

FAQ第一期 | 学习大规模数据处理需要什么基础？

2019-05-20 蔡元楠 来自北京

《大规模数据处理实战》



你好，我是蔡元楠。

专栏上线已经一个月了，在这里我要先感谢大家的留言，留言的对答可以使我们互有补益。

这段时间，我发现留言中的很多问题都很有价值，希望你也可以看到。所以，我根据已发布的文章中的思考题，从留言中摘录了一些典型的、常见的问题做出答疑集锦，最终成为了今天你看到的“特别福利篇”。

“🔖开篇词”问题精选

问题一：学习大规模数据处理需要什么基础？



风之伤

学习这专栏需要什么基础知识



hua168

老师，学习这个需要什么知识为提前？



As Sunshine

这个课程对学员的要求有没有什么限制



KingSwim

需要哪些先修知识？

这是一个很好的问题，虽然专栏已经更新了一个月，我还是要把这个开篇词中的提问放进来。就像你看到的那样，有好几位读者都问了类似的问题。

其实在最开始做专栏的内容设计时，我并没有对读者的知识背景作任何假设。

所以，即使是一些基础的技术概念，我也会举例解释一下（如果你已经会了可能会觉得啰嗦，这时候就需要你照顾一下其他同学了）。如果你有一些语言的编程经验（任何语言都可以）的话，看文章的理解速度会快一点。文章中会有一些示例代码，是用 Python 编写的。

但是在设计类型的案例中，我不觉得它对读者有特别的技术要求。

希望你在后面的阅读中提出建议，告诉我有哪里地方我讲得不够清楚，或者解释的过多，我会适当调整内容。

问题二：小型公司程序员学习大规模数据处理的意义？



韩程

老师你好，你上文提到的 AI 落地的基础是大规模的数据和高质量的标注，目前能满足的这个条件是否只有一些超大规模的一线互联网公司。那是否意味着大数据处理也只有在这些公司中才能发挥真正的价值，那对于在小型互联网公司工作的程序员，学习大数据处理的意义在哪里呢？

这个问题问得很好。以客观条件来看，韩程的说法没有问题。

大规模的互联网公司天生数据量是要大一些的。但是，这并不意味着大数据处理只在大公司才能发挥价值。你也要考虑其他方面。

第一，对于公司来讲，小型互联网公司或者传统企业，并不是不需要数据处理技能，而是他们还没有从数据中挖掘 business insight 的意识，没有数据驱动决策的意识，甚至没有收集数据的意识。

举个我工作中见到的例子。比如，有些饲养奶牛的农户，他们几十年来根本不知道什么是数据。但是，当我们帮他们细致地搜集奶牛每天的活动数据，比如饮食、运动、作息、产奶，他们就能从中找到最经济（最优）的饲料投放方式。

第二，对于个人来讲，你就一定要看长期的职业发展，公司会从小变大，职位会从低变高。当你需要影响决策的时候，当你面临的数据量变多的时候，当你准备跳槽的时候，数据的处理能力都是至关重要的。

“第一讲”问题精选

思考题：如果你在 Facebook 负责处理用户数据，你会选择什么样的分片函数来保证均匀分布的数据分片？

我发现有很多精彩的回答。比如下图中的 CountingStars 同学，他的思路非常有意思。是把年龄的数值前后颠倒进行分片。



_CountingStars

把年龄倒过来比如 28 岁 变成 82 来分片

还有这位 Mark Lee，他认为可以使用身份证后面的随机数来进行分片，纯技术上看起来似乎可行。但要使用用户的身份 ID 的话，你还需要考虑是否符合法律、道德、隐私方面的问题。



Mark Lee

把身份证后面的随机数截取到前面，或者有规律的加 3 到 4 位的随机数

而 Freud 的想法是引用随机标记来保证数据分片的随机性。但这里要保证数据的均匀可重复才行。如果你在 shard2 上的任务失败，你需要能够还原出错的任务并进行重试。



Freud

元楠老师，关于思考我认为引用随机标记的思想，第一次 map 输入时，在 key 上拼接随机数，经过第一个 mrjob 的处理后，再将标记去掉，这样可以大大减小数据倾斜。可以基本保证第一次数据分片的随机性。这个思想是我在 Hive 优化中学到的，期待您的指导意见

摇山樵客把这几个回答可能出现的问题做了个总结。他的回复是一切有效降低十位数权重的哈希算法都是可行的。



摇山樵客™

年龄是值域在 0-120（假定）之间的数值，难以分片的原因正是因为年龄的十位数权重过大，所以我觉得一切有效降低十位数权重的哈希算法应该都是可行的。

1. 对于年龄 ABC，比如倒置 CBA，或 $(C * \text{大质数} + B * \text{较小质数} + C) \% \text{numPartitions}$ ，这类方法应该可以明显改善分布不均，但是对某些单一热点无解，比如 25 岁用户特别多；

2. 随机分区，可做到很好均衡，对 combine，io 等优化不友好

3. 先采样 + 动态合并和拆分，实现过于复杂，效果可能不稳定

这是我的想法，请老师指正。

倒置年龄可以明显改善分布不均的问题，但是也可能对某些单一热点无解，比如 25 岁的用户特别多的话还是会出问题。

随机分区可以做到均衡，但对 combine、io 等优化不够友好。还有一个缺点，是当分区任务失败，需要重新分区的时候，分区结果不再是 deterministic 的。如果某一台机器宕机了，你要如何重新分配原本属于这台机器上的用户数据？

先采样，再动态合并和拆分的实现过于复杂，效果可能不够稳定。

像他一样，在每个答案里都分别给出这个答案所存在的不足，这一点是我非常赞赏的。在开发设计中没有哪个答案是特别完美的，我们能做的是分析哪一个才是最符合自身应用需求，进而改善。

“第二讲”问题精选

第二讲中，我留下的思考题是“你现在正在使用的数据处理技术有什么问题？你有怎样的改进设计？”。

mjl 在回答中阐述了他比较了解的 Spark 和 Flink，总结得很好。



mjl

Unify platform 和批流统一已经是主要趋势了，而我个人目前只对 spark、flink 有一定的了解。对于 spark 来说，无疑是很优秀的一个引擎，包括它的 all in one 的组件栈，structured streaming 出来后的批流 api 的统一，目前在做

的 continues Mode。而 flink，的确因为阿里的运营，在国内火了。但也展现了它的独有优势，更加贴近 dataflow model 的思想。同时，基于社区以及阿里、华为小伙伴的努力，flink 的 table/sql 的 api 也得到了很大的增强，提供了流批统一的 api。虽然底层然后需要分化为 dataset 和 datastream 以及 runtime 层的 batchTask 和 StreamTask，但是现在也在 rethink the stack，这个 point 在 2019 SF 的大会也几乎吸引了所有人。但就现状而言，flink 的确有着理念上的优势（流是批的超集），同时也有迅猛上升的趋势。

虽然原生 Spark Streaming Model 和 Dataflow Model 不一样，但是 Cloudera Labs 也有根据 Dataflow Model 的原理实现了 Spark Dataflow，使得 Beam 也可以跑 Spark runner。

而对于 Flink 来说的话，在 0.10 版本以后，它的 DataStream API 就已经是根据 Dataflow Model 的思想来重写了。

现在 Flink 也支持两套 API，分别是 DataStream 版本的和 Beam 版本的。其实 data Artisans 一直都有和 Google 保持交流，希望未来两套 Beam 和 Flink 的 API 能达到统一。

最后赞一点，批处理是流处理的子集，这个观点我在第一讲的留言中也提到过。

🔗 [第三讲](#)和🔗 [第四讲](#)中问题较为开放，与读者自身的工作内容强相关，很多都是大家在分享自己的经验，内容很丰富，这里篇幅不足，建议大家去原文的留言中看一看。

“🔗 [第五讲](#)”问题精选

第五讲中讲的主要是分布式处理系统的三个重要指标：扩展性，一致性和持久性。根据这个内容，3SKarl 同学提问弱一致性和最终一致性的区别是什么。



3SKarl

老师，我不太明白弱一致性和最终一致性的区别在哪里，文章对弱一致性提到

“经过不一致时间窗口这段时间后，后续对该数据的读取都是更新后的值”

这句描述的不就是最终一致性么

是不是说弱一致性允许数据始终不一致，而要求最终结果一致的弱一致性叫做最终一致性

这是个很棒的问题。简而言之，弱一致性是个很宽泛的概念，它是区别于强一致性而定义的。广义上讲，任何不是强一致的，而又有某种同步性的分布式系统，我们都可以说它是弱一致的。

而最终一致性是弱一致性的一个特例，而且是最常被各种分布式系统用到的一个特例。

其他的比如因果一致性、FIFO 一致性等都可以看作是弱一致性的特例，不同弱一致性只是对数据不同步的容忍程度不同，但是经过一段时间，所有节点的数据都要求要一致。

学习专栏时，重要的是理解它们的区别。这部分知识是为了后边讲 CAP 理论服务的，实际的工作中也不会像考试考概念题一样，让你背写这些一致性的定义。



hua168

老师我想问一下，所谓的强一致性，它允许的误差是多少范围？

比如金融股票大厅显示屏数据，应该是强一致性的吧，全球显示误差不会超过 0.2s 吧？

这样强一致性怎达到？网络传输，路由处理都不止了，更何况还有网络延迟

hua168 同学问的是强一致性的误差范围。这个问题非常有趣，强一致性并没有误差可言的，强一致性简单地说指的就是如果更新一条数据，那所有用户读取数据的时候必须都看到这条更新了的数据。

在这里我也想借着 FAQ 分享一个自己当年在面试 Bloomberg 的面试经历。

面试官给我出的题目是这样的：如果要设计 Bloomberg 的股票信息系统中的数据库系统，系统需要实时更新股票价格，而数据更新的写入量非常大，用户也需要读取最新的股票资讯，你会如何设计这套系统。

这个问题其实有很多的未知区域需要我们去和面试官去阐明，例如用户的 Use Cases 是什么？在此我就不一一展开了，在这里我只想分享一个和一致性相关的内容。

在和 Bloomberg 的 Tech Lead 讨论时我发现，原来他们的股票系统显示的股价并不是强一致性的，延迟范围是 1 分钟左右。

因为应用场景上，普通股民并不会需要实时关心每秒钟股票价格的动态，更多的是关心大盘走势。而金融巨头在操作股票的时候，更多只关心特定的几只股票，所以这些股票的价格通常对于他们来说会更新快一点。

所以说，很多现实生活上的实际应用和我们本来想象的并不太一样。

到这里，我们的第一期答疑就结束了。

就像我在专栏一开始的时候与你说的一样，我希望你能够积极与我互动。其实很多同样的问题会在不同的人身上重复出现，你不表达出来的话，可能永远也不知道，原来有那么多人曾经和你遇到过同样的困境。

如果你觉得有所收获，欢迎你把文章分享给你的朋友。

精选留言 (6)



HomeyLiu

2019-05-20

数据均匀分片的核心是 哈希函数的设计。

如果你数据结构和算法不错的话，我觉得这是一个很简单的问题。

通过hashFunchiton (key) 函数，输入key，输出hash值。

哈希函数设计的特点：

1》输入的key一样，得到的hash值肯定一样

2》输入的key不一样，得到的hash值可能一样，也就是hash冲突。

这个是评判一个哈希函数的好坏的重要标准。

冲突概率大的哈希函数肯定会引起严重的数据倾斜。极端的例子，所有的key的hash值都一样，都跑到一个桶里面去了。

所以衡量一个哈希函数的好坏：

1》冲突要小。（例如用素数，还有模拟10进制，弄个26进制，abc可以编码为 $0 \times 26^0 + 1 \times 26^1 + 2 \times 26^2$ ）

2》计算要快。常用位运算。

3》key哪怕很小的变动，输出的hash值差距越大越好。

有很多很经典的hash算法。

但是如果key一样hash值肯定一样。

所有key重复的数据很多的话，哈希函数是解决不了问题的。

必须对key进行组合，只要 组合后的key的重复的比率 不要比 哈希冲突的概率 大太多就行。



10



朱同学

2019-05-21

刚入行时，师傅曾指导我，hash可以做随机，但是不能做key，因为不同平台hash算法可能是不一样的，类似需求推荐使用md5。



4



2019-05-20

老师，请以你专家级的视角，推荐5篇将来可能影响大数据发展趋势的论文吧！

作者回复：谢谢你的留言！可以参考我在第22讲里所讲到的论文呀。



👍 3



sunsweet

2019-05-23

但是比特币交易平台就是实时的，那是怎么实现呢



👍 2



时间是最真的答案

2019-05-22

感觉不是做大数据领域的同学，读这个专栏还是比较吃力的。专栏设计知识的很广，提升了大家的认识，但不懂大数据相关技术，没法实践，比如spark不懂如何部署，然后用自己所熟悉的需要去实践



👍 2



listen

2019-05-21

老师你好，我们是做学生学习情况的，现在要做实时，就是一节课的信息，是一个大json，1-10+M，其中嵌套多个json，由于各个子json的耦合性太强没办法分离，使用kafka的话一条数据太大了，数据是在OSS上，现在是先拉取到hdfs，现在是发现3中方法，

- 1、java put到hdfs时，mq发送位置信息，sparkstreaming订阅，根据位置拉取
 - 2、put 到hbase,sparkstreaming 扫描
 - 3、使用sparkstreaming的textFileStream算子监控路径
- 三种方法没种都有很大的缺陷，老师能指点一下吗

共 1 条评论 >

