PDF

-
- C++
- go
- Java
-
-
-
-
-

01-

1. redis
2.
3. sql
4.
5. python import
6. nginx
7.
8.

9. easy

02-

> 1

1. 5 2.
2.
3. mysql
4. mvcc
5.
6. RAII
7. tcp udp
8. tcp
9.
   - O(1)
   - LRU
10.
11.
   -
   -

Java

Java 01-

> Bao

- b b- b+ (b– b)
- redis (key)
- (or)
- confirm broker ack broker ack id (uuid leaf)
- aqs (pv) aqs (volatile state) key (scan bigkey string hash)
- (explain es)
- jdk (17)
- ( )
- mac
- docker ( )
- :

:

Java 02- ICBU

-
-
-
- JVM “ ”
- MySQL
-
-
-
-
-
-
- JVM
-
- redis
-
-
-
-
-
-
-

Java -03-

Bao

- (25min)
-
- mvcc
-
-
- redis
- jdk1.8
- fullgc
- Spring
- SpringBoot Spring
- : LRU

Java -04-CVET

> Bao

- (20min)
- ArrayList
- HashMap remove
- CMS
-
- mvcc
- mysql
-
- Quartz Spring @Schedule xxl-job(xxx-job,)
- RabbitMQ
- RabbitMQ
-

Java -05-

> Echo

+HR +sql+

1.
2. Redis
3. Redis
4. Redis
5. Redis
6. Redis
7. Redis MQ MQ
8.
9. sql
   - sql
   - 59
     92 62 sql
10.
11. SpringBoot dao api

1.
2.

3.
4.
5.

1.
2. sql

й

l ● Redis sql

Java -06-

Java

1.

1.
2.
3.

«

3.

11. hashmap
12. redis
13.
14.
15.
16.
17. mysql RR MVCC
18. netty
19. netty
20. netty
21. netty
22. kryo ?
23.

7.27 79

1. Springboot
2. Redis,
3. RPC
4. RPC HTTP Rest
5. MySQL
6. MYSQL
7. RPC
8. RPC gRPC,Dubbo
9. Python
10. Solid,
11. if get
12.
    -
    - 1asdSD2513sdFD,
13. yyyy-MM-dd YYYY-MM-dd
14.
15. SQL LIMIT OFFSET
16. join inner join Union union all
17. linux grep head tail?)
18. java

19.
20. (CPU IO
21. threadlocal)
22. AOP AOP
23.

Java 08-

JUC

- Java ConcurrentHashMap
- ConcurrentHashMap
  64
- cas java synchronized
- java synchronized
- synchronized ReentrantLock
- Atomic
- CAS
  ?
- A B C wait
  notify boolean A boolean wait nofity
  sychroned?

JVM

- JVM ?

MySQL

- sql where a > 1 and b = 1 and c > 1 bca bac)
- b

>

>

3.

1 http https

1

HTTP URL “http://” 80
HTTPS URL “https://” 443

2

HTTP TCP

HTTPS SSL/TLS HTTP SSL/TLS TCP

HTTP HTTPS HTTPS HTTP ;

3

DES AES ;

4

RSA DSA

2 https

3

4 dns www.google.com dns

DNS

1 IP ;

2 ;

3 com ;

google IP

:

com -> google.com -> www.google.com

www.google.com.

. . .

DNS

:

. -> .com -> google.com. -> www.google.com.

5 tcp time-wait close-wait

1. FIN
2. ACK
3. FIN
4. ACK

4

1 FIN

FIN_WAIT_1

2 FIN

CLOSE_WAIT ACK +1 SYN FIN

3 FIN

LAST_ACK

4 FIN

TIME_WAIT ACK +1 CLOSED

CLOSE_WAIT

Q: CLOSE_WAIT

A: Sever Client FIN

Q: CLOSE_WAIT

A: Server Client FIN

6 Tcp

1. TCP
2.
3. TCP
4. TCP

7 TCP

1

2

8 http 1xx-5xx 301 302

9 http

10 DNS,HTTP,HTTPS,TCP,UDP UDP

11 DES,AES,RSA

12 xss cors

4.

1 Mysql mysql ACID undo-log,redo-log, ,mvcc MVCC

Atomicity Consistency Isolation Durability

A 50 B ACID

1

Q

OK A 50 B 50 50 ~

2

Q

OK A 200 B 0 A B 50

ironartisa*

A B 50

3

Q

Mysql CPU Mysql

(redo log)

4

Q

:

A 200 300 A -100

0

A 200 50 B A B
A+B

2

Mysql

OK

1

ACID C( ) A( )
I( ) D( ) AID

B

3

2 Mysql undo log

OK Innodb undo log

undo log sql

delete insert ;

update update ;

insert delete .

undo log rollback undo
log

3 Mysql (redo log)

OK Innodb redo log
Mysql

Q

A

Q

A 16kb
16kb

SQL IO
IO

redo log redo log
redo log (redo log)
redo log undo log binlog

Q redo log

A redo log :

redo log

redo log IO IO

4 Mysql (MVCC)

OK MVCC

t_balance

id	user_id	balance
1	A	200
2	B	0

ironartisa*

id user_id balance

隔离级别为Repeatable Read

由于user_id列没有索引
将t_balance整表锁上

事务1	事务2
mysql> begin; Query OK, 0 rows affected	mysql> begin; Query OK, 0 rows affected
mysql> update t_balance set balance=balance-50 where user_id = 'A'; Query OK, 1 row affected	
	mysql> update t_balance set balance=balance-50 where user_id = 'A'; //由于获取不到表锁，阻塞住
mysql> update t_balance set balance=balance+50 where user_id = 'B'; Query OK, 1 row affected	
mysql> commit; Query OK, 0 rows affected	
	//由于事务1提交，获取到锁 Query OK, 1 row affected
	mysql> update t_balance set balance=balance+50 where user_id = 'B'; Query OK, 1 row affected
	mysql> commit; Query OK, 0 rows affected

ironartisa*

MVCC, (Multi Version Concurrency Control)
undo log

DELELE UPDATE

MVCC (Repeateable Read) (Read Commited) MVCC

(Read Commited)
(Repeateable Read)

MVCC

MVCC
mvcc mysql trid rollbackpointer

MVCC

undo.log trid
rollbackpointer

1. RC
2. RR

1 Snapshot Read

MySQL InnoDB MVCC
Consistent Nonlocking Read

2 Read Committed

1. RC
2. “ ”
3. “ ” read

3 Repeated Read

1. RR
2. “ ” “ ” read

Q 6 RC RR

1. update /insert /delete
2. RC
3. RR read T T
read

1. read
2.

7 DB_ROW_ID 10 update DB_ROLL_PTR
DB_ROLL_PTR

TODO

8 b+ MEMORY hash

MySQL B+Tree B-Tree

1 B-Tree

1. data
2. data
3.

2 B+Tree

1. data
2. data
3.

mysql

B+ hash

InnoDB B+Tree

InnoDB Hash B+Tree Hash

9 Mysql int string b+ String

TODO

10 Mysql

TODO

5.

1 [“hello”,”max”,”aello”,”world”], search(String s), s1 s 1 s1

2

Offer 04.

target target

1. arr[] n n 0 O 1 O n
2. arr[] 2 k
3.
4. 90
5. 1 LeetCode 300

-
-
-
-

6.

mysql

https://www.yuque.com/renyong-jmovm/ds/zgpx1l#985980fa
https://www.yuque.com/noahnyy/mysql/yl3xrb#edb07147

Java 11-

> @author ironartisa

1. Java

1.1 volatile

1

;

1 volatile CPU CPU
CPU CPU cache

2 volatile
CPU cache

2

1 JVM
;

2 ;

3 latile volatile
.

3 synchronized

1. volatile volatile synchronized ;
2. volatile synchronized ;
3. volatile synchronized ;
4. volatile synchronized

1.2 threadLocal key

JDK ThreadLocal

ThreadLocal ThreadLocal

ThreadLocal
ThreadLocal
get() set()

ThreadLocal Thread
ThreadLocalMap ThreadLocalMap key ThreadLocal value ThreadLocal
set

ThreadLocalMap key ThreadLocal , value

ThreadLocal key value
ThreadLocalMap key null Entry value
GC

ThreadLocalMap set() get() remove() key null
ThreadLocal remove()

1.3 java 4

JDK1.2

Java reference
;

JDK1.2

Java

1 (StrongReference)

;

;

Java OutOfMemoryError

2 (SoftReference)

ReferenceQueue JAVA

3 (WeakReference)

ReferenceQueue Java

4 PhantomReference

" "

ReferenceQueue

JVM

OutOfMemory

2. JVM

1 CMS

CMS Concurrent Mark Sweep

CMS Concurrent Mark Sweep HotSpot

Mark Sweep CMS - ”

1

root ;

2

GC

GC

;

3

;

4

GC

1. CPU
2.
3. -“ - ”

2

1

- 1
- 1

0

2

“GC Roots”
GC Roots

3

1

- GC ;
- GC ;
- GC

2 -

Mutation 3 3

Root ;

;

3

1. ;
2. ;
3.

GC

4 cpu 100%

CPU100% -

3.

1

2

1

1.
2.
3.
4.

2

1.
2.

3

1. socket
2.
3. RPC
4. HTTP

3

wait() waitpid()

;

(init) exit() (Zombie)

exit()

ps “Z”;

kill

4

CPU MMU

5

5

CPU

1 (FCFS)

CPU

2 (SJF)

CPU

3

RR(Round robin)

4

UNIX

5

FCFS

4.

1

Client
Server ;

Client Server ;

Client Server
;

2 Q

ESTABLISHED
ESTABLISHED

ESTABLISHED
CLOSED

3 time-wait

2

1. TCP
2.

4 tcp

1 TCP

2 TCP

3

TCP
TCP

4 TCP

5

TCP TCP
TCP
TCP

6

7 ARQ

8

TCP

5.

1 B+ B+

B+ B B B+

B+ B+

B+ B+

B+ B+

6.

1

2

Java 12

> @author

1

2 id

3 volite JVM

4 synchronize

5 spring

response request

6 tcp 3

100

7 G1 CMS GC 3 - - -

> CMS

8

G1 CMS

jvm java .... jvm volite

6 7

😂

Java 13

> @author weikunkun

二面（40分钟）

1. 自我介绍，项目中如何维护状态机的
    1. 然后详细阐述了每个状态的转化
    2. 异常状态的处理
    3. 还有些细节忘了，实习期间没有做好笔记，然后面试官说这个感觉状态机流转的有些问题
2. 然后就看到我简历上写了了解Linux命令（说面了那么多实习生，终于遇到一个写熟悉Linux命令的。。。），然后后面30分钟就开始共享屏幕写各种命令
    1. 查看自己电脑运行了多少idea进程
        - ps -ef | grep idea，然后数了数
    2. 然后问了idea启动的进程是哪个
        - 瞅了了一眼，说路径最短的那个。。。
    3. 进入这个目录里面
        - cd XXXX (空格需要转译)
        - 查看文件大小
            - 返回上一级，ls -h
                - 第一列代表啥意思
                - 当前用户、用户组、其他用户权限?
                - 可执行、可读、可写？1，2，4?
                - 反正能说的都说了
        - 查看文件类型
            - 直接vim看来里面内容，然后感觉很RDB文件很像，说是二进制文件，同时是可执行文件。。。
            - 原来是想问file命令
    4. 把这个路径配置到当前的环境变量里面
        1. export PATH = $PATH:/xxxx
    5. 如何直接通过命令行执行idea
        1. 输入 idea（和配置MySQL到当前会话一个道理）
3. awk的问题，一个服务端业务逻辑耗时统计日志，统计服务端的耗时情况
    1. 不咋会awk，所以就说明了大概思路。
    2. 然后说先 cat xxxx.log | (xxxxxx) | printf(xxxx) > result.log
    3. 最后如果要排序的话，sort xxx result.log > result.log
    4. xxx表示不会写的东西。。。
4. 反问环节
    1. 有每周的技术分享会吗
    2. 对于codereview的流程，怎么避免需要频繁的rebase操作
    3. 业务场景，一些较为复杂的业务场景使用了哪些技术栈，会技术栈做一些自定义的横纵向拓展吗
    4. shell编程咋个练
        1. 不需要太过了解，但是对于一些提高效率的操作可以多试试，玩玩儿
    5. 本来还想问能不能进入下一轮，最后还是忍住了，随缘吧
5. 评价：
    1. 作为实习生，通过刚才的阐述能感觉到开发能力是属于还不错的类型，八股文这些，一面已经问过了，所以就不问你了
    2. 经验不太够，本来挺期待关于命令操作的，shell编程有待提高，一些基础命令用的还行。

Java 14

> @author

2 Java 2
45

Mybaits
Java JVM 5

1,2

b+
Hibernate
ArrayList LinkedList

4

2,3

Java
“ ”
18

Java 15

> @author

HR

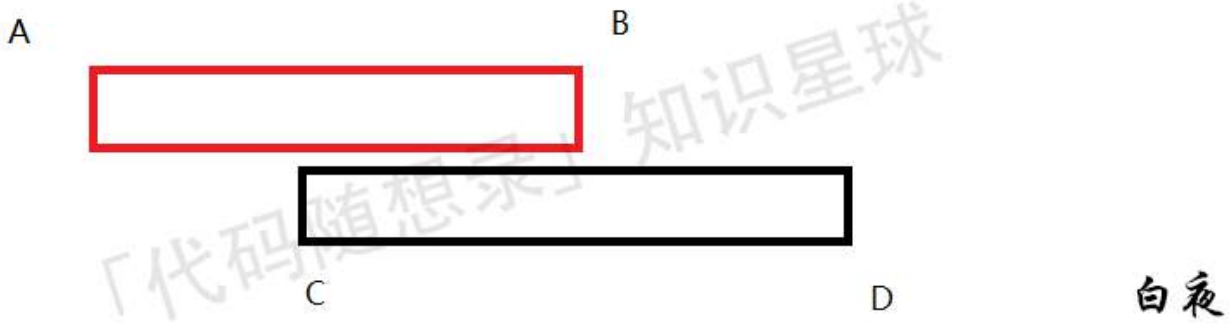
1.

2. memcpy
3.

```
class SafeSingle {
    public:
        SafeSingle* Instance() {
            Wait(event);
            if (_ins == nullptr)
                _ins = new SafeSingle;
            Release(event);
            return _ins;
        }
    private:
        static SafeSingle* _ins;
        static HANDLE event;
        SafeSingle() { event = CreateEvent(); }
}
```


_ins
nullptr new wait
nullptr new wait

src nullptr



A-B C-D

C++ C++11 decltype
C++ 11
shared_ptr unique_ptr weak_ptr

weak

set vector list map list map list map

Windows

offer

Java 16

> @author weikunkun

一面（50min，终于记得录音了！！！）

自我介绍。

为什么投递Android，看简历更适合后端嘛，然后说了HR联系说没有HC了，问是否愿意转岗

1. HashMap的阐述 ⚠️
    1. 先笼统的说了整体上采用的什么数据结构，然后解决hash冲突的方式
    2. 然后详细阐述了HashMap的具体实现细节，节点的构成、触发扩容的情况、红黑树、CRUD的逻辑、equals、hashcode关系等
    3. 面试官调侃说讲的挺细的，然后自己感觉有些地方说的有些问题🐵
2. ArrayList的阐述 ⚠️
    1. 底层实现、扩容机制、尽量初始化时制定容量大小，避免频繁扩容的开销，
    2. 说了头部插入、中间插入、尾部插入的性能区别
3. 线程安全
    1. 线程安全需要解决的核心问题：
        1. 原子性
        2. 可见性
        3. 有序性
    2. 实现线程安全的方式
        1. 阻塞同步：加锁
        2. 非阻塞同步：cas
        3. 无同步方案：本地存储
4. final关键字
    1. 修饰
        1. 类
        2. 方法
        3. 成员变量
    2. 然后讲了保证了可见性
    3. 然后说了看了一些源码里面的finaly的修饰、譬如AQS的release、acquire方法、String类本身等
        1. 主要是为了保证自身的安全性的机制
5. Integer、int的区别 ⚠️
    1. 集合类的不支持基础数据类型存储
6. 问了泛型⚠️
    1. 泛型说的比较含糊

2. 避免不必要的类型强转
3. 设计的类或者说数据结构，更加具有通用性，支持各种数据类型：JDK定义的、自定义的

7. String、StringBuilder、StringBuffer
    1. 对与需要频繁进行拼接、截取的操作，使用StringBuilder、StringBuffer
    2. StringBuffer相比于StringBuilder，保证了单个API操作的线程安全
8. TCP、UDP的区别
9. 问了拥塞控制 ⚠️
    1. 有些模糊了，所以说的极其凌乱。。。
10. 单例模式
    1. 基础饿汉模式
    2. 基础懒汉模式
    3. DCL 注意指令重排
    4. 还想说其他的实现，被打断了，说差不多了
11. 单元测试
    1. 复杂逻辑
    2. 模拟请求，输出格式是否符合要求
    3. 避免线上出现问题，消耗的排查时间吧
12. 算法：求两个有序数组是否有交集
    1. 数据类型会明确吗，如果不明确的话，需要使用泛型。然后补充说int类型。
    2. 然后和面试官说了几个思路，以及对应的时间、空间复杂度
        1. 双指针
        2. set
            1. 判断是否存在
            2. 判断长度是否相同
        3. map
            1. 统计次数
    3. 说的思路里面挑了一个实现，然后验证

Java 17

> @author weikunkun

1. 自我介绍，说了自己的Github有个100多⭐的爬虫仓库，然后就开始问了爬虫相关的问题。。。
    1. 目前爬过的最难爬的内容
        1. 微博，讲了除了单纯的爬取之外，自己还添加了Cookies池和IP代理池，提升爬虫的抓取效果
        2. 问了对Scrapy了解多少
    2. 阐述先爬虫的核心内容（直讲了当初自己接触过的内容）
        1. 只讲了web端的爬虫
            1. 爬
                1. 直接网页渲染的，使用正则或则一些基于标签的第三方库来抓内容
                2. Ajax类型的，首先需要分析接口，需要哪些参数，如果复杂的话，还要分析一些参数是怎么生成的，通常会和网页的js有关
            2. 其他
                1. 怎么和反爬斗智斗勇
    3. 反爬措施了解哪些
        1. 说了基于IP的策略，封IP
        2. 根据请求头来封策略，譬如User-Agent、Refer、Host这种
        3. 根据Cookie来封
        4. 图片识别、文字识别、9宫格这些（当初没说）
2. 项目内容介绍
    1. 就根据：业务介绍、实现细节、难点解决这几个步骤讲的
    2. 然后仔细问了线程池、Redis分布式锁、Kafka
3. JMM阐述下，有多少说多少
    1. 定义：一种内存模型，用来屏蔽各种硬件和操作系统的不同，保证Java程序在各个平台下对内存的访问都能达到一致的结果。
    2. 目的：解决由于多线程通过共享内存进行通信时，存在的原子性、可见性（缓存一致性）以及有序性问题
    3. 接着讲了主存和工作内存，以及JMM是怎么规定对主存中变量的读写
    4. 然后详细展开来讲了原子性、可见性、有序性。以及对应的支持
4. MySQL索引说一下，有多少说多少
    1. 基于InnoDB而言，索引的类型：hash索引、B+树索引、全文索引（这个没有深入了解）
    2. 然后作中将了B+树索引
        1. 先阐述了B+树的概念
        2. 然后阐述了磁盘如何利用局部性原理做的磁盘的预读，然后将了MySQL是怎么巧妙的利用磁盘的预读来设计B+树节点的大小，从而降低IO的次数
    3. 然后将了索引的类型
        1. 聚集索引、辅助索引、联合索引、唯一索引
    4. 使用索引的注意事项
        1. 选择区分度大的字段 count(distinct(field) / count(*))
        2. 最左匹配原则
        3. = 可以乱序
        4. 索引列保持干净，不参与运算，注意类型的匹配
        5. 尽量横向拓展，而不是新增索引

Java　　18

> @author weikunkun

一面（1h）

1. 面试岗位的了解
    1. 简单的阐述了下部门，然后说了面试的Java开发岗位
2. 自我介绍
3. 项目介绍
    1. 介绍项目的背景以及解决了什么问题
    2. 这个业务需求的痛点在哪儿
    3. 重难点
        1. 业务难点
        2. 技术难点
4. 介绍下策略模式、其他的设计模式了解那些
    1. 阐述了策略模式
    2. 其他的直白了说了没实际应用过，只写过demo，就没有细问
5. 分布式锁的介绍
    1. 为什么出现分布式锁
    2. 实现分布式锁需要注意的事项
6. Redis的底层 String的实现，让自己设计，用C如何实现。(当初没仔细看SDS的设计，就按照ArrayList来设计的)
    1. 如何设计数据结构
    2. 扩容策略
    3. 缩容策略
        1. 什么时候执行
    4. 频繁阈缩容如何解决
        1. 惰性删除 ⚠️
    5. 过期策略
        1. 有哪些过期策略
        2. 如何设计过期策略的执行，value设置什么值?
            1. map? ？ 存时间戳? ⚠️
7. 买卖股票
    1. 时间复杂度、空间复杂度
    2. 给了两种思路
8. 36匹马求前3名
9. topK
    1. 根据数据量和k的值来去定方法，整体方法有
        1. 基础排序
        2. 堆排序
        3. 快速选择
        4. 基数排序
        5. 分治

10. 场景题：
    1. 业务场景：
        1. 关于钱的问题，在数据库中的存储，以及实际编码阶段上对钱的操作
            1. 因为精度问题，一般钱的存储不直接使用float、double这种以二进制形式表示的数据类型，而是使用字符串表示的数据类型。譬如使用Decimal、Numeric。`salary DECIMAL(9,2)` (能表示的最大面额，以及保留几位小数)
            2. Java编码，用BigDecimal，根据业务需求选择保留1、2位，是否需要四舍五入，或者其他的舍弃形式
    2. MySQL什么情况下不走索引，给出具体的示例场景
        1. select * from student where score = 100; --- 回表
        2. select xxxx, xxx from student where score + 1 = 100; --- 索引列不干净
11. 反问：
    1. 部门职能划分介绍
    2. 技术栈

weikunkun

二面（1h）

1. 自我介绍
2. 实习为什么不考虑转正?
3. 岗位申请是成都，计划留在成都?
4. 深入项目
    1. 项目背景、解决了什么问题
    2. 分支太多了，如何解决，之前有写过博客，然后就说的挺全的
        1. 提前return，去除不必要的else
        2. 使用三元运算
        3. 使用枚举
        4. 使用optional
        5. 表驱动法，Map<?, Function<?> action> actionMappings = new HashMap<>();
        6. 优化逻辑，让正常流程走主干
        7. 策略模式+工厂方法消除if-else
    3. 不同接口不同，这个是怎么做的
        1. 厂商的参数变了，如何满足
            1. 把所有的参数封装成一个对象，然后在对象里面获取， 最后再根据配置中心的内容，执行不同的逻辑
            2. 重载
            3. 所有的api抽象成一个配置，然后修改某个字段，再结合配置中心来实现
5. 算法：二叉树转链表，要15min中做完。。。
    1. 最开始没听清楚，直接前序遍历，然后再构造ListNode，尾插法做了
        1. 递归、非递归都写了一遍
    2. 后面说不要新建ListNode，然后直接在TreeNode上做
        1. 后序遍历解决了
6. 有没有什么喜欢的框架、组件这种，分享下
    1. spring
        1. 就把最近看Spring揭秘的内容都讲了一遍
7. Object有哪些方法
    1. hashcode、equals、wait、notify、notifyAll
    2. wait如何实现的
        1. ？？？ 不是native方法吗
        2. 然后开始尝试分享了两种思路（还好心态没崩，面试官也在鼓励我，就开始天马行空了，最后感觉就是monitor的实现）
            1. 对象头做标志位
            2. monitor的实现原理
7. 反问
    1. 技术分享
    2. 为了能胜任这个岗位，需要对哪些内容做深入学习
8. 评价
    1. 项目上还可以深挖，能做的更好
    2. 学习建议
        1. 带着问题去学习，主要是思想
        2. 长期来看需要对某个方向有一个深入的了解，作为自己的竞争点

三面（40min）

1. 学校经历讲一些，有成就感的
    1. 本科实验室的内容
    2. 研一数学建模比赛
    3. 实习经历
2. 问了数学建模比赛的内容
    1. 问了建模比赛的内容，使用了什么模型这些
    2. 问了线性回归的大概内容
3. 问了算法相关的课程
    1. dfs、bfs、贪心、动态规划这些
    2. 详细阐述下动态规划
4. 设计模式了解吗
    1. 详细阐述了策略模式
    2. 单例模式、代理模式、工厂模式带过
5. IoC、AOP
    1. 结合Spring来讲了一大坨
6. 问了自己的GitHub的相关的内容，然后讲了获得⭐最多的项目
7. 分享出来之后，大家都拿来干什么
    1. 学习
8. 对我们的部门有了解吗
    1. 阐述了之前在面试官那里了解的信息
9. 如果愿意接受百度的机会的话，有意愿来百度实习吗
    1. 来，肯定来，必须来
10. 什么时候毕业?
11. 之前实习还在做吗
12. 反问
    1. 问了技术分享相关的内容
    2. 询问对个人的评价
13. 问了我对百度的认识
    1. 讲了技术驱动、2019年的春晚红包历程
14. 确认了应聘岗位

19-Java-

> 211 Java
>
> 2-17 3 8

1.
2.
3.
4.
5. TCP/UDP
6. JVM
7. java
8. ArrayList,LinedList,HashMap
9. ArrayList LinkedList
10. HashMap + 8 —>
11. MySQL
12. B+
13. B+
14. BeanFactory
15.
16.

1.
2.
3.
4. (KPI)
5.
6. Redis
7. (Redis ,
)
8. HTTP (HTTP1.0/1.1/2.0/3.0)
9. HTTP1.0/1.1/2.0/3.0( )
10. UDP
11. HTTP HTTP
12. ( )
13. ( )
14. SQL( )
    60 ( )
15. (leetcode)
16.

1.
2.
3. ( )
4.
5. md5
6.
7.
8. ( )
9.
10. (Redis zset)
11. zset(zset score)
12. Redis (RDB,AOF)
13.
14.
15. IP HashMap (
)
16. ( )
17.

20-Java-

1. JVM
2. nginx
3. HashMap
4.
5. JVM
6.
7. IO
8.
9. static final
10. static
11. string
12. synchonized locked
13. synchonized
14.
15.
16.
17. InnoDB Mysam
18. Innodb
19. Innodb
20. Redis
21. redis
22. Kafka consumer
23.

21-Java-

1.
2.
3.
4. Redis
5. Redis
6. Redis
7. Redis
8.
9. Java
10. MySQL B hash
11. MySQL
12. SpringBoot Spring Cloud
13. Spring
14. MongoDB
15. RPC
16. NIO
17.
18.
19. Java JUC
20. CountDownLatch CyclicBarrier
21.
22.
23. JDK
24.
25.
26.
27.
28.

1.
2.
3.
4. MyBatis
5. Redis MyBatis
6. Redis
7. Redis
8.
9. MySQL
10. CMS
11. synchronized ReentrantLock
12.
13.

14.
15.
16.

1.
2.
3.
4.
5.
6.

HR

1.
2.
3.
4.
5. offer
6.

22-Java-

JVM

1

class

2

jvm QAQ

3 Java 8

1

2 Java

cas syn Lock

3

actom

4

5

Thread callable runnable callable Future

6

MySQL

1 B+

MySQL B+

+

2

ID IO

ID

3 B+ B

4 where a=,b=,c= a b c

/

char

5

++

6 union /union all

sql

1

2

3

1

2

3

4 2MSL

++

1 10

1

23-Java-

> @author

1

Java

JAVA HashMap put

HashMap JDK1.7

Entry table Entry next
Entry HashMap
Entry

HashMap null null hashCode()
HashMap 0 null

```
int hash = hash(key);
int i = indexFor(hash, table.length);
```

```
final int hash(Object k) {
    int h = hashSeed;
    if (0 != h && k instanceof String) {
        return sun.misc.Hashing.stringHash32((String) k);
    }
    h ^= k.hashCode();
    // This function ensures that hashCodes that differ only by
    // constant multiples at each bit position have a bounded
    // number of collisions (approximately 8 at default load factor).
    h ^= (h >>> 20) ^ (h >>> 12);
    return h ^ (h >>> 7) ^ (h >>> 4);
}
public final int hashCode() {
    return Objects.hashCode(key) ^ Objects.hashCode(value);
}
```

`x = 1 << 4` x 2 4

```
x : 00010000
x-1 : 00001111
```

y x-1 y 4

```
y   : 10110010
x-1  : 00001111
y&(x-1) : 00000010
```

y x

```
y : 10110010
x : 00010000
y % x : 00000010
```

key hash hash%capacity capacity 2 n

```
static int indexFor(int h, int length) {
    return h & (length-1);
}
```

-

HashMap table M N
N/M O(N/M)

N/M M table
HashMap N M

capacitytable 16

capacity 2 n size thresholdsize size
threshold

loadFactor table threshold = (int)(capacity* loadFactor)
capacity

```
void addEntry(int hash, K key, V value, int bucketIndex) {
    Entry<K,V> e = table[bucketIndex];
    table[bucketIndex] = new Entry<>(hash, key, value, e);
    if (size++ >= threshold)
    resize(2 * table.length);
}
```

resize() oldTable newTable

-

HashMap
hash%capacity

HashMap capacity 2 n

capacity 16 new capacity 32

capacity : 00010000
new capacity : 00100000

Key hash 5

1. 0 hash%00010000 = hash%00100000
2. 1 hash%00010000 = hash%00100000 + 16 + 16
3.

HashMap 2 n 2 n

10010000 11111111

`mask |= mask >> 1 11011000`

`mask |= mask >> 2 11111110`

`mask |= mask >> 4 11111111`

mask+1 2 n

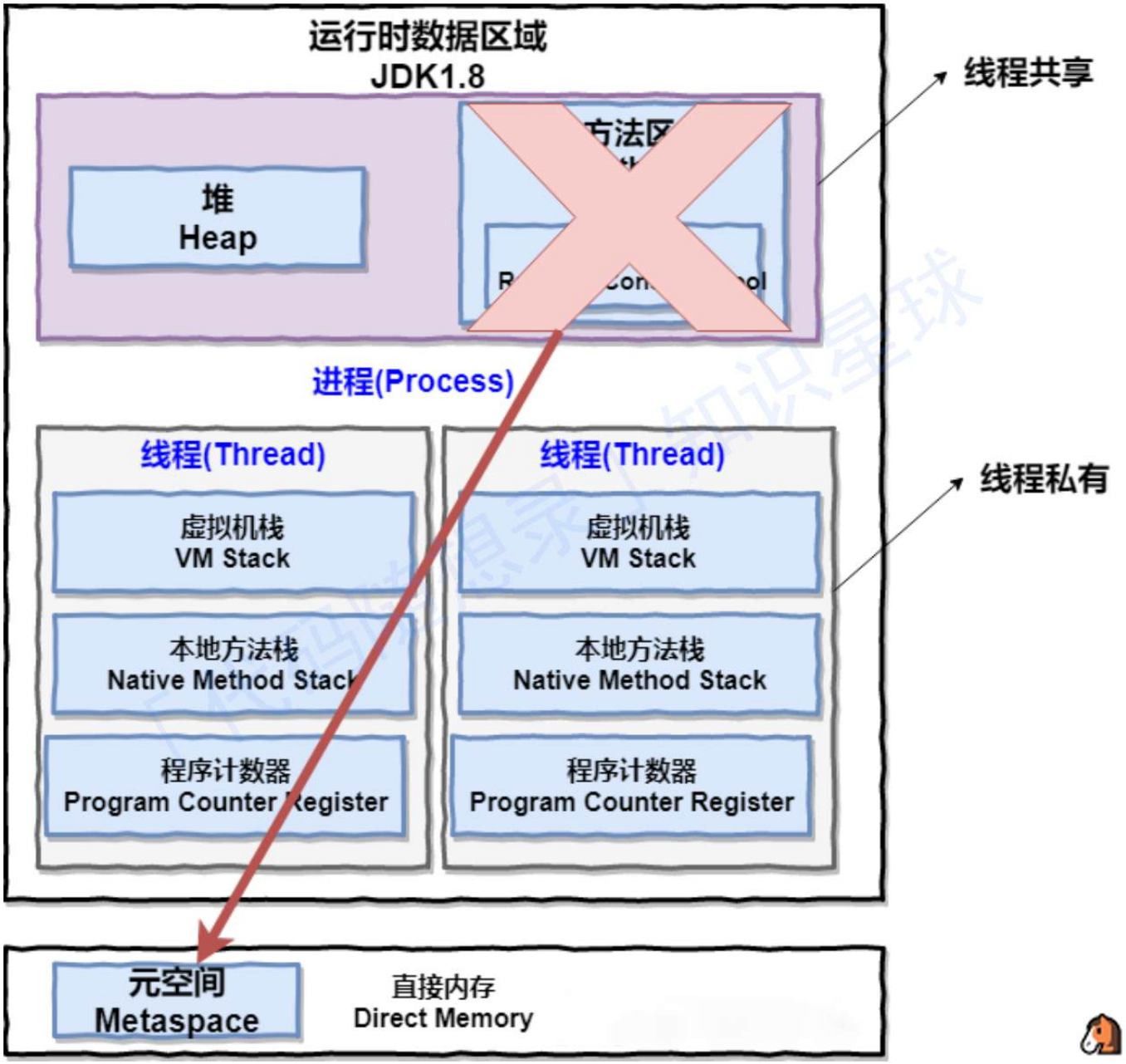
num 10010000

mask+1 100000000

HashMap

```
tatic final int tableSizeFor(int cap) {
    int n = cap - 1;
    n |= n >>> 1;
    n |= n >>> 2;
    n |= n >>> 4;
    n |= n >>> 8;
    n |= n >>> 16;
    return (n < 0) ? 1 : (n >= MAXIMUM_CAPACITY) ? MAXIMUM_CAPACITY : n + 1;
}
```

JDK 1.8 8


运行时数据区域
JDK1.8
线程共享
堆
Heap
进程(Process)
线程(Thread)
线程(Thread)
线程私有
虚拟机栈
VM Stack
虚拟机栈
VM Stack
本地方法栈
Native Method Stack
本地方法栈
Native Method Stack
程序计数器
Program Counter Register
程序计数器
Program Counter Register
元空间
Metaspace
直接内存
Direct Memory

(JDK1.8)

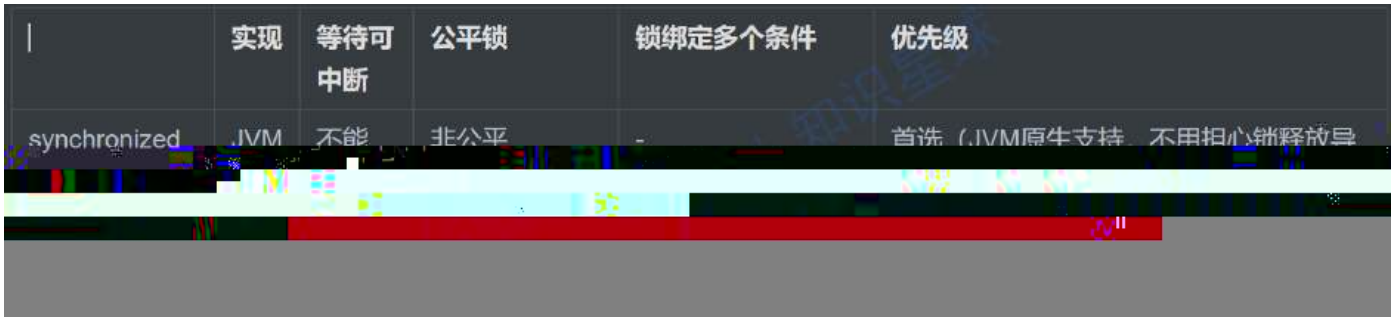
,

LINUX

Java

Java JVM synchronized
JDK ReentrantLock

ReentrantLock synchronized



CAS

CAS
CAS

1

Compare-and-Swap CAS CAS 3 V A B
V A V B

2 CAS

```
package sjq.mylock;

import java.util.concurrent.atomic.AtomicReference;

public class SpinLock {
    private AtomicReference<Thread> owner = new AtomicReference<>();

    public void lock(){
        Thread currentThread = Thread.currentThread();
        while(!owner.compareAndSet(null, currentThread)){  // owner == null
compareAndSet    true       false
            //     owner
        }
    }

    public void unLock(){
```

```
            owner.set(null);
            //
            /*
            Thread cur = Thread.currentThread();
            owner.compareAndSet(cur, null);
             */
        }

    }
```

MySql InnoDB

InnoDB MySQL

REPEATABLE READ

MVCC + Next-Key Locking

Redis

Redis NoSQL

Redis

MySql

MySql B+

1 B+ Tree

MySQL

B+ Tree

InnoDB B+Tree data

2

O(1)

InnoDB “ ” B+Tree
B+Tree

3

MyISAM
MATCH AGAINST WHERE

InnoDB MySQL 5.6.4

4

MyISAM R-Tree

GIS

B+

B+

B B+

B B+ MySql B+ B

MySQL B+

1 B+

B+ B

IO

2 B+

3 B+

B+ B
B+
B+

4 B+

B IO B+
B+ B

HTTP

URL
Header
Body

```
GET http://www.example.com/ HTTP/1.1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,a
pplication/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cache-Control: max-age=0
Host: www.example.com
If-Modified-Since: Thu, 17 Oct 2019 07:18:26 GMT
If-None-Match: "3147526947+gzip"
Proxy-Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 xxx

param1=1&param2=2
```

HTTP

状态码	类别	含义
1XX	Informational（信息性状态码）	接收的请求正在处理
2XX	Success（成功状态码）	请求正常处理完毕

POST， Respond

200 OK

```
HTTP/1.1 200 OK
Age: 529651
```

```
Cache-Control: max-age=604800
Connection: keep-alive
Content-Encoding: gzip
Content-Length: 648
Content-Type: text/html; charset=UTF-8
Date: Mon, 02 Nov 2020 17:53:39 GMT
Etag: "3147526947+ident+gzip"
Expires: Mon, 09 Nov 2020 17:53:39 GMT
Keep-Alive: timeout=4
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Proxy-Connection: keep-alive
Server: ECS (sjc/16DF)
Vary: Accept-Encoding
X-Cache: HIT

<!doctype html>
<html>
<head>
    <title>Example Domain</title>
 //    ...
</body>
</html>
```

TCP UDP

24-java-

> @

50

1.
2.
3.
4.
5. String Stringbuilder Stringbuffer String s = "abc" String s = new String("abd")
6.
7.
8.
9.
10. Sleep Wait
11. Java
12. JVM OOM
13. Java
14. ?tomcat

15. ?
16. full GC
17. Spring
18. Spring @Autowired @Resource
19. Spring bean
20. AOP AOP AOP
21. Redis

40

1.
2.
3. Tomcat
4. ,

Java

1. volatile
2. Hashmap
3. sycronized CAS
4.
5. Mysql
6. MVCC Mvcc
7.
8. 123456
9. JVM

spring

25

HR HR HR

Offer

1.
2.
3.
4.
5.
6. Offer

25-Java-

> @author

4 35

spring

lock

hashmap
concurrentHashMap

4 3

50 0-9 6-15

a. 3 111234 2333456
b. 3 123456 123457 123458
c.
d.

26-Java-

> @author

8 10

Java JVM

36
+ + + +

Java

synchronized

JVM

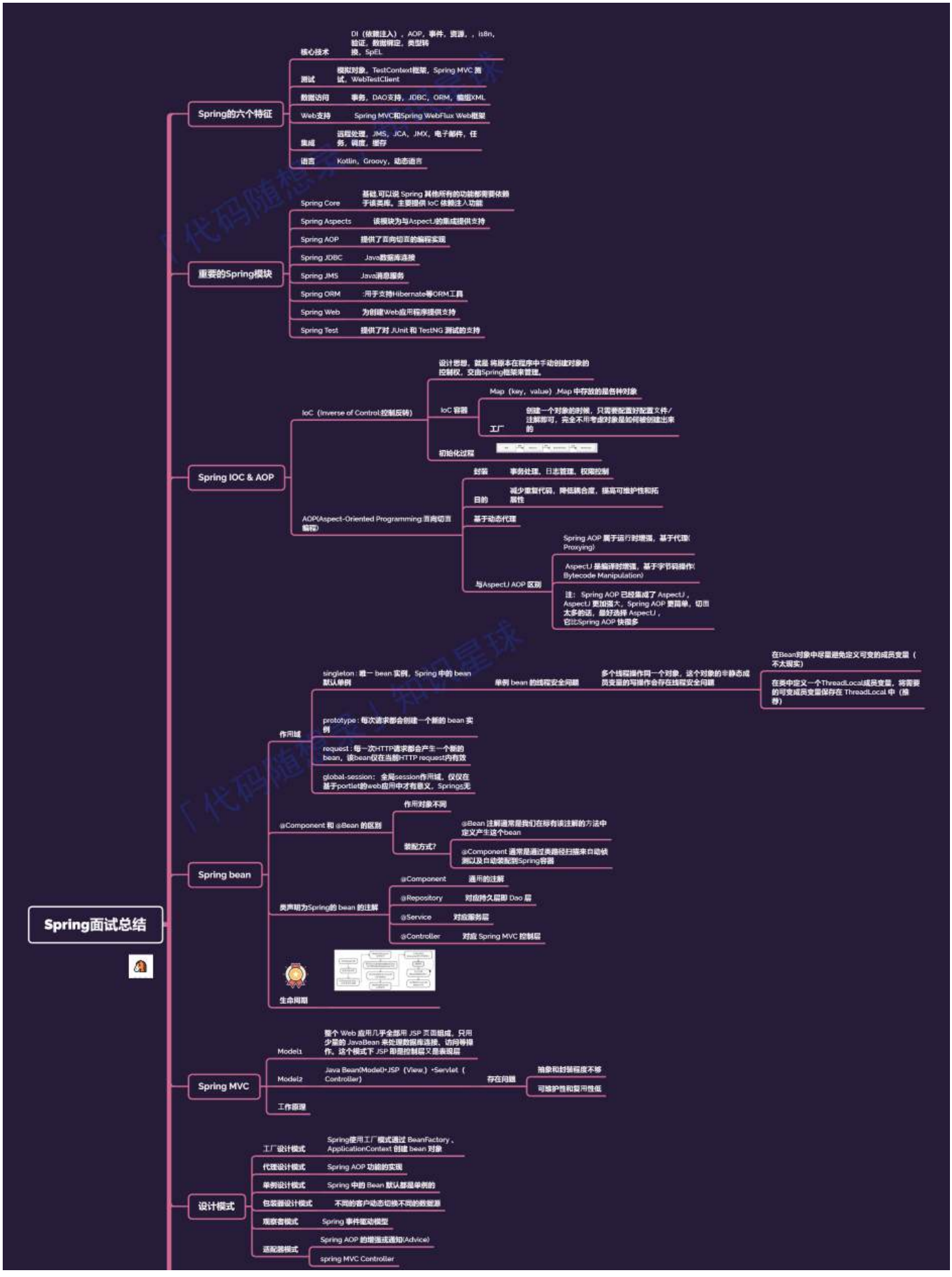
MySQL
B+ list map

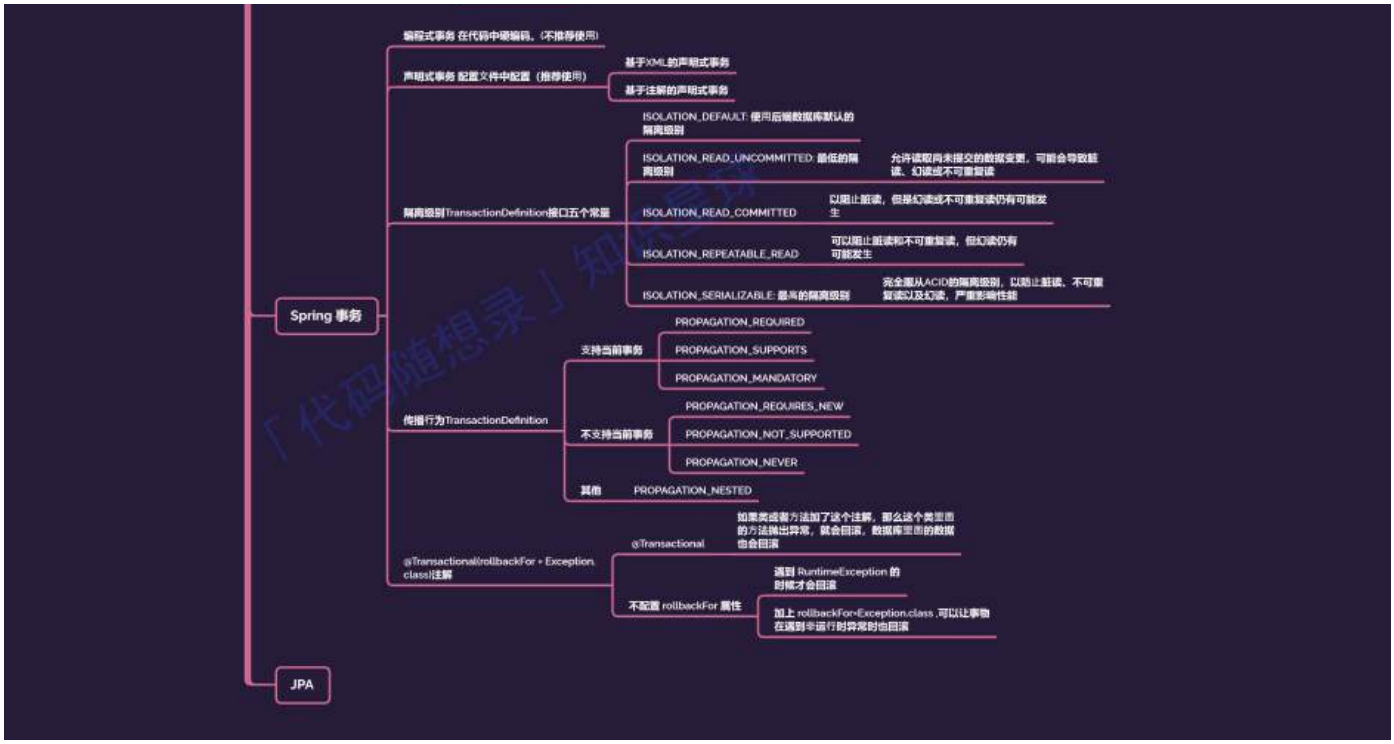
A B G 4G A + B +
C

2 Java

27-Spring

> @author





28

> @author

1 B+ B 11

B+ B

2 B+ 12

1

B B+ B+ range-query
B B+

2 B+tree

B+tree ( ) B

IO

3 B+tree

3 13

1

2

3

1.
2.
3.
4.

4 MySQL 14

undo log

1. ROLLBACK
2.

5 InnoDB MyISAM 15

: InnoDB Commit Rollback

InnoDB MyISAM

: MyISAM InnoDB

: InnoDB

: InnoDB

: MyISAM InnoDB

: MyISAM

29

> @author

1.
2.
3.
4.
5.
6.
7.

1. HTTP
2. HTTP

1

1. , , ,
2. ,
3. , ,
4. , , ,
5. . , ,
6. , ,

2

1.
2.
3.
4.
5.

1.
2.

3.

3 [ ]

1.

2.
3. IO IO
4.

4

1.
2.
3. , .Innodb
4.

1.
2.
3.

5

1.
2.
3. != <>
4. like ('%abc')
5.
6. or

6

1

MyISAM CREATE TABLE ALTER TABLE CREATE INDEX
CHAR VARCHAR TEXT

HASH 100% HASH
, “=” “in”

B

BTREE
root node leaf MySQL

R

RTREE MySQL geometry MyISAM BDb InnoDb
NDb Archive BTREE RTREE

2

,

3

, , ,
.

( )
InnoDB + “ ”
,

7

,

1.
2. text

2 HTTP

状态码	类别	含义	具体实例
1XX	Informational（信息性状态码)	接收的请求正在处理	100 Continue： 表明到目前为止都很正常，客户端可以继续发送请求或者忽略这个响应。
2XX	Success（成功状态码） 请	请求正常处理完毕	200 OK
3XX	Redirection（重定向状态码） 需	需要进行附加操作以完成请求	301 Moved Permanently： 永久性重定向 302 Found： 临时性重定向
4XX	Client Error（客户端错误状态码)	服务器无法处理请求	400 Bad Request： 请求报文中存在语法错误。 403 Forbidden： 请求被拒绝。 404 Not Found
5XX	Server Error（服务器错误状态码)	服务器处理请求出	500 Internal Server Error： 服务器正在执行请求时发生错误。 503 Service Unavailable： 服务器暂时处于超负载或正在进行停机维护，现在无法处理请求。

3 HTTP

HTTP/1.1 Cache-Control

1

no-store

2

no-cache

Cache-Control: no-store [ ]/no-cache [ ]

Cache-Control:private [ ]/public[ ]

30

> @author

0

1 MySQL

2 SQL

3

4

5

1

1 ——

2

3

2

Linux Windows

API

API C

1.
2.

1

1

2

3

f' = f/a /

4

ɓ Ǐ

1

0 0

6 LFU

+1

FCFC SJF

SRTN

1 first-come first-serverd FCFS

2 shortest job first SJF

3 shortest remaining time next SRTN

4

FCFS CPU

CPU

5

6

100 100

1,2,4,8,..

7

MySQL

1. join
2. MySQL
3.
4. Memcached Apc
5.
6. SQL SELECT * FROM TABEL SELECT field_1, field_2, field_3 FROM TABLE

SQL

1. DDL
2. DML
3. DQL
4. DCL :

类别	DDL	DML	DQL	DCL
创建对象	表、视图、索引同义词、聚簇			
包含语句	Create Drop Alter	1) 插入：INSERT 2) 更新：UPDATE 3) 删除：DELETE	Select From Where	GRANT 授权 ROLLBACK 回滚
关于是否能回滚	隐性提交的！不能rollback；操作立即生效	必须**提交**才能生效！执行的操作会放到**回滚段**，可回滚		

1

1.
2.
3.
4.
5.

1.
2. ;
3.

2

IO,

1.
2. text,blob
3.

1.
2.
3.

1.
2.
3.
4.
5.

31

> @author

1

1NF
2NF
3NF

2

1NF ,
2NF
3NF

1

2

Java synchronized
MySQL

CAS

3

1

CAS

CAS CAS synchronized
query 1 update 2

2

( )

( )

CPU

3

SELECT ... FOR UPDATE

SELECT ... FOR UPDATE select select for update

Java AtomicFieldUpdater CAS

1 CAS Compare And Swap

1 CAS 3

1. (V)
2. (A)
3. (B)

2

V A B
CAS

3

CAS Compare Swap

CAS CPU

CAS ----

ABA

—— 1 2

1. 1 A
2. 2 B
3. 2 A
4. 1 CAS

4 A CAS 2 ABA

ABA ( )

+1 CAS

CAS

Java AtomicStampedReference ABA

CAS CPU

2

CAS CAS

(1) Java volatile

(2) ( ) CAS

CAS

version 1

1 (mutex)--

sleep-waiting

CPU

2 (cond)--

1.
2.

3 (spin)--

busy-waiting

A pthread_spin_lock

B A core 0
B

1 CPU

CPU CPU

2

3 SMP

CPU

4

1.
2.
3.

1

5.2

5.3

5.3.1

5.3.2

5.3.3

5.3.4

5.4

5.5

5.5.1

Unix Linux Windows

5.5.2

1.

2.

1. Pi A
   - C i A
     1
   -

5.5.4

1.

2.

3.

4.

5.5.5

1 cwnd

2

3 swnd = cwnd

4 ssthresh

cwnd < ssthresh
cwnd > ssthresh
cwnd = ssthresh

1

TCP

cwnd 1 MSS
ACK cwnd
RTT(Round-Trip Time) cwnd 2
ssthresh slow start threshold cwnd >= ssthresh “ ”

2

cwnd ssthresh

TCP

1

ACK

2 cwnd 1

3

20%

3

cwnd
ssthresh cwnd ssthresh = cwnd
Fast Recovery

4

1 cwnd = cwnd + 3 MSS 3 MSS 3 ACK

2 DACKs

- DACKs cwnd
- ACK cwnd ssthresh

cwnd cwnd cwnd 1 cwnd ssthresh

32

> @autuor

1. union join
2.
3.
4.

union

SELECT

1. SELECT col
2.
3. UNION SELECT
4. UNION UNION Select Select

join

JOIN ( )

A B

B LISTEN

A B SYN=1 ACK=0 seq = x

B A SYN=1 ACK=1 ack = x+1
seq = y

A B B ack = y+1 seq = x+1

SYN = 1
ACK
socket connect()

1

SYN SYN

SYN TCP

SYN

SYN socket

SYN

2

“ ” 3 “ ”
“ ” Tcp

3

TCP 3
4 SYN ACK SYN
SYN-ACK

TCP

SYN
SYN

ESTABLISHED

33

> @author

1

2

3

1

2

JOIN

4

ID Name

1

2

1

IO,

2

B+

3

count

count

count

分类	Hash路由	范围路由	路由表
概念	选择表中的某一列，然后进行 Hash 运算，将 Hash 运算得到的结果再对子表数进行取模，这样就能均匀的将数据分到不同的子表上。	表示一定的范围；可以基于时间或者地址	专门搞个表来记录路由信息
优点	数据分布均匀	容易扩展	灵活‘迁移数据’，直接迁移后修改路由表即可
缺点	增加子表的时候比较麻烦，对HashMap的扩容，需要迁移数据	数据可能分布不均匀【例如：某个月搞了活动，日志特别大；BJ用户特别多】	多一次查询；每次都要查询访问路由表，有对应的缓存设计。
关于拆分键【Sharding-Key】设计问题	采用冗余数据的方式，无论如何对应拆分的列都不可能满足所有需求		

3 ID 1 2 3 ID 1 4 7 2 5 8
3 6 9

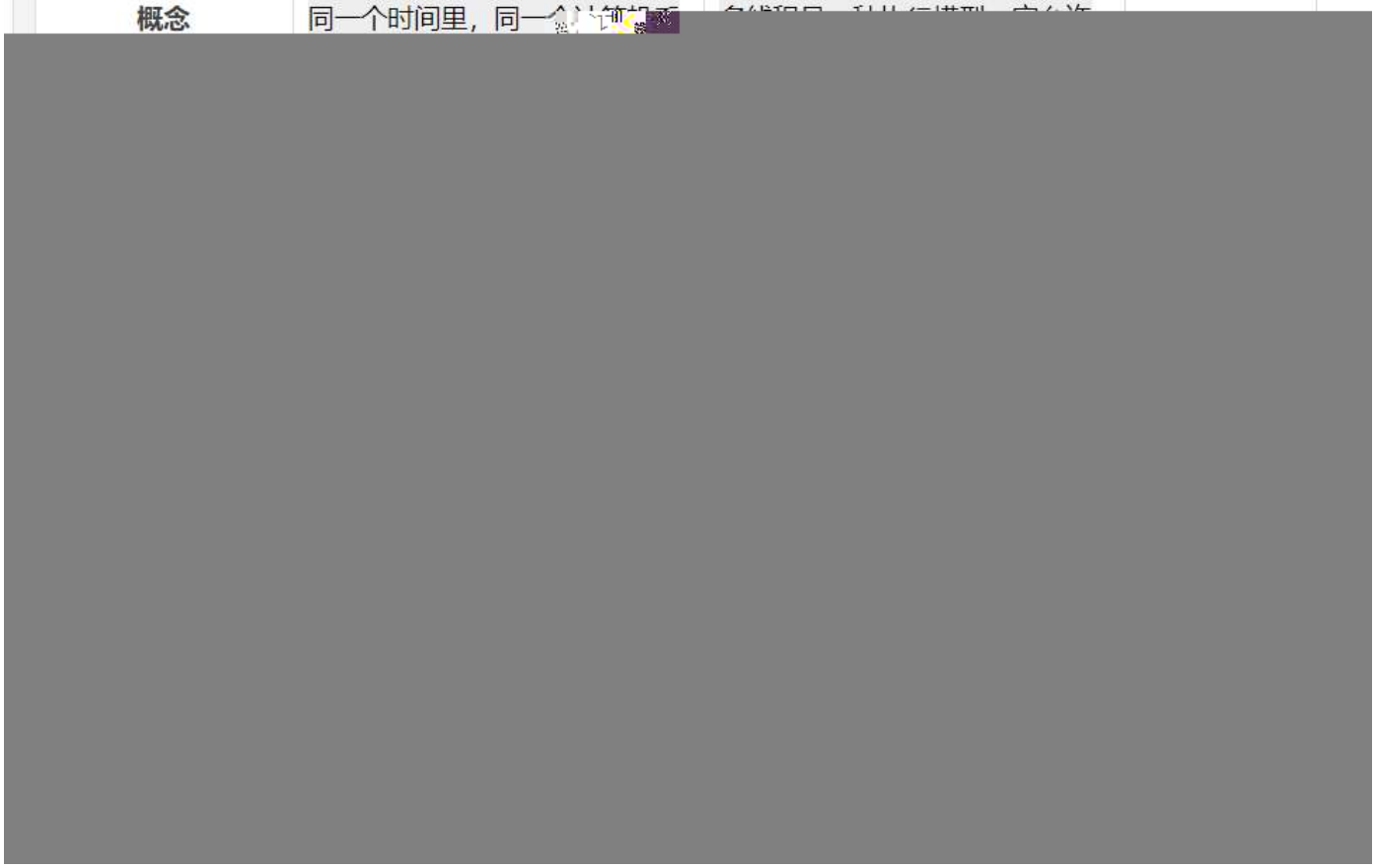
UUID

ID Twitter sonwflake redis

34

> @author

维度	多进程	多线程	总结
概念	同一个时间里，同一个计算机系		



1.

2. down up P V down up
down 0 -1 0 0 up +1
down
3. 0 1 0 1

1.
2.
3.

35-

> @author Mona

url

1 dns

2 http

(1) post get
(2) get post

3 TCP UDP

4

65535

5 10 tcp

2

3

...

Cpp

🤫

C++

1-C++- IEG

> Q E

1. map unorder_map

2. K8s Go
3. TCP
4. write
5. MySQL B+
6. rpc
7.
8. TCP HTTP
9. C++

2-C++-

> xyFei

-
-
-
-
- new new[] delete
- move
- map unordermap
-
- n*n k

3-C++-

> xyFei
>
> 1h

1.
2. 1
3. 2 webServer fd Reactor Epoll
4. Linux
5. 3
6.
7.
8. C++
9.
10. STL list
11.

4-C++-

> Mao

put put

camerax

api

!

5-C++-

1. balabala
2. C++

- c++ [ ]
-
-
-

5

- B C A D B C A public B C

D B C

- A public B C A D B C D public

3.

- malloc new
- new malloc new new
- new new

- ```
  struct {
      char A;
      char B;
      int C;
  }

  struct {
      char A;
      int C;
      char B;
  }
  ```

  sizeof

4. STL
   - STL
   - shared_ptr unquie_ptr
   - shared_ptr

     C++ JAVA JS JAVA
   - STL vector
   - 2 2 3
   - vector 0 0 1
5.
   -
   - 3
  ```

    -
    -
6.
    - HTTPS
7.
    - webserver
    - webserver Buffur
    - buffer fd
8.
    -
9.
    - C++ shared_ptr C++ JAVA JS
    -
10.
    - C++ C++ C++
    -
    -

6-C++-

- select poll epoll
-
- gbd
- STL
- QT
- LT ET
-
- SOCKET OSI
- ping
- 2MSL

7-c++ -

1. webserver
2.
3.
4. QPS
5. 1

1. TCP UDP
2. TCP
3. TCP
4. TCP
5. TCP
6. / /
7.
8. B B+

1.

C++

1. C++
2.

8-c++-

1.
2.
3.
4.
5. 16 32
6. epoll
7.
8. epoll
9. LRU
10.
11. CPU CPU
12.
13. HTTP
14. HTTP
15. HTTP GET POST

16. Linux CPU
17. C++
18.
19.

1. CPU
2.
3.
4. webserver

9-C++-Momenta

1.
2. webserver
3.
4.
5.
6.
7.
8.

9. 887.
10.
11.
12. —— + +
13. —— C++ Python MySQL

1.
2. vector clear insert

10-C++- |

1. static
2.
3.
4.
5. Linux

6. struct
7. C++
8. 32 unsigned int 0 1 1

VR

11-C++-OPPO

VR

1.
2.
3. app runtime
4.
5. openxr api
6. tcp udp
7. epoll

VR/AR

1. SXR SDK OpenXR SDK
2. OpenXR
3. OpenXR
4. ATW
5. app OpenXR runtime

1.
2.
3. app runtime
4.
5. openxr api
6. tcp udp
7. epoll

VR/AR

1. SXR SDK OpenXR SDK
2. OpenXR
3. OpenXR
4. ATW
5. app OpenXR runtime

12-C++-

.

1. , (, , , ,)
2.
3. , , main add (CSAPP)
4.
5. , ,
6. .bss ,
7.
8. ,
9. C++ vector
10. C++ map ( )
11. ,

, ,

- vector emplace_back() , reserver() ? ?
- , ?

12. linux ( ) task_struct , , epool , , , ,
13. ,

HR , , github , , CUDA , , , , .

1.
2. , .

,

13-C++- /SelectDB

> Outlook

1. MVCC
2.
3. ACID
4. B+
5. redo
6. redis (, ,)
7. redis
8. C++

, , , ,

1. ( )
2.

SelectDB

1.
2.
3. ,
4.
5.
6.
7. TCP
8. C++ ,
9.
10.
11. socket

14-C++-

> @
>
> 2022.8.19 19.00-19.40

1.
2.
3.
4. new malloc malloc
5.
6.
7.
8.
9.
10. http
11. tcp udp
12. tcp
13.
14. dfs
15.
16.
17.
18.
19. poll epoll select
20.

2022.8.22 15.40-16.40

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22. c++
23.
24.
25.

2022.8.29 15.10-15.30

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.

15-C++-

> @

2022.6.23 14:30-15:00

1.
2.
3.
4.

5. bug log debug
6. Python
7. Python2 Python3
8. Python
9. dict list
10. list tuple
11. stl
12. map
13.
14.
15. tcp udp
16.
17. Socket
18.
19.
20. mqtt
21.
22.
23.
24.

2022.7.1 10:30-11:00

1.
2. c++
3. c++
4.
5. Static
6. Const
7.
8. new/malloc delete/free

9. stl
10. stl sort
11.
12. 2 n n&(n-1)
13. p -
14.
15.
16. cad
17. cad c++ qt opengl mfc
18. 1 ...
19.
20. ......

2

21.
22. cad
23.
24.

HR 2022.7.6 17:25-17:30

1.
2.
3.
4.
5.
6.
7.

16-C++-

> @ PCJ600

1. define const
2.
3.
4.
5. vector push_back
6.
7.
8.
9.
10.
11.
12.

13.
14. select a, b, c, d from t where a = 1 and b = 2 order by c desc
15.
16. LRU

17-C++-

> @ PCJ600

1. C++
2.
3. vector vector
4.
5.
6.
7.
8. TCP UDP
9. TCP
10. kv
11. topK
12.
13.
14.

1. lambda std::function
2. lambda lambda
3. LC.735

18-C++-

> @ PCJ600

1.
2. 20
3.
4. showmebug [(0, 0), (1, 2),.....]
5.
6. time_wait

1.
2. 10
3.

4.
5. C++
6.
7.
8. C
9.
10. MySQL Redis
11.
12.

19-C++-

> @ PCJ600

1.
2. Java Java C++
3. MySQL innoDB B+
4.
5. ACID
6. MVCC
7.
8.
9. Redis Redis Redis
10.
11. url
12.
13. easy
14. Java

1.
2.
3. mongo
4.
5. MySQL
6.
7.
8.
9. SSL
10.
11.

20-C++-

> @ PCJ600

1.
2.
3. Go channel
4. HTTP2.0 HTTP1.1 1.1 1.0
5.
6. docker k8s
7.
8. topsort
9. DFS BFS
10.
11. - LRU LFU LRU LFU
    - LRU
12. LRU
13.
14.
15.

- 5 Java Go Python

- Base

C++ Github Python

-
-
-
-

21-C++-TP-Link

> @ PCJ600

1.
2.
3.
4.
5. C++
6.
7. IO

1.
2. 10
3. C++
4. static
5.
6.
7.
8. TCP 2MSL
9. IDE

leader

1.
2.
3.
4.

10 /

5.

22-C++-

> @ PCJ600

1.
2. weak_ptr
3.
4.
5. _____
6.

1. TCP UDP
2.
3.
4. C++
5. C++

23-C++-

> @ PCJ600

1.
2. C++
3. HTTPS
4. HTTPS
5.
6. epoll LT ET
7. SQL group by order by limit
8. B+ B
9. MySQL 4kb
10.
11. K
12.

1.
2. 2 3
3. close_wait
4. MySQL B+ B+
5. MySQL Gap-Lock
6. MySQL
7.
8. LC.153 LC.154
9. LC.33

24-C++-

> @ PCJ600

1.
2.
3. 3 2
4. close_wait
5. HTTPS HTTP
6. SSL
7. HTTPS

8.
9.
10.
11. I/O
12.
13. MySQL explain
14. MySQL SQL
15.
16.
17.
18. Redis
19.
20. C++
21.
22.
23.
24.
25. Go GMP
26. MongoDB Redis
27. MongoDB blabla

28. int long long long
29.
30.
31. 5

1.
2.
3.
4.
5.
6.
7. .cpp ->
8.
9. vector size/capacity
10.
11.
12. showmebug

25-C++-

> @ PCJ600

1.
2. STL sort

3. AVL
4. B+ AVL
5. B+ I/O
6.
    - K partition /
    - (9^999999)%5=?
    - a^b %n a b n int long

1.
    -
    -
    -
    -
2.
3.
    - n
    -

26-C++-

> @

C++ Web

1

2

3

4

5

6

7 C++14 17

8 redis

9 redis

10 HTTP

11 HTTPs

12 int* const

13 C++11

14 swap

1

2 web

3 api

4

5 detach join

6

7 timewait

8 closewait

9 closewait

10 close

12

13

14

15

16 redis

17

18 string

19 gdb

20 gdb core

21 api

22

23 epoll

24 ET LT

25 select epoll

26 static

27

28 C++11

29 weakPtr sharedPtr

30 weakPtr

31 lambda

32

33 C++11

34 C++11

35

1

2

3

4 C++

5

6

7 redis

8 redis

9

10 sql id,name

11 topk

12 linux

13 IP url topk IP url

14 AWS

15

16

17

18

19

20

21

22

23

27-C++-ZEKU

> @

ZEKU

> 30 min

1

2 912.

3

4 nlogn

5

6

7

8

9

10 C++

11

12 C++11

13 lambda

14

15

16

17

18

ZEKU

> 40 min

1

2 web

3

4

5

6

7

8

9

10 epoll select poll

11

12

13 SIGPIPE ET EAGAIN

14

15 redis

16

17

18

19

ZEKU HR

> 20 min

1

2

3

4

5

6

7

8

9

10 C++

11

12

13 ZEKU

14

15

28-C++-VIVO

> @

VIVO

1

2

3

4

5 MySql B+

6

7 MySql

8 linux

9

10 VPN

11

VIVO HR

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

29-C++-TPLINK

> @

17min

1.
2. Tcp Udp
3. tcp udp
4. C++ struct
5.
6.
7.
8.
9. struct

30min

1.
2.
3. yolov5
4.
5.
6.
7.
8.

1.
2.
3.
4.
5.
6.
7.
8.

30-C++-

1.
2.
3. Web IO
4.
5.

6.
7. Nginx Nginx
8.
9.
10.
11.
    -
12.
    -
    - IO
    - IO IO
13.
14.
15.
16. C++
17.
18. unordered_map
19.
20. https http
21. https
22. https
    -
    -
23.
24.
25.
    - 1 1
26.
27. atoi

1.
2.
   -
3. Web
4. keep-alive
   -
   - keep-alive
5.
6.
7.
8. Web
9.
10. C++ inline
11. CPU
12. CPU
    -
13.
    -

1.
2.
3.
4. unicode utf-8
   - “Unicode “ ”,UTF-8 “ ”
5.
6.
7.
8.
9.
10.
11. EPOLL
    - EPOLL EPOLL

12.
    -
    -

1 10

13.
14.
15. ip ip ip
16. NAT
17. udp
    - udp ip

18. DNS
19.
    -
    - HTTPDNS
20. HTTP3 HTTP1 HTTP3
21.
22.
    -
    -
    -
23. C++
24. C++ new delete malloc free
25. C++ extern "C"
26. C++
    - O2
27. `push_back()` `emplace_back()`
28. DFS
29.
    -

HR

1.
2.
3. ACM
4. ACM
   -
   -
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18. mentor
19.
20.
21.
22.
23.
24.
    -
    - offer

31-C++-

> @author Nx

1.
2.
3.
4.

5.
6.
7. TCP UDP
8. HTTP
9. HTTP
10. HTTPS

1.
2. C++, Java

20

32-C++

> @author

1. TCP
2. TCP
3.
4.
5. C++ volatile/const/static
6.

TCP

CP /

1

TCP

2 /

TCP ACK ACK

3

4

5

TCP

6

A B B B

B

B A A

TCP

TCP

1

TCP TCP TCP

TCP

2 ?

1. TCP TCP
2.
3. TCP
4. MSS TCP TCP -TCP > MSS

3

1.
2.
3.

1

PC

2

CPU

3

4

PC

task_struct

1

2

1 CPU

CPU

2

64 Linux 8MB 64MB

3

4

IO IO

33-C++-

> @author

1.
2.
3.
4.
5. mutable volatile
6. vector resize reverse
7. map
8. inline define
9.
10.
11. select epoll io
12. malloc
13.

14. mysql redis
15. redis
16. mysql
17. mysql

4

18.
19. if else
20.
21.

34-C++

> @author

1.
2.
3.
4. Nagle
5.
6. MVCC

C++

1.

1.1

1.2

1.3

N " %N=0

+

1.4

char 1 short 2 int,float,double 4

#pragma pack (value) value

2.

1.
2.
3. NULL
4.
5. sizeof sizeof
6.
7. 4
8. typename refname &varname
9.
10.

1.

1.1

OSI

TCP/IP

TCP/IP TCP/IP
IP

1.2

1

DNS HTTP
SMTP

2

3

4

1 TCP

2 UDP

5

IP

6

IP

7

2. Nagle

Nagle

2.1

TCP/IP ACK
TCP Nagle

2.2

“ ” MSS “ ”
ACK

2.3

MSS

FIN

TCP_NODELAY

TCP_CORK MSS

200ms

2.4

Linux 40ms
ACK ACK
ACK

Nagle Nagle

1. ACID

(atomicity)

(consistency)

(isolation)

RU RC RR SERIALIZABLE

(durability)

2. MVCC-

1

2

1 LBCC

2 MVCC

1. MVCC SELECT
2. MVCC
ACID

35-C++

> @author

1.
2. inline deifine
3.
4. SYN

C++

1.

2. inline deifine

1.
2.

1. inline
2.

1.
2.

1.

TCP CLOSED LISTEN ,

1. SYN
2. ACCEPT

SYN

TCP SYN ACK SYN LISTEN
SYN_RCVD SYN

ACK,
ACCEPT

2. SYN

1

DoS (Denial of Service) IP , SYN
SYN+ACK ACK
TCP

TCP CPU

2

- IP SYN

3

1

SYN ACK
SYN TCP

2 TCP/IP

3 SYN cookies

SYN Cookies TCP SYN

SYN cookie
SYNACK SYN+ACK ACK

36

> @author

1. ACK
2.
3.
4.
5. / /

1 ACK

ACK TCP SYN_RECV, TCP 3
6 12 SYN+ACK ACK
ACK

RST Reset

2

A B ACK 1

A FIN=1 TCP
B TCP B A A B
B FIN=1
A TIME-WAIT 2 MSL
B A

3

FIN
ACK .

FIN
ACK FIN

TCP

日志类别	逻辑层架构	文件空间	作用	写入方式	内容	恢复数据的效率
redo log	InnoDB引擎特有	空间大小固定	保证事务的持久性的，是事务层面的	循环写入和擦除	物理日志；是数据页面的修改之后的物理记录。	高
binlog	通过MySQL的Sever层实现，所有引擎都可以使用	空间不固定，写完会切换下一个文件	还原的功能，是数据库层面的	追加写入，不会覆盖已写文件	逻辑日志；记录的就是sql语句。	低
两者日志产生的时间，可以释放的时间，在可释放的情况下清理机制，都是完全不同的。						

1 redo log --

mysql redo log

/

checkpoint write_pos

write_pos = checkpoint

redo log checkpoint checkpoint buffer

1.
2.

fsync

2 binlog --

3 IP IP

IP MAC

4 ARP

IP MAC ARP

2 A B

1 A ARP ARP

2 ARP ARP

3 B ARP IP A ARP

4 A B ARP ARP B IP

MAC

ARP ARP ARP

MAC

ARP MAC

38

> @author

ICMP

IP IP ICMP

1. IP
2. IP

1.
2.

ping

1.
2. TTL(IP Time To Live)

1. ping ICMP Echo Request
2. , ICMP Echo Reply
3.

traceroute

3

1

2

3

4

5 memcpy

39-C++- +

> @author

K

ps

41-C++-

> @author

7.27

1.
2.
3. STL vector list vector
4. vetor
5.
6. STL
7.
8.
9. git
10. leetcode.138.

7.27

1.
2.
3. I/O
4. UDP TCP
5. TCP
6. vector print
7.
8. shared_ptr +1 -1 STL
9.
10. +
11. GDB
12.
13. + / leetcode 206.

7.30

1.
2. new
3. ball
4.
5.

7.28

1.
2.
3.

4.
5. shared_ptr
6.
7.
8.
9.
10. : ClosestStorage

    1. insert(item) item
    2. delete(item) item
    3. getClosest() item
       int64

shopee 7.29

1. C++
2.
3.
4.
5. TCP
6. HTTPS HTTP
7.
8.
9.
10.

7.31

1.
2.
3. leetcode33.

ps

42

> @author

32 4

CPU

CPU

43

@author


自我介绍
不流畅
DH算法的主要实现

44-

> @author Nx

20

1.
2.
3.
4.

5. Linux
6. Linux 100 abc
7. Linux ”|“ TCP
8.
9.
10.

711

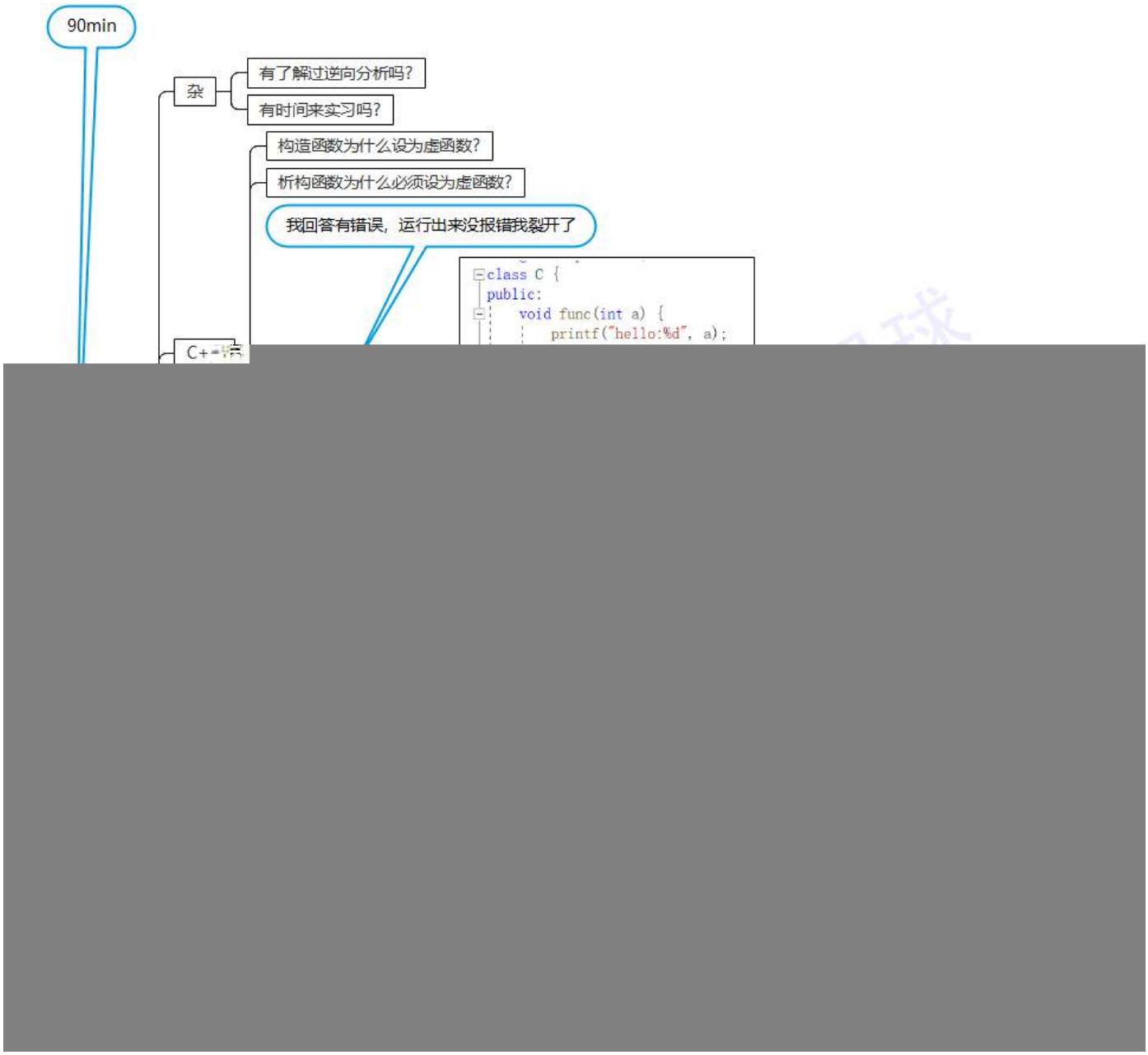
Linux

down

Linux

45

> @author



Go

1-Go-

1.
2. https
3.
4. go channel
5. go interface GMP
6. goroutine
7. channel ->
8. slice 1024
9. MySQL

10. Redis
11. Redis
12. Redis
13. Redis
14.

-
-
-
- ( )
-

0.
1.
2. DNS
3. https
4. go channel
5.

-
-
-

1.
2.
3.
4.
5.
6.
7.
8.

2-Go-

1.
2.
3. redis hash
4.
5.

6.
7.
8.
9. redis hash
10.
11. MYSQL MVCC
12. Golang

13. Linux
14. CPU
15.
16.
17.
18.
19. new

linux
20. Python / /
21. Python map
22.
23. -K

3-Go-

> @author A N K

blog

985 2019 Golang Web
curd leetcode200

- - base 2021-12-12 12-20 12-21

> 2021-12-22 1h10min

1

2 OSI 7

OSI TCP/IP - yinrw - (cnblogs.com)

3 tcp udp

TCP UDP - Fundebug - (cnblogs.com)

4 MTU 1500

ip 20~60 1500-20=1480

5 tcp

TCP (baidu.com)

6 golang

Golang - Go - Golang (studygolang.com)

channel csp gmp gmp

7 goroutine

goroutine SlagSea-CSDN goroutine

8 linux

Linux R S D T Z X( )sdkdlwk -CSDN d

9 golang

mutex rwmutex

10 go

11 go gc

Golang - (zhihu.com)

12 mysql ACID

, kayla -CSDN

13 cap

CAP - (ruanyifeng.com)

14 cap c acid c

cap c acid c mysql

acid c

15 mysql

mysql - andy - (cnblogs.com)

16 innodb MyISAM

MyISAM InnoDB 9 -CSDN innodb myisam

17 innodb

index

18

mysql mysql weixin39660931 -CSDN

19 mysql

20 mysql

pc

21 mysql

> eg. A A B B

>

5

6

mysql MySQL - UCloud

7

curd

8 golang gmp

Go GPM - (juejin.cn)

9 gmp m p

10 go gc

Golang - (zhihu.com)

11 docker

image

12 docker

docker

Docker Linux (Namespace,Cgroup) -CSDN

13 k8s

14 tcp

TCP “ ” “ ” (baidu.com)

timewait closewait

15 tcp

TCP genzld -CSDN tcp

16 Linux

Linux (baidu.com)

17

fff -CSDN

18 os

csdnkou -CSDN

19

+ res

10min

12.27 40min

1

2

3

Linux - (zhihu.com)

4

5 golang goroutine

6 golang goroutine

gmp m os

7 gmp m

M OS M M M

G P Go

-- Go

8 mysql

MySQL - - (cnblogs.com)

9

10

11 ut

12 docker XXX

13 go

```
func calc(base int) (func(int) int, func(int) int) {
  add := func(i int) int { base += i
  return base }
  sub := func(i int) int { base -= i
  return base }
  return add, sub
}
func main() {
  f1  f2 := calc(10)
    fmt.println(f1(1),f2(2))
    fmt.println(f1(3),f2(4))
    fmt.println(f1(5),f2(6))
    fmt.println(f1(7),f2(8))
}
```

?

```
11 9
12 8
13 7
14 6
```

14 list

:

1

2

3

4

5

leader

g

> go go

hr hr

HR

> 12.31 15min

1

2

3

4

1. 2.

4-Go-

1.
2. defer
3. leetcode
4. Go panic recover panic
5. panic recover
6. map
7. go map
8. map sync.map cmap
9. goroutine
10. gmp g m p go m
11. goroutine
12. channel
13. Go gc
    STW
14.
15.
16. https
17. time_wait 2 MSL
18. websocket
19. tcp http why http
20. rpc http
21.
22.

23.
24. Linux
25. Linux
26. Linux
27.
28. MySQL
29. mvvc
30. MySQL
31.
32.
33.
34. gin
35. Nosql redis mongodb
36. docker k8s
37. pod deployment service
38. deployment pod
39.
40. orm

5-Go-

1.
2.
3. Raft Raft
4. Raft
5. Raft
6. Raft id
7. Gossip
8.

web

1. jwt
2. session token token
3. token
4. token

go

1. go routine
2. go routine
3. go routine ?
4. defer

1.

2.
3. B+ B
4.
5. c++ map go map
map hmap bmap hash

6. delete drop truncate
7. MVCC mvcc redo log read review db_trx_id
read review rr rc rr
8.
9. binlog redo log undo log

1. http tcp
2.
3.

25mins

1. leecode 3. medium
2. leetcode 200. medium

1.
2. B B+ B+ B
3.

1.
2.
3.
4.
5.
6.
7. linux
8.

1. tcp
2. https
3. golang
4. go channel hchan

k

1.
2.
3.
4.
5. paper
6.
7.
8.
9.
10.
11.
12.
13. kill
14. kill
15.
16. golang map
17. map
18. sync.Map
19. sync.Map
20. sync.Map map
21. sync.Map

01- -

> The Knight

1.
2.
3.
4. -
5. [9,98,123,32] -
6. web
7. - cv

8.

02- -

1.
2.
3. java
4. hashmap
5. java
6. == equals ==
7.
8. http
9.
10. tcp
11.

03- -

dreamer

1.
2.
3. jmeter
4.
5.
6.
7. :
8. linux
9.
10.
11.

04- -

1. (LC 76)

P.S: O(N) substr O(N^2)

2. 2
   - o
   - o sql

table1 table2

Q1 : 80

Q2:

79 90

Q1

Q2 Q2 Q1

3. 3
   - DNS
   - linux ps netstat, ls, more...
4. 4
   - linux cp 5 3
   -
5.
   -
   -
   -
   -

FTP

TCP

HTTP

MySQL
MySQL --> B+ --> B+ (O(logN) O(N)??
)

-->

-->

05- -

, , , , , ,
, , .

, , ,

1. , ,
2. (, , , „ ,
) , , (offer,
,)
3. , k , , JAVA HashMap
, C++ , C++ .C++ map v
. map , , ,
4. SQL , count grpup by . , , ,
, ,
5. , , (gtest ,
, , , , ,
,)
6. (, , , ,
)
7. , , , , , ,
,

(3.22)

1.
2.
3. ( )
4.
5. redis
6.
7. SQL , ,
8. , , debug ,

06- -

@

1.
2.
3.
4.

: # ( )

1.
2.
3.
4.
5.
6.

hr + 25min

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.

07- -

> @

1h10min

1.
2.
3.
4.
5.
6.
7.
8.
9. cookie session
10.
11. get post
12. mysql
13.
14.
15.

08- -

> @

37min

1.
2.
3.
4.
5.
6.

7.java buffer builder

8.

9.

10.

11.

50min ..

1.
2. Java C+ Python
3.
4.
5.
6.

1h

1.
2.
3.
4.
5. “
” ...)
6. split

09- -

> @

35min

1.
2.
3.
4.
5.
6. alpha beta
7.
8.
9.
10.

39min

1.
2.
3.
4.
5.

6.
7.
8.

10- -ZEKU

> @

35min

1.
2.
3.
4.
5.
6.
7.
8.

45min

1.
2.
3.
4.
5.
6.
7.
8.

11-

> @

30min

1.
2.
3.
4.
5.

20min

1.
2.
3.
4.
5.
6.

12- -

> @

40min

1.
2.
3. http
4. get post
5.
6.
7.

13- -

> @

8.25 12:00

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11. 1 3 5 7 9 5
13579 5 11s

14- -

> @

1.

2.
3.
4.
5.
6.
7.

8.
9.

10. static static static
11. tcp udp
12. tcp
13. tcp udp
14.
15.
16.

1.
2.
3.
4.

5.

1.
2.
3.
4.
5.
6.
7.
8. Java
9. Java
10. SQL Redis
11.
12.
13.

Java IDE linux SQL postman

1.
2.

3.

4.

5.
6. + A

7.

15- -

@

3.16

, , , , , ,
, , .

, , ,

1. , ,
2. (, , , „ ,
) , , (offer,
,)
3. , k , , JAVA HashMap
, C++ , C++ .C++ map v
. map , , ,
4. SQL , count grpup by . , , ,
, ,
5. , , (gtest ,
, , , , ,
,)
6. (, , , ,
)
7. , , , , , ,
,

3.22

1.
2.
3. ( )
4.
5. redis
6.

7. SQL , ,
8. , , debug ,

16- -

64 4

Tuple List Python2/3 List Set

Pytest

AI AI BP

Python Yield Nonlocal Global

Np.array List Linux

TCP/IP HTTP HTTPS

Nvidia

Nvidia

AI AI MLP BP

Nvidia

Python

AI BP

Nvidia

AI CN CV Pytorch BP GPU CPU Linux

Nvidia

Pytest Fixture Dependency Jenkins

C++ C++ C C++ Python

Nvidia

Delay

Intel

Intel

Python KNN MLP KNN
MLP Reward

Intel

Linux Vim Log

Python Pytest

Intel

Git

Python Yield Return

AI

Intel

MLP GBDT Attentiom MLP DDQN Tricks PG AC
Reward

1- -

> syrup

1. 30px flex 100px
2.
3. 6px
4. 1,2,3 4,5,6 1 4 ...
5. reduce

2- -

60

1.
2. https://todomvc.com/
3. `React` `ToDoList`

- 15min

-
  -
-
  -
-
  - `storage` `localStorage`
  - API `GPT`
4.

(50min)

1.
   -
2.
3. `JWT`
   -
4.
   - `cookie` `session`
5. `HTTP`
6.
7. `git` `rebase` `merge`
8.
   - `stash`
9.
10. `bug`
11.
12.
    -
13. n
    - `dfs`

3- -

> 1. freestyle
> 2. (STAR)

1.
2. js
3.
4.
5.
6.
7.
8. HTTP
9. HTTP HTTPS
10. vue
11. vue
12.
13.

 25

1. freestyle
2. (STAR)

 25

1.
2. 15
3.
4.
5.
6. TCP
7. TCP
8. TCP
9. TCP
10.
11.
12.
13. MDN JS
14. ...
15. async/await
16. async/await
17. await

18.

4- -


腾讯一面
自我介绍
能力（比赛、项目）
实习——参与项目：监控、运维
性格
细心、开朗、前公司的同事相处的比较
l. js 考察
防抖
t内，频繁->最后一次为准
setTimeout
应用场景：在输入框需要不断响应的时候
节流
t内，频繁->只一次
时间戳、定时器
滚动条
代码
js闭包
在一个函数中，若里面有一个新的函数，新的函数可以访问外面这个函数的值，叫闭包。（有权访问另一个函数作用域中的变量）
计时器的var和let
事件循环
2分钟视频，队列
题目
原型链
也去稀土看看

5- -

> ø

1.
2. TCP
   -
3. TCP
   ACK
4. HTTP
5.
6. HTTP2
7.
8.
9. HTTPS .....
10.
    - bind call apply
11. this ....

```
name = 'a'
const obj = {
  name: 'b',
  fn: () => {
    return this.name
  },
  fn2(){
    return this.name
  }
}
const fn = obj.fn
const fn2 = obj.fn2
// console.log(obj.fn())
// console.log(fn())
// console.log(obj.fn2())
// console.log(fn2())


//a
//a
//b
//a
```

12.

[0~25] [a~z], 11258

....

13.

....

6- -

1.
2.
3.
4.
5.
6. MVVM
7.
8.

1.
2.
3.
4.
5. vue
6.
7.
8.
9. CSS
10. HTTP2.0 3.0
11. quic
12. tcp UDP
13.

TCP UDP

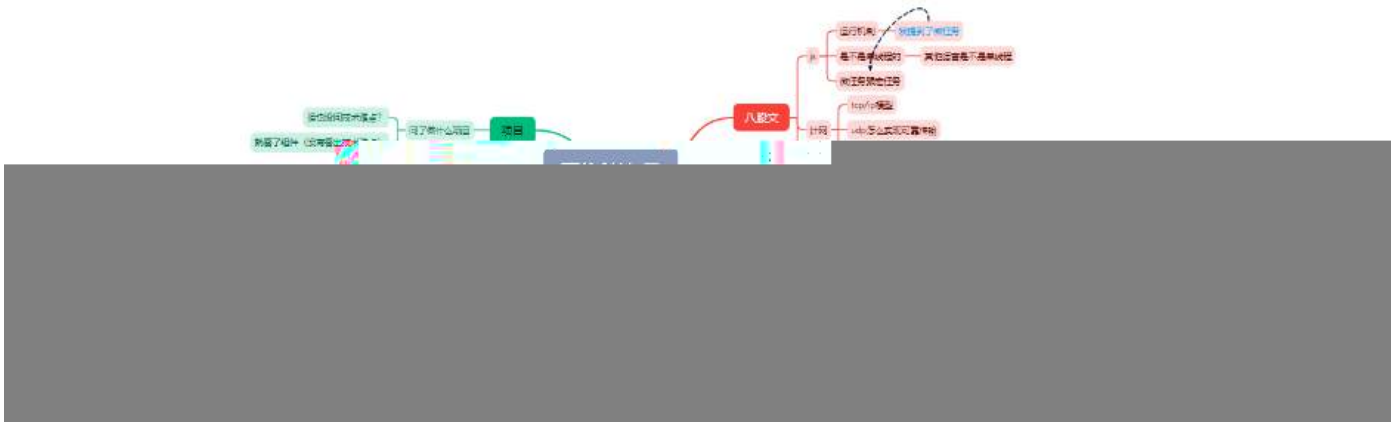
HR

7- -

30

1. URL
2. TCP
3.
4. CSS
5.
6.
7.
8.
9.
10. JS
11.
12. typeof null
13. JS
14. JS
15. JS
16.
17. Vue3 Vue2 API setup
18.

8- - /



9- -

> Noah

1.
2. Monorepo
3. gulp rollup
4. Vue scope-CSS
5.
6. git rebase git merge
7. display
8. CSS
9. v-if v-show v-if dom v-show
   - v-show css--display:none,dom
   - v-if dom
10. React useCallback useMemo
11. React useEffect useLayoutEffect
12. http
13. tab js JS CSS

leader

1.
2.
    1.
    2.
    3. review
    4.
3.
    1. vitest
    2. DDD
4.
    1. cors
        1. express
    2. jsonp
        1.
    3. websocket
        1.
        2.
        3.
    4. iframe
5. vue react
    1. mvvm mvc
    2. diff
    3. Vue React UI
    4. Vue api React JSX JS Vue3 JSX hook
6.
    1. Nginx
7. docker
    1. docker file compose docker
8.
9.
    1. React+TS
    2. React
    3. Vue
10.
    1.
    2.
    3.
    4. +1

10- -

1.
2.
3.
4. , ,
5. nodejs java
6. ,
7. jwt
8. js
9. v8
10. post
11. post ContentType (application/x-www-form-urlencoded/multipart/form- data/text/plain)
12. jiwt
13. http https
14. https tls
15. ( ),
16. axios ,
17. ,
18. css
19. box-sizing border-box content-box

1. nginx ,
2. linux
3. node ,
4. nodejs
5.
6. koa2 compose
7. Scheduler ,

1.
2. (react+node+)
3. , ?
4. node
5. , node?
6. , node java ?
7. node
8. http ? ?(https)
9. https ?(tls)
10. tls (, , +)
11. ?(AES)

12. ? ?(base64,cors,xss,csrf,cookie httponly samesite)
13. ?
14. 1: s
15. 2: , ,
16. 3: 1
17. 4:
18. :

11- - / /oppo

> WU

7.23 (1h)

1.
2.
3.
4.
5. url
6. http1.0112.0
7. http nttps
8. vue3
9. vue2 vue3
10. vue3
11. ts js
12. ts
13. Let const var
14. JSONstringfy
15.
16.

7.23 (50min)

1.
2.
3.
4.
5.
6. (Object.create())
7. vue3
8.
9. promise
10. shuffle
11.
12.

1. css
2.
3.
4. this
5. apply call bind
6. async await
7. top await?( )
8. vue3
9. ts
10. interface type
11. boolean
)
12.
13.
14.
15. +once
16.
)
17.
18. jwt
19. webpack loader plugins
20.
21. nextTick
22.
23.

OPPO

1.
2. vue2 vue3
3.
4.
5.
6. diff
7.
8. class
9. es5
10. class
11. promise
12. cors
13. cookie
14. XSS
15. csrf
16.

17. gzip
18.
19.

12- -

1. ➕
2.
3. vue3.0 vue2.x vue3.0 vue2.x
4. react react
5. react hook
6. webassembly js (js audio 😭
7. vue
8. vue-module vue

css

1.
2.flex
3.flex:1
4.BFC

js

1.
2. promise promise.all
3.
4. js
5. ES5 ES6
6. TS TS js TS
7. TS
8.
9.sessionstorage localstorage
10. https http

1. ➕
2. react vue
3. vue3.0
4. BFC
5. js
6.
7.
8.
9.

13- -

1.
2.
3.
4.
5. JS
6. JS
7. Vue.nextTick
8.

```
//a.js
const b = require('./b.js');
console.log(exports.x);
exports.x = 'x';
require('./c.js');
//b.js
const a = require('./a.js');
console.log(a);
a.x='y';
//c.js
const a = require('./a.js');
console.log(a.x);
//     node a.js
```

9. UDP TCP
10. HTTP

11.
12. mentor

1.
2.
3.
4.
5. B
6.

+ HR

1.
2.
3.
4.
5. HR
6.
7.

1- -

1.
2.
3. int int32
4.
5.
6. int double double
7. C++
8. Sizeof
9.
10. int bool int bool
11.
12.
13.
14.
15. Vector list
16. Vector
17.
18.
19.
20.
21.
22. camera

23. 45
24. Camera
25. update
26.
27.
28.
29. lua index newindex
30.
31.
32. npc
33. Npc
34.
35. unity
36.
37. gpu
38. unity
39.
40.
41.
42.
43.
44. UGUI UI anchor Pivot
45.
46.

2-Android -

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13. Linux
14. Linux
15. data bss
16. bss data
17.

18. fork exec
19. shell ls fork exec
20.
21. TCP
22.
23. epoll epoll select
24.
25.
26. Android
27. Android

1.
2.
3. app
4.
5.
6.
7.
8.
9.
10.
11.
12.

2-Android

> @author weikunkun

一面（50min，终于记得录音了！！！）

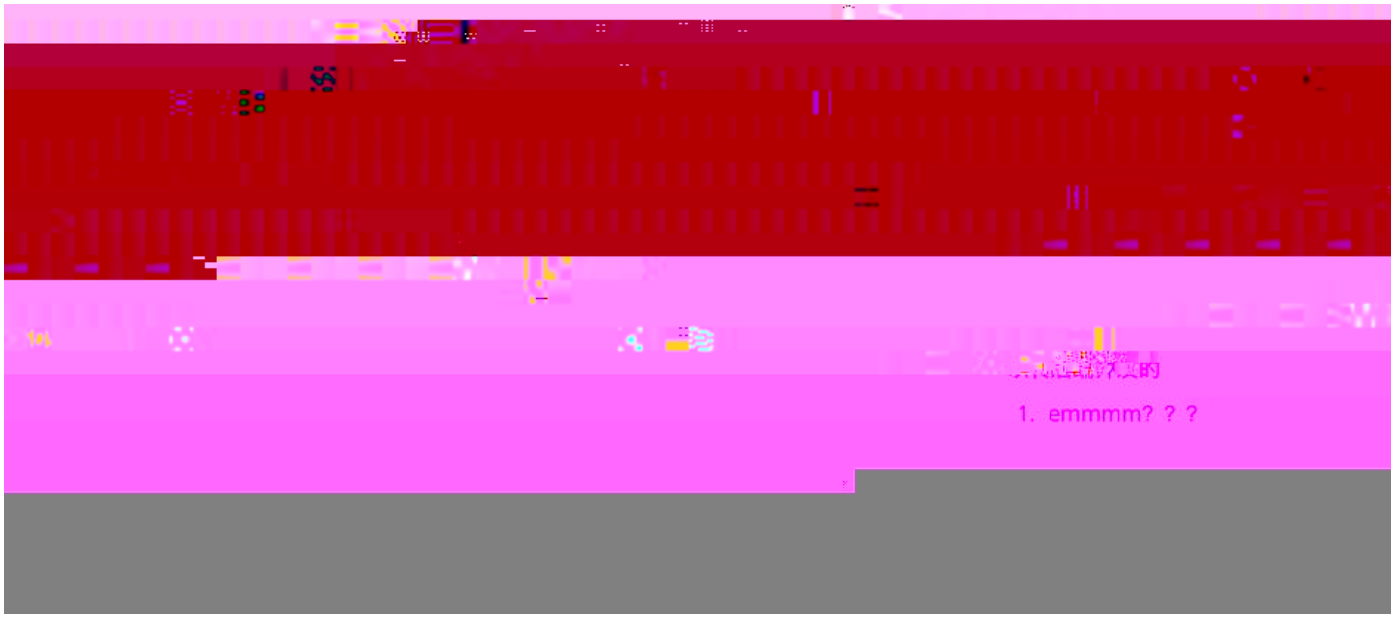
自我介绍。

为什么投递Android，看简历更适合后端嘛，然后说了HR联系说没有HC了，问是否愿意转岗

1. HashMap的阐述 ⚠️
    1. 先笼统的说了整体上采用的什么数据结构，然后解决hash冲突的方式
    2. 然后详细阐述了HashMap的具体实现细节，节点的构成、触发扩容的情况、红黑树、CRUD的逻辑、equals、hashcode关系等
    3. 面试官调侃说讲的挺细的，然后自己感觉有些地方说的有些问题🐵
2. ArrayList的阐述 ⚠️
    1. 底层实现、扩容机制、尽量初始化时制定容量大小，避免频繁扩容的开销，
    2. 说了头部插入、中间插入、尾部插入的性能区别
3. 线程安全
    1. 线程安全需要解决的核心问题：
        1. 原子性
        2. 可见性
        3. 有序性
    2. 实现线程安全的方式
        1. 阻塞同步：加锁

    1. 阻塞同步：加锁
    2. 非阻塞同步：cas
    3. 无同步方案：本地存储
4. final关键字
    1. 修饰
        1. 类
        2. 方法
        3. 成员变量
    2. 然后讲了保证了可见性
    3. 然后说了看了一些源码里面的finaly的修饰、譬如AQS的release、acquire方法、String类本身等
        1. 主要是为了保证自身的安全性的机制
5. Integer、int的区别 ⚠️
    1. 集合类的不支持基础数据类型存储
6. 问了泛型⚠️
    1. 泛型说的比较含糊
    2. 避免不必要的类型强转
    3. 设计的类或者说数据结构，更加具有通用性，支持各种数据类型：JDK定义的、自定义的
7. String、StringBuilder、StringBuffer
    1. 对与需要频繁进行拼接、截取的操作，使用StringBuilder、StringBuffer
    2. StringBuffer相比于StringBuilder，保证了单个API操作的线程安全
8. TCP、UDP的区别
9. 问了拥塞控制 ⚠️
    1. 有些模糊了，所以说的极其凌乱。。。
10. 单例模式
    1. 基础饿汉模式
    2. 基础懒汉模式
    3. DCL 注意指令重排
    4. 还想说其他的实现，被打断了，说差不多了
11. 单元测试
    1. 复杂逻辑
    2. 模拟请求，输出格式是否符合要求
    3. 避免线上出现问题，消耗的排查时间吧
12. 算法：求两个有序数组是否有交集
    1. 数据类型会明确吗，如果不明确的话，需要使用泛型。然后补充说int类型。
    2. 然后和面试官说了几个思路，以及对应的时间、空间复杂度
        1. 双指针
        2. set
            1. 判断是否存在
            2. 判断长度是否相同
        3. map
            1. 统计次数
    3. 说的思路里面挑了一个实现，然后验证

题目：第一次遇到，还挺有趣的哈哈哈



3- -

1.
2.
3.
4.
5. HTTP
6. get post
7. HTTP
8. UDP TCP
9. TCP
10. time wait
11. select epoll
12. reactor
13. proactor reactor
14.
15. PCB
16.
17.
18.
19.
20. C++ +
21.
22.
23.
24. STL
25. vector list
26. map unordered_map
27. target 34.

1.
2. http1.0 1.1 2.0 3.0
3. io
4. http2.0
5. HTTP TCP UDP TLS
6. LeetCode 3

1.
2.
3.
4.
5. swift C++
6.
7.
8.
9.

4- -

> 211 985 CS-Notes
> C++

(1h)

1. 1
2. 5 Android C++
3.
4. inline
5. inline
6. main va_list
7.
8.
9. const mutable
10. const* *const
11.
12. new malloc
13. map unordered_map
14. avl
15. hash
16.
17.
18.
19. URL

20. tcp udp
21. socket
22. socket http
23. http2.0
24. quic http3.0
25. python python
26. :

(1h)

1. 1
2. 15
3. Main
4.
5.
6.
7. c++ jni ndk
8. i+1 i
9.
10.
11. dll
12. cmake
13. 32 64 int
14. LRU

(1h)

1. 1
2. 20
3.
4.
5. c++
6. c++ static
7.
8.
9.
10. C++
11. STL
12.
13. target

HR

1.
2.
3.
4.
5.

6.
7.
8. Java c++
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.

5- -

> @Woods Lin

55min

1.
2.
3. IO
4. select
5.
6.
7.
8.
9. i++
10. HTTP HTTPS
11. HTTPS
12. SSL
13. DNS
14. DNS
15. TCP UDP
16.
17. TCP
18. 0
19. TCP
20. C++ class struct
21. C struct

22. class private private
23.
24. malloc new
25.
26.
27. class
28. C++
29.
30. map unordered_map
31.
32.
33.
34. B+ B
35. dfs
36.

1h10min

1.
2. ok

3. web
4. mysql http cookie session
5. cookie session - GET POST

6.
7. 1.1 DELETE
8. HTTP 2345
9. QQ Ping
10. ping ping ping ping
11. ping block
12. qq
13. HTTP TCP UDP TCP
14.
15. HTTP
16.
17. IP
18. DNS DNS
19. DNS ping ping ip
20.
21.
22. a a 2
23.
24.

25. DNS ping

6- -

> @author

字节三面(1h20min)

1. 写一道算法题吧：看了下是力扣两数之和的变体

   排序 + 双指针简单解决，未考虑极端情况，然后面试官没有让我分析代码说明思路，直接问：如果让你设计测试案例，你会怎么做？

   接下来我说了几种比较常见的边界：

   1. 数组为空或者元素个数小于2
   2. 数组元素大数情况（即做加法的时候会发生溢出）
   3. 存在重复项满足条件的情况，如 arr = [3, 3, 5, 5], target = 8, 最终的结果应该是四个
   4. 不存在合适解，返回空

   接下来面试官让我检查算法是否能够处理这些情况，但是由于未考虑到重复项，前面的方法作废了，然后想了想可以用哈希表实现，哈希key记录元素的值，value记录值为key元素的索引（可能有多个，所以用了数组vector去存储），然后就是排序加双指针大框架去做遍历，只是在找到答案的时候需要进行O(N^2)的组合（全组合问题，需要二重循环）

2. 第二道算法：一共n个人（编号 0 ~ n - 1），初始每个人手中都有糖果，交换他们手中的糖果，保证均匀分布，并且每个人不会拿到原来的糖果（重点是如何保证均匀性）

(40+min)

hr

( )

wo a (15min

) ( )

(hr hr)

字节六面

1. 快排（写了个简单的）
2. 优化一下快排（减少了函数调用）
3. 智能指针?
4. shared_ptr循环引用的小demo
5. 面试官在编辑框写代码让判断是否出错，为什么？（考了智能指针引用普通指针的错误、前向声明）
6. 循环数组下一个更大的元素

hr hr

hr

-
-
-
-
-
- (10)
-
-
-
-

HR

hr

1-NLP-OPPO

> seven

OPPO NLP OPPO NLP

ChatGPT

-)
- BERT
- BERT
-
- Transformer Attention

- Attention 3 query
- RNN
- Word2Vec
- Skip-Gram ?
-
-

- +
-
- token limit input
-
- /
- bq
-

hr

-
-
-
-
- OPPO
-
-
-

3-NLP-

> xyFei

NLP

-
-
- NLP
-
-
-
-

(

)

4-NLP-

> seven

-
- Python
- Python list
- Java ArrayList, LinkedList
- B+ Tree + Index
-
-

NLP
Java B+ Tree
ML DL

5-NLP-

> seven

1.
2. NLP
3.
4. BERT
5. BERT GPT
6.
7. P-tuning-V2
8. Word2Vec
9. LayerNorm BatchNorm
10. target

6- -

> seven

50min

-
-
- Kmeans Kmeans++
- SVM Logistics Regression
-
- XGBoost
- Bagging Boosting
- L1 L2
-
-
-
- SQL

edge case

7- -

seven

1.
2.
3. Lora
4.
5. QLora
6.
7. Git
8. cuda
9. pytorch
10.
11. cpu gpu

8- -

&

1. Python c++ Python Python
2. list [1,2,3,4,5] 2
3. Python is ==
4.
5. ? ?
6. ?
7. base ?
8. ?

9- -

&

: : ?

1.
2.
3. yolov5
4.
5.
6. ? ?
7. Python : ? ?
8. opencv
9. ? ?

10.

10-

> @author Douglas

question:

DBNet
DBNet
DBNet

psenet east
crnn ctc
crnn

ocr

YOLO3 YOLO5
anchor
yolo anchor
RPN
FPN
yolo

yolo
centernet
one stage
IOU

OpenCV
OpenCV

ncnn :

FPS

tricks

:

-
-
-
-
-

17K

C++

11- - pnc

1. dqn ddpg ppo
2.
3. KL PPO
4. python
5. map val map key sort pair vector

sort cmp l.second > r.second

6. c++
7. vector list queue
8. push*back* *emplace*back
9.
10. Linux --

63. _ . . ll m x n “Start”
“Finish”
——

1. RL MADDPG VDN QMIX COMA MAPPO QMIX MADDPG
2. MARL——
3.

dfs

1. SAC RL AC
2. Linux Ubuntu
3. python
4. numpy
5.
6.
7.
8.
9.
10.

12- -

> C9 23 20
>
> 3 15 21 24 29 31 HR 4 2 offer

1.
2. AUC
   word2vec
3. - hash map l l

1. embedding PCA
2.
3.
4. LeetCode easy

1.
2.

13- -

cnn-bilstm

mape

python

mysql

left join inner join

k

offer

0

0

+

1 lstm

2 lstm

( )

+ balal

3

sigmoid

4 sigmoid

0 1

5

sigmoid logistic tanh

relu

6 cnn

7

bala

8

nice

1

2

3

4

5

6

7

8 arima

9 p,q

10 prophet

11

12.1 PT

12.2

12.3

12.4

13

14

15

16 EM

17 EM

18 EM

19 doc2vec

bert GPT

20 GPT

21 GPT

debug

ROI

22

23

24

nice

1

2.1 KNN

2.2 kd

3.1

3.2

3.3

4 SVM

5

6 Adam

7

8 tensorflow

9 LSTM

10.1

10.2 sigmoid

10.3 sigmoid tanh

11

11.1 1-100 log(N)

11.2

xxxx

1- -

> Frank

45min

1.
2.
3.
4.
5.
6. PV
7. nuttx inux
8. linux
9. C++
10.
11.
12.
13.

2- -

> (´ ω `)

1. ROS 1 ROS2?
2. C C++ (C c)
3. volatile
4.
5.
6.
7.
8.

9.
10. uart i2c i2c i2c
11. usb
12.
13.

3- -

> @

37w+10w BASE

gdb git gdb git

Q1:Arm bl

A: arm TI X86 B
,bl X86 jl x86 long jump

Q2:static C

A data

Q3:C

A: stack data heap bss

Q4

A:stack 0.

Q5:

A: free

Q6

A DMA

Q7: gcc hello.c

A:gcc hello.c -o hello.out

Q: gdb

A:

Q:

A:

Q1: fft

A:
fft

Q2:

A:

Q3 cache

A: DSP DSP 0 1
cache cache
cache_invalid cache_writeback
cache, writeback

Q4:

A: X86
spinlock

Q5: linux

A: linux

Q6:

A: i386 qemu Sytemcall

Q7: Boot

A i386 bios 0x7c80
entry
page directory, 4kb
linux fork page directory

round-robin hash-map

Q8:

A: bss end, cmos

Q9: 4kb

A: i386 MMU 4kb 12

Q10: fork

A: idt
isr cr2 isr
fork

Q11:

A: eip systemcall

4- -

> @

46w

HR

Q1: SRIO

A: DSP+FPGA SRIO SRIO
PCIE PCIE SRIO FPGA xilinx
IP DSP

Q2:FPGA

A: Verilog, FPGA biss

Q3: cache

A:

Q4:

A:

Q5:

A:

C C text data stack

C volatile

5- -

> @

Q1:

A:

ISR

Q2:

A:

Q3:

A: TI sys-bios linux linux sys-bios

Q4:

A: systemcall

Q5:

A:

6- -

> WoodsLin

1.
2.
3. web
4.
5.
6.
7. Linux
8. malloc
9.
10.

7- -

-
- C++11
- lambda -
- shapeptr
-
-

- i +1 -1
- -
- STL queue
- queue
- pop
  queue
- +1 -1 0
-
-
-
- map unordermap
- static
- IO
- epoll ET ET
- ET
- TCP
-
-
-
- HTTP TCP
-
-
-
-
-
- + +
-
-