异步io调度框架的实现

峰云就她了
http: //xiaorui. cc

网卡收到数据之后

- 包少，硬中断
- 包多，屏蔽硬中断，DMA+软中断

socket详细流程

- tcp三次握手
- setblocking
- backlog多大合适?
- syn, accept队列
- accept(), Why new socket
- 协议栈
- 粘包
- buffer缓冲区大小

IO阻塞的本质

- 读缓冲，写缓冲区
- send，recv原理
- 满加锁，空解锁
- tcp ack 滑动窗口角色
- 那么8k的写缓冲区，我有80k要写，流程?

疑问

- 怎么确定recv的结构是完整的?
  - 短连接
  - 协议约定
- 粘包是个伪命题！
- 长连接、短连接的区别
- socket维护长短连接的手段
  - 屏蔽close()
  - IO复用模型
  - 用thread、process活跃上下文
- keepalive心跳包谁来维护？ 自定义意义

疑问

- socket so_keepalive 参数的作用
  - 检测对端的存活
    - keepalive_probes
    - keepalive_time
    - keepalive_intvl
- 如果没配置so_keepalive, client挂了,服务端在recv阶段呆很久...
- 如果配置了so_keepalive, 通过协议栈来探测存活

高并发模型

- fork模型
- 进程池模型 or 线程池模型
- io复用模型

疑问

- 惊群?
- 饥饿?
- so_reuseaddr
- so_reuseport

上下文

- 什么是上下文
- 什么是上下文切换
- 为什么要上下文切换
- 什么时候会上下文切换

执行单元

- 进程
- 线程
- 协程
- 堆、栈
- 抢占、协作

io

- 同步阻塞
- 同步非阻塞
- 异步阻塞
- 异步非阻塞

fd

- 一个线程如何多个fd？
- 忙轮询？要不线程池，堵塞等唤醒

io多路复用

- select
- poll
- epoll

select

- 用法?

epoll

- epoll_create
- epoll_ctl
- epoll_wait
- 水平触发 vs 边缘触发
- tornado、nginx的选择

跨平台

- libevent
- libev
- libuv

非堵塞客户端

- connect_ex()
- sock.setblocking
- eagain == EWOULDBLOCK
- O_NONBLOCK
- eintr

连接问题

- 协程可以共用一个连接么 ？
- 连接池 vs call create ？

调度器组成部分

- 核心 event loop
- 类生成器
- map{fd: object}
- 信号处理
- IO状态
- 文件属性变化
- 定时器
- periodic
- timeout

prefork + epoll

- Master Worker工作模型
- max_requests
- add、reduce
- worker reload
- log reload
- socketpair

epoll wait惊群

某个进程 —
配置文件启动了accept_mutex:
    是否超负载
    开始尝试拿锁，非堵塞
    如果拿到:
        flags |= NGX_POST_EVENTS；优先处理accept事件
        释放锁
        处理正常socket事件
    没有拿到:
        返回下次的epoll_wait的超时时间,而且该tiemout缩短，意味着加大机会拿到锁

别的进程在某个进程释放mutex和epoll_wait超时后，就可以有机会拿到锁了。

疑问?

- c10k
- 服务端没有65535port限制

"END"

-xiaorui.cc