19:00开始讲座

现场赛进行中：

联系工作人员有线接入局域网
IP地址可以DHCP或联系工作人员分配
访问 http://192.168.1.80:10000/board.txt 查看赛况

高频交易与计算机科学

什么是高频交易、需要什么技术、什么样的人适合参与?

主讲人：莫涛 | 时间：2022年3月19日

题面

给定一个无限长的数字流，从中找出长度不超过 N 的连续数字串，满足其组成的正整数是任意 M 的倍数

样例输入

```
123456789987654321.....
N=6
M1=823
M2=108
```

样例输出

```
12345  # 823 的倍数
3456   # 108 的倍数
9876   # 823 的倍数
432    # 108 的倍数
```

示例代码

GX INVEST

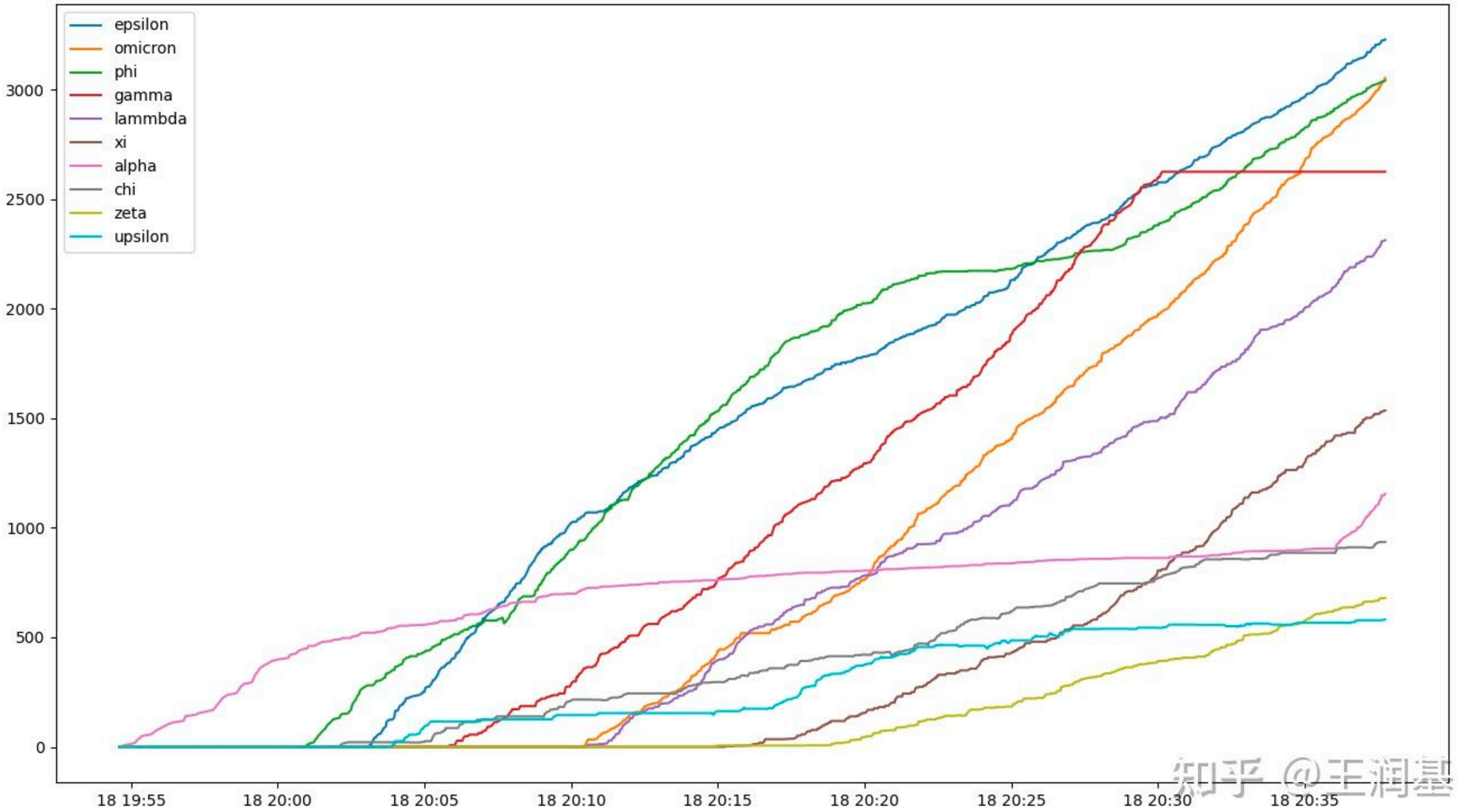
```
import requests

N = 256

M = 20220311122858

s = b""

with requests.Session().get("http://192.168.1.80:10001", stream=True,headers=None) as fin:
    for c in fin.iter_content():
        s += c
        if len(s) > N:
            s = s[-N:]
        for i in range(len(s)):
            if s[i] != ord("0") and int(s[i:]) % M == 0:
                requests.post("http://192.168.1.80:10002/submit", data=s[i:])
                print("submit", s[i:].decode("ascii"))
```

决赛结果

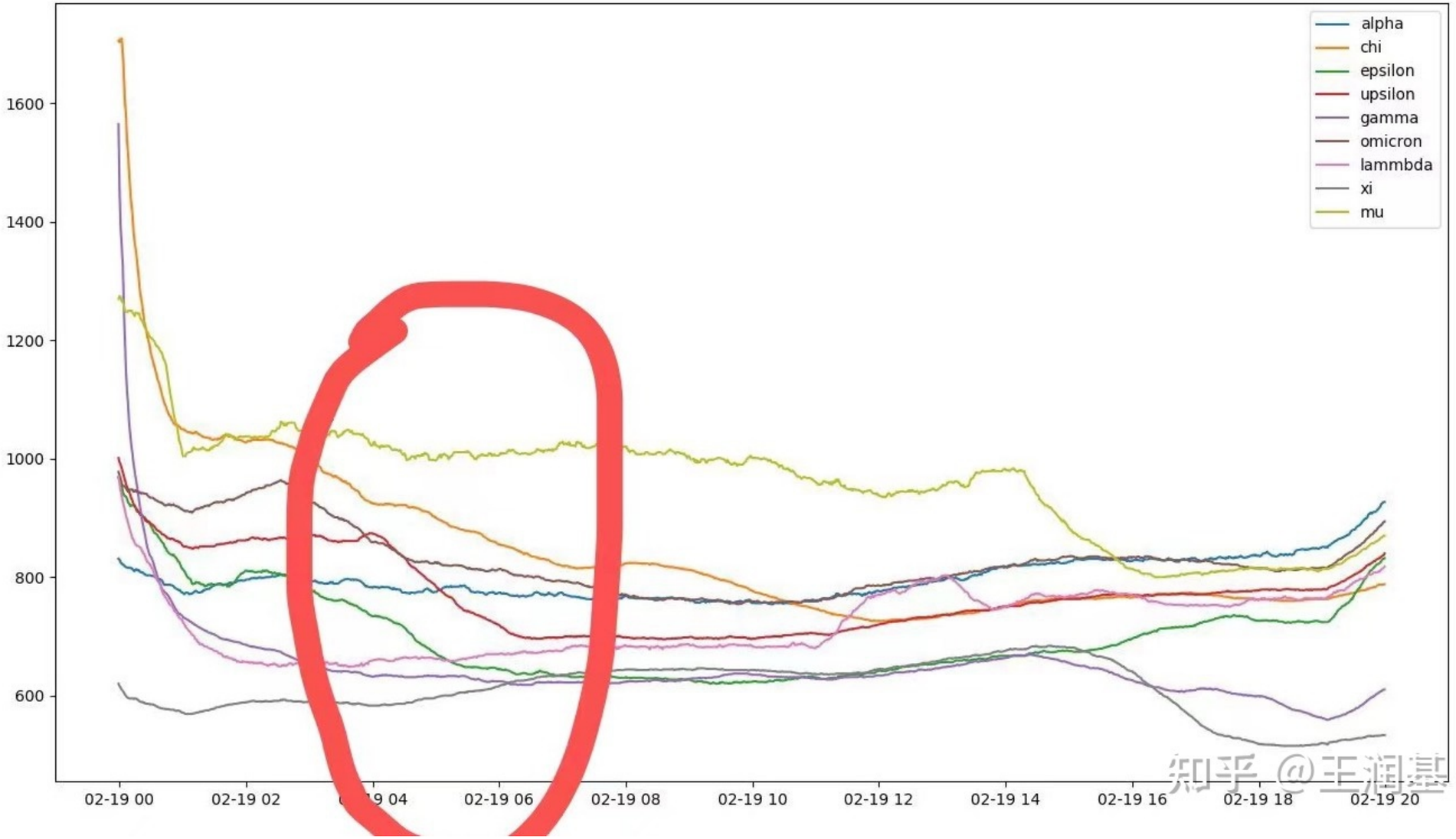
GX INVEST

```
N  = 256
M1 = 20220217214410
M2 = 104648257118348370704723119
M3 = 125000000000000140750000000000052207500000000006359661
M4 = a hidden but fixed integer, whose prime factors include and only include 3, 7 and 11

----------------------------------------------------------------      Leader Board     ----------------------------------------------------------------
(last update 2022-02-20 20:00:38.856106)
           +8     +4      +2     +1  >5s  wrong    dup  submit  10%*  50%*   90%*    99%*  mean*  std* frequent IP(ping)   score rank
xi      76695  98965   26218   3194  142   1255      2  117106   213   270    344     466    234    25  172.1.1.125(137)  947944    1
gamma   70579  86993   39447   7213   52   3954    118  118853   204   260    334     442    225    24       172.1.1.138  879858    2
lammbda 20146  48235   59813  64975   81   6492   3330  124148   224   298    428     563    251    32  172.1.1.126(121)  414561    3
alpha   17519  39724   52556  84316    8  20532    514  139975   235   513    983    1266    317    90  172.1.1.122(167)  348501    #
epsilon 11747  35102   71831  60204  238   4548     16  108049   245   372    525   37079    292    51  172.1.1.128(146)  330201    4
chi      8328  35380   78035  51039   95   1945      2   92845   188   333    459     565    240    54  172.1.1.135(288)  322408    5
omicron  4082  26368  103676  70520    1     21      0  117378   304   451    582     742    358    57  172.1.1.134(131)  298622    6
mu       6349  26972   82277  84780    1    156      6  111903   263   454    839   22913    338    74  172.1.1.124(130)  296111    7
phi      7606  32489   47039  77453  936   7273   2219  105050   305   473  25153  383550    365    66  172.1.1.123(131)  257285    8
upsilon  1742  19472   82889  85144   14  86981    416  194990   345   518    731     904    412    72  172.1.1.133(119)  147756    9
zeta     3250  10370   34991  86165    0  10586  14054   98108   370   927   3116  172621    521   164  172.1.1.129(131)  125519   10
*: latency distribution in us of recent 2000 correct submissions except front run
#: author code which has no effect on scoring and ranking

----------------------------------------------------------------     Recent Solved     ----------------------------------------------------------------
122163757577725314503711252438178689534847586743810750774857393361977215591430796984239725043439452444442947168040864783454462793050799675 7
1645358369.545511 send
1645358369.545707 epsilon@172.1.1.128
1645358369.545737 xi@172.1.1.125
1645358369.545738 chi@172.1.1.135
1645358369.545786 gamma@172.1.1.136
1645358369.545804 lammbda@172.1.1.126
1645358369.545821 phi@172.1.1.123
1645358369.545923 omicron@172.1.1.134
1645358369.545986 upsilon@172.1.1.133
1645358369.546038 zeta@172.1.1.131
1645358369.546124 mu@172.1.1.124
1645358369.546446 alpha@172.1.1.122

554817298759102745517130785994482140168270439763057402530026524692166717 ..... 64678531575302470429557309008095152762682740003307444932673014799156761 9
1645358366.973448 send
1645358366.321687 xi@172.1.1.125
1645358366.343523 gamma@172.1.1.136
1645358366.973696 chi@172.1.1.135
1645358366.973741 lammbda@172.1.1.126
1645358366.973762 zeta@172.1.1.131
1645358366.973763 omicron@172.1.1.134
1645358366.973820 mu@172.1.1.124
1645358366.973849 epsilon@172.1.1.128
1645358366.973885 upsilon@172.1.1.133
1645358366.974335 alpha@172.1.1.122
```

决赛第一小时分数走势



决赛第一日延迟走势



算法

注1: L为每次输入的长度

注2: 详见 王润基@知乎

算法

GX INVEST

注1: L为每次输入的长度

注2: 详见 王润基@知乎

三次方枚举 O(N x L x N)　注1

算法

GX INVEST

注1: L为每次输入的长度
注2: 详见 王润基@知乎

三次方枚举 O(N x L x N)　注1

枚举所有结尾在新收到的数据中，且总长不超过N的数字串
依次检查是否能被M整除，由于检查需要高精度，复杂度为N

算法

GX INVEST

注1: L为每次输入的长度
注2: 详见 王润基@知乎

三次方枚举 O(N x L x N)　注1

枚举所有结尾在新收到的数据中，且总长不超过N的数字串
依次检查是否能被M整除，由于检查需要高精度，复杂度为N

平方递推 O(N x L)

算法

GX INVEST

注1: L为每次输入的长度
注2: 详见 王润基@知乎

三次方枚举 O(N x L x N)　注1

枚举所有结尾在新收到的数据中，且总长不超过N的数字串
依次检查是否能被M整除，由于检查需要高精度，复杂度为N

平方递推 O(N x L)

维护以当前位置结尾的长度不超过 N 的数字串模 M 的余数
每新增一个数字递推计算新的余数
rem[i+1] = (rem[i] x 10 + digit[i+1]) % M
若某个rem=0则说明找到了答案

算法

GX INVEST

注1: L为每次输入的长度
注2: 详见 王润基@知乎

三次方枚举 O(N x L x N)　注1

枚举所有结尾在新收到的数据中，且总长不超过N的数字串
依次检查是否能被M整除，由于检查需要高精度，复杂度为N

平方递推 O(N x L)

维护以当前位置结尾的长度不超过 N 的数字串模 M 的余数
每新增一个数字递推计算新的余数
rem[i+1] = (rem[i] x 10 + digit[i+1]) % M
若某个rem=0则说明找到了答案

批处理递推 O(N x L / C)

算法

GX INVEST

注1: L为每次输入的长度
注2: 详见 王润基@知乎

三次方枚举 O(N x L x N)　注1

枚举所有结尾在新收到的数据中，且总长不超过N的数字串
依次检查是否能被M整除，由于检查需要高精度，复杂度为N

平方递推 O(N x L)

维护以当前位置结尾的长度不超过 N 的数字串模 M 的余数
每新增一个数字递推计算新的余数
rem[i+1] = (rem[i] x 10 + digit[i+1]) % M
若某个rem=0则说明找到了答案

批处理递推 O(N x L / C)

在平方递推的基础上，每次新增连续的C个数字并计算余数
若余数长度不超过C且符合一定的条件则说明大概率找到了答案

算法

GX INVEST

注1: L为每次输入的长度
注2: 详见 王润基@知乎

三次方枚举 O(N x L x N)　注1

枚举所有结尾在新收到的数据中，且总长不超过N的数字串
依次检查是否能被M整除，由于检查需要高精度，复杂度为N

平方递推 O(N x L)

维护以当前位置结尾的长度不超过 N 的数字串模 M 的余数
每新增一个数字递推计算新的余数
rem[i+1] = (rem[i] x 10 + digit[i+1]) % M
若某个rem=0则说明找到了答案

批处理递推 O(N x L / C)

在平方递推的基础上，每次新增连续的C个数字并计算余数
若余数长度不超过C且符合一定的条件则说明大概率找到了答案

线性递推 O(L)　注2

算法

GX INVEST

注1: L为每次输入的长度
注2: 详见 王润基@知乎

三次方枚举 O(N x L x N)　注1

枚举所有结尾在新收到的数据中，且总长不超过N的数字串
依次检查是否能被M整除，由于检查需要高精度，复杂度为N

批处理递推 O(N x L / C)

在平方递推的基础上，每次新增连续的C个数字并计算余数
若余数长度不超过C且符合一定的条件则说明大概率找到了答案

平方递推 O(N x L)

维护以当前位置结尾的长度不超过 N 的数字串模 M 的余数
每新增一个数字递推计算新的余数
rem[i+1] = (rem[i] x 10 + digit[i+1]) % M
若某个rem=0则说明找到了答案

线性递推 O(L)　注2

使用乘法逆元，将区间和转化为两个前缀和的差值
使用哈希表保存前 N 个前缀和
每新增一个数字递推出新的前缀和，若出现在哈希表中则说明找到了答案

Trick

GX INVEST

GX INVEST

分析数据规律：

M3=500000000000000221 x 500000000000000231 x 500000000000000243

预测: 枚举接下来可能的若干位判断是否整除

Trick

GX INVEST

分析数据规律：

M3=500000000000000221 x 500000000000000231 x 500000000000000243

预测: 枚举接下来可能的若干位判断是否整除

关键路径优化

M的顺序

放弃部分答案

预处理所有可能的递推

更新递推起点

Trick

GX INVEST

分析数据规律：

M3=5000000000000221 x 5000000000000231 x 5000000000000243

预测: 枚举接下来可能的若干位判断是否整除

关键路径优化

M的顺序

放弃部分答案

预处理所有可能的递推

更新递推起点

快速过滤

结尾0

答案长度下限

答案不重叠

计算


第三方库
高精度: libgmp、boost
向量化: AVX2、AVX512、Rust Portable SIM
数据结构
有损哈希表
bitmap
trie
并行
多线程计算不同的长度
多进程计算不同M
语言
u128
O3
arch=native
常量传播
多次减法替代取模
循环展开
避免动态内存

系统

Tune Operating System

GX INVEST

Tune Operating System

- 禁用超线程
- 隔离
- 禁用中断
- 实时调度

GX INVEST

Tune Operating System

- 禁用超线程
- 隔离
- 禁用中断
- 实时调度

Non Blocking Programming

GX INVEST

Tune Operating System

- 禁用超线程
- 隔离
- 禁用中断
- 实时调度

Non Blocking Programming

- spin lock / while true
- epoll / select
- socket
- avoid context switch

系统

GX INVEST

Tune Operating System

- 禁用超线程
- 隔离
- 禁用中断
- 实时调度

Non Blocking Programming

- spin lock / while true
- epoll / select
- socket
- avoid context switch

Kernel Bypass

系统

GX INVEST

Tune Operating System

- 禁用超线程
- 隔离
- 禁用中断
- 实时调度

Non Blocking Programming

- spin lock / while true
- epoll / select
- socket
- avoid context switch

Kernel Bypass

- DPDK
- 内核注入
- Solarflare

网络

网络

GX INVEST

HTTP

提前发Header

手工构造报文

zero copy

网络

GX INVEST

HTTP

提前发Header

手工构造报文

zero copy

TCP

多个连接接收数据

预先建立提交答案连接

禁用Nagle

长连接复用

网络

GX INVEST

HTTP

提前发Header

手工构造报文

zero copy

TCP

多个连接接收数据

预先建立提交答案连接

禁用Nagle

长连接复用

硬件

服务器位置

十连抽

TCP echo实验

省钱技巧
- 抢占式(Spot)
- 预留(Reserved)

周边设施

GX INVEST

数据抓取

数字流: 统计规律，验证猜想
Board: 获得M4，分析难度，其它选手表现

监控报警

错误率&分数变化
自动重启

开发测试环境

dev server
mock service

为什么叫莫队"交易"赛?

行情数据 -＞商品期货示例

沪金2206M<au2206>

卖五	396.16	7
卖四	396.14	3
卖三	396.12	12
卖二	396.10	13
卖一	**396.08**	**6**
买一	**396.06**	**21**
买二	396.04	6
买三	396.02	4
买四	396.00	21
买五	395.98	8

最新	396.04	昨结	392.62
今开	396.38	昨收	395.44
涨跌	3.42	最高	398.58
幅度	0.87%	最低	395.32
总手	120039	涨停	424.02
持仓	133907	跌停	361.20
增仓	658	结算	-
杠杆	10.0	振幅	0.83%

时间	价位	现手	增仓	开平	买价	卖价
:24	396.04	10	5	空开	396.04	396.08
:27	396.06	2	2	双开	396.04	396.08
:32	396.04	9	-3	多平	396.02	396.08
:33	396.04	4	-2	多平	396.02	396.06
:41	396.06	4	1	多开	396.02	396.08
:44	396.08	2	0	多换	396.04	396.08
:45	396.08	4	0	多换	396.04	396.08

策略示例

小美便开始介绍基于**反转因子和不均衡因子**的策略，给定一个特定标的物，交易所每秒钟会推送最近一秒内的全市场关于这个标的的所有成交的汇总信息

- 平均成交价格Price
- 主动买入成交量Buy（主动成交的定义请自行了解）
- 主动卖出成交量Sell

对于每一个时刻t，统计过去4秒即 $t-3, t-2, t-1, t$ 的最大的成交价格 $MaxP$ 和最小的成交价格 $MinP$ ，计算得到反转因子

$$A_4 = (P(t) - MinP)/(MaxP - MinP) \times 2 - 1$$

表示当前价格在过去一段时间的价格的相对位置。该因子越接近1表示当前价格越靠近窗口内的最高价，接近-1则表示靠近最低价。同样的方法，对于60秒和300秒分别计算 A_{60} 和 A_{300} 。

类似地，统计过去4秒内的总主动买入成交量 $SumBuy$ 和总主动卖出成交量 $SumSell$ ，计算不均衡因子

$$B_4 = (SumBuy - SumSell)/(SumBuy + SumSell)$$

表示买和卖的比例关系。该因子越接近1表示窗口内买的比例越高，接近-1则表示卖的比例高。同样的方法，对于60秒和300秒分别计算 B_{60} 和 B_{300} 。

计算最终的信号值

$$Signal = k_0 \times A_4 + k_1 \times A_{60} + k_2 \times A_{300} + k_3 \times B_4 + k_4 \times B_{60} + k_5 \times B_{300}$$

若Signal超出某个上涨阈值则下单买入，若Signal低于某个下跌阈值则下单卖出，否则持仓不动。所有的系数 k_i 和阈值均由策略在历史数据上统计给出。

time	price	by	sell	A_4	B_4	signal
1	100	1	1			
2	101	3	0			
3	101	1	2			
4	99	2	4	-1	0	0.2
5	96	3	3	-1	0	0.2
6	97	9	1	-0.6	0.2	0.28
7	100	2	1	1	0.28	0.024
8	101	1	2	1	0.363636	0.090909

注: 详见 莫涛@知乎

API示例

```
def insert_limit_order(price, side)

class Strategy:
    def on_md_depth(bid1, ask1 ...):
        update_model()

    def on_md_trade(price)
        update_model()

    def update_model():
        ...
        if signal > threshold:
            insert_limit_order(ask1, buy)
        if signal < -threshold:
            insert_limit_order(bid1, sell)
```

什么是高频交易

GX INVEST

下注次数

波动 vs 趋势

市场容量 & 订单簿

交易成本 & 手续费

低延迟交易

Tick To Trade

GX INVEST

Eurex&CME （180ns）

ASIC　　微波塔

同交换机

国内期货 （3~5us）

FPGA　　定制网卡

裸光纤　　同机房

数字货币 （50us）

云服务器　　虚拟化

同地域

A股

？

更多的低延迟技术

GX INVEST

更多的低延迟技术

GX INVEST

进程间通信

更多的低延迟技术

进程间通信

服务漏洞发现

更多的低延迟技术

GX INVEST

进程间通信

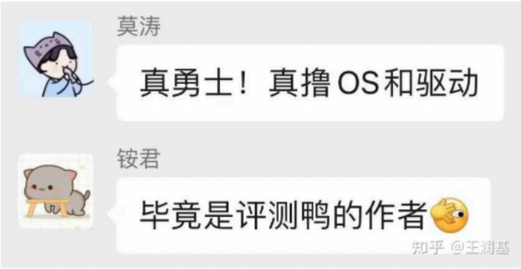
服务漏洞发现

Proxy

什么样的人适合做高频交易?

在第 29 步优化中我简单介绍了绕过内核的四种方案：Raw Socket，DPDK，内核模块，自己写 OS。猛男松爷选择了最硬核的手撸 OS 并成功上线，让我们直接看 @松 自己写的解题报告吧：

> ……（前三条是算法就不再贴了）
> 4. 此时操作系统已成瓶颈（计算 30us，内核协议栈 ??us……），因此考虑任何一种 kernel bypass 的方案（DPDK？eBPF？Linux 内核模块？自己写个 OS？）
> 5. 曾经写过一个带 UDP/IP 协议栈的纯内核态 OS，还缺个 TCP 和云服务器的网卡驱动（virtio-net），搞了两个通宵之后放弃了 TCP，写好了网卡驱动然后接了 lwIP 库上去，好用
> 6. 想办法如何在云服务器这种困难环境下部署：用 grub 启动，且设置 GRUB_DEFAULT=saved 及使用 grub-reboot 命令来保证从自己的 OS 重启后能回到 Linux，以防止丢失控制权
> 7. 应该还可以优化，lwIP 很慢（应自行实现 TCP 协议栈）、上述离线算法也很慢（启动一次附带 O(N) 时间）



我来具体解读一下：松之前自己写过一个操作系统内核 “评测鸭”，为了上云也曾尝试写过 virtio-net 网卡驱动，但是当时没有成功。这次比赛松又拿出了自己的鸭子，修好了网卡驱动，接上了嵌入式网络协议栈 lwIP，然后把计算逻辑塞了进去。为了在没有控制台的云服务器上部署调试，松首先在网络上做了特殊配置，当收到某种特定的网络包时就立刻重启（称为 GG reboot）。然后松把编译好的内核放在 grub 中，使用 grub-reboot 进入自己的内核，通过 VNC 远程连接获取屏幕输出，之后想离开时就发送特殊的网络包触发 GG reboot，接着就回到了 Linux！这一系列操作让我直接跪倒在地 _(:з」∠)_

还有一个我特别想在这里讨论的话题：什么是真正的计算机系统能力？

最近几年教育部陆续举办了多场“全国大学生计算机系统能力大赛”，分别面向 CPU、编译器和操作系统（其中 CPU 赛就是同学们比较耳熟的“龙芯杯”）。虽然我有些遗憾没有参加过这些比赛，但我认识很多同学他们都在这些比赛的历练中展示出了极为惊人的能力水平。在我看来，“莫队交易赛”在某种意义上来说也是一种“计算机系统能力大赛”，而在本次比赛中 @松 的表现向我们诠释了“系统能力”的真正内涵，那就是：利用计算机系统的原理和手段 **去解决实际问题的能力**。

解决问题！而不是自己造轮子玩儿。这才是做系统的终极目的——这也是让我感触最深的一点。

莫队交易赛所体现出的实际问题就是，怎样以最快的速度做交易，从而在市场上赚钱。这是一个很现实的问题，也是一个很有诱惑力的问题，同时是一个很有挑战的问题。要解决这个问题，除了算法上的优化以外，还需要懂网络的原理、CPU 的原理、体系结构的原理、操作系统的原理，而这些都不是比赛规则会告诉你的。更重要的是，你需要在有限的时间里，依据这些原理，选择适当的工具，运用合理的手段，改造一个系统或者做一个新系统出来，使得你比别人快。只有做到这一点，我认为才算掌握了真正的计算机系统能力。

详见 王润基@知乎

1. 思维定势

很明显，我们都陷入了自己的思维定势之中。由于长期做系统相关方向，我们拿到一个问题自然会想用系统的方法解决。而恰好这个比赛的背景又是量化投资公司组织的一场“交易赛”，很容易让人联想到低延迟高频交易系统，而这又是我们比较擅长的方向。所以当球向着系统的方向踢出去之后，很容易就一条路走到黑，再也停不下来了。

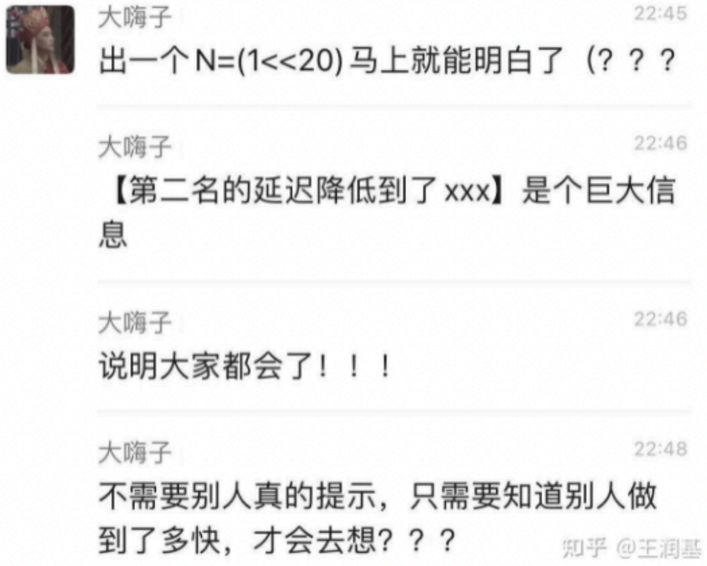
2. 信息茧房

在参加比赛的选手中，我和两位群友平时很熟，会经常在群里讨论比赛话题。而剩下还有一半的选手据说都来自公司内部，想必他们之间也会有一个群来讨论这些话题。然而我们两拨人之间却互相不认识，不知道对方是怎么做的。这样就形成了各自独立的讨论圈，或者叫信息茧房，因为你获得的信息是不断被周围环境所强化的：既然我周围的人都在讨论系统，那我就更加相信这是系统题，并且以为系统就是整个世界，完全没有意识到还有另一个世界的存在。

详见 王润基@知乎

3. 知其可为

所以事情的关键不在于知道它怎么做，而是**知道这件事可以做**。如果莫队一开始就告诉我们这是算法题，或者有人告诉我们存在线性时间的算法，那么凭借我们的知识储备总是能把它想出来的。



而事实上在比赛过程中，大部分时候我们也是看到了排行榜上别人能做到多快，才开始逼自己去想怎么才能做到这么快。最初大家都在毫秒级别互啄的时候，我是万万想不到存在微秒级别的解法的。或许有一天会出现一位大佬能把时延做到 1us 以下，我想到那时仅仅是知道这条消息就足以让人万分激动了。

详见 王润基@知乎

我们在寻找这样的人

GX INVEST

计算机科学知识

动手实践

打破常规出奇制胜

追求极致

欢迎了解罡兴投资

HR:

公众号:

罡兴投资是一个专注于二级市场量化交易的团队，通过统计和机器学习模型进行程序化自动交易。

扫描左方二维码了解更多量化资讯与岗位信息

电话：13001286513
地址：北京市海淀区清华科技园启迪大厦c座2001室
（距清华大学仅800米）

谢谢观看

地址：北京市海淀区清华科技园启迪大厦c座2001室
电话：13001286513