

High Performance Web Server



版权声明

本文件/视频中含有的信息为上海道客网络科技有限公司（简称“道客”）专有的，且为保密信息，如果泄漏可能会给提供相同服务的竞争对手带来实质的好处。本材料中含有对战略规划、咨询方案、创意思路、架构设计、技术蓝图、文字和照片素材、案例、方法论，（概念的描述是通过道客的实质性的研究和开发的努力所获得的）。因此，在未经道客事先书面认可的情况下，本文件/视频的任何部分都不得被复制或传递。道客将对非法复制或传递者保留采用法律手段追究的权利。

COPYRIGHT NOTICE

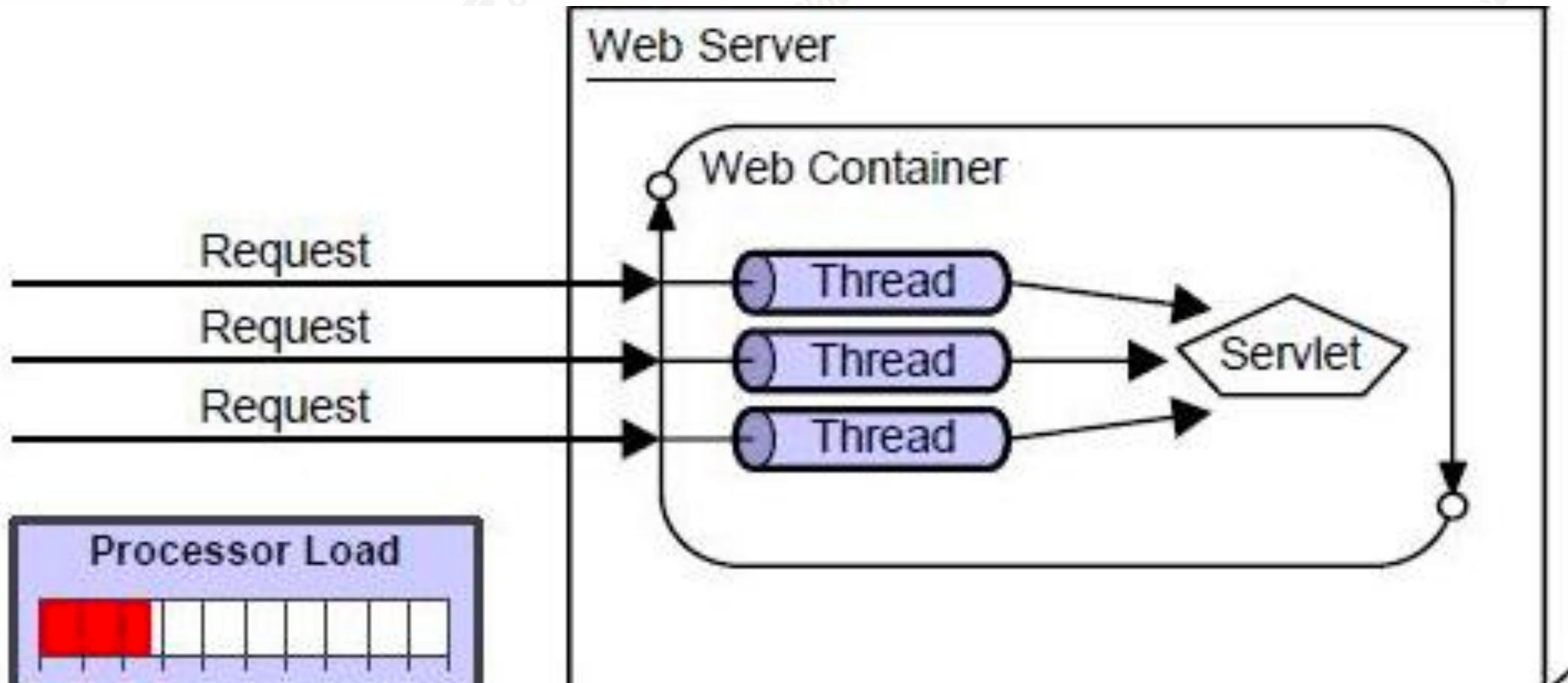
The information contained in this document / video is proprietary to Shanghai DaoCloud Network Technology Co.,Ltd. (DaoCloud).

This information is confidential and if it is leaked, it will bring substantial benefits to the competitors who provide the same service. This material contains strategic planning, consulting solutions, creative ideas, architectural design, technical blueprints, text and photo materials, cases, and methodology. Therefore, no part of this specification may be reproduced in any form or by any means, without prior written consent of DaoCloud. DaoCloud reserves all legal rights against illegal copying or delivery.



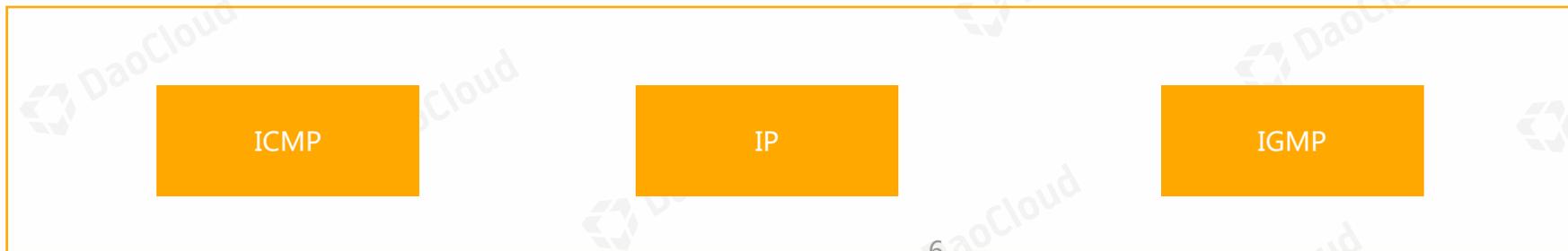
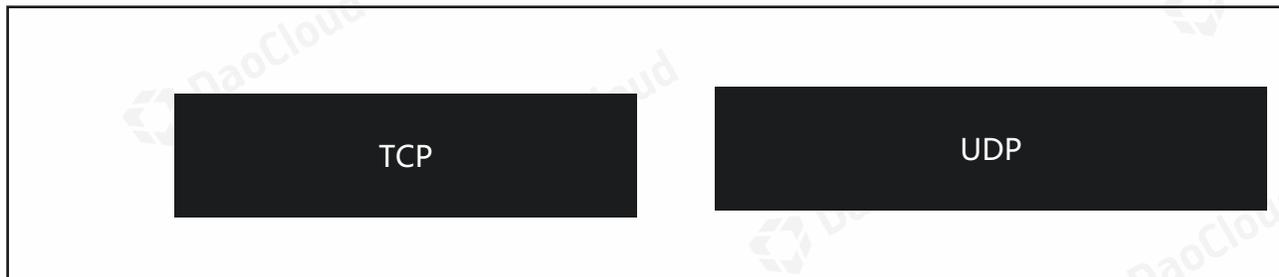
Chapter 1: Who Blocking Me!

传统 Application Web Server

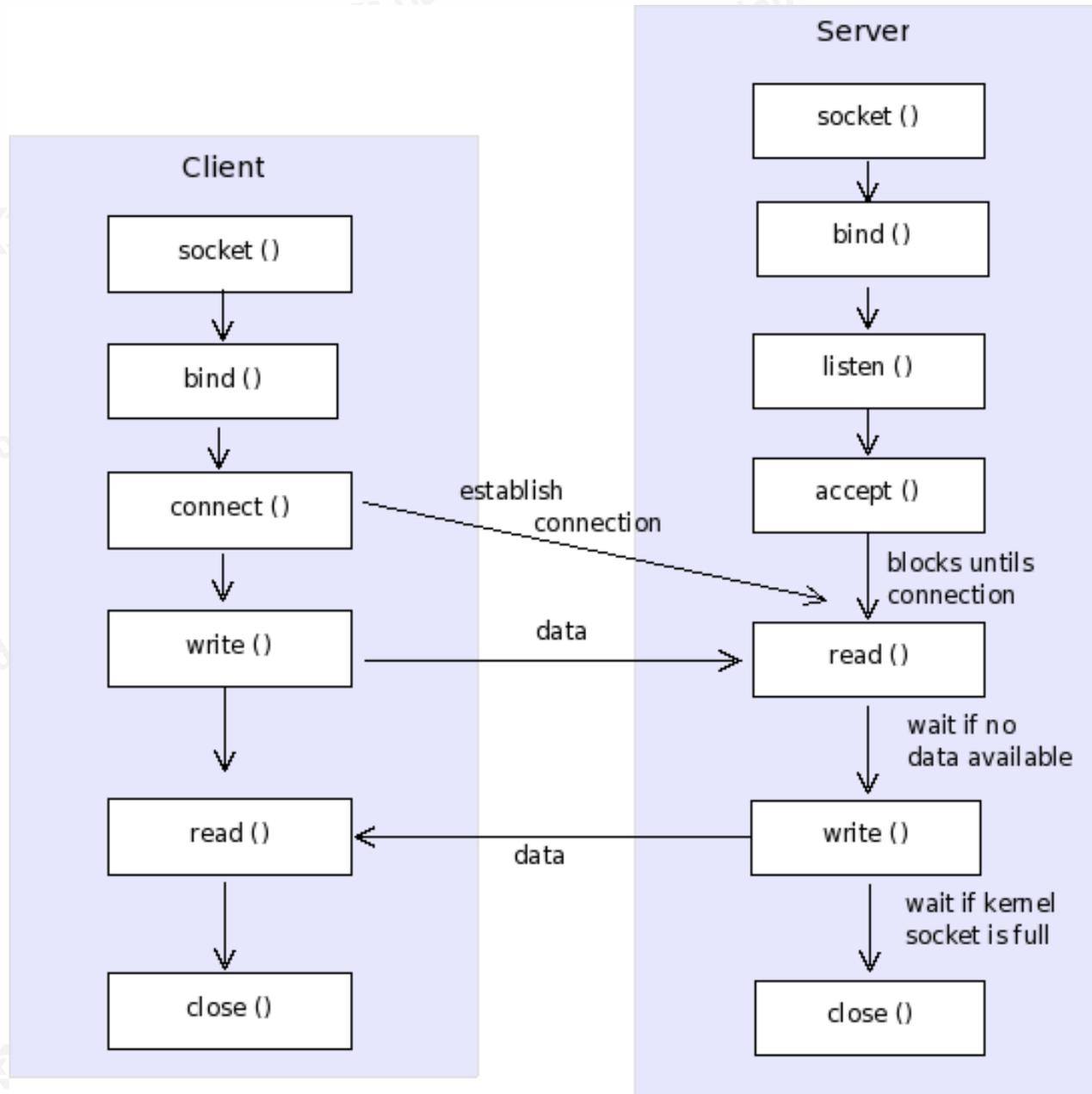


TOMCAT <= 7

Socket !



Socket!



Echo example

```
int main()
{
    listen_fd = socket(AF_INET, SOCK_STREAM, 0);
    bind(listen_fd, (struct sockaddr *) &servaddr, sizeof(servaddr));
    listen(listen_fd, 10);
    comm_fd = accept(listen_fd, (struct sockaddr*) NULL, NULL);

    while(1)
    {
        read(comm_fd, str, 100); // 重点1
        printf("Echoing back - %s", str);
        write(comm_fd, str, strlen(str)+1); // 重点2
    }
}
```

Read() Def

NAME

read - read from a file descriptor

SYNOPSIS

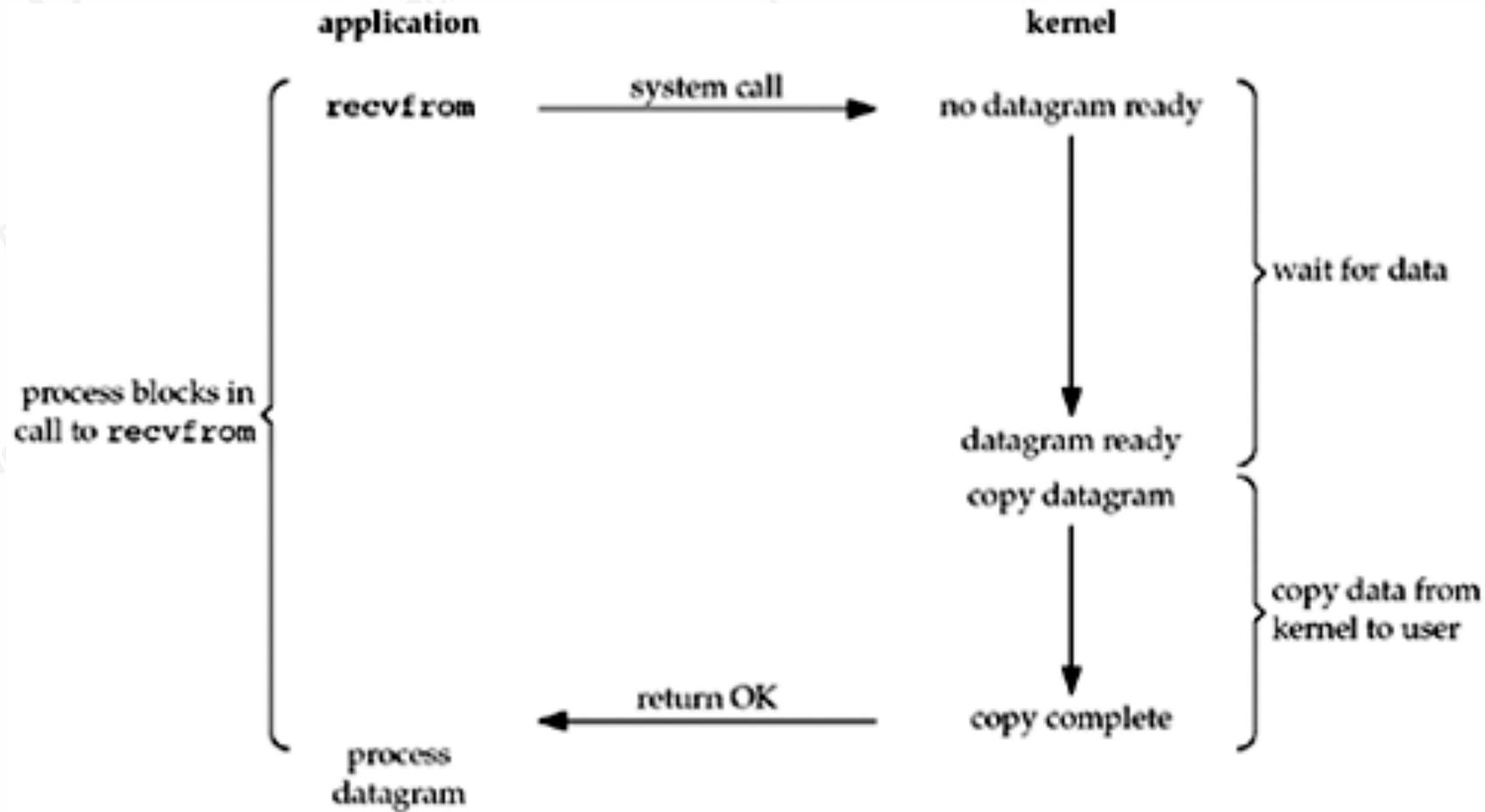
```
#include <unistd.h> ssize_t read(int fd, void *buf, size_t count);
```

ERRORS

EAGAIN The file descriptor *fd* refers to a file other than a socket and has been marked nonblocking (**O_NONBLOCK**), and the read would block. See [open\(2\)](#) for further details on the **O_NONBLOCK** flag.

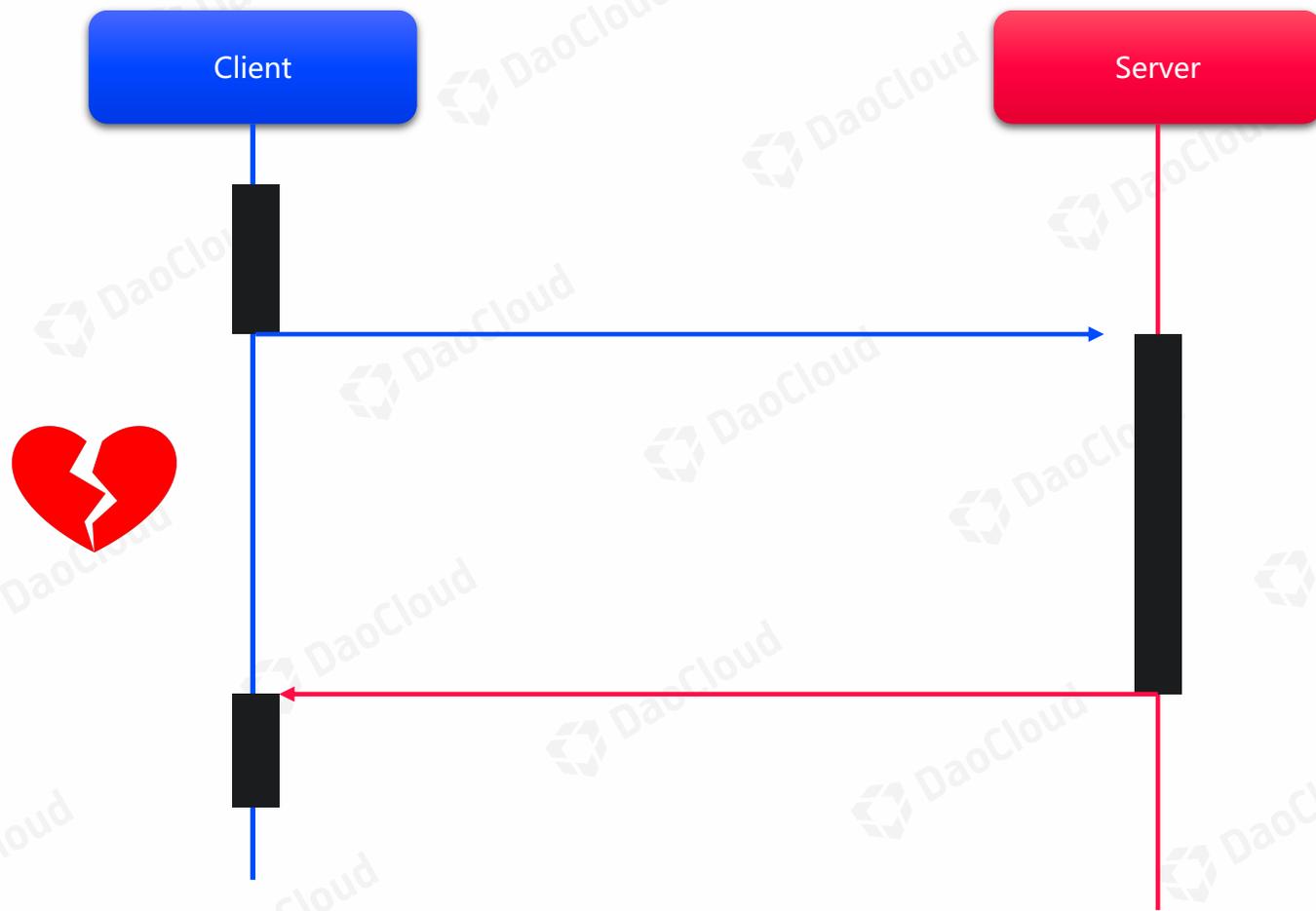
Blocking

IO Model

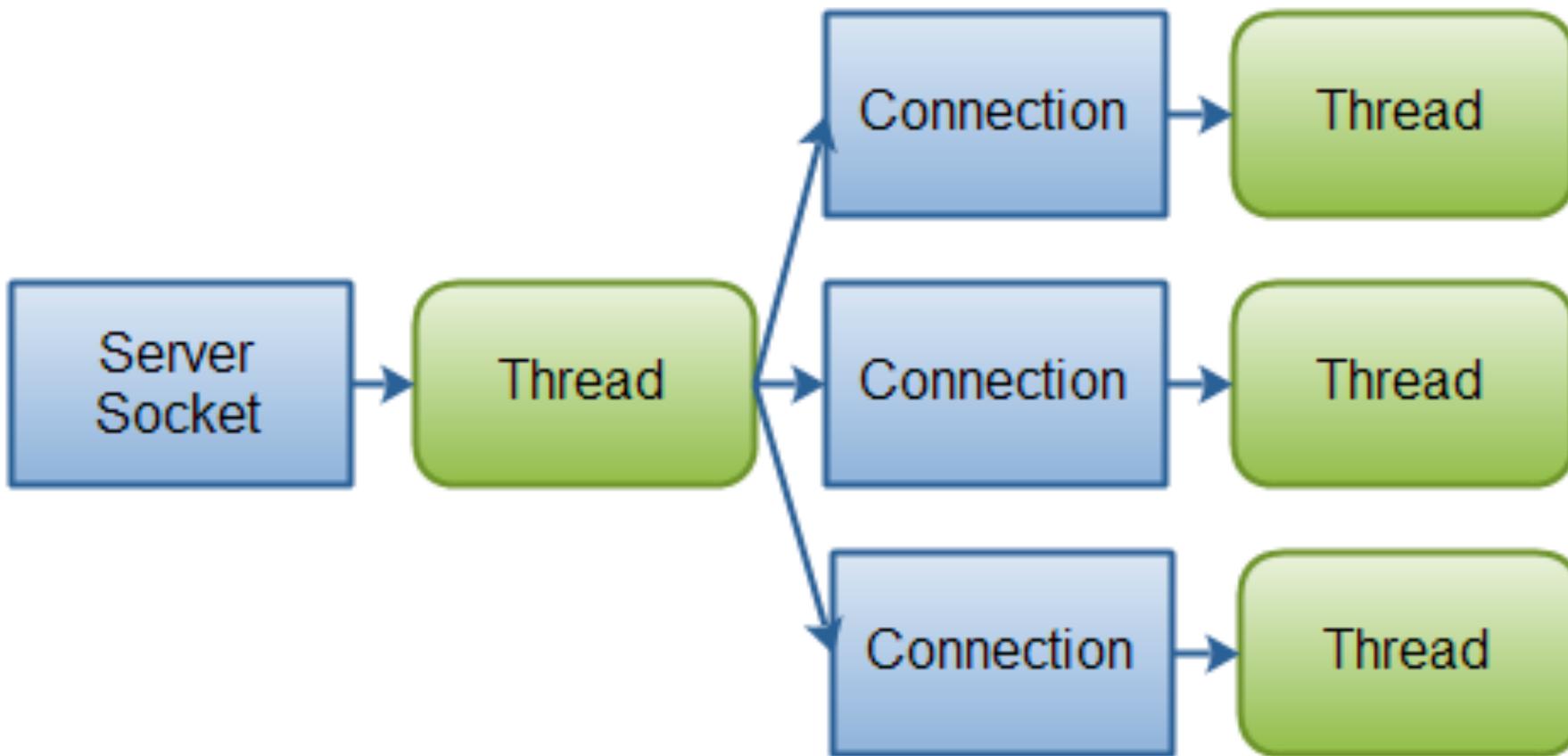


Blocking I/O model.

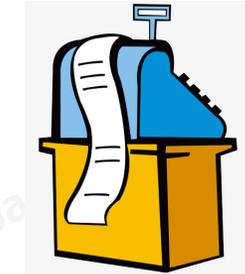
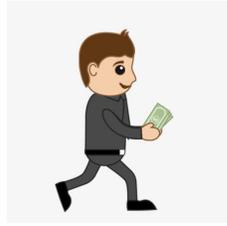
Wait & Response



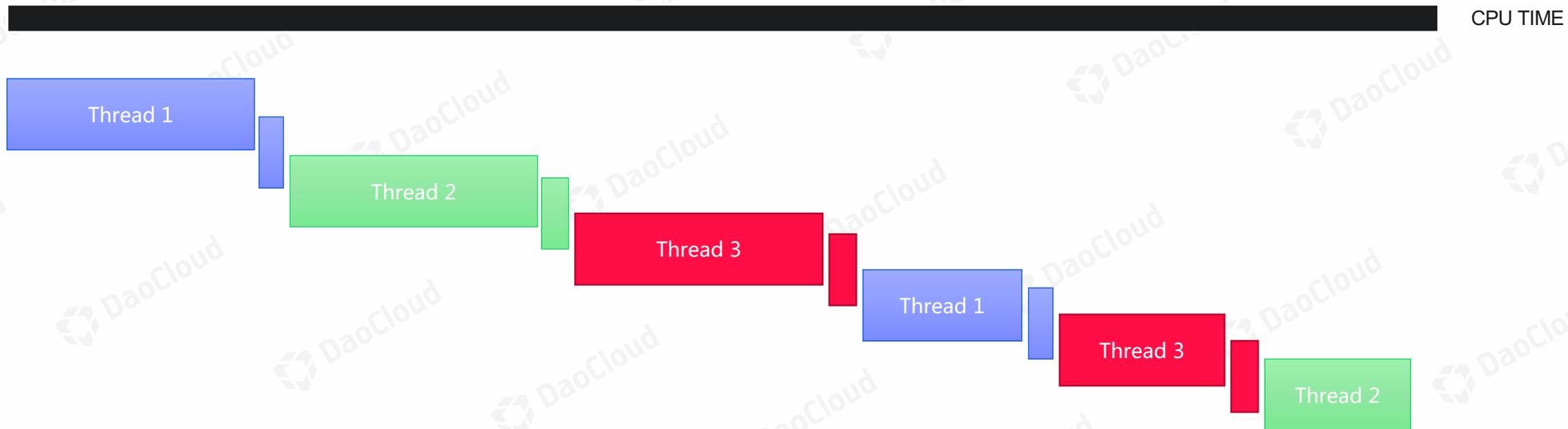
Blocking Threading Model



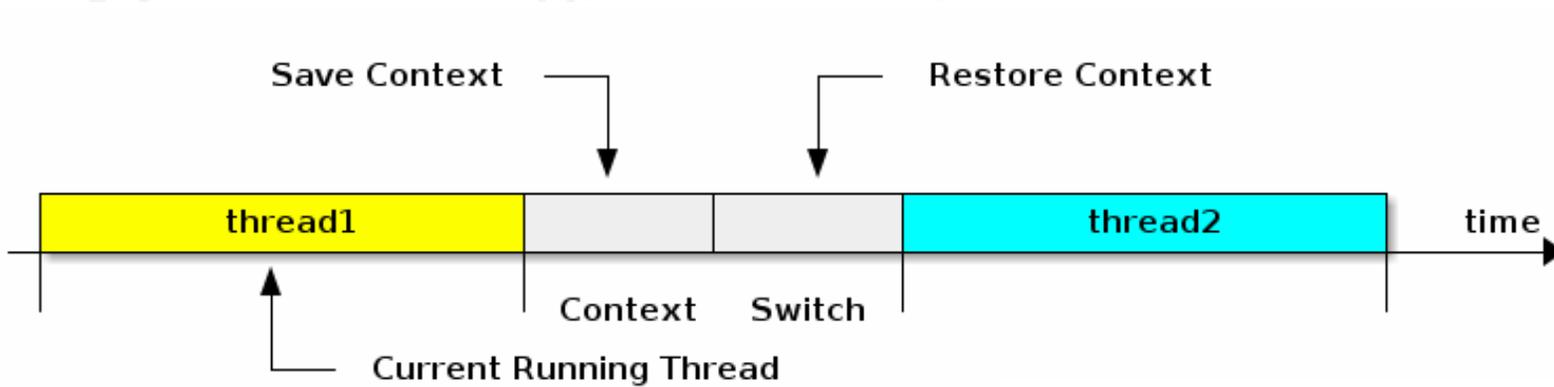
高阶比喻



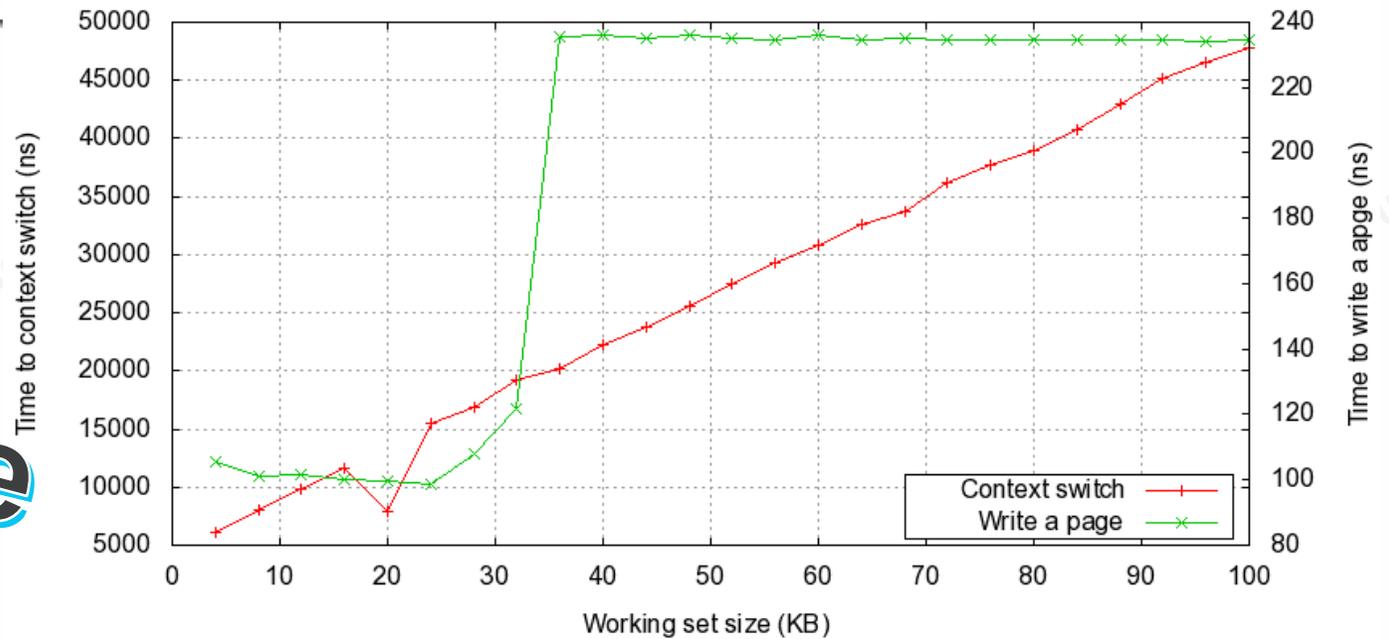
Blocking is Wasted



Threading Context Switch



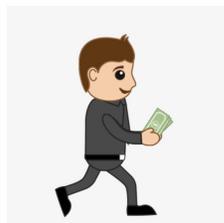
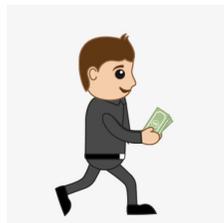
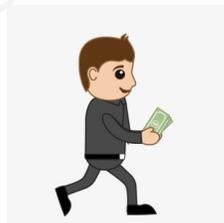
Cost of context switching on a dual Intel 5150



Switch Not Free

高阶比喻

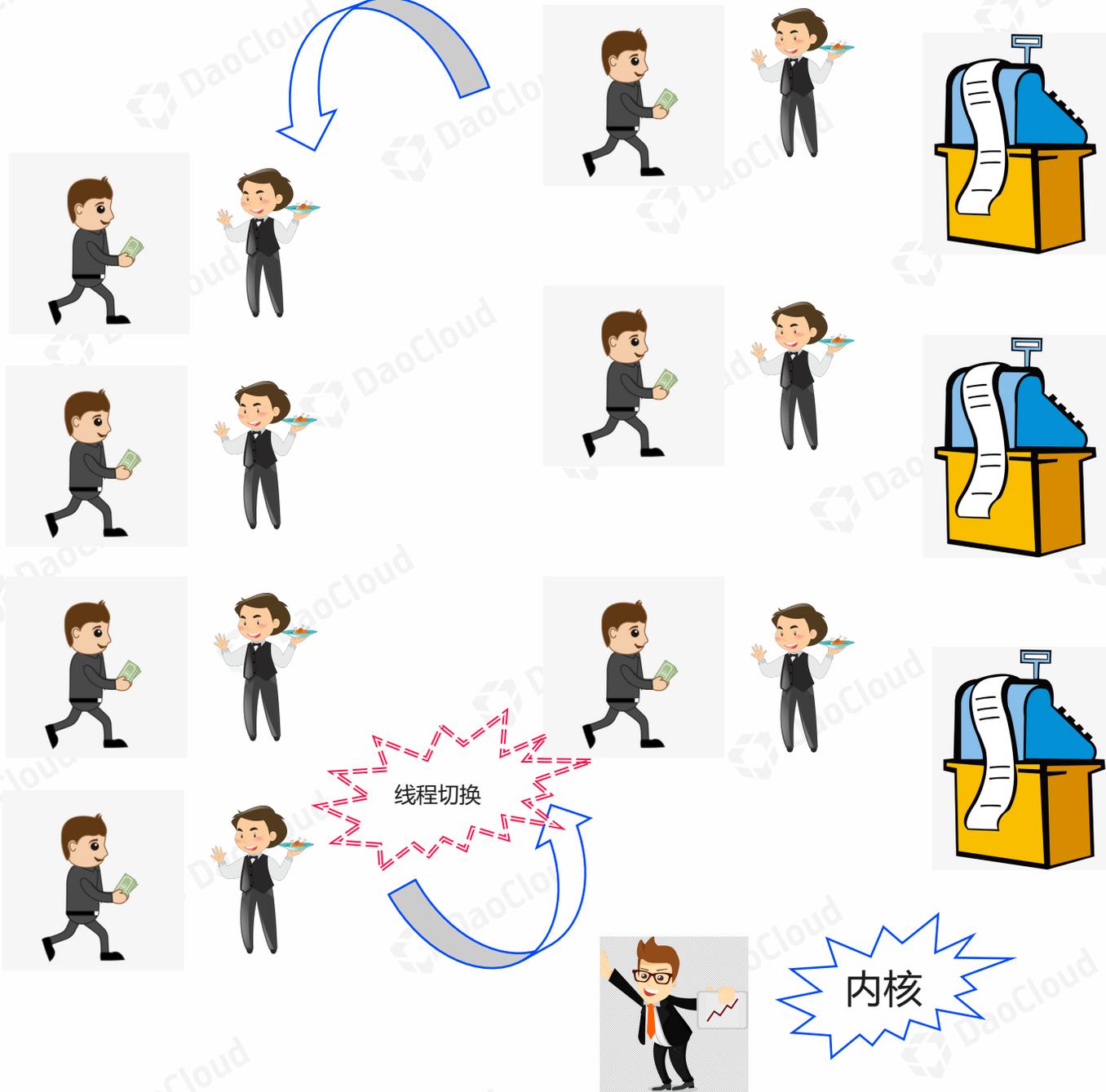
所有还在犹豫点单的先去
旁边凉快下



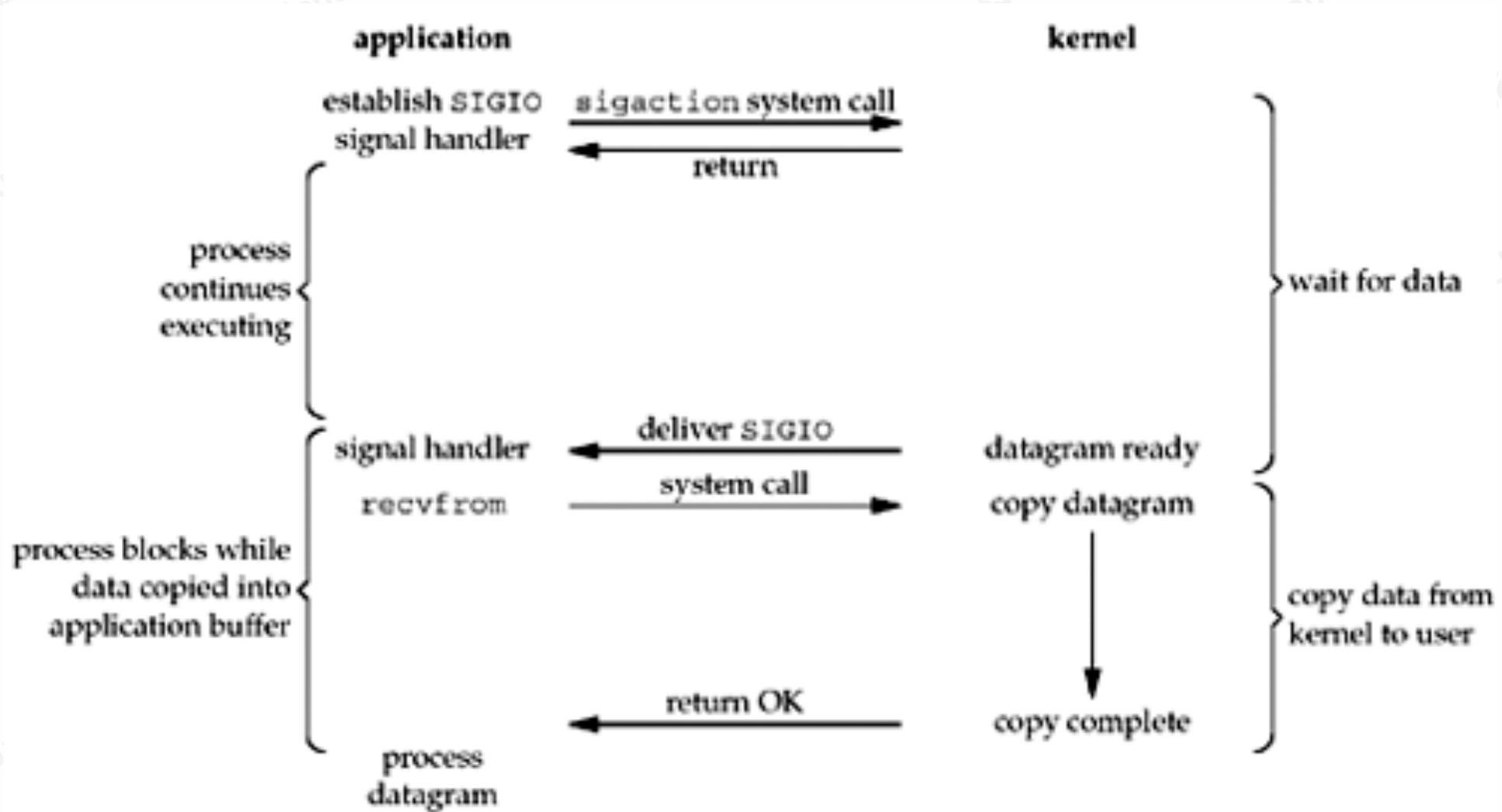
内核

高阶比喻

看看有没有其他人想好了先下单



I/O multiplexing model



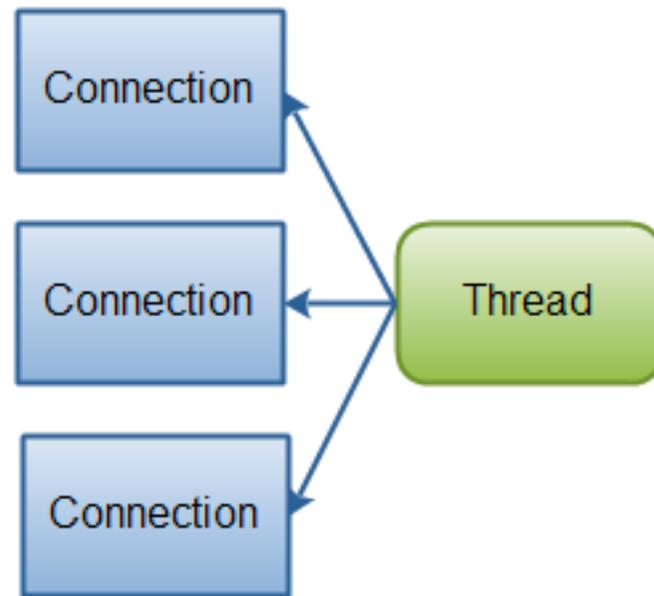
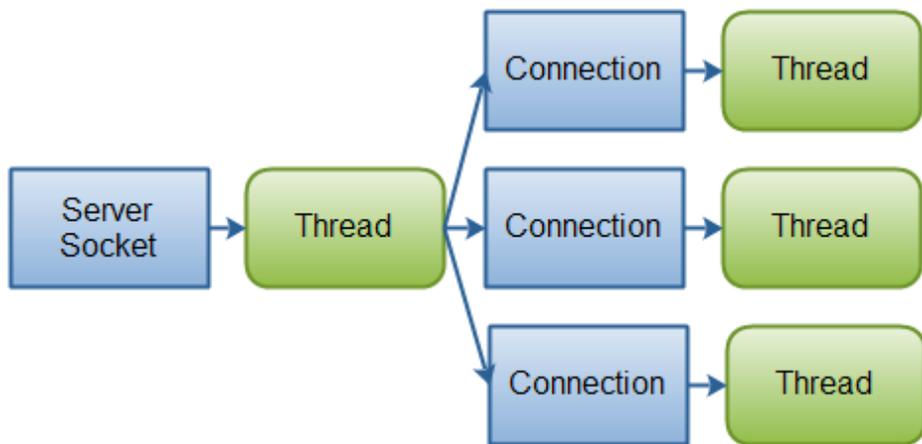
NonBlocking



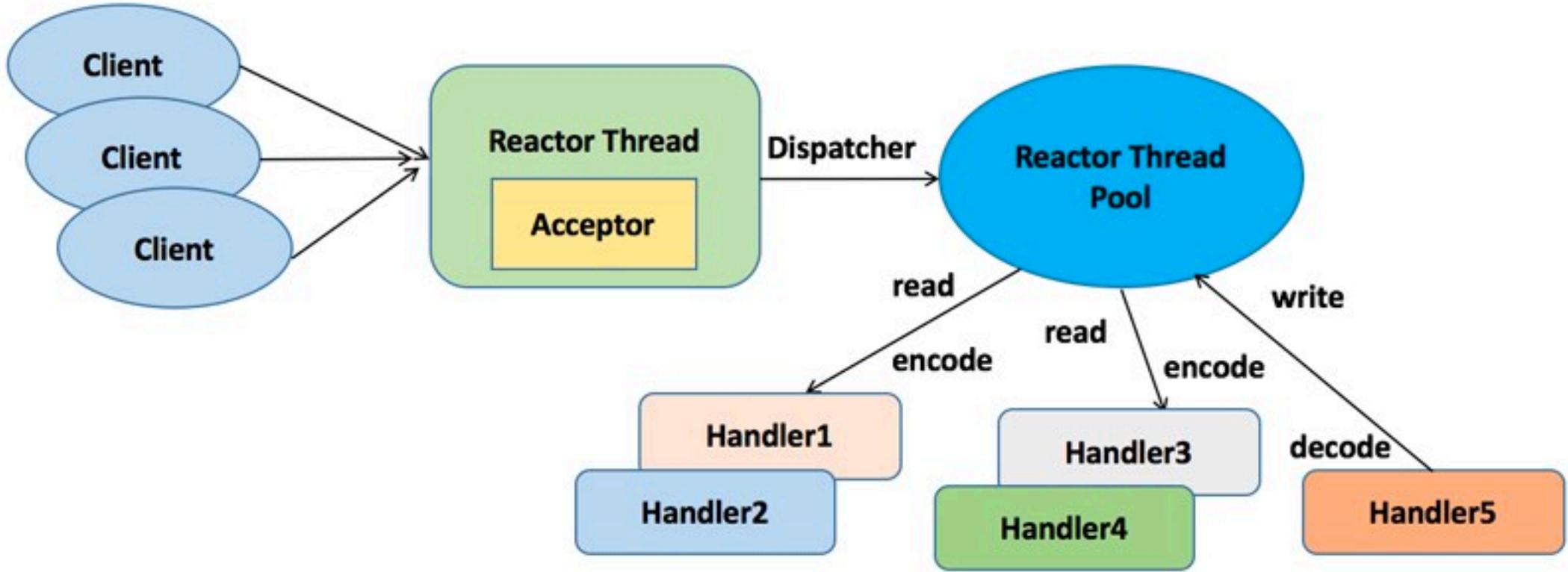
Epoll

```
void server_run(){
    bind(listen_sock, (struct sockaddr *)&srv_addr, sizeof(srv_addr));
    setnonblocking(listen_sock);
    listen(listen_sock, MAX_CONN);
    epfd = epoll_create(1);
    epoll_ctl_add(epfd, listen_sock, EPOLLIN | EPOLLOUT | EPOLLET);
    for (;;) {
        nfd = epoll_wait(epfd, events, MAX_EVENTS, -1); //epoll_wait is blocking
        for (i = 0; i < nfd; i++) {
            if (events[i].data.fd == listen_sock) {
                conn_sock = accept(listen_sock, (struct sockaddr *)&cli_addr, &socklen);
                setnonblocking(conn_sock);
                epoll_ctl_add(epfd, conn_sock, EPOLLIN | EPOLLET | EPOLLRDHUP | EPOLLHUP);
            } else if (events[i].events & EPOLLIN) {
                for (;;) {
                    bzero(buf, sizeof(buf));
                    n = read(events[i].data.fd, buf,
                            sizeof(buf));
                    printf("[+] data: %s\n", buf);
                    write(events[i].data.fd, buf, strlen(buf));
                }
            }
        }
    }
}
```

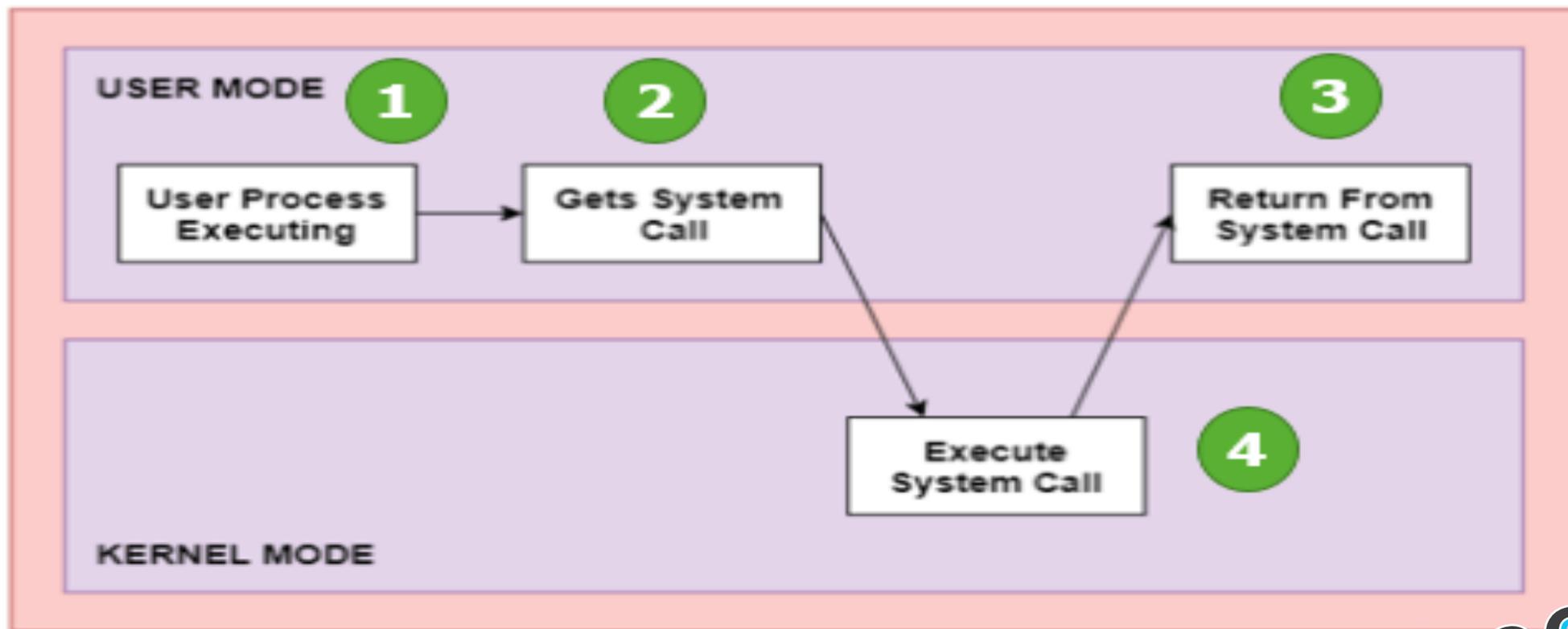
Threading mode switch



Netty



用户态 & 内核态



由于需要限制不同的程序之间的访问能力, 防止他们获取别的程序的数据, 或者获取外围设备的数据, 并发送到网络, CPU划分出两个权限等级 -- 用户态和内核态。

Switch Not Free x 2

一次系统调用开销到底有多大?

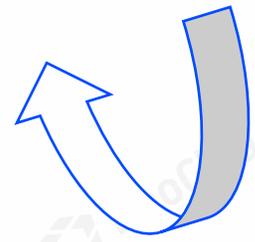
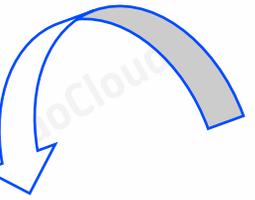
高阶比喻



提交
订单
→



不断询问

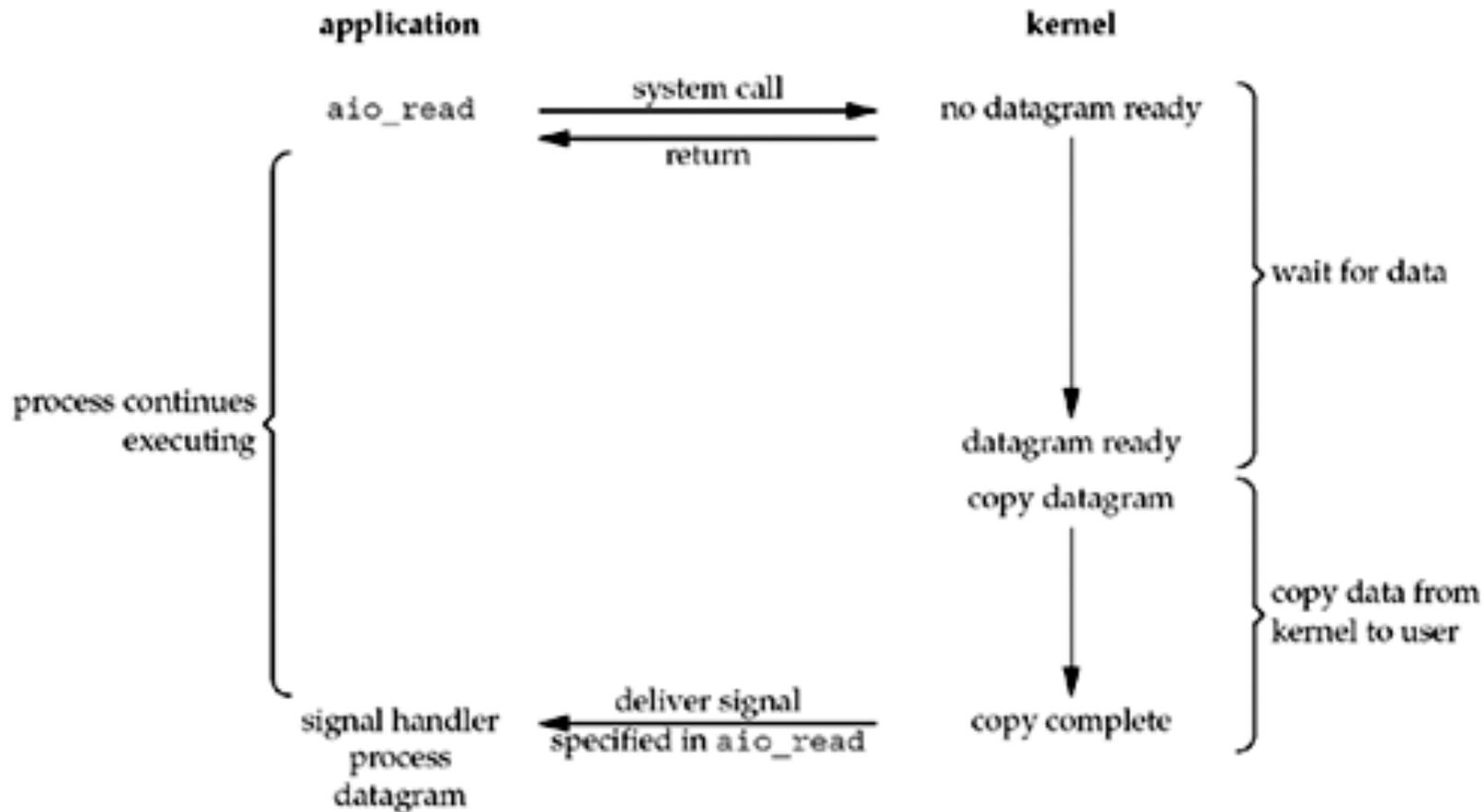


用户态
-> 内核态



内核一个人记
录所有人的请
求

不怎么成熟的 AIO



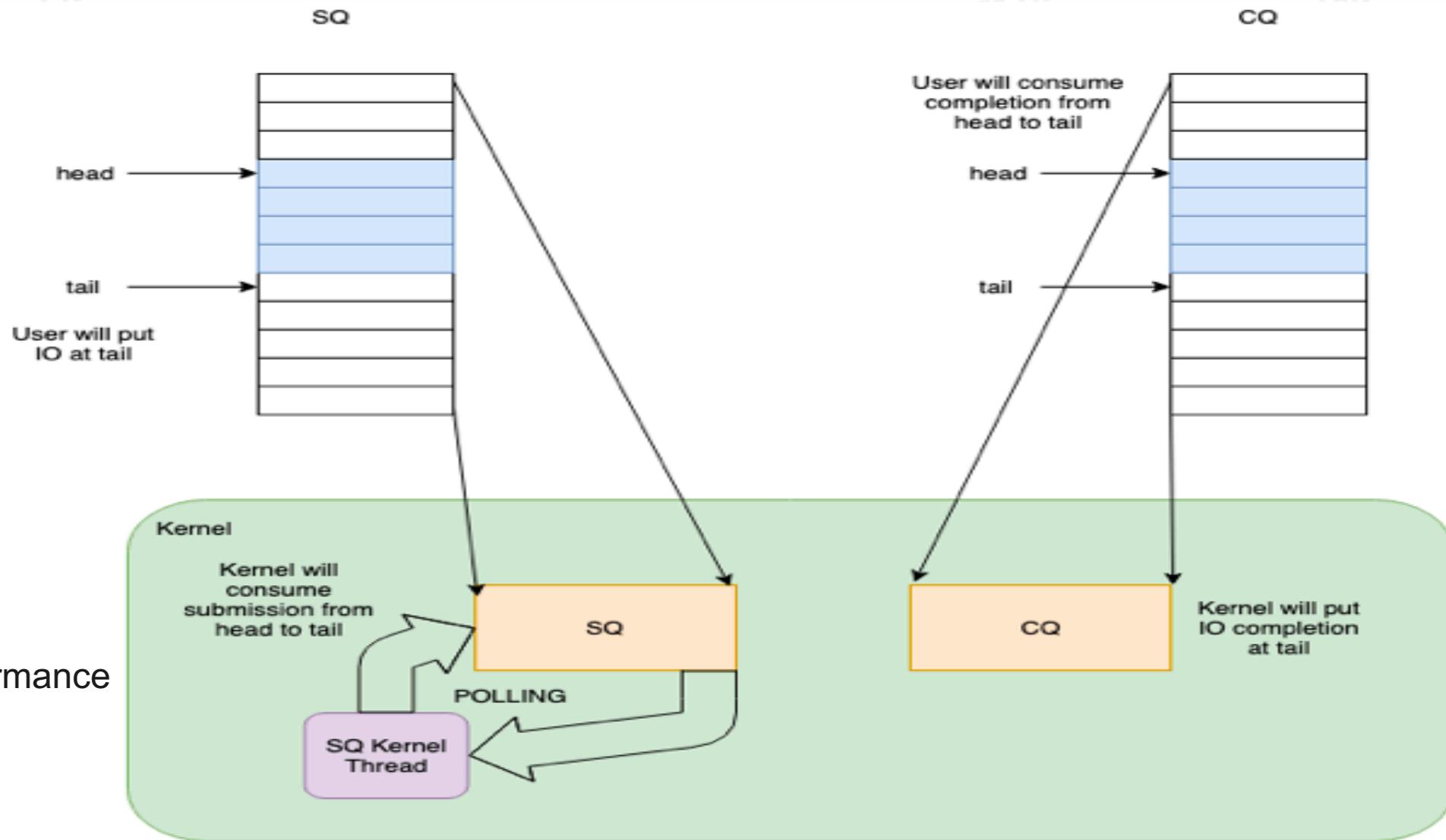
New AIO: io_uring

最近几年一直流行 kernel bypass，从网络到存储，各个领域开花，内核在性能方面被各种诟病。

Interface	QD	Polled	IOPS
io_uring	128	1	1620K
libaio	128	0	608K
spdk	128	1	1739K

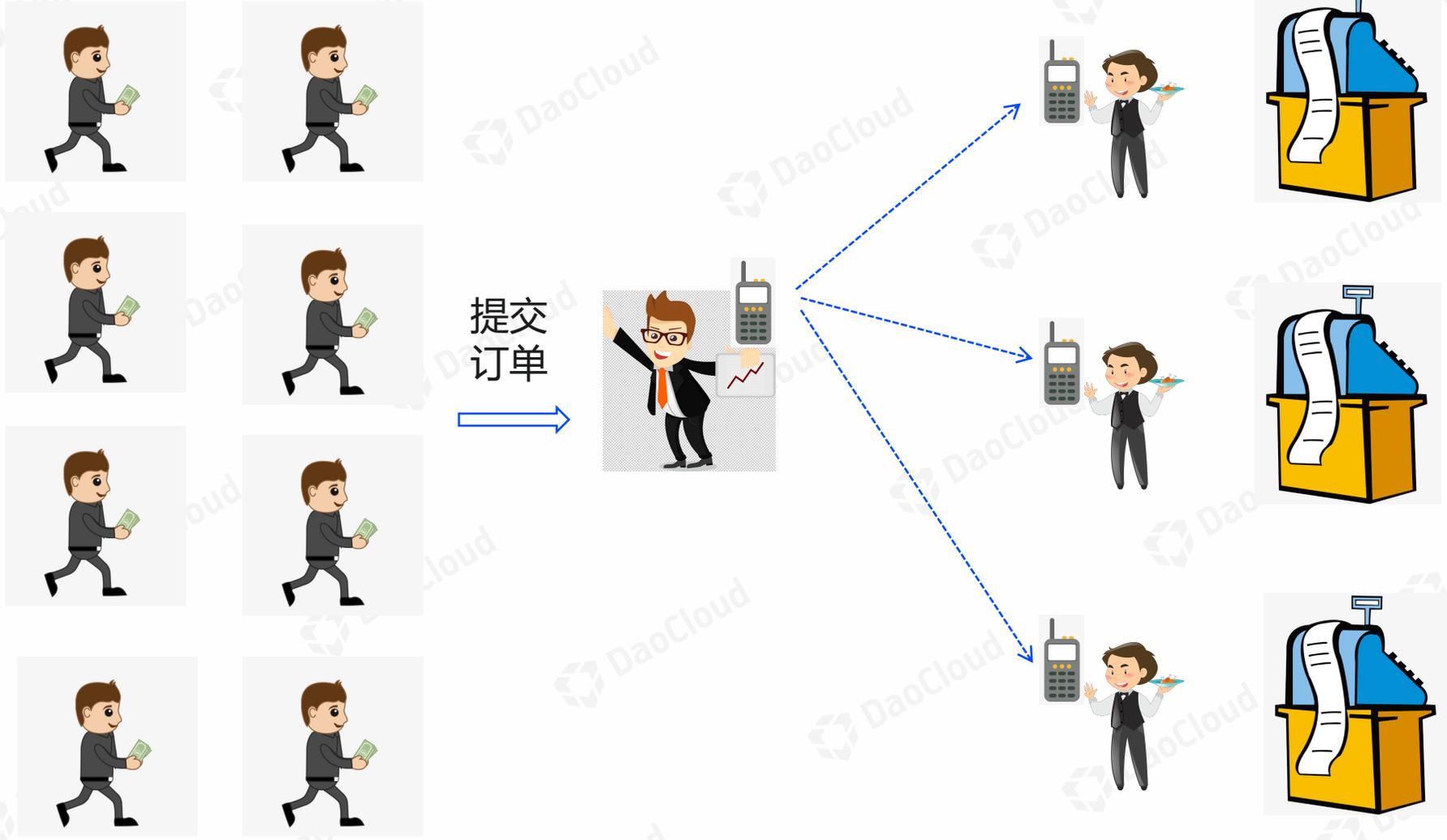
- 用户态和内核态共享提交队列（ submission queue ）和完成队列（ completion queue ）
- IO 提交和收割可以 offload 给 Kernel，且提交和完成不需要经过系统调用（ system call ）
- 支持 Block 层的 Polling 模式
- 通过提前注册用户态内存地址，减少地址映射的开销

IO_URING



+99% Performance
-45% CPU

高阶比喻



感谢聆听！



驾驭数字方程式
THE FORMULA DIGITAL