

31 | 大数据架构：大数据技术架构的思想和原理是什么？

2020-02-03 李智慧

后端技术面试38讲

[进入课程 >](#)



讲述：李智慧

时长 12:54 大小 10.35M



我在开篇词讲到，任何新技术都不是凭空产生的，都是在既有技术的基础之上，进行了一些创新性的组合扩展，应用到一些合适的场景之中，然后爆发出来巨大的生产力。后面几篇我要讲的大数据技术，区块链技术都是如此。

大数据技术其实是分布式技术在数据处理领域的创新性应用，本质和我们此前讲到的分布式技术思路一脉相承：用更多的计算机组成一个集群，提供更多的计算资源，从而满足更大的计算压力要求。



前面我们讨论的各种分布式缓存、负载均衡、分布式存储等都是讲如何在高并发的访问压力下，利用更多的计算机满足用户的请求访问压力。而大数据技术讨论的是，如何利用更多的计算机满足大规模的数据计算要求。

大数据就是将各种数据统一收集起来进行计算，发掘其中的价值。这些数据，既包括数据库的数据，也包括日志数据，还包括专门采集的用户行为数据；既包括企业内部自己产生的数据，也包括从第三方采购的数据，还包括使用网络爬虫获取的各种互联网公开数据。

面对如此庞大的数据，如何存储，如何利用大规模的服务器集群处理计算大量的数据，就是大数据技术的核心关键。

分布式文件存储 HDFS 架构

大规模数据计算首先要解决的是大规模数据的存储问题。如何将数百 T，数百 P 的数据存储起来，通过一个文件系统统一管理，这本身就是一个极大的挑战。

我曾在专栏 [第 5 篇](#) 讲过，分布式文件系统 HDFS 的架构。

HDFS 可以将数千台服务器组成一个统一的文件存储系统，其中 NameNode 服务器充当文件控制块的角色，进行文件元数据管理，即记录文件名、访问权限、数据存储地址等信息，而真正的文件数据则存储在 DataNode 服务器上。

DataNode 以块为单位存储数据，所有的块信息，比如块 ID、块所在的服务器 IP 地址等，都记录在 NameNode，而具体的块数据则存储在 DataNode 上。理论上，NameNode 可以将所有 DataNode 服务器上的所有数据块都分配给一个文件，也就是说，一个文件可以使用所有服务器的硬盘存储空间，达到数百 P 的大小。

此外，HDFS 为了保证不会因为硬盘或者服务器损坏而导致文件损坏，还会对数据块进行复制，每个数据块都会存储在多台服务器上，甚至多个机架上。

大数据计算 MapReduce 架构

数据存储在 HDFS 上的最终目标还是为了计算，进行数据分析或者机器学习，从而获得有益的结果。但是如果像传统的应用程序那样，把 HDFS 当做普通文件，从文件读取数据，进行计算，那么对于需要一次计算数百 T 数据的大数据计算场景，就不知道要算到什么时候了。

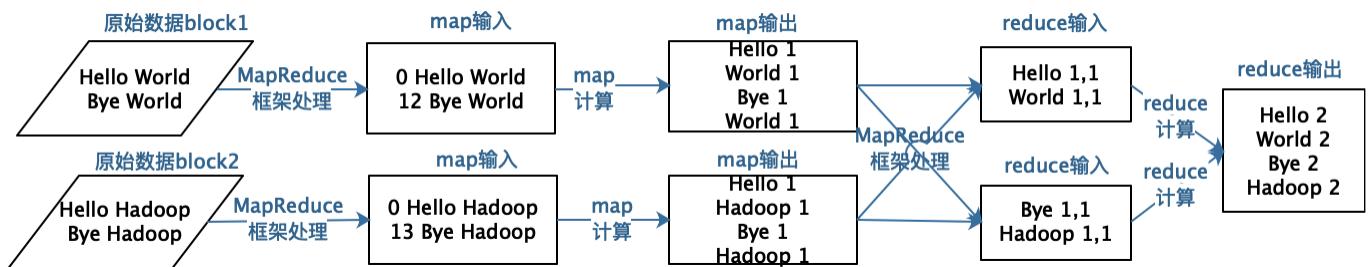
大数据处理的经典计算框架是 MapReduce。MapReduce 的核心思想是对数据进行分片计算。既然数据是以块为单位分布存储在很多台服务器组成的集群上，那么能不能就在这些

服务器上针对每个数据块进行分布式计算呢？

事实上，MapReduce 将同一个计算程序启动在分布式集群的多台服务器上，每个服务器上的程序进程都读取本服务器上要处理的数据块进行计算，因此，大量的数据就可以同时进行计算了。但是这样的话，每个数据块的数据都是独立的，如果这些数据块需要进行关联计算怎么办？

MapReduce 将计算过程分成两个部分，一个是 map 过程，每个服务器上会启动多个 map 进程，map 优先读取本地数据进行计算，计算后输出一个 `<key, value>` 集合。另一个是 reduce 过程，MapReduce 在每个服务器上都启动多个 reduce 进程，然后对所有 map 输出的 `<key, value>` 集合进行 shuffle 操作。所谓 shuffle 就是将相同的 key 发送到同一个 reduce 进程，在 reduce 中完成数据关联计算。

我们以经典的 WordCount，也就是统计所有数据中相同单词的词频数据为例，看看 map 和 reduce 的处理过程。



假设原始数据有两个数据块，MapReduce 框架启动两个 map 进程进行处理，分别读入数据。map 函数对输入数据进行分词处理，然后针对每个单词输出 `<单词, 1>` 这样的 `<key, value>` 结果。然后，MapReduce 框架进行 shuffle 操作，相同的 key 发送给同一个 reduce 进程，reduce 的输入就是 `<key, value 列表>` 这样的结构，即相同 key 的 value 合并成一个 value 列表。

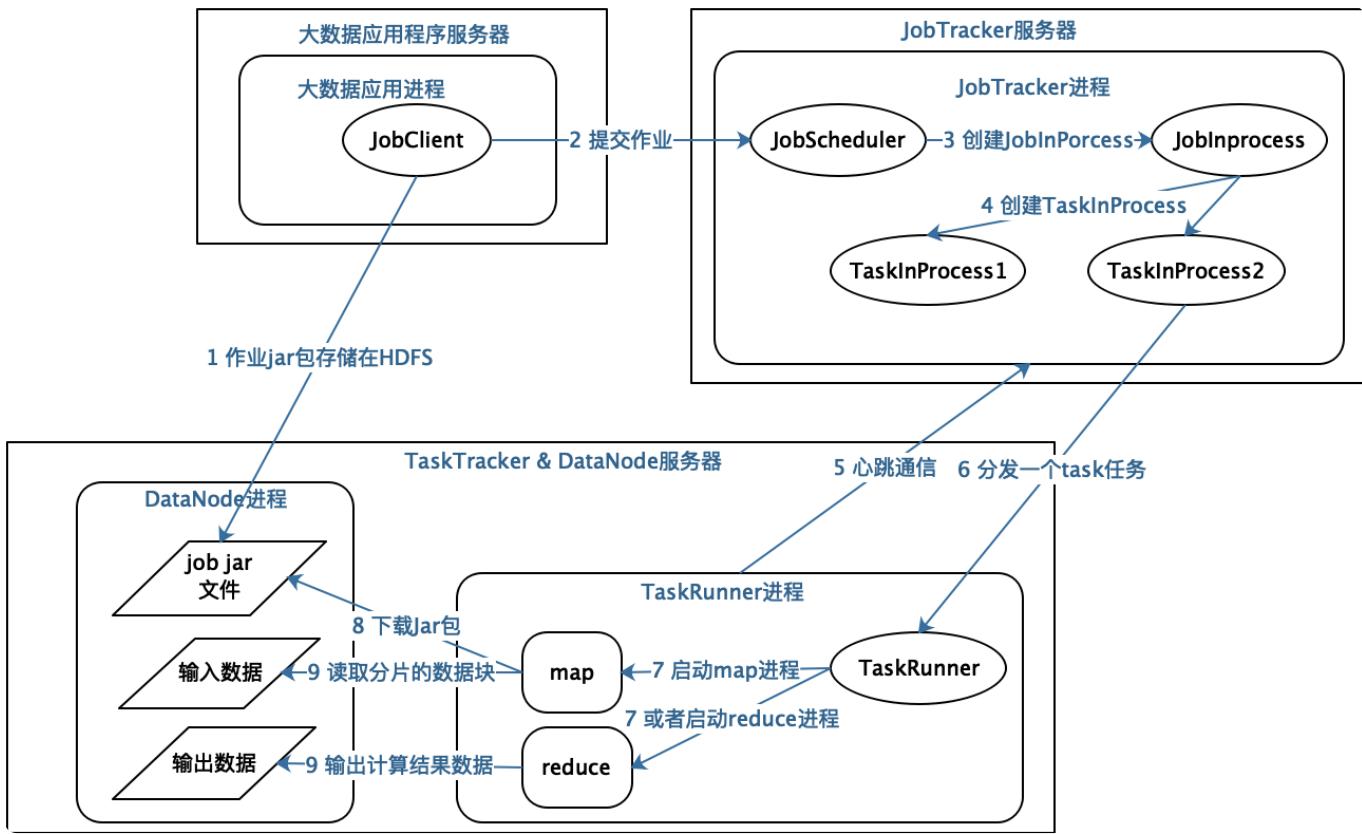
在这个例子中，这个 value 列表就是很多个 1 组成的列表。reduce 对这些 1 进行求和操作，就得到每个单词的词频结果了。具体的 MapReduce 程序如下：

复制代码

```
1 public class WordCount {
2
3     public static class TokenizerMapper
4         extends Mapper<Object, Text, Text, IntWritable>{
```

```
5  private final static IntWritable one = new IntWritable(1);
6  private Text word = new Text();
7
8  public void map(Object key, Text value, Context context
9                  ) throws IOException, InterruptedException {
10    StringTokenizer itr = new StringTokenizer(value.toString());
11    while (itr.hasMoreTokens()) {
12      word.set(itr.nextToken());
13      context.write(word, one);
14    }
15  }
16}
17
18 public static class IntSumReducer
19   extends Reducer<Text, IntWritable, Text, IntWritable> {
20   private IntWritable result = new IntWritable();
21
22   public void reduce(Text key, Iterable<IntWritable> values,
23                      Context context
24                      ) throws IOException, InterruptedException {
25     int sum = 0;
26     for (IntWritable val : values) {
27       sum += val.get();
28     }
29     result.set(sum);
30     context.write(key, result);
31   }
32 }
```

上面讲述了 map 和 reduce 进程合作完成数据处理的过程，那么这些进程是如何在分布式的服务器集群上启动的呢？数据是如何流动，最终完成计算的呢？我们以 MapReduce1 为例看下这个过程。



MapReduce1 主要有 JobTracker 和 TaskTracker 两种进程角色，JobTracker 在 MapReduce 集群中只有一个，而 TaskTracker 则和 DataNode 一起，启动在集群的所有服务器上。

MapReduce 应用程序 JobClient 启动后，会向 JobTracker 提交作业，JobTracker 根据作业中输入文件路径分析，需要在哪些服务器上启动 map 进程，然后就向这些服务器上的 TaskTracker 发送任务命令。

TaskTracker 收到任务后，启动一个 TaskRunner 进程下载任务对应的程序，然后反射加载程序中的 map 函数，读取任务中分配的数据块，进行 map 计算。map 计算结束后，TaskTracker 会对 map 输出进行 shuffle 操作，然后 TaskRunner 加载 reduce 函数进行后续计算。

HDFS 和 MapReduce 都是 Hadoop 的组成部分。

大数据仓库 Hive 架构

MapReduce 虽然只有 map 和 reduce 两个函数，却几乎可以满足任何大数据分析和机器学习的计算场景。不过复杂的计算可能需要多个 job 才能完成，这些 job 之间还需要根据其先后依赖关系进行作业编排，开发比较复杂。

数据分析传统上主要使用 SQL 进行分析，如果能根据 SQL 自动生成 MapReduce，那么可以极大降低大数据技术在数据分析领域的应用门槛。

Hive 就是这样一个工具。我们看下，对于如下一条常见的 SQL 语句，Hive 是如何将其转换成 MapReduce 计算的。

复制代码

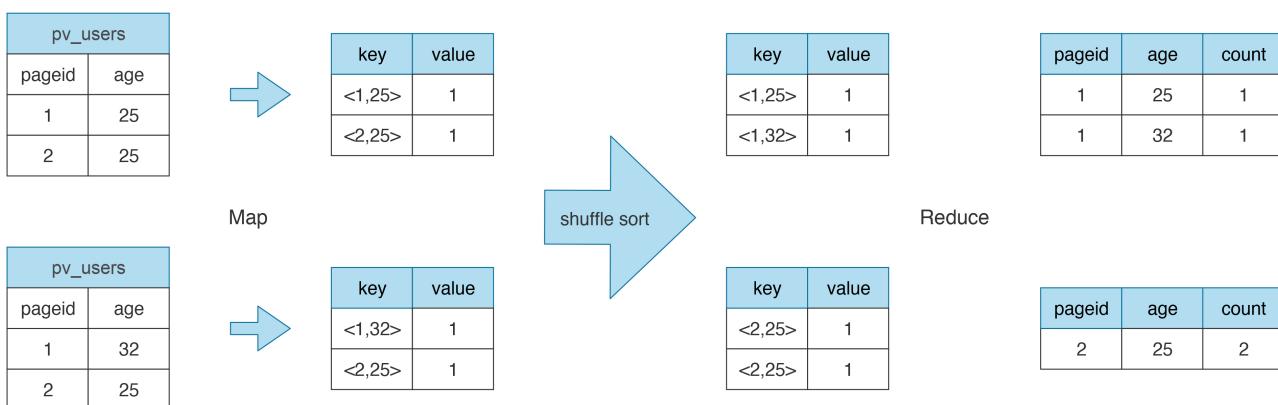
```
1 SELECT pageid, age, count(1) FROM pv_users GROUP BY pageid, age;
```

这是一条常见的 SQL 统计分析语句，统计不同年龄的用户访问不同网页的兴趣偏好，具体数据输入和执行结果示例如下。

pv_users	
pageid	age
1	25
2	25
1	32
2	25

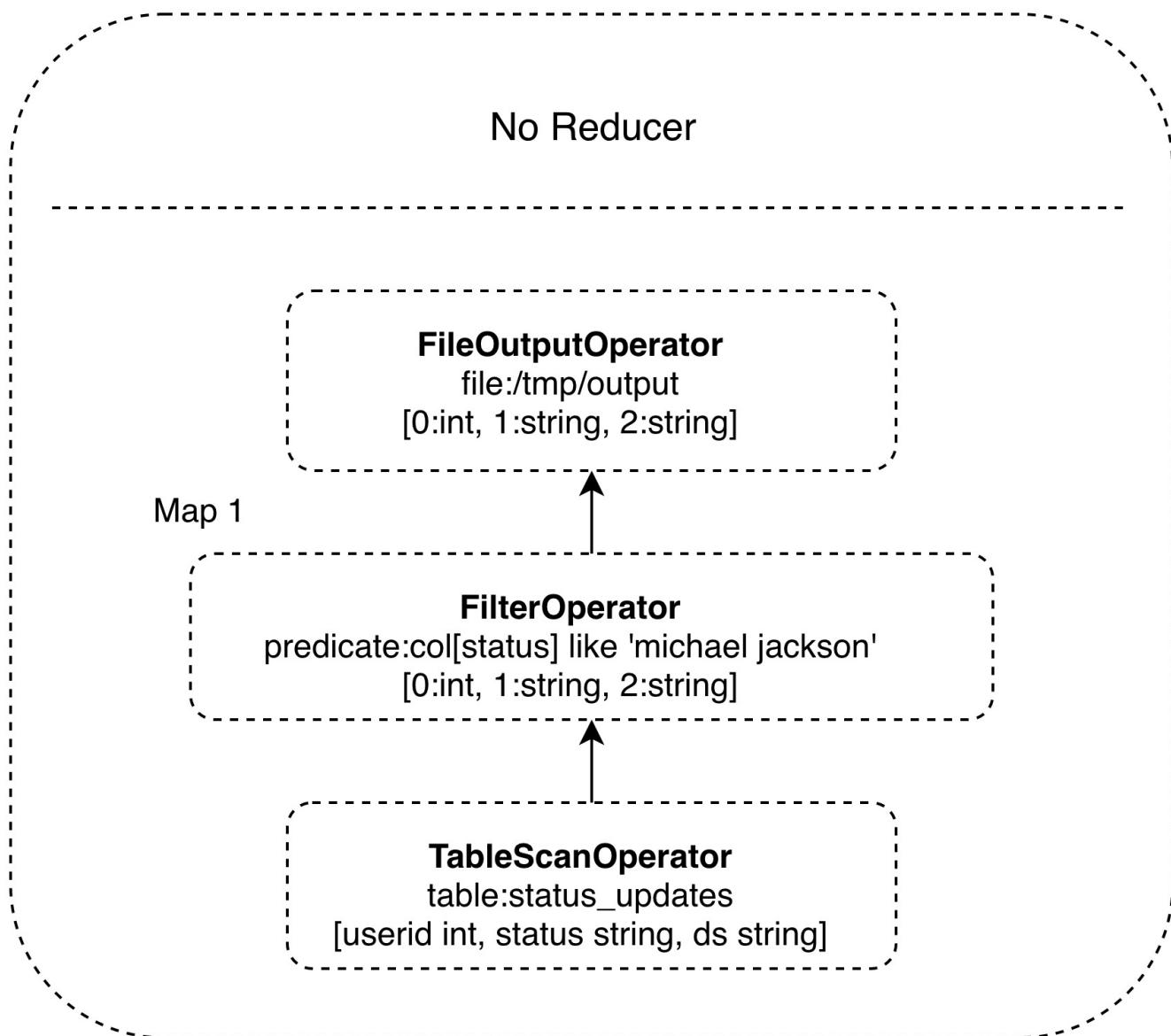
pageid	age	count
1	25	1
2	25	2
1	32	1

我们看这个示例就会发现，这个计算场景和 WordCount 很像。事实上也确实如此，我们可以用 MapReduce 的计算过程完成这条 SQL 的处理。

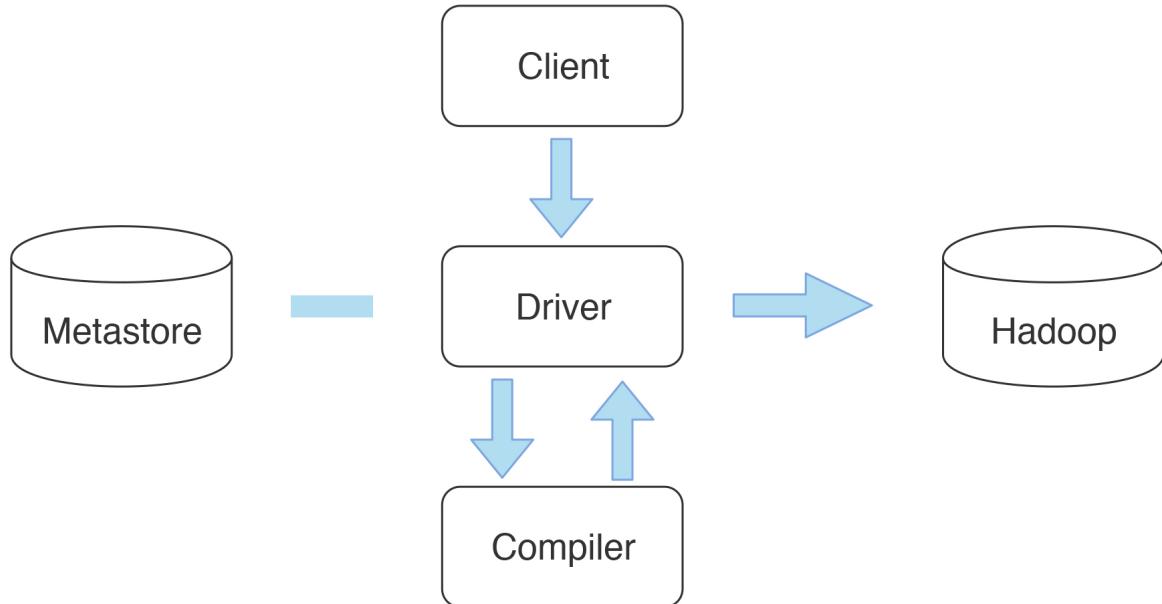


map 函数输出的 key 是表的行记录, value 是 1, reduce 函数对相同的行记录, 也就是相同的 key 的 value 集合进行求和计算, 就得到最终的 SQL 输出结果了。

那么 Hive 要做的就是将 SQL 翻译成 MapReduce 程序代码, 实际上, Hive 内置了很多 Operator, 每个 Operator 完成一个特定的计算过程, Hive 将这些 Operator 构造成一个有向无环图 DAG, 然后根据这些 Operator 之间是否存在 shuffle 将其封装到 map 或者 reduce 函数, 就可以提交给 MapReduce 执行了。Operator 组成的 DAG 图示例如下, 这是一个包含 where 查询条件的 SQL, where 查询条件对应一个 FilterOperator。



Hive 整体架构如下, Hive 的表数据存储在 HDFS。表的结构, 比如表名、字段名、字段之间的分隔符等存储在 Metastore。用户通过 Client 提交 SQL 到 Driver, Driver 请求 Compiler 将 SQL 编译成如上示例的 DAG 执行计划, 然后交给 Hadoop 执行。



快速大数据计算 Spark 架构

MapReduce 主要使用硬盘存储计算过程中的数据，这样虽然可靠性比较高，但是性能其实比较差。此外，MapReduce 只能使用 map 和 reduce 函数进行编程，虽然能够完成各种大数据计算，但是编程比较复杂。而且，受 map 和 reduce 编程模型简单的影响，复杂的计算必须组合多个 MapReduce job 才能完成，编程难度进一步增加。

Spark 在 MapReduce 基础上进行改进，主要使用内存进行中间计算数据存储，加快了计算执行时间，在某些情况下，性能可以提升上百倍。Spark 的主要编程模型是 RDD，弹性数据集。在 RDD 上定义了许多常见的大数据计算函数，利用这些函数，可以用极少的代码完成较为复杂的大数据计算。前面举例的 WorkCount，如果用 Spark 编程，只需要三行代码：

复制代码

```

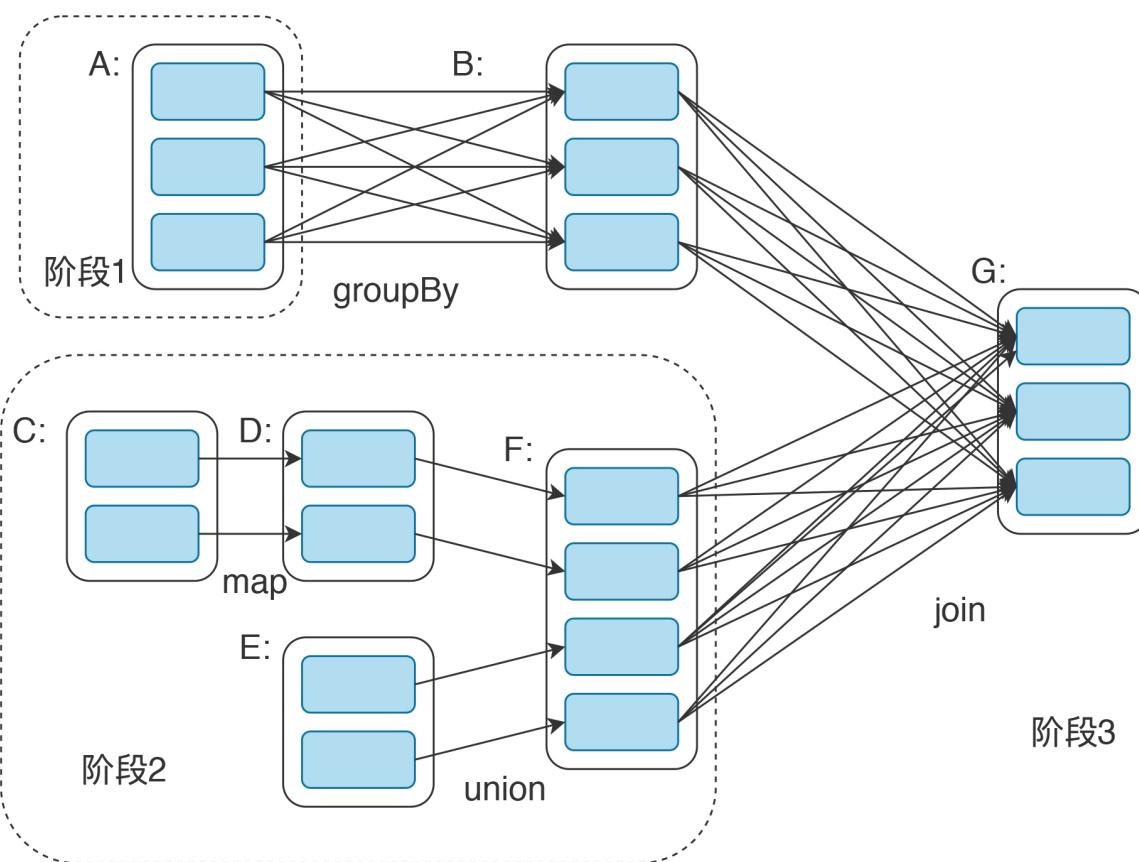
1 val textFile = sc.textFile("hdfs://...")
2 val counts = textFile.flatMap(line => line.split(" "))
3           .map(word => (word, 1))
4           .reduceByKey(_ + _)
5 counts.saveAsTextFile("hdfs://...")
  
```

首先，从 HDFS 读取数据，构建出一个 RDD `textFile`。然后，在这个 RDD 上执行三个操作：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)`；最后，使用 `reduceByKey` 函数将所有具有相同单词的值相加，从而得到单词计数。

1), 生成 $\langle \text{Key}, \text{Value} \rangle$ 的结构; 相同的 Key 进行统计, 统计方式是对 Value 求和。最后, 将 RDD counts 写入到 HDFS, 完成结果输出。

上面代码中 flatMap、map、reduceByKey 都是 Spark 的 RDD 转换函数, RDD 转换函数的计算结果还是 RDD, 所以上面三个函数可以写在一行代码, 最后得到的还是 RDD。Spark 会根据程序中的转换函数生成计算任务执行计划, 这个执行计划就是一个 DAG。Spark 可以在一个作业中完成非常复杂的大数据计算。

Spark DAG 示例如下:



如上所示, A、C 和 E 是从 HDFS 上加载的 RDD, A 经过 `groupBy` 分组统计转换函数后得到 RDD B, C 经过 `map` 转换函数后得到 RDD D, D 和 E 经过 `union` 合并转换函数后得到 RDD F, B 和 F 经过 `join` 连接转换函数后得到最终结果 RDD G。

大数据流计算架构

Spark 虽然比 MapReduce 快很多，但是大多数场景下，计算耗时依然是分钟级别的，这种计算一般被称为大数据批处理计算。而在实际应用中，有些时候需要在毫秒级完成不断输入的海量数据的计算处理，比如实时对摄像头采集的数据进行监控分析，这就是所谓的大数据流计算。

早期比较著名的流式大数据计算引擎是 Storm，后来随着 Spark 的火爆，Spark 上的流式计算引擎 Spark Streaming 也逐渐流行起来。Spark Streaming 的架构原理是将实时流入的数据切分成小的一批一批的数据，然后将这些小的一批数据交给 Spark 执行。由于数据量比较小，Spark Streaming 又常驻系统，不需要重新启动，因此可以毫秒级完成计算，看起来像是实时计算一样。



最近几年比较流行的大数据引擎 Flink 其架构原理其实和 Spark Streaming 很相似，随着数据源的不同，根据数据量和计算场景的要求，可以灵活适应流计算和批处理计算。

小结

大数据技术可以说是分布式技术的一个分支，都是面临大量的计算压力，采用分布式服务器集群的方案解决问题。差别是大数据技术要处理的数据具有关联性，所以需要有个中心服务器进行管理，NameNode、JobTracker 都是这样的中心服务器。

思考题

SQL 生成 MapReduce 计算的时候，如果遇到 join 这样的 SQL 操作，map 函数和 reduce 函数该如何设计？以下 SQL 和输入数据为例：

[复制代码](#)

```
1 SELECT pv.pageid, u.age FROM page_view pv JOIN user u ON (pv.userid = u.userid)
```

page_view:

pageid	userid	time
1	111	9:08:01
2	111	9:08:13
1	222	9:08:14

user:

userid	age	gender
111	25	female
222	32	male

点击参加 21 天打卡计划 ↗

搞定后端技术基础

扫一扫参与小程序话题



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 30 | 安全性架构：为什么说用户密码泄漏是程序员的锅？

下一篇 32 | AI与物联网架构：从智能引擎到物联网平台

精选留言 (1)

 写留言



黄海峰

2020-02-03

shuffle把相同key发送给同一个reduce，那岂不是还是要传输大量数据？还是实际是把相同key放到相同hdfs文件reduce进程读取？

展开 ▼

作者回复: 确实要传输很大量数据。

shuffle不通过HDFS，即使通过HDFS，也解决不了数据传输的问题。

