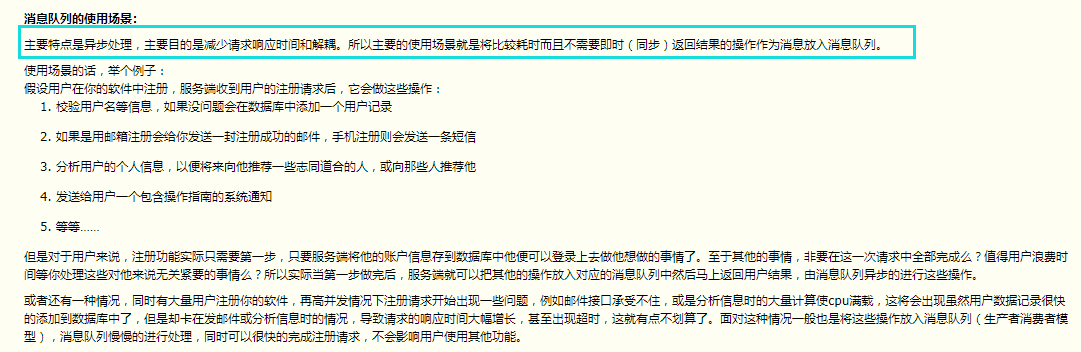
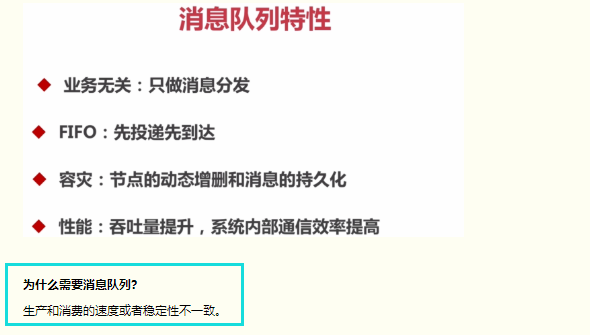
# 高性能组件

## &&&如何设计内存池（看之前的）

## 高并发场景下的消息队列





## \*\*\*协程框架的实现

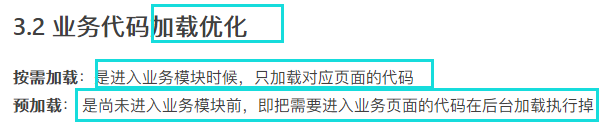
RN在公司的使用情况

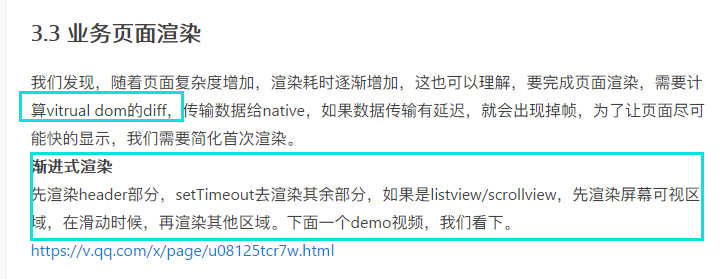




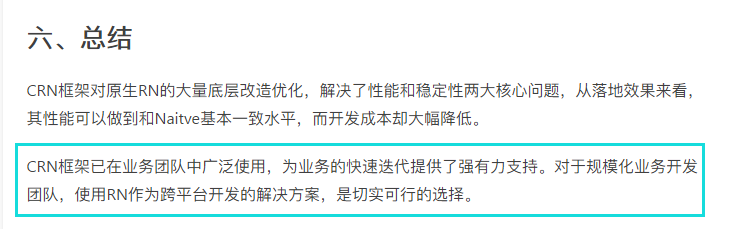






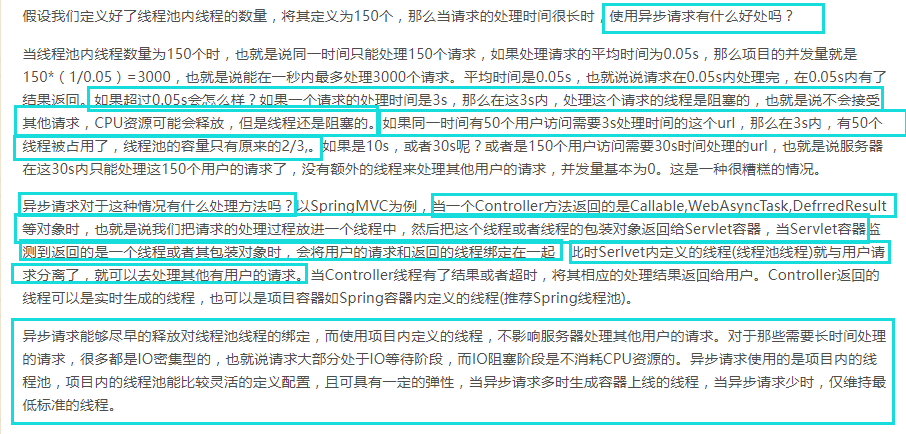




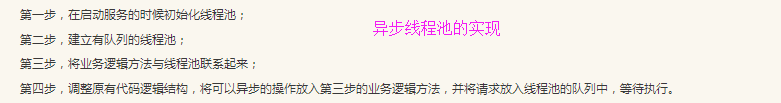


## &&手写线程池（看之前的）

## 异步请求池

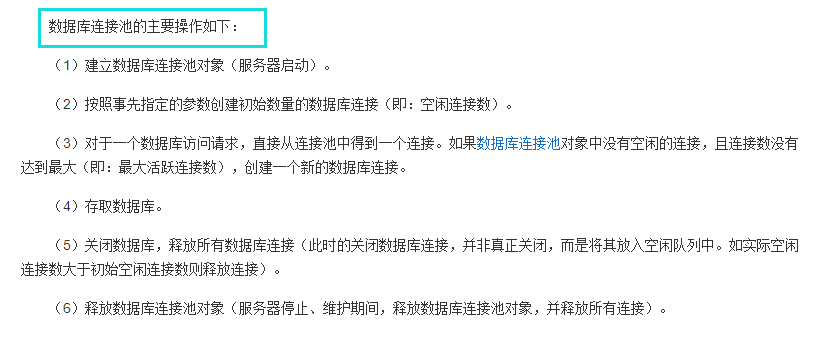


## 异步线程池的实现



## 连接池

连接池是创建和管理一个连接的缓冲池的技术，这些连接准备好被任何需要它们的线程使用。



# 高并发网络

## select,poll和epoll模型的区别

(1)select==>时间复杂度O(n)

它仅仅知道了，有I/O事件发生了，却并不知道是哪那几个流（可能有一个，多个，甚至全部），我们只能无差别轮询所有流，找出能读出数据，或者写入数据的流，对他们进行操作。所以select具有O(n)的无差别轮询复杂度，同时处理的流越多，无差别轮询时间就越长。

(2)poll==>时间复杂度O(n)

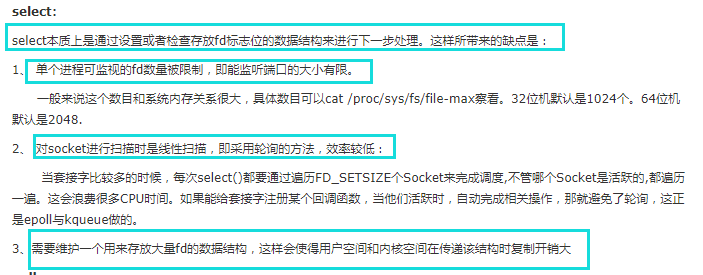
poll本质上和select没有区别，它将用户传入的数组拷贝到内核空间，然后查询每个fd对应的设备状态， 但是它没有最大连接数的限制，原因是它是基于链表来存储的.

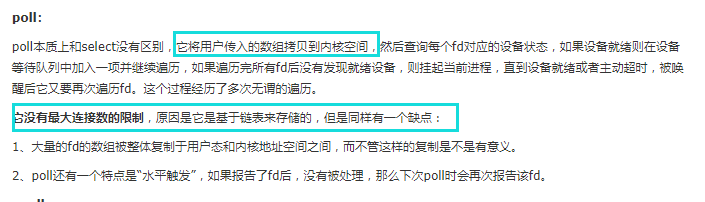
(3)epoll==>时间复杂度O(1)

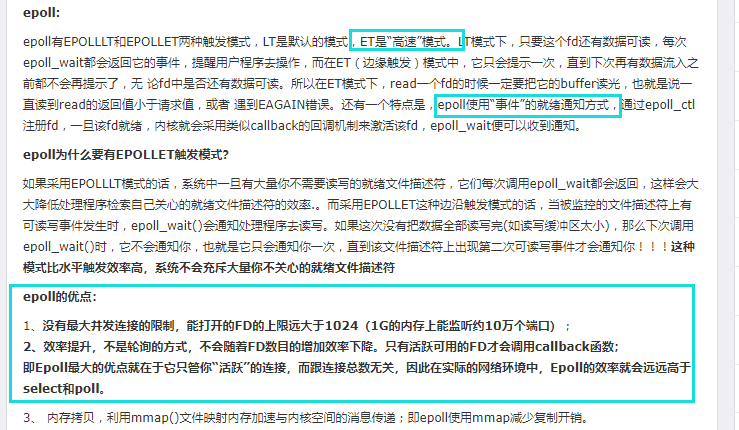
epoll可以理解为event poll，不同于忙轮询和无差别轮询，epoll会把哪个流发生了怎样的I/O事件通知我们。所以我们说epoll实际上是事件驱动（每个事件关联上fd）的，此时我们对这些流的操作都是有意义的。（复杂度降低到了O(1)）

select，poll，epoll都是IO多路复用的机制。I/O多路复用就通过一种机制，可以监视多个描述符，一旦某个描述符就绪（一般是读就绪或者写就绪），能够通知程序进行相应的读写操作。但select，poll，epoll本质上都是同步I/O，因为他们都需要在读写事件就绪后自己负责进行读写，也就是说这个读写过程是阻塞的，而异步I/O则无需自己负责进行读写，异步I/O的实现会负责把数据从内核拷贝到用户空间。

epoll跟select都能提供多路I/O复用的解决方案。在现在的Linux内核里有都能够支持，其中epoll是Linux所特有，而select则应该是POSIX所规定，一般操作系统均有实现



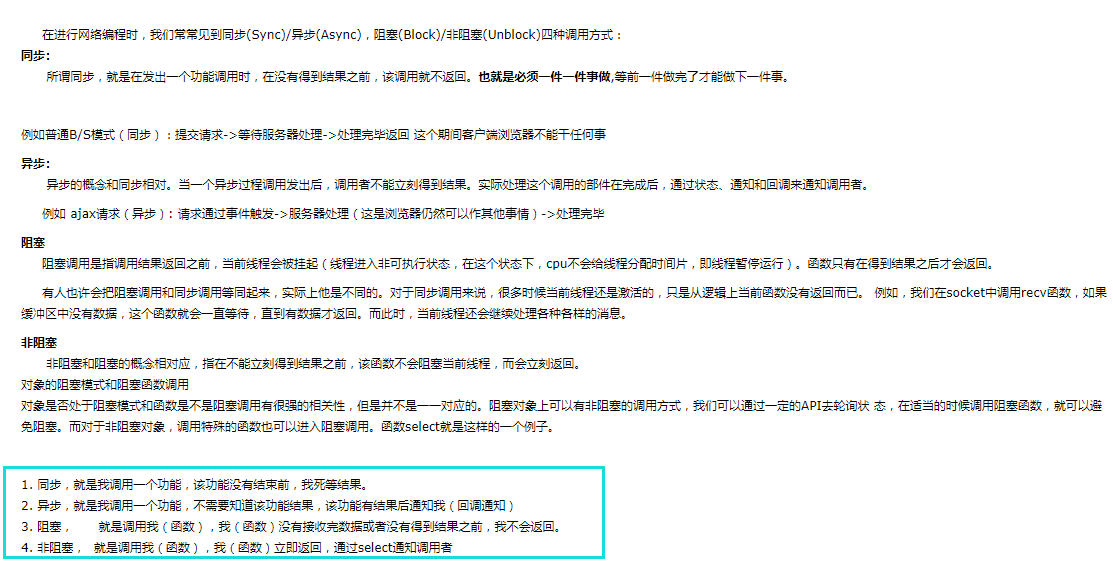






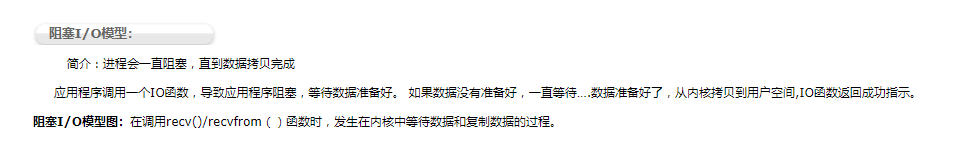
## Linux系统IO模型

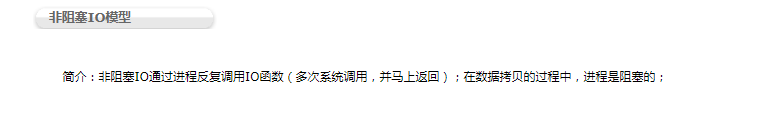


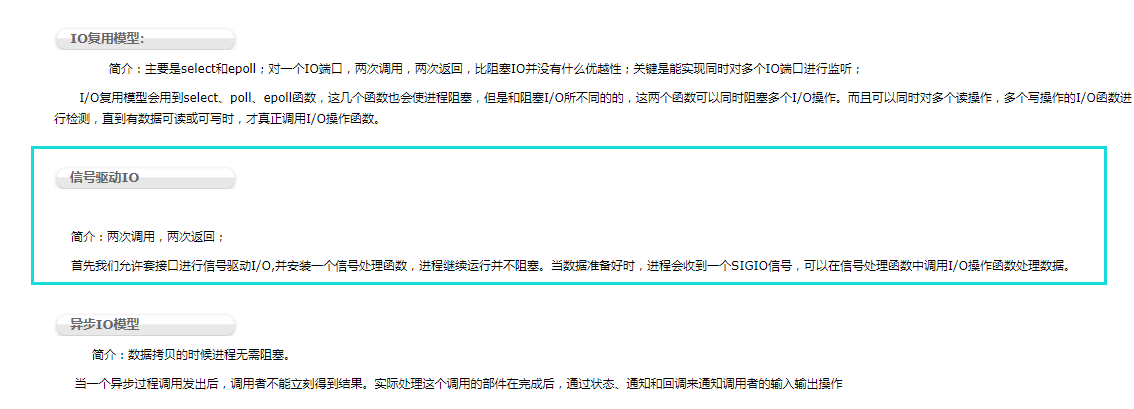


同步IO和异步IO的区别就在于：数据拷贝的时候进程是否阻塞！

阻塞IO和非阻塞IO的区别就在于：应用程序的调用是否立即返回！







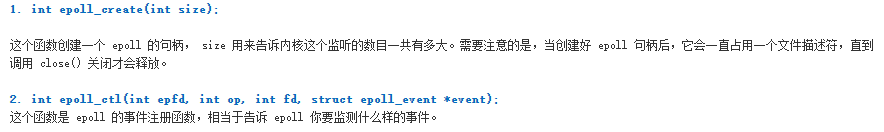
## 理解linux服务器高并发编程

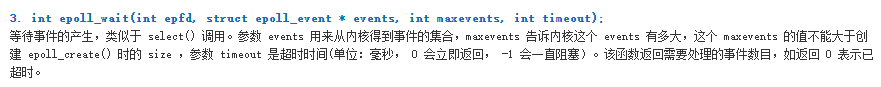
所以在编写TCP协议网络通信中，通常采用多路复用和多线程/多进程的方式实现服务器的并发，即一个服务器同时服务多个客户端。

## 网络接入层架构设计

接入层：反向代理，负载均衡，限流（简单计数法，滑动窗口法，漏桶算法，令牌桶算法），降级，熔断，超时

## 你必须懂的epoll玩法





# 并发性

&&多线程与线程安全

&&事务的隔离级别

$$公平锁&非公平锁

&&CopyOnWrite容器

$$悲观锁&乐观锁&CAS&ABA问题

$$事务的ACID特性

# Web服务

## $$反向代理nginx

$$最广泛的web服务器Httpd

## 高吸能Web平台OpenResty



# 定时调度

## 触发器按时启动原理



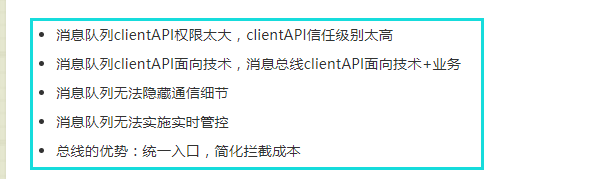
## Linux定时任务cron配置(定时调度需要时间触发，使用cron来配置触发时间)

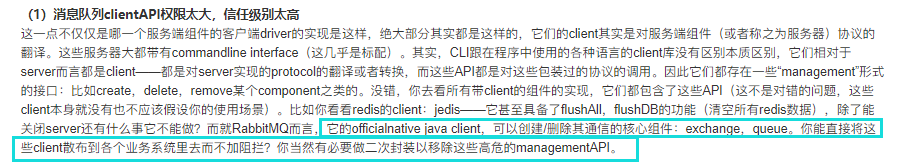
 cron是服务名称，crond是后台进程，crontab则是定制好的计划任务表。

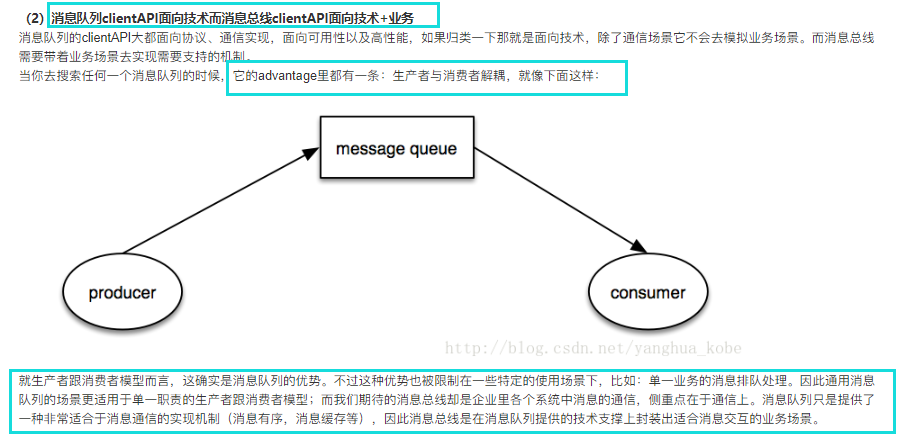
# 消息队列

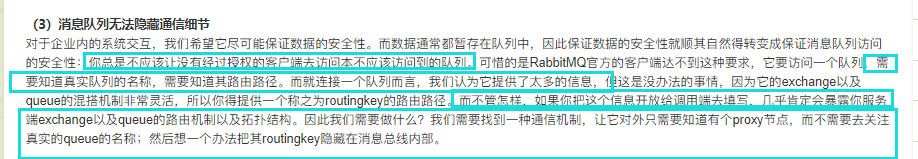
## 消息总线和消息队列

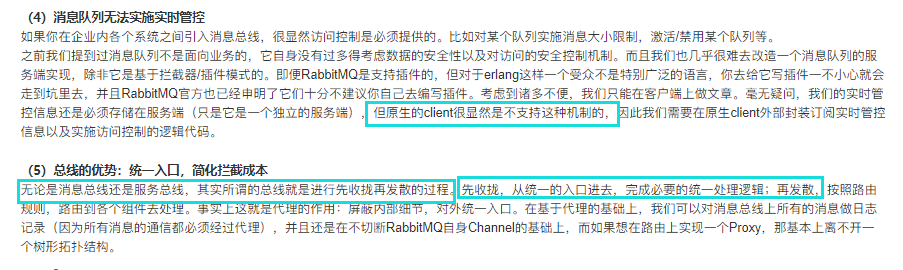
消息总线跟消息队列有何区别？如果有人问你这个问题，你的答案是什么？如果你的消息总线是基于一个已经相当成熟的消息队列或者消息系统做二次封装。为什么需要用你的客户端，而不直接用原始的（这是一个大家都相信权威的时代，请注意这里用的是相信，而不是迷信，你确实应该相信权威，至少比相信一个新手来得靠谱，当然我这里指的权威，是正面的意思）



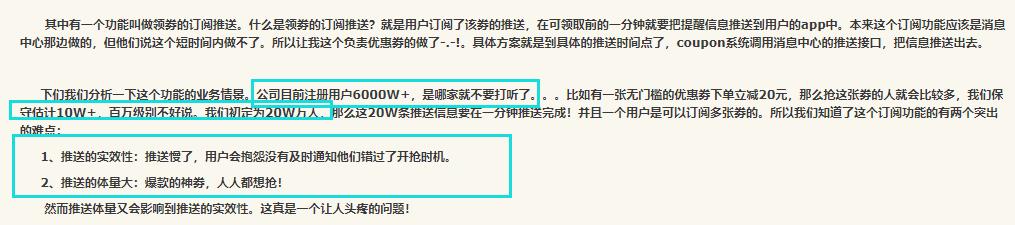








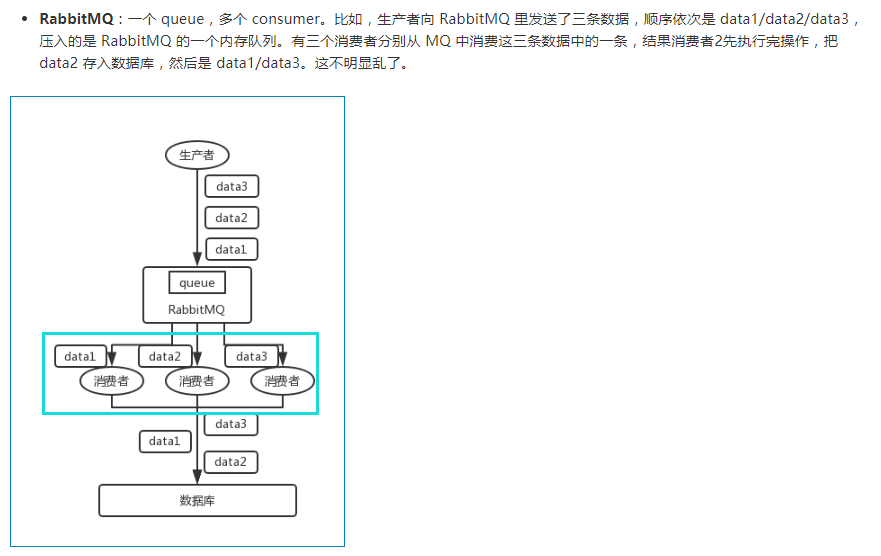
## redis消息推送

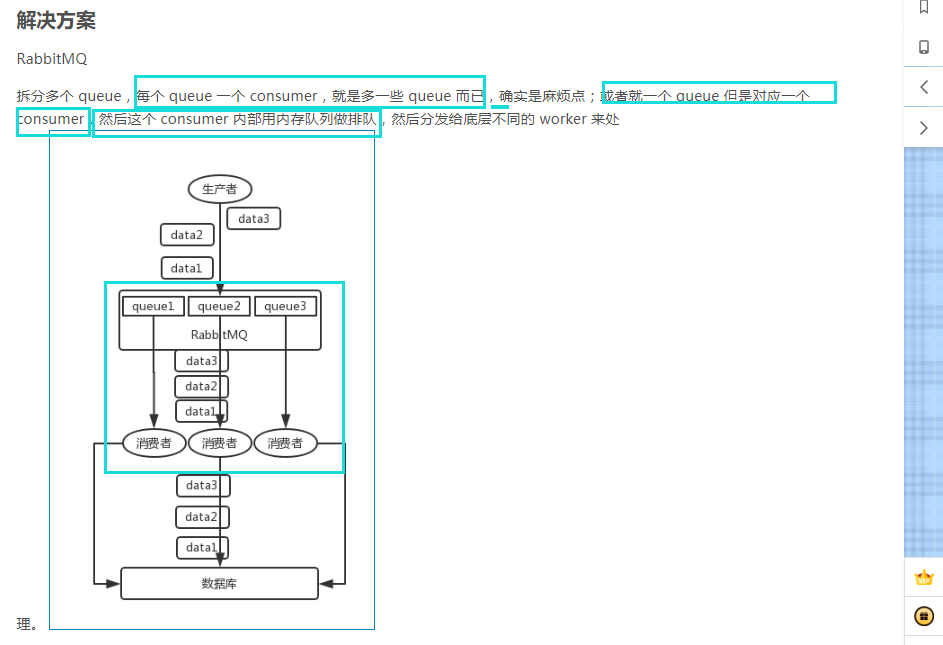


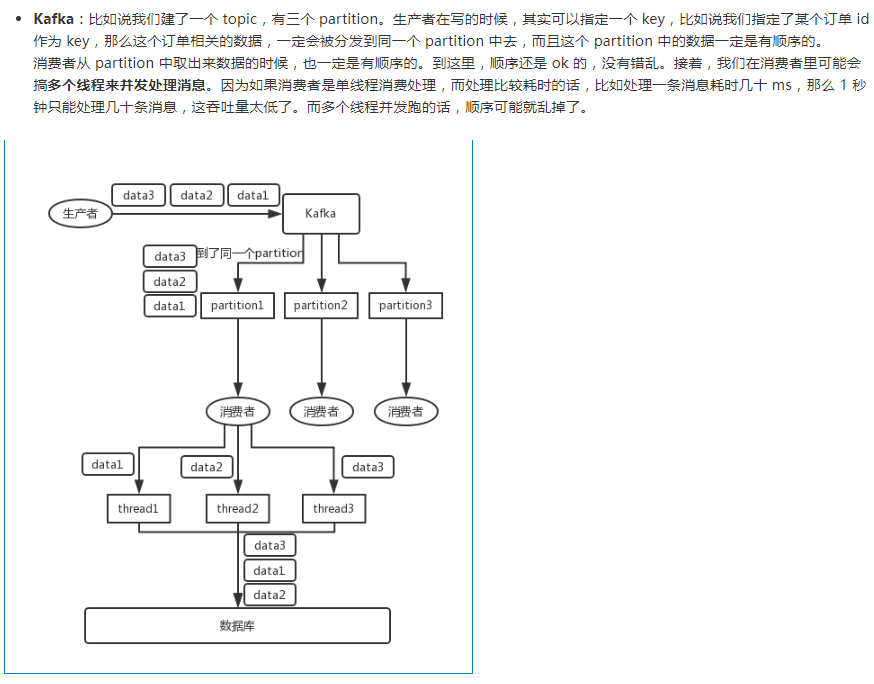


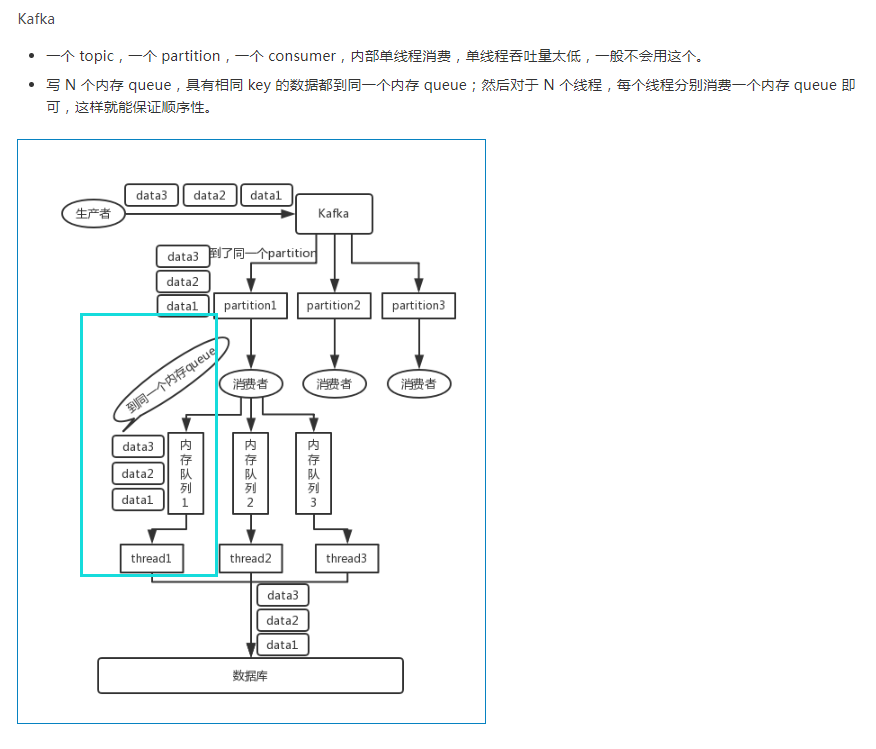


## 如何保证消费者接受消息的顺序





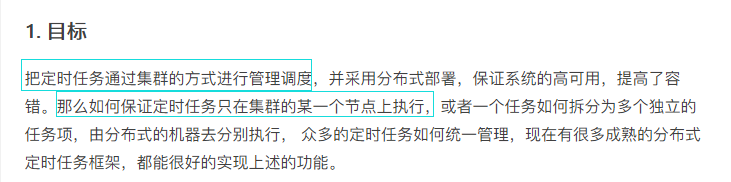




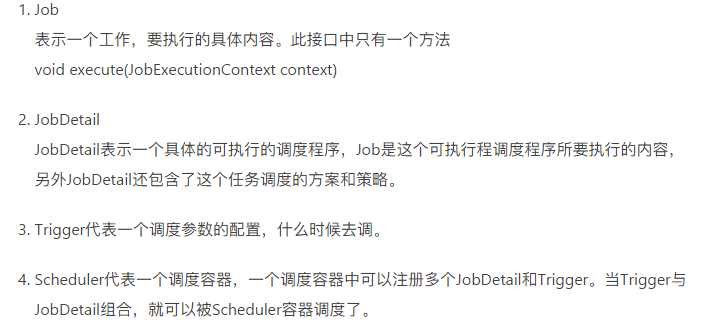
# 分布式定时调度（opencron）

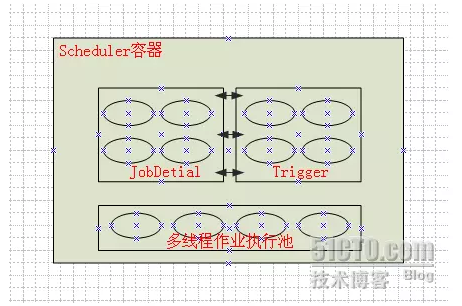


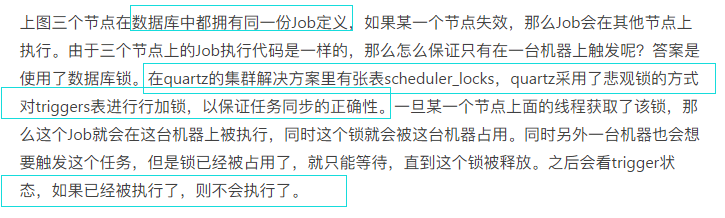
# 分布式定时任务调度框架

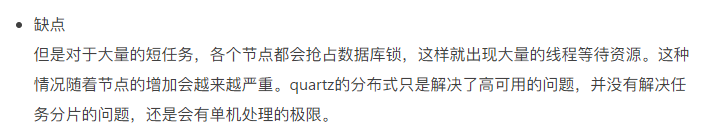












# Web安全

## CSRF（cross site request forgery）原理及防范

a)攻击原理：

i.用户C访问正常网站A时进行登录，浏览器保存A的cookie

ii.用户C再访问攻击网站B，网站B上有某个隐藏的链接或者图片标签会自动请求网站A的URL地址,例如表单提交，传指定的参数

iii.而攻击网站B在访问网站A的时候，浏览器会自动带上网站A的cookie

iv.所以网站A在接收到请求之后可判断当前用户是登录状态，所以根据用户的权限做具体的操作逻辑，造成伪造成功

b)防范措施：

i.在指定表单或者请求头的里面添加一个随机值做为参数

ii.在响应的cookie里面也设置该随机值

iii.那么用户C在正常提交表单的时候会默认带上表单中的随机值，浏览器会自动带上cookie里面的随机值，那么服务器下次接受到请求之后就可以取出两个值进行校验

iv.而对于网站B来说网站B在提交表单的时候不知道该随机值是什么，所以就形成不了攻击

我的理解：搞清楚三个点

1、什么是csrf？（cross-site request forgery）简称跨站请求伪造，首先要搞明白它是一种行为，什么行为？

伪造你的请求的行为。

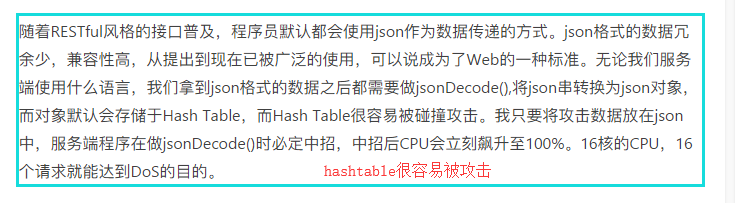
简单来说就是: 你访问了信任网站A,然后A会用保存你的个人信息并返回给你的浏览器一个cookie，然后呢，在cookie的过期时间之内，你去访问了恶意网站B，它给你返回一些恶意<br>请求代码，要求你去访问网站A，而你的浏览器在收到这个恶意请求之后，在你不知情的情况下，会带上保存在本地浏览器的cookie信息去访问网站A，然后网站A误以为是用户本身的<br>操作，导致来自恶意网站C的攻击代码会被执行：发邮件，发消息，修改你的密码，购物，转账，偷窥你的个人信息，导致私人信息泄漏和账户财产安全受到威胁。

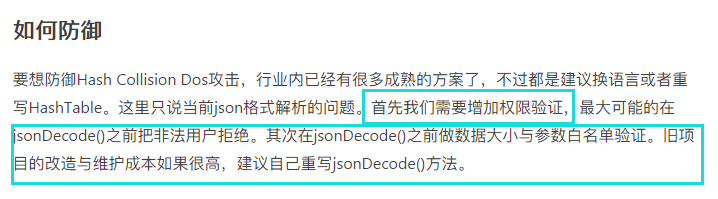
2、如何解决？在post请求时，form表单或ajax里添加csrf\_token（实际项目代码里就是如此简单）

解决原理：添加csrf\_token值后，web框架会在响应中自动帮我们生成cookie信息，返回给浏览器，同时在前端代码会生成一个csrf\_token值，然后当你post提交信息时，web框架<br>会自动比对cookie里和前端form表单或ajax提交上来的csrf\_token值，两者一致，说明是当前浏览器发起的正常请求并处理业务逻辑返回响应，那么第三方网站拿到你的cookie值<br>为什么不能通过验证呢，因为他没你前端的那个随机生成的token值啊，他总不能跑到你电脑面前查看你的浏览器前端页面自动随机生成的token值吧。

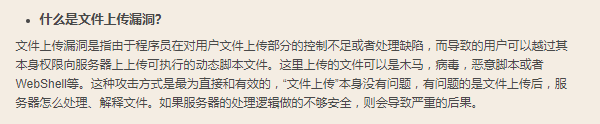
注意：你打开浏览器访问某个url（页面），默认是get请求，也就是说，你只要访问了url，对应的视图函数里只要不是if xx == post的逻辑就会执行，所以你打开页面，他会先生<br>成cookie（token）值，返回给浏览器，然后你提交表单，或者发ajax请求时，会将浏览器的cookie信息（token值）发送给服务器进行token比对，这个过程相对于你发起了两次请<br>求，第一次是get，第二次才是post，搞清楚这个，你才能明白csrf\_token是怎么比对的。

## 高级dos攻击+Hash碰撞攻击





## 脚本注入文件上传漏洞

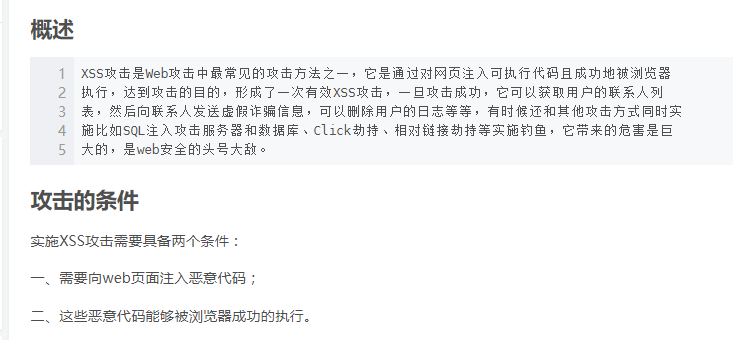




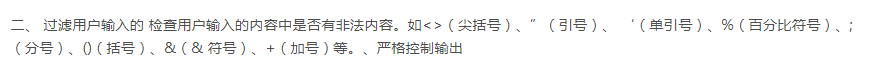
## sql注入



## xss攻击原理与解决方案







# 搜索引擎

## 搜索引擎原理

搜索引擎的基本工作原理包括如下三个过程：首先在互联网中发现、搜集网页信息；同时对信息进行提取和组织建立索引库；再由检索器根据用户输入的查询关键字，在索引库中快速检出文档，进行文档与查询的相关度评价，对将要输出的结果进行排序，并将查询结果返回给用户。

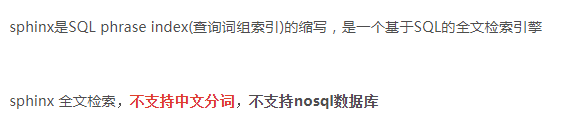
## 分布式全文搜索ElasticSearch

ES也使用Java开发并使用Lucene作为其核心来实现所有索引和搜索的功能，但是它的目的是通过简单的 RESTful API来隐藏Lucene的复杂性，从而让全文搜索变得简单。

## 全文检索Solr

solr是一个现成的全文检索引擎系统， 放入tomcat下可以独立运行， 对外通过http协议提供全文检索服务（就是对索引和文档的增删改查服务），企业中可以通过solrJ(solr的客户端的jar包)来调用solr服务。

## 基于MySql，Postgre全文检索Sphinx



## 全文检索引擎工具包Lucene

它不是一个完整的全文检索引擎，而是一个全文检索引擎的架构，提供了完整的查询引擎和索引引擎，部分[文本分析](https://baike.baidu.com/item/%E6%96%87%E6%9C%AC%E5%88%86%E6%9E%90/11046544" \t "_blank)引擎. Lucene是当前以及最近几年最受欢迎的免费Java信息检索程序库.

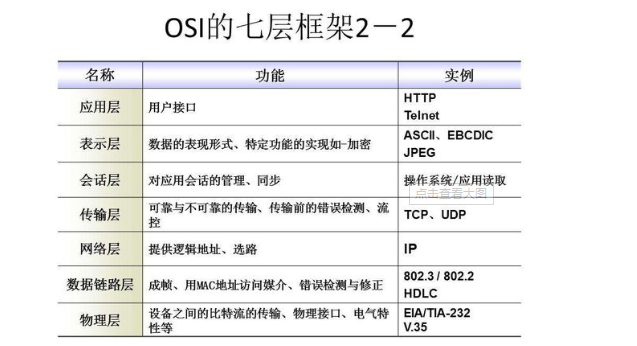
缺点

Lucene只是一个库。想要使用它，你必须使用Java来作为开发语言并将其直接集成到你的应用中，更糟糕的是，Lucene的配置及使用非常复杂，你需要深入了解检索的相关知识来理解它是如何工作的。

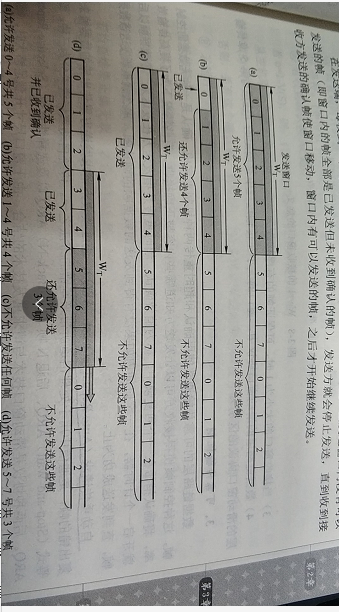
# 网络协议栈

tcp/udp/ip/eth

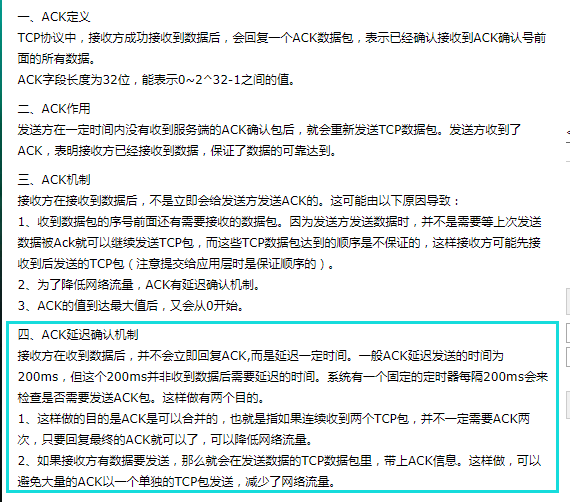




## 滑动窗口



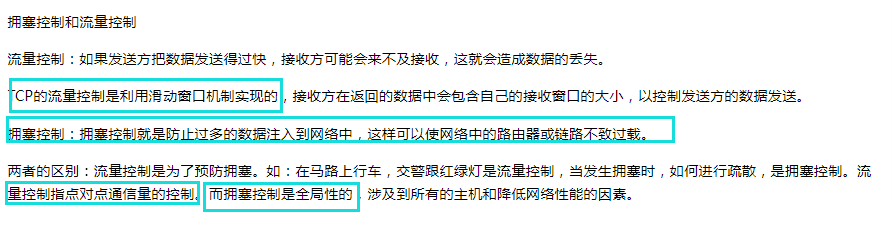
## 延迟确认

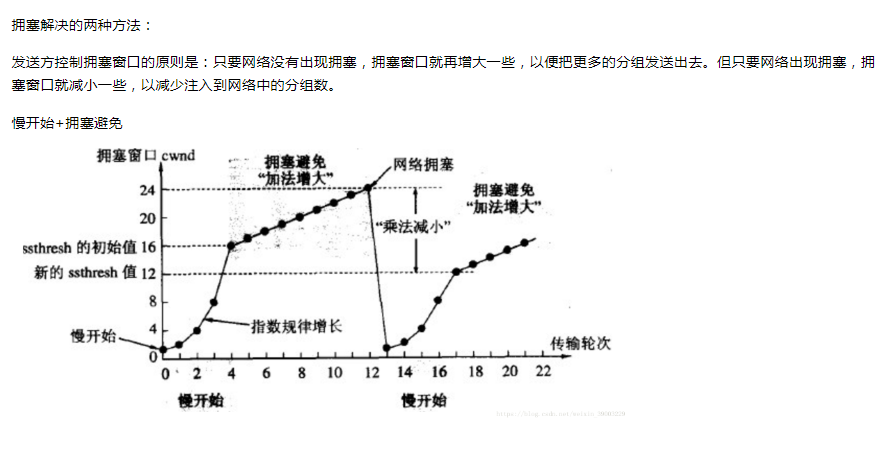


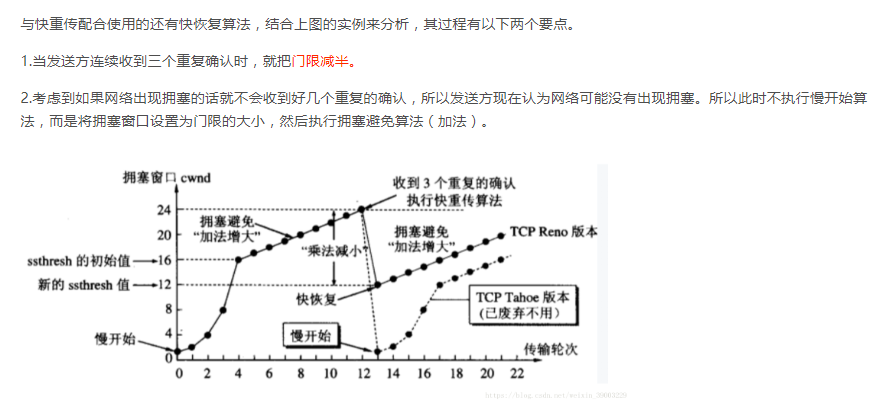
## 定时重传

每条TCP连接都会维护一个超时重传定时器，该定时器是TCP保证可靠性的一个非常重要的手段，一旦该定时器超时，那么就会重传还未收到ACK的报文

## 流量控制和拥塞控制







## tcp握手与挥手

