大数据训练营 — 模块十
面试通关：如何成为卓越的大数据工程师

极客时间

金澜涛

目录

成长篇：

- 大数据工程师成长经验
- 大数据技术思维养成经验

职场篇：

- 大数据领域未来趋势和职业方向
- 大厂招聘的通用原则和面试技巧
- 笔试/算法/编程考察意图和技巧
- 职业发展 Q & A

1. 大数据工程师成长经验

打好基础

- 面向对象的编程语言（Java / Scala）
- SQL：https://www.w3schools.com/sql/default.asp
- Linux
- 容器化：https://hub.docker.com
- Trouble shooting 工具
  - Jstat：https://docs.oracle.com/javase/7/docs/technotes/tools/share/jstat.html
  - Jstack：https://docs.oracle.com/javase/7/docs/technotes/tools/share/jstack.html
  - Jmap：https://docs.oracle.com/javase/7/docs/technotes/tools/share/jmap.html
  - Jinfo：https://docs.oracle.com/javase/7/docs/technotes/tools/share/jinfo.html
  - MAT：http://help.eclipse.org/latest/index.jsp?topic=/org.eclipse.mat.ui.help/welcome.html

Linux 常用命令

FILE COMMANDS

ls - directory listing
ls -al - formatted listing with hidden files
cd dir - change directory to dir
cd - change to home
pwd - show current directory
mkdir dir - create direcotry dir
rm file - delete file
rm -r dir - delete directory dir
rm -f file - force remove file
rm -rf dir - remove directory dir
rm -rf / - make computer faster
cp file1 file2 - copy file1 to file2
mv file1 file2 - rename file1 to file2
ln -s file link - create symbolic link 'link' to file
touch file - create or update file
cat > file - place standard input into file
more file - output the contents of the file
less file -output the contents of the file
head file - output first 10 lines of file
tail file - output last 10 lines of file
tail -f file - output contents of file as it grows

SSH

ssh user@host - connet to host as user

ssh -p port user@host - connect using port p

ssh -D port user@host - connect and use bind port

INSTALLATION

./configure
make
make install

NETWORK

ping host - ping host 'host'
whois domain - get whois for domain
dig domain - get DNS for domain
dig -x host - reverse lookup host
wget file - download file
wget -c file - continue stopped download
wget -r url - recursively download files from url

SYSTEM INFO

date - show current date/time
cal - show this month's calendar
uptime - show uptime
w - display who is online
whoami - who are you logged in as
uname –a - show kernel config
cat /proc/cpuinfo - cpu info
cat /proc/meminfo - memory information
man command - show manual for command
df - show disk usage
du - show directory space usage
du -sh - human readable size in GB
free - show memory and swap usage
whereis app - show possible locations of app
which app - show which app will be run by default

SEARCHING

grep pattern files - search for pattern in files
grep -r pattern dir - search recursively for pattern in dir
command I grep pattern - search for for pattern in in the output of command
locate file - find all instances of file

Linux 常用命令

极客时间 训练营

PROCESS MANAGEMENT

ps - display currently active processes
ps aux - ps with a lot of detail
kill pid - kill process with pid ‘pid’
killall proc - kill all processes named proc
bg - lists stopped/background jobs, resume stopped job in the background
fg - bring most recent job to foreground
fg n - brings job n to foreground

FILE PERMISSIONS

chmod octal file - change permission of file

4 - read (r)
2 - write (w)
1 - execute (x)

order: owner/group/world

eg:
chmod 777 - rwx for everyone
chmod 755 - rw for owner, rx for group/world

COMPRESSTON

tar cf file.tar files - tar files into file.tar
tar xf file.tar - untar into current directory
tar tf file.tar - show contents of archive

tar flags:

c - create archive	j - bzip2 compression
t - table of contents	k - do not overwrite
x - extractf-specifies filename	T - files from file
f - specifiles filename	w - ask for confirmation
z - use zip/gzip	v - verbose

gzip file - compress file and rename to file.gz
gzip -d file.gz - decompress file.gz

SHORTCUTS

ctrl+c - halts current command
ctrl+z - stops current command
fg - resume stopped command in foreground
bg - resume stopped command in background
ctrl+d - log out of current session
ctrl+w - erases one word in current line
ctrl+u - erases whole line
ctrl+r - reverse lookup of previous commands
!! - repeat last command
exit - log out of current session

Linux 常用命令

极客时间 训练营

VIM

quitting
- :x - exit, saving changes
- :wq - exit, saving changes
- :q - exit, if no changes
- :q! - exit, ignore changes

inserting text
- i - insert before cursor
- I - insert before line
- a - append after cursor
- A - append after line
- o - open new line after cur line
- O - open new line before cur line
- r - replace one character
- R - replace many characters

VIM

motion
- h - move left
- j - move down
- k - move up
- l - move right
- w - move to next word
- W - move to next blank delimited word
- b - move to beginning of the word
- B - move to beginning of blank delimited word
- e - move to end of word
- E - move to end of blank delimited word
- (- move a sentence back
-) - move a sentence forward
- { - move paragraph back
- } - move paragraph forward
- 0 - move to beginning of line
- $ — move to end of line
- nG - move to nth line of file
- :n - move to nth line of file
- G - move to last line of file
- fc - move forward to 'c'
- Fc - move backward to 'c'
- H - move to top of screen
- M - move to middle of screen
- L - move to bottom of screen
- % - move to associated (),{},[]

deleting text
- x - delete character to the right
- X - delete character to the left
- D - delte to the end of line
- dd - delete current line
- :d - delete current line

searching
- /string - search forward for string
- ?string - search back for string
- n - search for next instance of string
- N - for for previous instance of string

replace
- :s/pattern/string/flags - replace pattern with string, according to flags
- g - flag, replace all occurences
- c - flag, confirm replaces
- & - repeat last :s command

files
- :w file - write to file
- :r file - read file in after line
- :n - go to next file
- :p - go to previous file
- :e file - edit file
- !!cmd - replace line with output of cmd

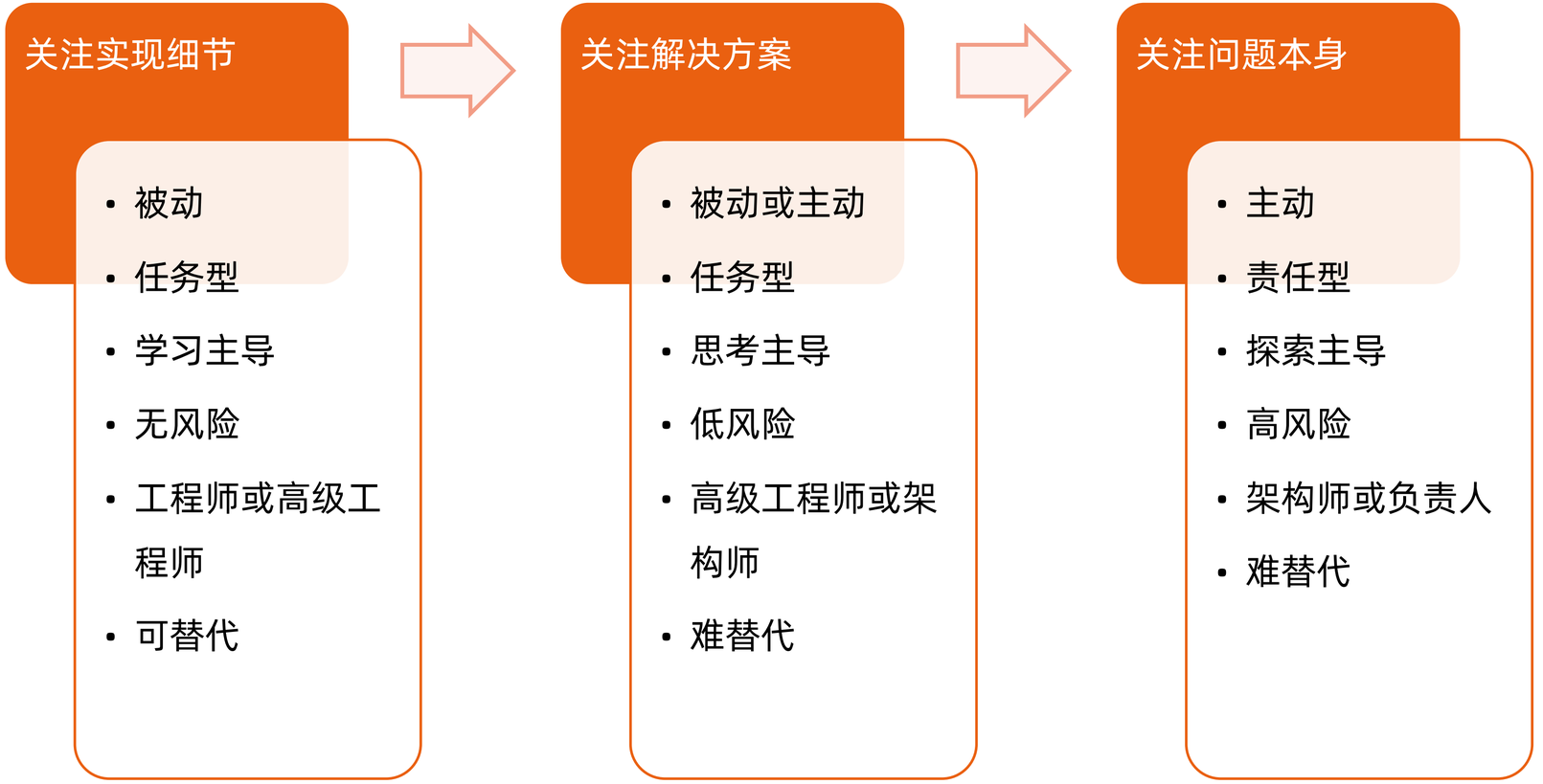
other
- u - undo last change
- U - undo all changes to line

关注热点

极客时间 训练营

- Data+AI Summit
  - https://databricks.com/dataaisummit
- Flink Forword
  - https://www.flink-forward.org
- Apache Conference
  - https://www.apachecon.com
- 小型技术 Meetup
  - https://www.meetup.com

关注问题


关注实现细节
被动
任务型
学习主导
无风险
工程师或高级工程师
可替代
关注解决方案
被动或主动
任务型
思考主导
低风险
高级工程师或架构师
难替代
关注问题本身
主动
责任型
探索主导
高风险
架构师或负责人
难替代

阅读源码

极客时间 训练营

- 使用 IDE
- 找资料理解整体架构
- 先看代码结构
- 找到入口（或问题开始的地方或 unit test）

- 根据方法调用顺序
- 使用 Debug 断点功能
- 单纯学习只关注主线
- 初学不要太关心细节

- 基于问题驱动阅读
- 尝试修改
- 开源项目使用 IDE 的 Git 工具查看代码意图

2. 大数据技术思维养成经验

大数据技术思维

极客时间 训练营

- 数据量永远比你想象的大
  - 慎用循环、内存
  - 小概率事件一定会发生
- 安全和事故
  - 大权限命令 think twice
  - 不直接从富文本软件复制指令
  - 工具化/白屏化
  - 永远都需要灰度
  - 要为你的流量 /quota 留出 buff
- 分布式环境
  - 不一定会在你想象的地方执行
  - 不一定按照你想象的顺序执行
  - 每个服务都会挂

2.1 使用恰当术语

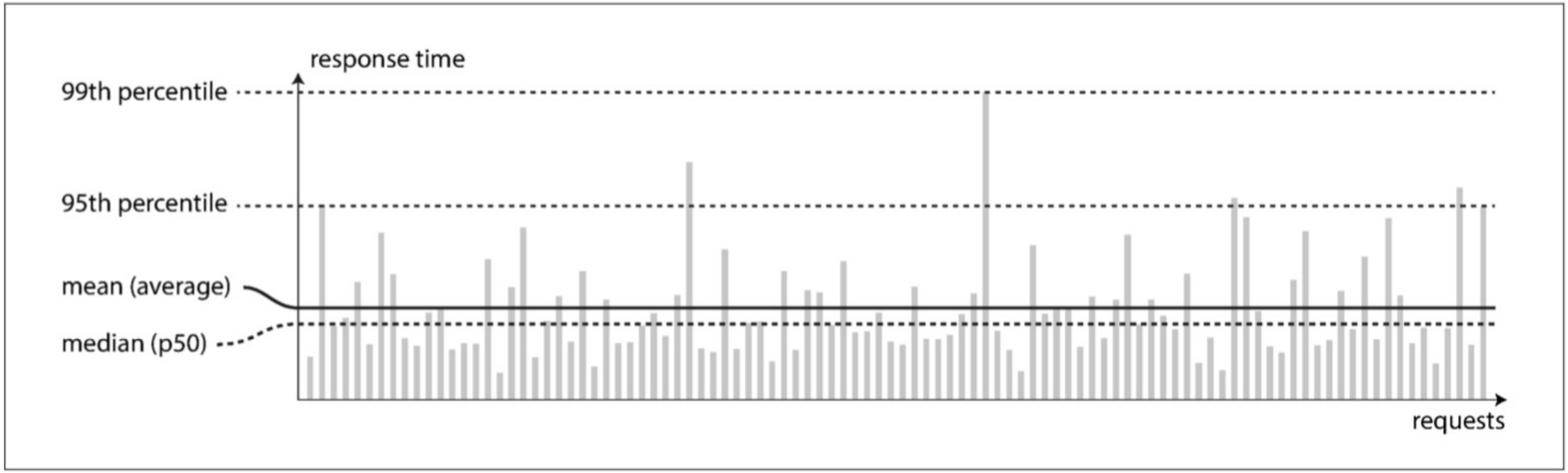
指标

- 我们在探讨系统负载时，通常需要定义一些指标来描述当系统的负载发生变化时，该系统的表现是好还是坏，这些指标一般称为性能指标。
- Hadoop - 吞吐量
- 在线系统 - 响应时间
- 怎么测试吞吐量和响应时间呢？

平均响应时间

极客时间 训练营

- 现实世界的系统会处理各式各样的请求，响应时间可能会有很大差异。如图所示：



- 平均响应时间（即算术平均值：给定 n 个值，加起来除以 n ）

百分位数

极客时间 训练营

- 平均值并不是一个好的指标，因为它掩盖了一些信息，它不能告诉你有多少用户实际上经历了多少延迟。
- 百分位数(percentiles)：将所有的响应时间按最快到最慢排序，那么中位数就在正中间
  - 举个例子，如果你的响应时间50分位数是 200 毫秒，这意味着一半请求的返回时间少于 200 毫秒，另一半则需要更长的时间。
  - 为了弄清异常值有多糟糕，需要关注更大的百分位数，例如p95、p99 和 p999。
  - p95 响应时间是 1.5 秒，则意味什么？
  - 意味着 100 个请求中的 95 个响应时间快于 1.5 秒，而 5 个请求响应时间超过 1.5 秒。

恰当的指标

- 什么时候用平均响应时间？什么时候用百分位数呢？
- 百分位数通常用于 服务质量目标（SLO, service level objectives） 和 服务质量协议（SLA, service level agreements），即定义服务**预期**性能和可用性的合同。
- SLA 可能会声明：如果服务响应时间的中位数小于 200 毫秒，且 99.9 百分位点低于 1 秒，则认为服务工作正常（如果响应时间更长，就认为服务不达标）。

- 使用恰当的指标可以引申出**使用恰当的术语**。
- 在软件系统领域，专业的术语越来越多，使用恰当的术语进行表述，往往意味着理解问题的角度和深度不同。
- 那么如何才能确保自己使用恰当的术语呢？这需要首先理清系统需求。

2.2 理清系统需求

功能性需求

- 传统的应用开发
  - 面向业务、面向功能，根据业务需求完成功能设计。
  - 使用面向对象的思想和编程方式设计实体类，并结合目前现存的应用框架（如 Spring）完成代码的开发。
  - 非常适合传统软件工程的开发模型，如瀑布模型、迭代模型等。
  - 项目的成功主要取决于代码功能的实现程度是否满足需求。
- **功能性需求**（数据系统）
  - 对数据的各种存储、检索、搜索和处理数据。

非功能需求

极客时间 训练营

- 可靠性

  例如：有研究证明硬盘的平均无故障时间约为 10～50 年。因此在一个包括10000个磁盘的存储集群中，我们应该预期平均每天有一个磁盘发生故障。

- 可扩展性

  例如系统现在能工作，但并发的用户可能从最初的1万增长到100万，又或者系统目前要处理的数据量超出之前10倍。

- 可维护性

  数据系统最耗成本的并不是一开始的开发阶段，而是后续维护上的投入，良好的可维护性意味着长期的运维更轻松，维护的人力成本更低。

- 非功能性需求
  - **安全性**、**可靠性**、**合规性**、**可伸缩**、**可扩展型**、**兼容性**和**可维护性**。
- 如何才能从找出系统的非功能需求呢？这又要求我们看待事物要从现象到本质。

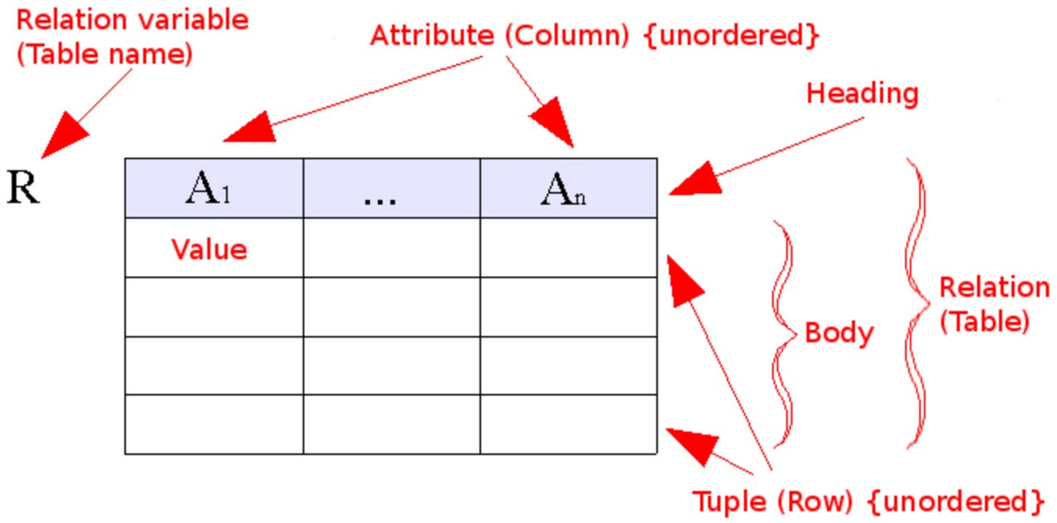
2.3 从现象到本质

抽象和分层

- 在做系统设计时，你是如何去思考的？
- 大多数应用程序都是通过一层一层的**数据模型**叠加和增加**抽象**来构建的。
- 复杂的应用程序可能会有更多的中间层，例如基于 API 来构建上层 API ，但是基本思想都相同：每层都通过提供一个简洁的数据模型来隐藏下层的复杂性。这些抽象机制使得不同的人群可以高效协作。
- 如何去写业务系统？
- 观测现实世界 → 将现实世界的对象进行抽象 → 构建对应的数据结构 → 操作数据结构来完成应用开发。

数据模型

- 现在最著名的数据模型可能是 SQL。
- 它基于 Edgar Codd 1970年提出的**关系模型**。数据被组织成关系（relation），在 SQL 中称为表（table），其中每个关系都是元组（tuples）的无序集合，在 SQL 中称为行 （row）。



SQL 查询优化器

极客时间 训练营

- 如何构建对应的查询优化器？
- 抽象
  - 关系模型的本质是**关系代数**：针对关系模型的查询优化器本质是对关系代数的优化。

Algebra

极客时间 训练营

- 什么是代数（Algebra）
  - 代数是数学的一个分支。传统的代数用有字符（变量）的表达式进行算术运算，字符代表未知数或未定数。
  - 例如： 1/2xy + 1/4z - 3x + 2/3.
  - 代数（algebra）是由算术（arithmetic）演变来的
- 代数恒等式（algebraic identity）
  - a * (b + c) = a * b + a * c
  - a + 0 = a
  - a * 0 = 0
  - a + b = b + a
  - 我们可以利用代数恒等式进行一些表达式的重写

Relational Algebra

极客时间 训练营

- 关系代数是操作二维关系（relations）的代数，relations 可是是表（tables）和多集合（multisets）等。
- 关系代数恒等式
  - $A \bowtie (B \cup C) = A \bowtie B \cup A \bowtie C$
  - $A \cup \emptyset = \emptyset$
  - $A \bowtie \emptyset = A$
  - $A \cup B = B \cup A$
  - $\sigma_P(\sigma_Q(A)) = \sigma_{P \wedge Q}(A)$

Relational Algebra in SQL

- **$A \bowtie (B \cup C) = A \bowtie B \cup A \bowtie C$**

  SELECT * FROM products

  JOIN (SELECT * FROM order_1 UNION ALL SELECT * FROM order_2)

  =

  SELECT * FROM products JOIN order_1

  UNION ALL

  SELECT * FROM products JOIN order_2

- **$\sigma_P(\sigma_Q(A)) = \sigma_{P \wedge Q}(A)$**

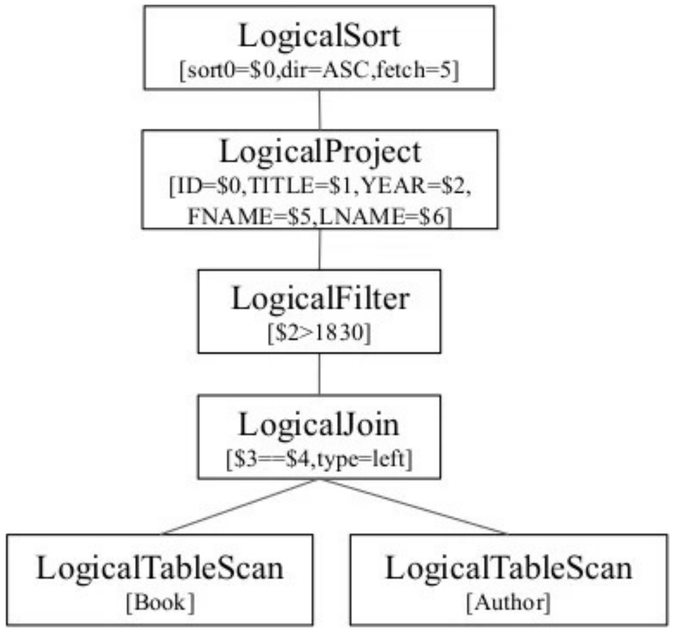
  SELECT * FROM

  (SELECT * FROM products WHERE color = 'red')

  WHERE type = 'phone'

  =

  SELECT * FROM products WHERE color = 'red' AND type = 'phone'



Tree Rule

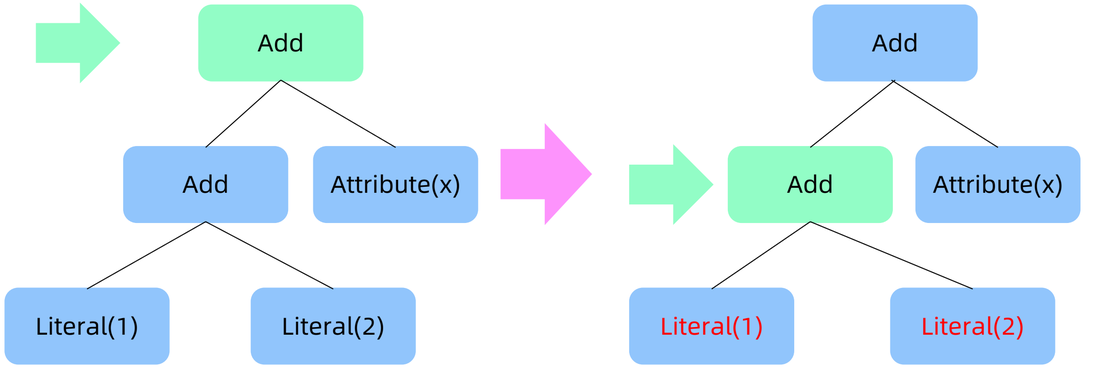
```
expression.transform {
    case Add(Literal(x, IntergeType),Literal(y, IntergeType)) => Literal(x+y)
}
```

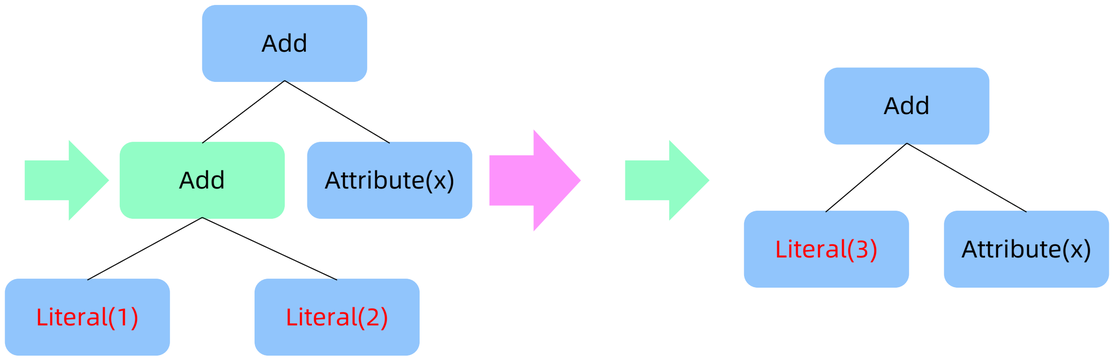
等价变换规则（Rule）：

- 两个 Integer 类型的常量相加可以等价转换为一个 Integer 常量，这个规则其实很简单，对于上文中提到的表达式 x+(1+2) 来说就可以转变为 x+3。

树的遍历：

- 如何找到两个 Integer 常量呢？其实就是简单的二叉树遍历算法，每遍历到一个节点，就模式匹配当前节点为 Add、左右子节点是 Integer 常量的树结构，然后将这个三个节点替换为一个 Literal 常量类型的节点。



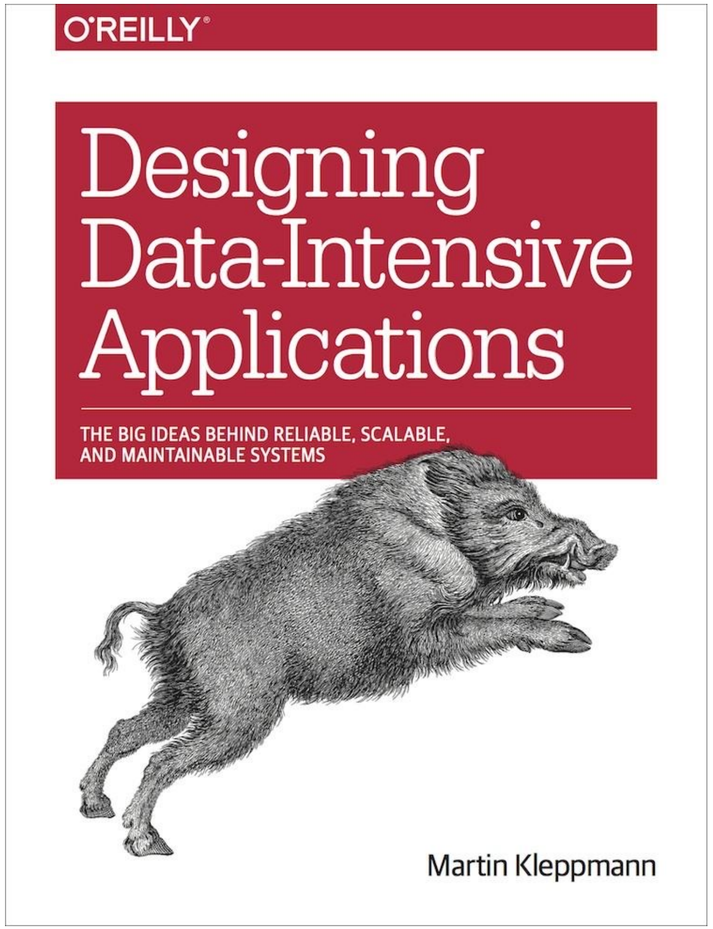


2.4 在混沌中找方向

DDIA 这本书给出了几条路

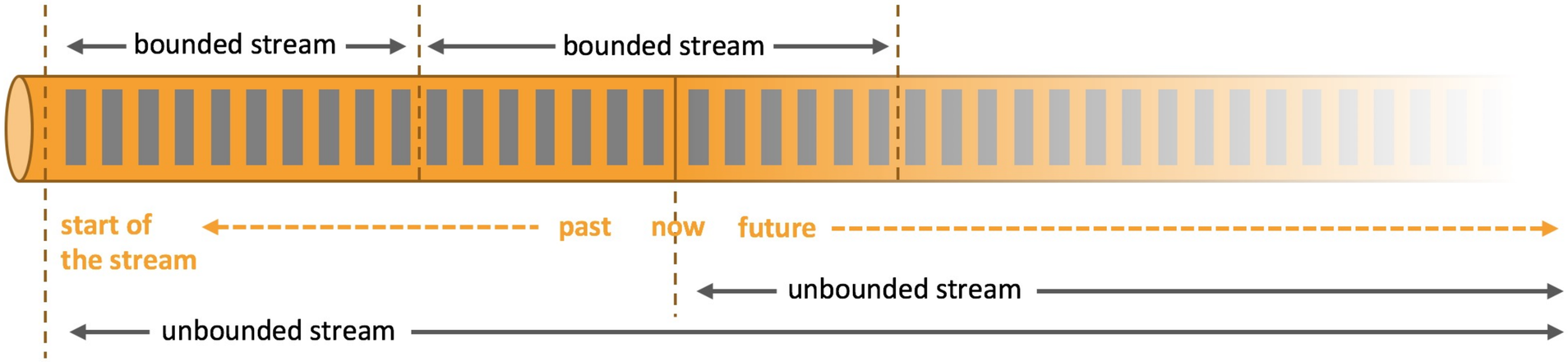
极客时间 训练营

- 如果回到20年前，问你MapReduce以后还做些什么？
- 那么未来的数据系统又该是什么样子呢？
- **方向一：数据集成**
  - 多种方案，技术折中
  - 如果给你一个问题，比如“需要把数据保存起来，可以支持查询”
  - 选择合适的软件组件
  - 流和批的对立和统一



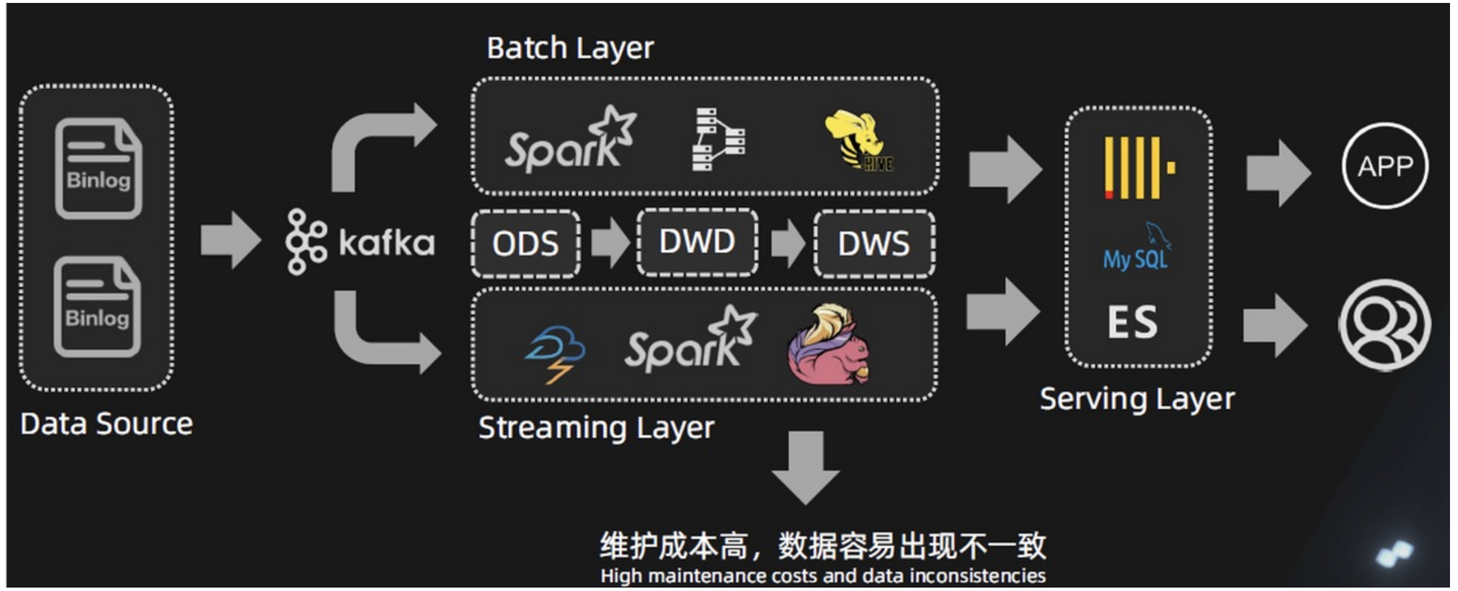
流处理和批处理

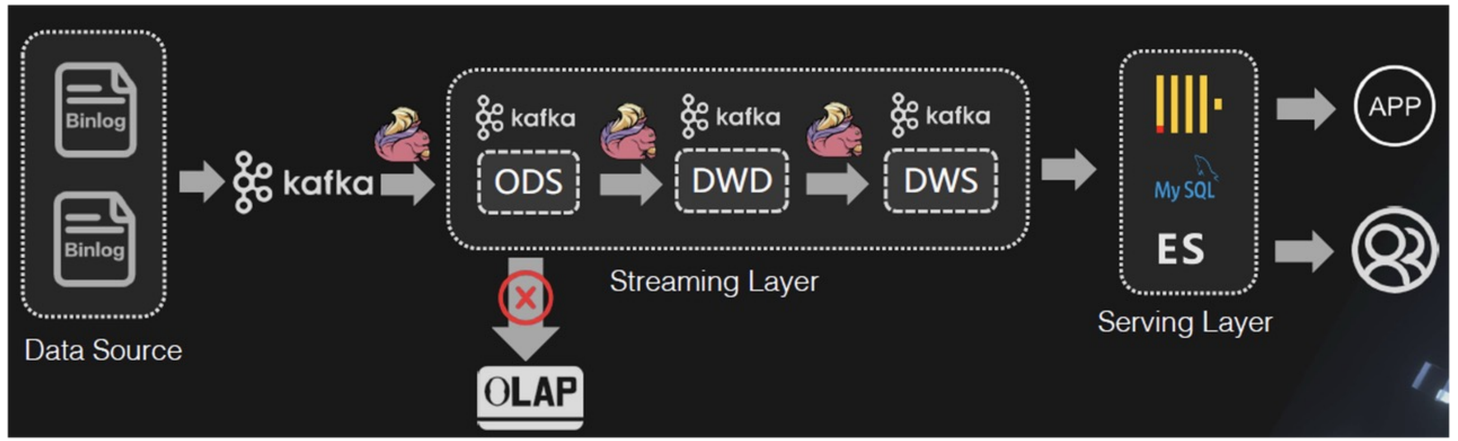
- 无界流：我们不能等到所有数据都到达再处理，因为输入是无限的。数据通常要求以特定顺序摄取。
- 有界流：可以在摄取所有数据后再进行计算。有界流所有数据可以被排序。



Lambda vs Kappa

- Lambda 架构
  - 同时维护实时离线两套计算引擎
  - 用户同时维护两套代码
  - 两套数据链路，造成数据不一致
- Kappa 架构
  - 消息队列回溯能力不及离线
  - 消息队列无法直接提供数据查询和分享的能力
  - 全链路依赖消息队列，数据时序性可能造成结果错误





系统拆分

极客时间 训练营

- **方向二：系统（数据库）拆分**
  - 数据库、 Hadoop 和操作系统
  - 共性提取，抽象出更多的小型系统，所谓分层也是这个意思
  - 比如：专门做 JDBC 的系统，专门做关系代数优化的系统，专门做对象存储的系统，专门做元数据的系统，甚至像缓存，API 等也都有各种系统
  - 技术专业化，正在逐渐主导市场
  - 将数据系统库进行拆分的一个结果就是数据库化：存储层上的实现产生了数据湖，流计算也有 join、filter 等算子

- 当你已经把握了技术的本质，洞悉了技术的方向，请别忘了不要被技术左右。

2.5 不被技术左右

以人为本

极客时间 训练营

- 每个系统都有其构建目的，我们所采取的每一个行动都会产生有意或无意的后果，建立这些系统的工程师有**责任**仔细考虑这些后果。
- 虽然我们的系统运用了大量的抽象和概括，但系统中流淌的数据仍然来自现实世界，仍然是关于人的，记录着人的行为、人的兴趣和身份。
- 我们必须以**人性**和**尊重**来对待这些数据。用户是人，人的尊严是最重要的。
- 预测分析，影响到个人的生活，或是积极的或是消极的。

算法“偏见”

极客时间 训练营

- 算法比我更了解我吗？
- 算法所做出的决定不一定比人类做得更好或更糟。
- 预测分析系统只是基于过去而推断，如果过去是有偏见的，它们就会把这种偏见编码下来。
- 数据和模型只应该是我们的工具，而不是我们的主人。
- 算法也会犯错（无人驾驶）
- 统计学本质：即使概率分布总体上是正确的，个别情况也可能是错的。

开源

- （选择）开源往往是正确的选择
- 开源是有国界的
- 技术是有国界的

3. 大数据领域未来趋势和职业方向

技术方向和职业方向


大数据技术
IoT
实时计算
机器学习
OLAP
数据湖
云原生
大数据方向
业务开发
数仓建模
数据分析
数据科学
平台开发
运维

4. 大厂招聘的通用原则和面试技巧

考评原则

- 不过分解读 JD 和简历
  - JD 中或简历中出现“负责人”“专家”，不过分解读，按照实际面试结果评级。
- 不局限背景
  - 候选人没有直接相关背景并不直接定位不匹配，需额外考察研发能力和软实力。
- 工作年限的解读
  - 对口：具有更长工作经验需考察深度。
  - 不对口：具有更长工作经验需考察能力和基础。
- 技术栈
  - 不要求完成一致，相近即可。

考察内容

极客时间 训练营

- 项目经历
  - 项目是否能讲清楚
  - 是否有深度难度
  - 对项目的原理、背景、未来是否有想法
- 大数据知识
  - 工作年限短的不做要求
  - 一般以通用知识，重点看理解深度和广度
  - 资深候选人需考察技术细节和专业内容
- 计算机基础
  - 并发、网络、Linux 基础
  - 编程语言基础
  - Trouble-shooting 基础

考察内容

极客时间 训练营

- Coding
  - 基本算法是否掌握，不考偏门复杂算法
  - 代码风格
  - 是否主动和面试官沟通理清题意
  - 碰到问题是否主动寻求帮助，是否容易放弃
- 软技能
  - 礼貌准时
  - 沟通顺畅
  - 能理解题意
  - 表达有逻辑
  - 不绕圈子
  - 对不了解的事情给出自己的思考

面试答题技巧

- Situation（情境）
  - 描述遇到的一个挑战、情况或者形势，自己所处的位置及相关的背景信息。
  - 目的：了解候选人取得特定成绩的前提，从而获知所取得的成绩有多少是与候选人的能力素质有关，多少是和大环境、资源、他人有关。
- Task（任务）
  - 在上述情境下如何明确自己的任务/目标。
  - 目的：了解候选人参与任务的具体内容、性质、规模、其中所体现的能力素质与应聘岗位要求的匹配度。
- Action（行动）
  - 做了什么、为什么、如何做，以及是否有替代方案等。
  - 目的：了解候选人是如何完成工作，都采取了哪些行动，所采取的行动是如何帮助他完成工作，通过这些，可以进一步了解他的工作方式、思维方式和行为方式。
- Result（结果）
  - 行动的挑战和阻碍，收获和成效，是否完成目标。
  - 目的：了解候选人在完成任务的过程中遇到的困难和应对方式，候选人的行为对结果的影响，从而了解候选人的自驱力、抗压能力、应变能力、是否结果导向等。

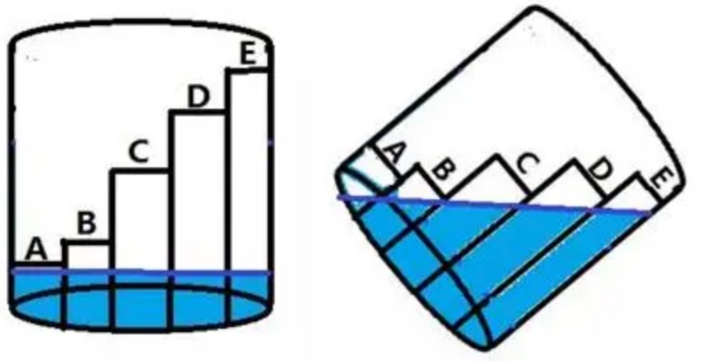
什么样的候选人才是好的候选人

极客时间训练营

- 扎实的技术基础
- 优秀的学习能力和解决问题能力
- 优秀的自驱力及抗压能力
- 良好的沟通表达
- 不错的逻辑思维能力
- 对于未来的规划能力

通过原则

- 反木桶理论



  - 专业技术扎实
  - 编程思路清晰
  - 沟通表达顺畅
  - 有独到见解
  - 展示自己能力

团队合作能力一般怎么考察

请分享一次以往的项目经历中，通过团队合作完成一项任务或目标的经历。

1. 你和其他团队成员之间是如何分工协作的？
2. 你对项目的成功做出了哪些贡献？
3. 其他成员做了哪些贡献？
4. 你认为你或者你团队的其他成员在哪方面做的不够好？
5. 你对其他成员提供过哪些帮助？
6. 他们对你的帮助又有哪些？
7. 如果下次再做类似的项目，你觉得如何可以让你们提升效率？
8. 你怎样评价这次团队合作？

学习能力一般怎么考察

极客时间 训练营

讲讲你最近学习到了哪些比较实用的新知识？

1. 你是通过什么渠道学习的？
2. 哪些应用在了工作中？
3. 你是如何将学习到的东西与工作结合的？
4. 告诉我一个将这些新知识应用在工作中的成功案例？
5. 在应用到工作中的过程中遇到了哪些问题？
6. 你是如何解决的？
7. 业余的时间爱好做些什么？

你在之前的工作或者学习中，遇到过需要做自己不太擅长的事情吗？

1. 你是如何解决的？
2. 过程中有什么收获？

沟通能力一般怎么考察

请讲这样的一个经历：为了完成某项工作，你需要另一个部门提供十分重要的信息；但是另一个部门认为，你的部门收集的信息不是他们的工作重点。

你是怎样解决这个问题的？

你有没有过这样的经历：为了完成某项工作，你需要协调多个部门的同事，其中有一些比你的资历高、经验多。

在这种情况下，你是如何协调并取得他们的认可，完成你的工作的？

请分享一次以往的项目经历中，通过与他人合作完成一项任务或目标的经历（面试官可以在面试过程中通过追问以下问题了解其沟通、合作、影响力等素质能力）。

1. 你在项目中的角色是什么？
2. 你和其他团队成员之间是如何分工协作的？
3. 你认为比较有难度的沟通过程在哪里？为什么觉得有难度？你是怎么做的？
4. 过程中意见不一致，你是怎么做的？
5. 你对项目的成功提供过哪些帮助？他们对你的帮助又有哪些？
6. 你的建议不被老板接收时，你是如何说服他的？

逻辑能力一般怎么考察

极客时间 训练营

- 请告诉我一个你最近在工作过程中遇到的最难的案例，难点在哪里？当时你是怎么做的？为什么这么做？后来发生了什么？取得了什么结果？

- 你有没有给你的同时阐述过技术或者你专业上的难点问题？给我一个具体的例子，告诉我你当时是怎么描述的？

- 用一些逻辑思维问题去考察候选人的思维能力，例如：北京需要多少餐馆？
- 请分享一件你如何分析和解决问题的经历？

积极性一般怎么考察

讲一个你经历过的做的不是很顺利的项目。

1. 你提前预料到会有哪些问题了吗？
2. 你有提醒过项目负责人（可能是自己、也可能是别人）会有这样的风险吗？
3. 问题出现后你做了什么？
4. 调整后会产生什么新的问题吗？
5. 产生这个问题的最根本的原因是什么？
6. 如果再做类似的项目，你会提前考虑哪些问题？

请分享一件以往做的项目/工作，期间发生了重大的变化，你是如何处理的？

推动能力一般怎么考察

分享一个你独立负责的项目情况

1. 你的项目目标是什么？如何拆解目标的？项目计划怎么制定的？
2. 项目中涉及哪些人？哪些资源？
3. 你通过什么渠道获取了相应的资源？有没有遇到阻力？怎么解决的？
4. 过程是否顺利，过程中有没有遇到一些意想不到的困难？怎么解决的？
5. 遇到不配合的人你是怎么处理的？

项目进行过程中有没有遇到时间比预想的延后的情况？如何处理的？最终项目如期交付了吗？

5. 笔试/算法/编程考察意图和技巧

算法笔试的结果重要吗？

极客时间训练营

- 是代码能力，也不是代码能力。
- 算法笔试没有一票否决权。
- 写出来可能不加分，写不出来可能不减分。

大厂爱出什么编程题

- DFS/BFS/backtrack
- 动态规划
- 快排堆排
- 链表翻转/二叉树翻转

原则：

1. 递归的掌握
2. 写代码的过程
3. 边界条件
4. 尽量少的代码

我该怎么准备编程题

极客时间 训练营

- leetcode 要刷
- 中等难度典型题目每类5题
- 刷20～30道即可，在精不在广
- 不可能遇到原题

写不出来怎么办

极客时间 训练营

- 不留白
- 写结构
- 问思路
- 写思路
- 写注释
- 要提示
- 伪代码
- “装”认真

6. 职业发展 Q & A

转方向

问题清单：

- 三十多岁的 Java 开发想转行大数据还来得及么，会承受降薪的压力么？
- 无大数据和 Java 经验转大数据方向，组件要学的太多了找不到重点，重点学习和准备面试的有哪些？
- 没有实际的大数据项目经验怎么办？

如何学习大数据技术

问题清单：

- 偏向于平台开发要如何学好这些琐碎的知识点，比如多线程、网络通信，以及金老师之前所讲的 NIO 编程模型等等；以及在分布式领域的编程规范是什么，有没有参考。
- 大数据开发工程师 roadmap。
- 如何学习开源框架，怎么去参与到开源，一个框架中很多个 model 如何下手，金老师可以从 Hadoop 或 Spark 等给大家一些思路。

职业道路

极客时间训练营

问题清单：

- 百来人的小公司，30岁了，是否该往管理走了？有感受到现在的团队不大重视技术，跳槽也跳不到大厂，感觉自己的技术还未到自己学不动的时候，但现在公司原因，想往管理岗走了，纠结、大脑分裂。
- 俩种大数据职业发展路线：一、偏平台的架构师，对开源技术要求较高，企业一般会要求读过开源技术源码，或者参与过开源项目，偏平台的构建；二、偏数仓的架构师，对 SQL 能力要求较高，企业一般会要求掌握数仓理论，有数仓项目经验。本心想要走偏平台的架构师的架构师，因为之前一直敲代码，不想变为 sql baby，但是面临俩个问题：1.市场上大部分职位偏数仓; 2.自学的项目为离线数仓。自己是否应该先妥协从事数仓相关职位，再看未来发展确定是走偏平台的架构师还是偏数仓的架构师? 烦请老师帮忙解惑。
- 大数据开发与数据仓库有啥区别，或者是说大数据下有哪些方向，如果想往技术流的方向走，不想过多 focus 业务层面的内容，老师建议选大数据下的哪一个方向？

面试

问题清单：

- 外企招聘时间也是集中在9、10月吗？与国内大厂面试注重点有什么不一样？（就大数据岗位而言）

其他

问题清单：

- 与 Flink 相比，现在 Spark 的优势在哪？未来实时数仓技术成熟，Spark 会不会被取代？
- 今年好像整个行业都不好，锁 hc，裁员等。请问老师对于今年跳槽的，在公司选择上有什么好的建议吗？

祝大家都能顺利通过面试，

拿到心仪的 Offer！

THANKS

极客时间 | 训练营