

12 | 我们并没有觉得MapReduce速度慢，直到Spark出现

2018-11-24 李智慧

从0开始学大数据

[进入课程 >](#)



讲述：李智慧

时长 11:25 大小 5.24M



Hadoop MapReduce 虽然已经可以满足大数据的应用场景，但是其执行速度和编程复杂度并不让人们满意。于是 UC Berkeley 的 AMP Lab 推出的 Spark 应运而生，Spark 拥有更快的执行速度和更友好的编程接口，在推出后短短两年就迅速抢占 MapReduce 的市场份额，成为主流的大数据计算框架。

读到这里请你先停一下，请给这段看似“没毛病”的引子找找问题。

不知道你意识到没有，我在这段开头说的，“Hadoop MapReduce 虽然已经可以满足大数据的应用场景，但是其执行速度和编程复杂度并不让人们满意”，这句话其实是错误的。这样说好像可以让你更加清晰地看到事物发展的因果关系，同时也可以暗示别人自己有洞察事物发展规律的能力。然而，这种靠事后分析的因果规律常常是错误的，**往往把结果当作了原因。**

事实上，在 Spark 出现之前，我们并没有对 MapReduce 的执行速度不满，我们觉得大数据嘛、分布式计算嘛，这样的速度也还可以啦。至于编程复杂度也是一样，一方面 Hive、Mahout 这些工具将常用的 MapReduce 编程封装起来了；另一方面，MapReduce 已经将分布式编程极大地简化了，当时人们并没有太多不满。

真实的情况是，人们在 Spark 出现之后，才开始对 MapReduce 不满。原来大数据计算速度可以快这么多，编程也可以更简单。而且 Spark 支持 Yarn 和 HDFS，公司迁移到 Spark 上的成本很小，于是很快，越来越多的公司用 Spark 代替 MapReduce。也就是说，因为有了 Spark，才对 MapReduce 不满；而不是对 MapReduce 不满，所以诞生了 Spark。真实的因果关系是相反的。

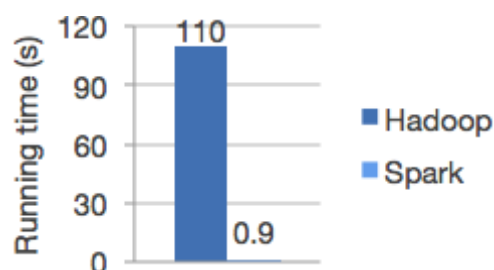
这里有一条关于问题的定律分享给你：**我们常常意识不到问题的存在，直到有人解决了这些问题。**

当你去询问人们有什么问题需要解决，有什么需求需要被满足的时候，他们往往自己也不知道自己想要什么，常常言不由衷。但是如果你真正解决了他们的问题，他们就会恍然大悟：啊，这才是我真正想要的，以前那些统统都是“垃圾”，我早就想要这样的东西（功能）了。


所以顶尖的产品大师（问题解决专家），并不会拿着个小本本四处去做需求调研，问人们想要什么。而是在旁边默默观察人们是如何使用产品（解决问题）的，然后思考更好的产品体验（解决问题的办法）是什么。最后当他拿出新的产品设计（解决方案）的时候，人们就会视他为知己：你最懂我的需求（我最懂你的设计）。

乔布斯是这样的大师，Spark 的作者马铁也是这样的专家。

说了那么多，我们回到 Spark。Spark 和 MapReduce 相比，有更快的执行速度。下图是 Spark 和 MapReduce 进行逻辑回归机器学习的性能比较，Spark 比 MapReduce 快 100 多倍。



除了速度更快，Spark 和 MapReduce 相比，还有更简单易用的编程模型。使用 Scala 语言在 Spark 上编写 WordCount 程序，主要代码只需要三行。

 复制代码

```
1 val textFile = sc.textFile("hdfs://...")
2 val counts = textFile.flatMap(line => line.split(" "))
3                       .map(word => (word, 1))
4                       .reduceByKey(_ + _)
5 counts.saveAsTextFile("hdfs://...")
```

不熟悉 Scala 语言没关系，我来解释一下上面的代码。

第 1 行代码：根据 HDFS 路径生成一个输入数据 RDD。

第 2 行代码：在输入数据 RDD 上执行 3 个操作，得到一个新的 RDD。

将输入数据的每一行文本用空格拆分成单词。

将每个单词进行转换，`word => (word, 1)`，生成 `<Key, Value>` 的结构。

相同的 Key 进行统计，统计方式是对 Value 求和，`(_ + _)`。

第 3 行代码：将这个 RDD 保存到 HDFS。


RDD 是 Spark 的核心概念，是弹性数据集（Resilient Distributed Datasets）的缩写。RDD 既是 Spark 面向开发者的编程模型，又是 Spark 自身架构的核心元素。

我们先来看看作为 Spark 编程模型的 RDD。我们知道，大数据计算就是在大规模的数据集上进行一系列的数据计算处理。MapReduce 针对输入数据，将计算过程分为两个阶段，一个 Map 阶段，一个 Reduce 阶段，可以理解成是**面向过程的大数据计算**。我们在用 MapReduce 编程的时候，思考的是，如何将计算逻辑用 Map 和 Reduce 两个阶段实现，map 和 reduce 函数的输入和输出是什么，这也是我们在学习 MapReduce 编程的时候一再强调的。

而 Spark 则直接针对数据进行编程，将大规模数据集合抽象成一个 RDD 对象，然后在这个 RDD 上进行各种计算处理，得到一个新的 RDD，继续计算处理，直到得到最后的结果

数据。所以 Spark 可以理解成是**面向对象的大数据计算**。我们在进行 Spark 编程的时候，思考的是一个 RDD 对象需要经过什么样的操作，转换成另一个 RDD 对象，思考的重心和落脚点都在 RDD 上。

所以在上面 WordCount 的代码示例里，第 2 行代码实际上进行了 3 次 RDD 转换，每次转换都得到一个新的 RDD，因为新的 RDD 可以继续调用 RDD 的转换函数，所以连续写成一行代码。事实上，可以分成 3 行。

 复制代码

```
1 val rdd1 = textFile.flatMap(line => line.split(" "))
2 val rdd2 = rdd1.map(word => (word, 1))
3 val rdd3 = rdd2.reduceByKey(_ + _)
```

RDD 上定义的函数分两种，一种是转换（transformation）函数，这种函数的返回值还是 RDD；另一种是执行（action）函数，这种函数不再返回 RDD。

RDD 定义了很多转换操作函数，比如有计算**map**(func)、过滤**filter**(func)、合并数据集**union**(otherDataset)、根据 Key 聚合**reduceByKey**(func, [numPartitions])、连接数据集**join**(otherDataset, [numPartitions])、分组**groupByKey**([numPartitions]) 等十几个函数。

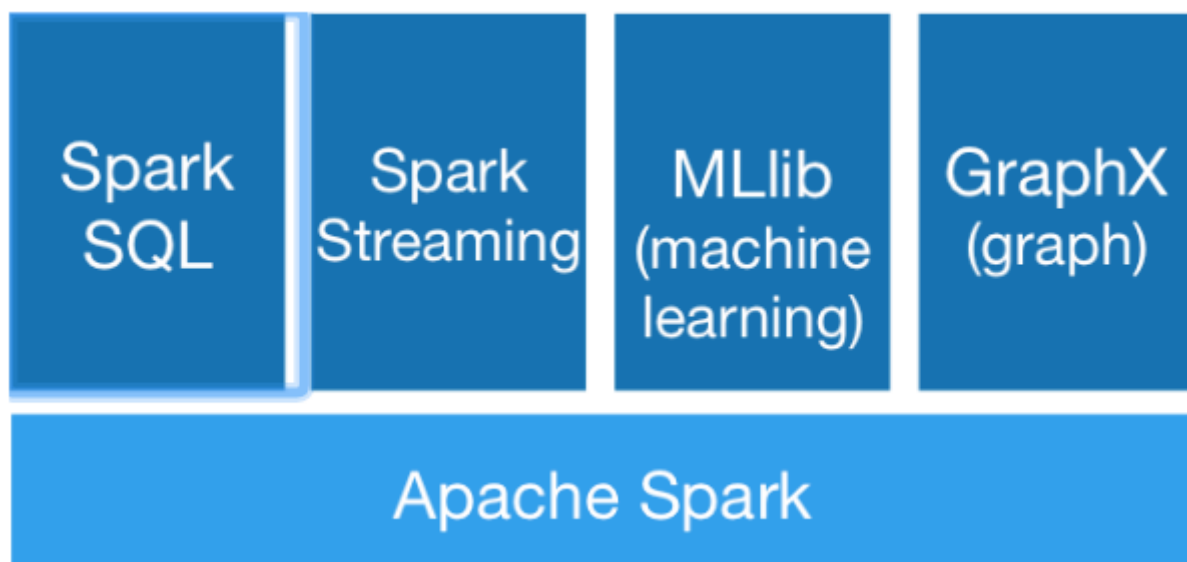
我们再来看看作为 Spark 架构核心元素的 RDD。跟 MapReduce 一样，Spark 也是对大数据进行分片计算，Spark 分布式计算的数据分片、任务调度都是以 RDD 为单位展开的，每个 RDD 分片都会分配到一个执行进程去处理。

RDD 上的转换操作又分成两种，一种转换操作产生的 RDD 不会出现新的分片，比如 map、filter 等，也就是说一个 RDD 数据分片，经过 map 或者 filter 转换操作后，结果还在当前分片。就像你用 map 函数对每个数据加 1，得到的还是这样一组数据，只是值不同。实际上，Spark 并不是按照代码写的操作顺序去生成 RDD，比如 `rdd2 = rdd1.map(func)` 这样的代码并不会在物理上生成一个新的 RDD。物理上，Spark 只有在产生新的 RDD 分片时候，才会真的生成一个 RDD，Spark 的这种特性也被称作**惰性计算**。

另一种转换操作产生的 RDD 则会产生新的分片，比如 `reduceByKey`，来自不同分片的相同 Key 必须聚合在一起进行操作，这样就会产生新的 RDD 分片。实际执行过程中，是否会产生新的 RDD 分片，并不是根据转换函数名就能判断出来的，具体我们下一期再讨论。

总之，你需要记住，Spark 应用程序代码中的 RDD 和 Spark 执行过程中生成的物理 RDD 不是——对应的，RDD 在 Spark 里面是一个非常灵活的概念，同时又非常重要，需要认真理解。

当然 Spark 也有自己的生态体系，以 Spark 为基础，有支持 SQL 语句的 Spark SQL，有支持流计算的 Spark Streaming，有支持机器学习的 MLlib，还有支持图计算的 GraphX。利用这些产品，Spark 技术栈支撑起大数据分析、大数据机器学习等各种大数据应用场景。



我前面提到，顶尖的产品设计大师和问题解决专家，不会去询问人们想要什么，而是分析和观察人们的做事方式，从而思考到更好的产品设计和问题解决方案。

但是这种技巧需要深邃的观察力和洞察力，如果没有深度的思考，做出的东西就会沦为异想天开和自以为是。要知道大众提出的需求虽然也无法触及问题的核心，但是好歹是有共识的，大家都能接受，按这种需求做出的东西虽然平庸，但是不至于令人厌恶。

而缺乏洞见的自以为是则会违反常识，让其他人本能产生排斥感，进而产生对立情绪。这种情绪之下，设计没有了进一步改进的基础，最后往往成为悲剧。这两年在所谓互联网思维的鼓吹下，一些缺乏专业技能的人，天马行空创造需求，受到质疑后公开批评用户，也是让人倍感惊诧。

我们在自己的工作中，作为一个不是顶尖大师的产品经理或工程师，如何做到既不自以为是，又能逐渐摆脱平庸，进而慢慢向大师的方向靠近呢？

有个技巧可以在工作中慢慢练习：**不要直接提出你的问题和方案**，不要直接说“你的需求是什么？”“我这里有个方案你看一下”。

直向曲中求，对于复杂的问题，越是直截了当越是得不到答案。迂回曲折地提出问题，一起思考问题背后的规律，才能逐渐发现问题的本质。通过这种方式，既能达成共识，不会有违常识，又可能产生洞见，使产品和方案呈现闪光点。

你觉得前一个版本最有意思（最有价值）的功能是什么？

你觉得我们这个版本应该优先关注哪个方面？

你觉得为什么有些用户在下单以后没有支付？

思考题

你在工作、生活中通过提问发现问题背后的本质、现象背后的规律的例子有哪些？或者你观察到同事、朋友这样的例子有哪些？

欢迎你写下自己的思考或疑问，与我和其他同学一起讨论。



从 0 开始学大数据

智能时代你的大数据第一课

李智慧

同程艺龙交通首席架构师
前 Intel 大数据架构师



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

上一篇 11 | Hive是如何让MapReduce实现SQL操作的？

下一篇 13 | 同样的本质，为何Spark可以更高效？

精选留言 (44)

写留言



光脚的sun

2018-11-24

36

老师，现在都说flink更优秀，是不是我们直接跳过mr和spark直接学flink就行了？

作者回复: 如果从MapReduce, spark, flink一路学过来，就会发现他们一脉相承的，如果理解了前面的，学习后面的很多地方甚至可以秒懂。

登山的时候，老司机们都走Z字形，有时候最长的路是最容易的路。



三木子

2018-11-24

17

比如学习机器学习，可能有很多人和我有同感，基本上是从入门到放弃。我自己也思考了原因。主要是恐惧心态，因为数学差，恐惧那些数学公式，而现在又崇尚几十天学会xxx，这会让人更加焦虑，更不能静下心来学习。所以我认为解决问题主要根本也就是调整心态，想象学数学公式就像谈恋爱，从陌生到熟悉，再到走入婚姻的殿堂，不是一蹴而就，罗马不是一天建成的。所以公式一遍看不懂就看两遍，三遍，刻意练习，逃离舒适区。念念...
展开



老男孩

2018-11-25

16

惭愧，我遇到的产品经理或者需求人员，基本上分为两类。一类经常说，这是客户的要求必须马上改，用客户来压制研发。一类比较以自我为中心，把自己的观点等同于用户的观点。常常想当然，结果用户一看不是我想要的。结果就是开发人员一次次的从坑里刚爬上来，又被产品一脚踹下去。有几次我真的无法克制，有一种想套麻袋然后一顿打的冲动。
非常赞同老师的观点，不管解决技术问题，还是设计产品都需要深刻的洞察力。想起...

展开 ▾

作者回复: ㊦

◀ ▶



小美

2018-11-25

👍 15

spark优势在于迭代式的内存运算，适合于做大数据分析，机器学习之类的，flink是流式计算框架，对于实时性任务也许更好，对于机器学习内任务，spark还是要好点

展开 ▾

作者回复: ㊦

◀ ▶



无心乐乐

2018-11-24

👍 13

第一次在极客写留言，感觉这个专栏的老师真的是用心良苦，不只是技术的教授，更是经验的传递。

我们常常意识不到问题的存在，直到有人解决了这些问题。

醍醐灌顶，受益良多！！

作者回复: 谢谢鼓励，共勉~

◀ ▶



Lambda

2018-11-26

👍 8

看了一些留言，感觉大家还是“面向工具”学习，对层出不穷的“工具”，感到困惑。但是归根结底，这些工具本身还是计算机科学中很多基础概念的具象化，因此，“面向思想”学习应该是更好的一种做法。先对一种最原始的实现透彻的研究，理解其背后的思想和设计理念，然后再逐步学习后期更为先进的技术，这种学习路径应该更为有效。

展开 ▾

作者回复: 赞

◀ ▶



hua168

2018-11-24

👍 5

看了这篇文章，这个价格，值了！很值！！超值得！！！不到教我们技术还教我们思想！大神真的很负责👍...

Spark可以代替mapReduce那我理解，是不是可以直接跳过mapReduce直接学习Spark？如果有一个东西比Spark更好，我也直接跳过Spark？

我先把您的大数据学完，再补学java大数据编程，可以吧？...

展开 ▾



Dream.

2018-11-26

👍 4

乔布斯还在世的时候就说过用户永远不知道他们想要的是什么，只有我们做出来了，他们才发现这就是他们想要的。

这几年来一直将这句话当做自己需求分析时的座右铭。

但一直不能领会其中的精髓，今天看完老师的专栏，有种拨开云雾的感觉。

其实我们大多时候把结果当成了原因，并且缺乏了深度的思考以及洞察力。...

展开 ▾



小辉辉

2018-11-25

👍 4

好的提问艺术更能帮助我们深入理解和思考问题

展开 ▾



呆猫

2018-11-26

👍 3

我要把这篇文章推荐给我们的产品经理去好好读读

展开 ▾



hunterlodg...

2018-11-26

👍 2

工作中，一个新的方案出现的时候，如果它在某个或某些方面优于当前最好方案，我一定会去思考它的catch(另一面)是什么？比如新方案更快，我就大概会看看它的空间使用率、可维护度、全面度。一般都会发现一些问题。生活里也是如此，对表面上只有好处而无需付出或者代价很低的东西永远保持警惕。说白了，世上没有免费的午餐，一些都是权衡利弊的结果

展开 ▾



张闯

2019-04-04

👍 1

理解了java8的lambda集合操作相比于传统集合操作的优势，就理解了spark相比于MapReduce的优势。

第一步，将集合对象封装成流式对象。

第二部，将函数传递给流式对象，在流式对象中执行内部循环。

spark之所以快，就是将外部循环替换成了内部循环。...

展开 ∨



黄吉吉

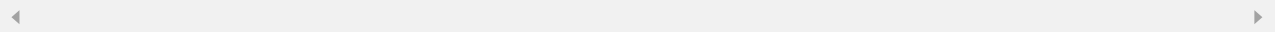
2018-11-25

👍 1

老师有开其他博客、公众号嘛？想听老师更多的分享

展开 ∨

作者回复: 谢谢支持，可以看看我写的书《大型网站技术架构：核心原理与案例分析》



Abyte

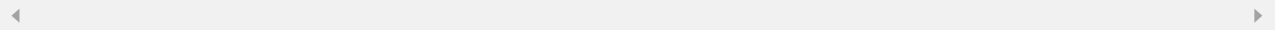
2018-11-25

👍 1

大数据开发处理数据过程中难免被领导提一些需求，做各种各样的报表统计。我们第一手接触数据，如果我们能再有精力投入业务，是不是也能主动做出一些老板需要的报表统计，提升自己的价值

展开 ∨

作者回复: 📖



REAL_MADIR...

2018-11-24

👍 1

养成穿透问题现象直达问题本质的能力真的很重要，老师有机会可以多给我分享下如何培养这种能力，或者学习编程过程中的一些方法论，我觉得有时这比学习本身更重要

展开 ∨



吴科

2018-11-24

1

老师，讲讲基于spark的数据分析和机器学习吧

展开



随缘_NJ

2018-11-24

1

发现用户的需求，用户永远不清楚想要什么，总结的真好

展开



张闯

2019-04-04

1

初始状态：

每个DataNode上存有全部表的分片数据，但每个表在这个DataNode上的分片区域是随机不相关的。

最终状态：

对所有表的数据进行了整合计算处理，处理结果分片存储在不同的DataNode上。...

展开

作者回复: 结果1和表2按照key进行partition，shuffle后相同的key在同一个节点上



记忆犹存

2019-03-30

1

所以顶尖的产品大师（问题解决专家），并不会拿着个小本本四处去...

受教了~

展开



盖饭

2019-03-19

1

这篇更多产品篇的。自己作为产品理解一下：

用户本身诉求的需求或痛点，是用户本身根据结果而产生的有意识的信息。这些信息更多是共性的、表面性的，正如本文作者所说，解决了这行表面、共性的问题，虽然平庸，但不会令人厌恶。

作者所主张的要通过观察和思考用户对问题的解决过程、行为或状态，更有助于直击问...
展开 ∨