而线性下降
- 使用mmap加速内核与用户空间的消息传递。



”维护等待队列”和“阻塞进程”分离，使效率提升



相比select，epoll拆分了功能

系统调用	select	poll	epoll
事件集合	用户通过 3 个参数分别传入感兴趣的可读、可写及异常等事件，内核通过对这些参数的在线修改来反馈其中的就绪事件。这使得用户每次调用 select 都要重置这 3 个参数	统一处理所有事件类型，因此只需一个事件集参数。用户通过 pollfd.events 传入感兴趣的事件，内核通过修改 pollfd.revents 反馈其中就绪的事件	内核通过一个事件表直接管理用户感兴趣的所有事件。因此每次调用 epoll_wait 时，无须反复传入用户感兴趣的事件。epoll_wait 系统调用的参数 events 仅用来反馈就绪的事件
应用程序索引就绪文件描述符的时间复杂度	O(n)	O(n)	O(1)
最大支持文件描述符数	一般有最大值限制	65 535	65 535
工作模式	LT	LT	支持 ET 高效模式
内核实现和工作效率	采用轮询方式来检测就绪事件，算法时间复杂度为 O(n)	采用轮询方式来检测就绪事件，算法时间复杂度为 O(n)	采用回调方式来检测就绪事件，算法时间复杂度为 O(1)

C1000K!!!! (十萬)

- 假设每个请求需要 16KB 内存的话, 那么总共就需要大约 15 GB 内存。
- 而从带宽上来说, 假设只有 20% 活跃连接, 即使每个连接只需要 1KB/s 的吞吐量, 总共也需要 1.6 Gb/s 的吞吐量。

本质上还是构建在 epoll 的非阻塞 I/O 模型上。只不过, 除了 I/O 模型之外, 还需要从应用程序到 Linux 内核、再到 CPU、内存和网络等各个层次的深度优化, 特别是需要借助硬件

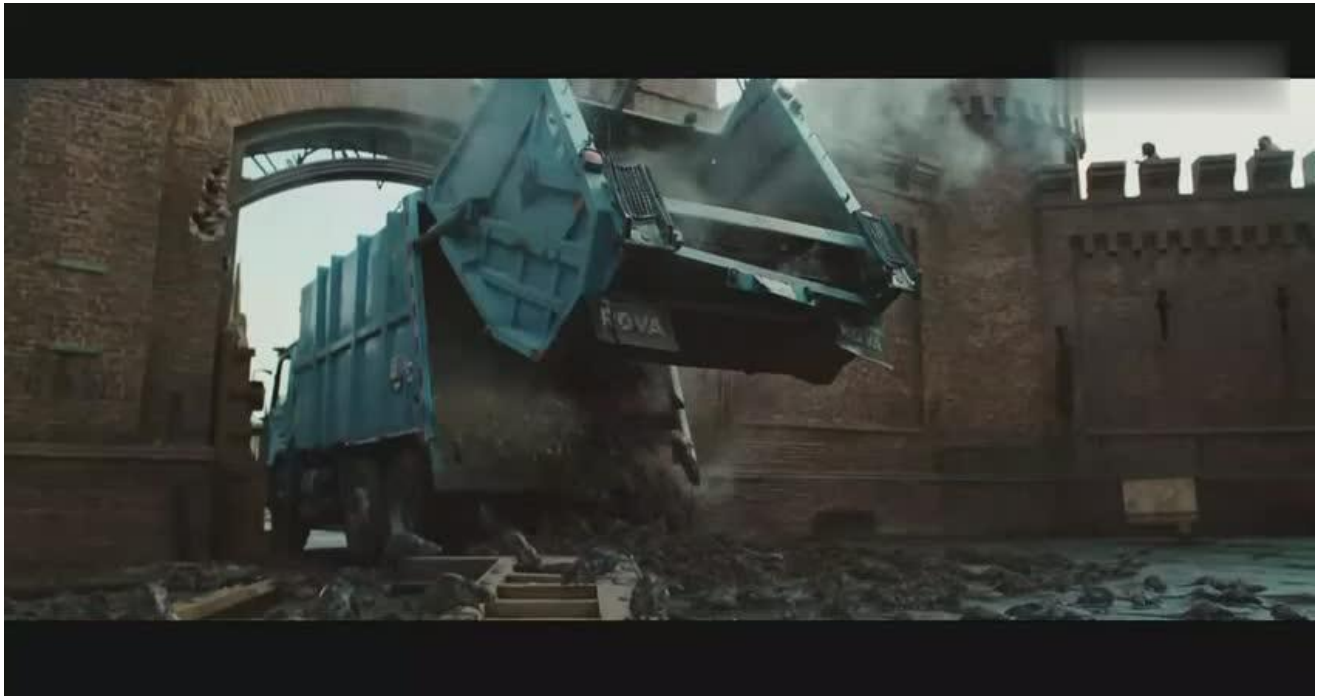
C10M (C10,000,000)（一千萬）

第一种机制，DPDK，是用户态网络的标准。它跳过内核协议栈，直接由用户态进程通过轮询的方式，来处理网络接收。

第二种机制，XDP（eXpress Data Path），则是 Linux 内核提供的一种高性能网络数据路径。

「抓一隻老鼠是一件事，
抓□千隻老鼠□是另一件事。」

–SRE社群前輩



思考討論題1：

1. 現行公司單一台主機撐流量C10K 有可能嗎？
2. 防火牆. 路由器也需要能承受C10K
3. 電商雙11流量的迷思
4. 館長求救網站事件

思考討論題2:

大流量問題, 除了調教硬體外還有無可解方法？

- 多開主機
- 快取很重要
- 加速資料讀取速度

參考資料

1. **select、poll、epoll之间的区别总结**

   https://www.cnblogs.com/anker/p/3265058.html

2. 分享一篇終於把epoll講明白的文章

   https://kknews.cc/zh-tw/code/oeorylq.html

3. DPDK and XDP

   https://cloud.tencent.com/developer/article/1484793

4. 五種IO模型：

   https://zhuanlan.zhihu.com/p/127170201