xb腾讯现场二面

自我介绍说了解Linux内核，上来说让讲两个点：讲了完全公平调度算法和内存slab分配。

100W个1亿内的数，找到第k个：字典树，计数，堆三种都说了。分析了一下时间复杂度

select和poll epoll的底层实现

epoll既然是对select和poll的改进，就应该能避免上述的三个缺点。那epoll都是怎么解决的呢？在此之前，我们先看一下epoll和select和poll的调用接口上的不同，select和poll都只提供了一个函数——select或者poll函数。而epoll提供了三个函数，epoll_create,epoll_ctl和epoll_wait，epoll_create是创建一个epoll句柄；epoll_ctl是注册要监听的事件类型；epoll_wait则是等待事件的产生。
select几大缺点：
1. 每次调用select都需要把fd集合从用户态拷贝到内核态，这个开销在fd很多时会很大。
2. 每次调用select都需要在内核遍历传递进来的所有fd，这个fd很多时开销也很大。
3. select支持的文件描述符太小了，默认是1024

epoll两种触发，什么情况下用哪种比较好。

水平触发和边缘触发，水平触发是所有三个模型都支持的，而边缘触发是epoll才有的，而且epoll默认的也是边沿触发，什么时候选择边沿触发比较好?
理论上边缘触发性能更高，但是使用更加复杂，任何意外的丢失都会造成请求处理错误。
ET方式比较危险，编程比较复杂。

服务器模型

1. 同步阻塞迭代模型
2. 多进程并发模型
3. 多线程并发模型
4. IO多路复用 select/poll/epoll
   在多进程模型和多线程(线程池)模型中，每个进程/线程只能处理一路IO，在服务器并发数较高的情况下，过多的进程/县城会使得服务器性能下降。而通过多路IO服用，能使得一个进程同时处理多路IO，提升服务器吞吐量。
   epoll模式下，没有客户端数量的限制，有100w个客户端同时与一个服务器保持TCP连接。而每一个时刻，通常只有几百上千个TCP连接时活跃的。如果时select则需要频繁的拷贝，而epoll效率就高一些。select一般只能处理几千的并发连接。
   https://blog.csdn.net/uestczshen/article/details/53057414

线程池模型中，如何与线程交互。消息队列和邮箱，共享内存，管道等哪种好。

ipc方式。

消息队列、管道、共享内存、信号量、信号、socket

了解高并发后台框架没，不了解

推导快排的时间复杂度

推到partition找第k元素的时间复杂度

稍微扯了一下项目

C++中虚函数表，虚函数指针，有三个虚函数的类的对象进行sizeof是多大，答虚函数指针的大小，8或者是4.

还问了各个排序的时间复杂度，按稳定性和时间复杂度分别说一下。

虚函数表存在C++内存哪个区？应该是常量区，不过是根据编译器的，我答了全局区，好像不太对。

C++内存分区包括哪些区?

栈、堆、自由存储区、全局静态存储区、常量存储区。

udp 中bind有什么用。

使用sendto和recvfrom需要一直绑定端口和ip，bind后可以使用send和recv，不需要一直绑定端口和ip。

kmalloc和vmalloc

从可否睡眠，地址是否连续，分配的内存块大小，效率方面来考虑。

slab如何解决内存碎片问题

析构函数能使虚函数吗？构造函数能是虚函数吗？为什么?