拒絕接口裸奔！開放API接口簽名驗證！

終端研發部 今天

轉載自業餘草

 # 接口安全問題

- 請求身份是否合法？
- 請求參數是否被篡改？
- 請求是否唯一？

AccessKey&SecretKey（開放平台）

 # 請求身份

為開發者分配AccessKey（開發者標識，確保唯一）和SecretKey（用於接口加密，確保不易被窮舉，生成算法不易被猜測）。

 # 防止篡改

參數簽名

- 按照請求參數名的字母升序排列非空請求參數（包含AccessKey），使用URL鍵值對的格式（即key1=value1&key2=value2…）拼接成字符串stringA；
- 在stringA最後拼接上Secretkey得到字符串stringSignTemp；
- 對stringSignTemp進行MD5運算，並將得到的字符串所有字符轉換為大寫，得到sign值。

請求攜帶參數AccessKey和Sign，只有擁有合法的身份AccessKey和正確的簽名Sign才能放行。這樣就解決了身份驗證和參數篡改問題，即使請求參數被劫持，由於獲取不到SecretKey（僅作本地加密使用，不參與網絡傳輸），無法偽造合法的請求。

 # 重放攻擊

雖然解決了請求參數被篡改的隱患，但是還存在著重複使用請求參數偽造二次請求的隱患。

timestamp+nonce方案

nonce指唯一的随机字符串，用来标识每个被签名的请求。通过为每个请求提供一个唯一的标识符，服务器能够防止请求被多次使用（记录所有用过的nonce以阻止它们被二次使用）。

然而，对服务器来说永久存储所有接收到的nonce的代价是非常大的。可以使用timestamp来优化nonce的存储。

假设允许客户端和服务端最多能存在15分钟的时间差，同时追踪记录在服务端的nonce集合。当有新的请求进入时，首先检查携带的timestamp是否在15分钟内，如超出时间范围，则拒绝，然后查询携带的nonce，如存在已有集合，则拒绝。否则，记录该nonce，并删除集合内时间戳大于15分钟的nonce（可以使用redis的expire，新增nonce的同时设置它的超时失效时间为15分钟）。

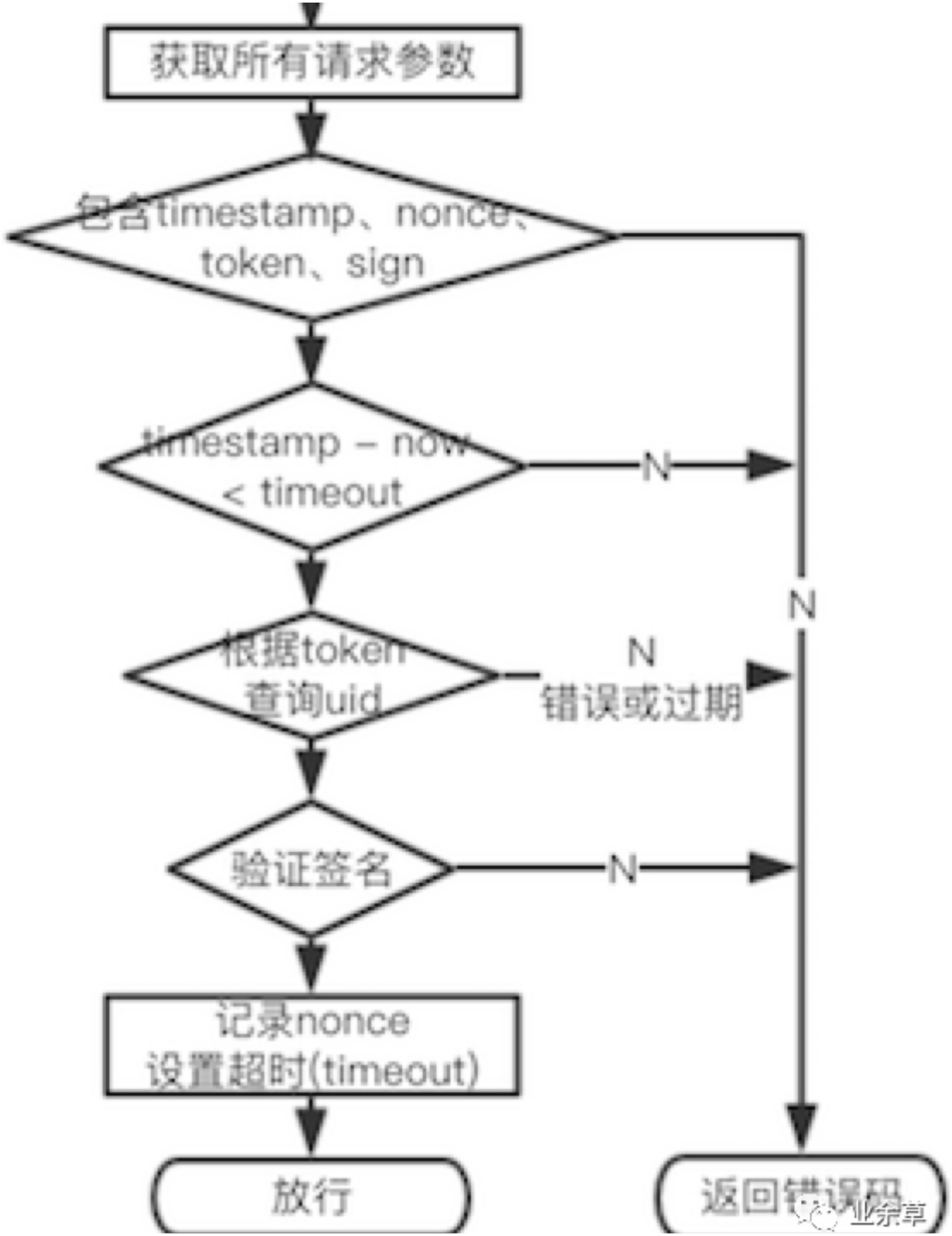
实现

请求接口：http://api.test.com/test?name=hello&home=world&work=java

客户端

- 生成当前时间戳timestamp=now和唯一随机字符串nonce=random
- 按照请求参数名的字母升序排列非空请求参数（包含AccessKey)stringA="AccessKey=access&home=world&name=hello&work=java×tamp=now&nonce=random";
- 拼接密钥SecretKeystringSignTemp="AccessKey=access&home=world&name=hello&work=java×tamp=now&nonce=random&SecretKey=secret";
- MD5并转换为大写sign=MD5(stringSignTemp).toUpperCase();
- 最终请求http://api.test.com/test?name=hello&home=world&work=java×tamp=now&nonce=nonce&sign=sign;

服务端


获取所有请求参数
包含timestamp、nonce、
token、sign
timestamp – now
< timeout
N
根据token
查询uid
N
错误或过期
N
验证签名
N
记录nonce
设置超时(timeout)
放行
返回错误码

 # Token&AppKey（APP）

在APP开放API接口的设计中，由于大多数接口涉及到用户的个人信息以及产品的敏感数据，所以要对这些接口进行身份验证，为了安全起见让用户暴露的明文密码次数越少越好，然而客户端与服务器的交互在请求之间是无状态的，也就是说，当涉及到用户状态时，每次请求都要带上身份验证信息。

Token身份验证

- 用户登录向服务器提供认证信息（如账号和密码），服务器验证成功后返回Token给客户端；
- 客户端将Token保存在本地，后续发起请求时，携带此Token；
- 服务器检查Token的有效性，有效则放行，无效（Token错误或过期）则拒绝。
- 安全隐患：Token被劫持，伪造请求和篡改参数。

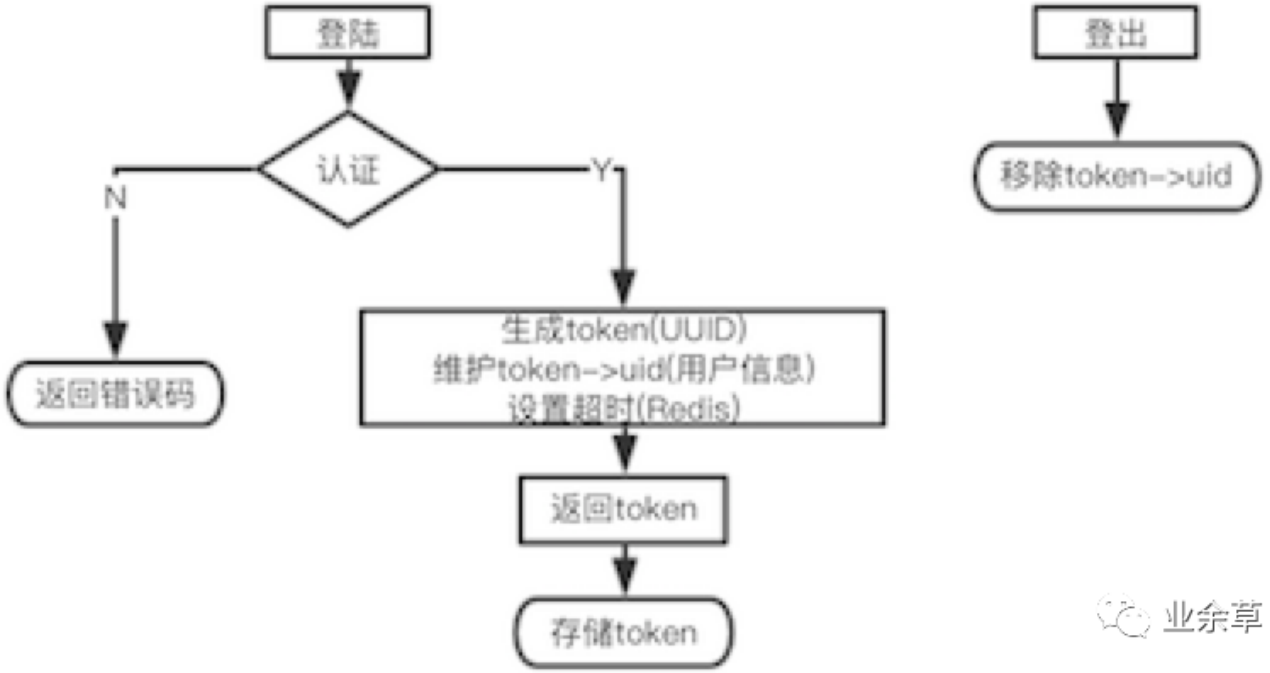
Token+AppKey签名验证

与上面开发平台的验证方式类似，为客户端分配AppKey（密钥，用于接口加密，不参与传输），将AppKey和所有请求参数组合成源串，根据签名算法生成签名值，发送请求时将签名值一起发送给服务器验证。

这样，即使Token被劫持，对方不知道AppKey和签名算法，就无法伪造请求和篡改参数。再结合上述的重发攻击解决方案，即使请求参数被劫持也无法伪造二次重复请求。

实现

登陆和退出请求



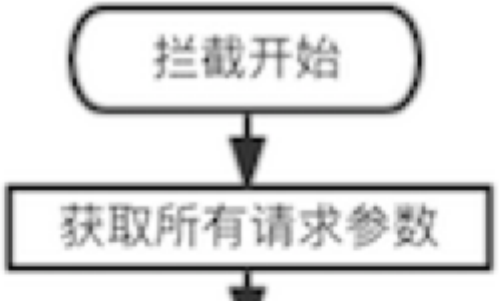
登陆和退出流程

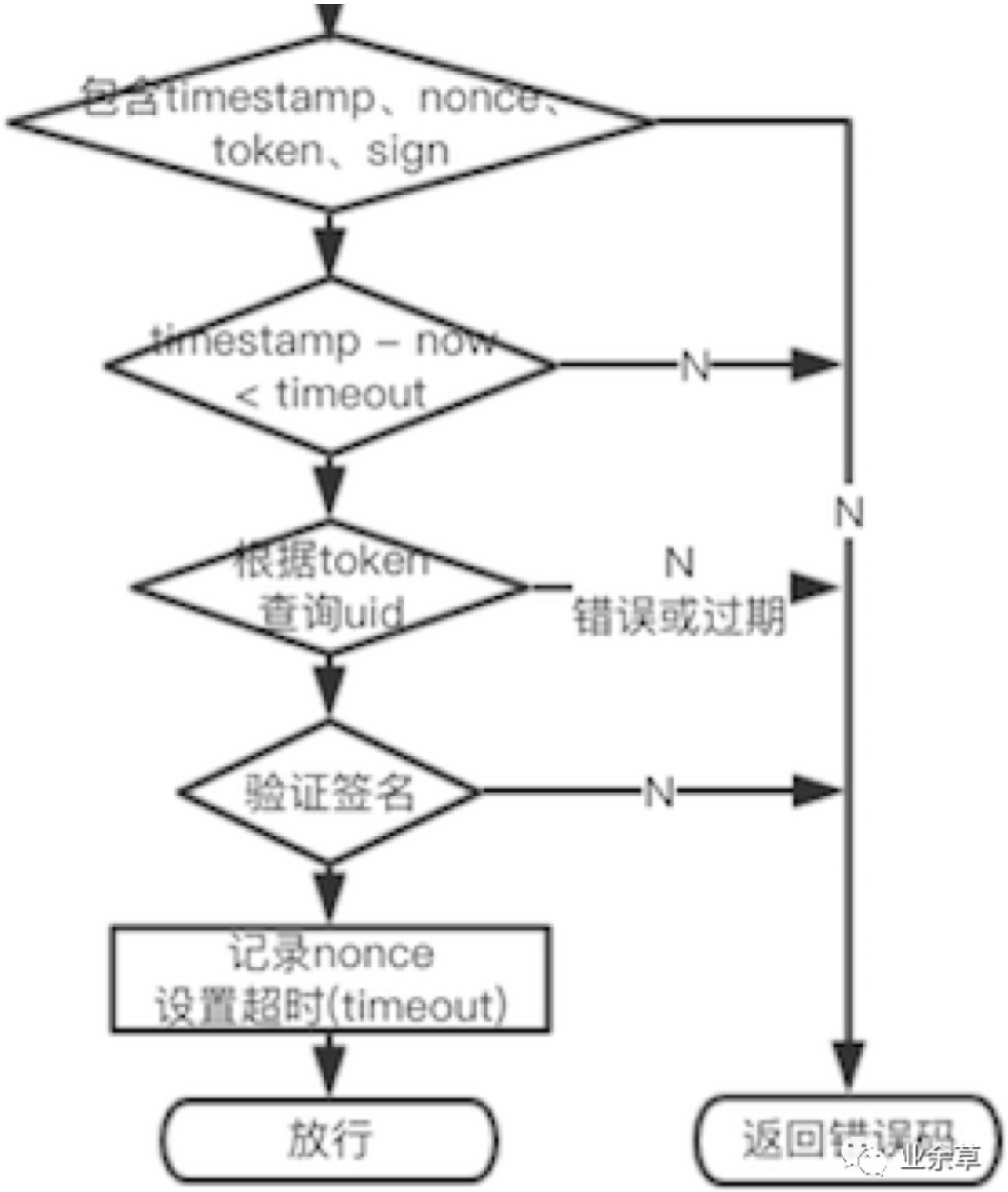
后续请求

客户端

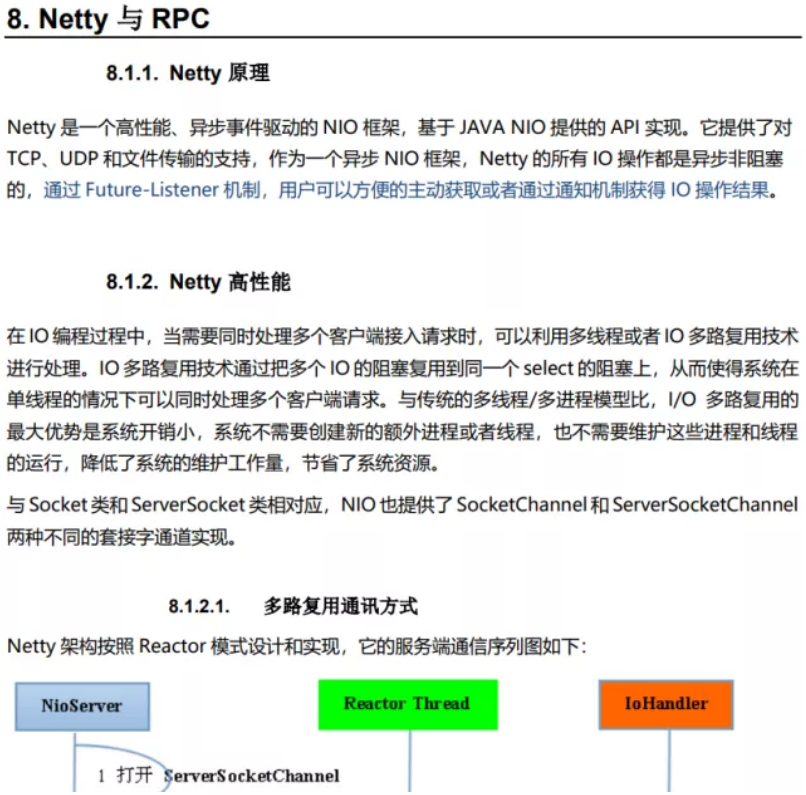
- 和上述开放平台的客户端行为类似，把AccessKey改为token即可。

服务端





BAT等大厂Java面试经验总结



8. Netty 与 RPC

8.1.1. Netty 原理

Netty 是一个高性能、异步事件驱动的 NIO 框架，基于 JAVA NIO 提供的 API 实现。它提供了对 TCP、UDP 和文件传输的支持，作为一个异步 NIO 框架，Netty 的所有 IO 操作都是异步非阻塞的，通过 Future-Listener 机制，用户可以方便的主动获取或者通过通知机制获得 IO 操作结果。

8.1.2. Netty 高性能

在 IO 编程过程中，当需要同时处理多个客户端接入请求时，可以利用多线程或者 IO 多路复用技术进行处理。IO 多路复用技术通过把多个 IO 的阻塞复用到同一个 select 的阻塞上，从而使得系统在单线程的情况下可以同时处理多个客户端请求。与传统的多线程/多进程模型比，I/O 多路复用的最大优势是系统开销小，系统不需要创建新的额外进程或者线程，也不需要维护这些进程和线程的运行，降低了系统的维护工作量，节省了系统资源。

与 Socket 类和 ServerSocket 类相对应，NIO 也提供了 SocketChannel 和 ServerSocketChannel 两种不同的套接字通道实现。

8.1.2.1. 多路复用通讯方式

Netty 架构按照 Reactor 模式设计和实现，它的服务端通信序列图如下：

想获取 Java大厂面试题学习资料

扫下方二维码回复「**BAT**」就好了

一些秘密

回复 【加群】 获取github掘金交流群

回复 【电子书】 获取2020电子书教程

回复 【C】 获取全套C语言学习知识手册

回复 【Java】 获取java相关的视频教程和资料

回复 **【爬虫】** 获取SpringCloud相关多的学习资料

回复 **【Python】** 即可获得Python基础到进阶的学习教程

回复 **【idea破解】** 即可获得intellij idea相关的破解教程

回复 **【BAT】** 即可获得intellij idea相关的破解教程

关注我gitHub掘金，每天发掘一篇好项目，学习技术不迷路！

回复 **【idea激活】** 即可获得idea的激活方式

回复 **【Java】** 获取java相关的视频教程和资料

回复 **【SpringCloud】** 获取SpringCloud相关多的学习资料

回复【python】获取全套0基础Python知识手册

回复【2020】获取2020java相关面试题教程

回复【加群】即可加入终端研发部相关的技术交流群

阅读更多

为什么HTTPS是安全的

因为BitMap，白白搭进去8台服务器...

《某厂内部SQL大全 》.PDF

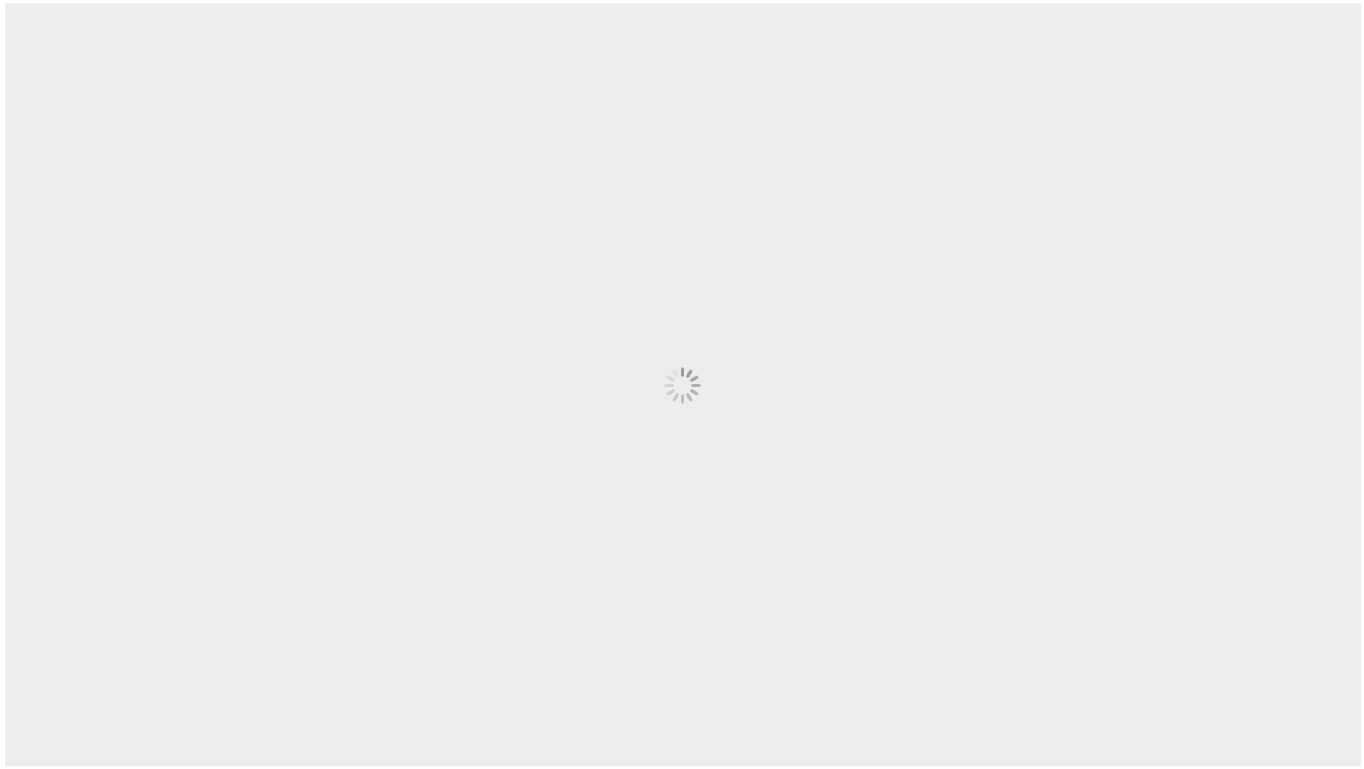
字节跳动一面：i++ 是线程安全的吗？

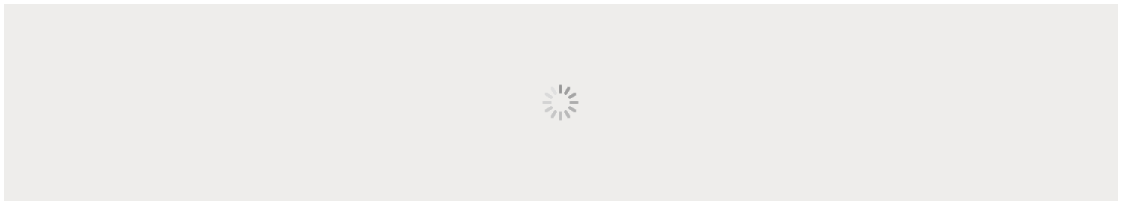
大家好，欢迎加我微信，很高兴认识你！

在華為鴻蒙OS 上嚐鮮，我的第一個“hello world”，起飛！

相信自己，沒有做不到的，只有想不到的

在這裡獲得的不僅僅是技術！





喜歡就給個“在看”

喜歡此內容的人還喜歡

男孩子在外打拼，如何識別女海王

我就BB怎麼了

藍綠部署、金絲雀發布（灰度發布）、AB測試.....

匠心零度

公司架構師常常提起的DNS負載均衡是個什麼鬼?

石杉的架構筆記