列一些常见的Spark面试题

浪尖聊大数据 6/7

收录于话题
#spark 10 #面试 16

浪尖聊大数据
主要分享大数据框架，如spark，flink，kafka，hbase原理源码，同时会分享数据仓...
366篇原创内容
Official Account

Spark

1. 通常来说，Spark与MapReduce相比，Spark运行效率更高。请说明效率更高来源于Spark内置的哪些机制?
2. hadoop和spark使用场景?
3. spark如何保证宕机迅速恢复?
4. hadoop和spark的相同点和不同点?
5. RDD持久化原理?
6. checkpoint检查点机制?
7. checkpoint和持久化机制的区别?
8. RDD机制理解吗?
9. Spark streaming以及基本工作原理?
10. DStream以及基本工作原理?
11. spark有哪些组件?
12. spark工作机制?
13. 说下宽依赖和窄依赖
14. Spark主备切换机制原理知道吗?
15. spark解决了hadoop的哪些问题?
16. 数据倾斜的产生和解决办法?
17. 你用sparksql处理的时候， 处理过程中用的dataframe还是直接写的sql？为什么?
18. 现场写一个笔试题
19. RDD中reduceBykey与groupByKey哪个性能好，为什么
20. Spark master HA主从切换过程不会影响到集群已有作业的运行，为什么
21. spark master使用zookeeper进行ha，有哪些源数据保存到Zookeeper里面

1. 通常来说，Spark与MapReduce相比，Spark运行效率更高。请说明效率更高来源于Spark内置的哪些机制?

spark是借鉴了Mapreduce,并在其基础上发展起来的，继承了其分布式计算的优点并进行了改进，spark生态更为丰富，功能更为强大，性能更加适用范围广，mapreduce更简单，稳定性好。主要区别

（1）spark把运算的中间数据(shuffle阶段产生的数据)存放在内存，迭代计算效率更高，mapreduce的中间结果需要落地，保存到磁盘

（2）Spark容错性高，它通过弹性分布式数据集RDD来实现高效容错，RDD是一组分布式的存储在节点内存中的只读性的数据集，这些集合石弹性的，某一部分丢失或者出错，可以通过整个数据集的计算流程的血缘关系来实现重建，mapreduce的容错只能重新计算

（3）Spark更通用，提供了transformation和action这两大类的多功能api，另外还有流式处理spark-streaming模块、图计算等等，mapreduce只提供了map和reduce两种操作，流计算及其他的模块支持比较缺乏

（4）Spark框架和生态更为复杂，有RDD，血缘lineage、执行时的有向无环图DAG,stage划分等，很多时候spark作业都需要根据不同业务场景的需要进行调优以达到性能要求，mapreduce框架及其生态相对较为简单，对性能的要求也相对较弱，运行较为稳定，适合长期后台运行。

（5）Spark计算框架对内存的利用和运行的并行度比mapreduce高，Spark运行容器为executor，内部ThreadPool中线程运行一个Task,mapreduce在线程内部运行container，container容器分类为MapTask和ReduceTask.程序运行并行度高

（6）Spark对于executor的优化，在JVM虚拟机的基础上对内存弹性利用：storage memory与Execution memory的弹性扩容，使得内存利用效率更高

2. hadoop和spark使用场景?

Hadoop/MapReduce和Spark最适合的都是做离线型的数据分析，但Hadoop特别适合是单次分析的数据量“很大”的情景，而Spark则适用于数据量不是很大的情景。

- 一般情况下，对于中小互联网和企业级的大数据应用而言，单次分析的数量都不会“很大”，因此可以优先考虑使用Spark。
- 业务通常认为Spark更适用于机器学习之类的“迭代式”应用，80GB的压缩数据（解压后超过200GB），10个节点的集群规模，跑类似“sum+group-by”的应用，MapReduce花了5分钟，而spark只需要2分钟。

3. spark如何保证宕机迅速恢复?

适当增加spark standby master
编写shell脚本，定期检测master状态，出现宕机后对master进行重启操作

4. hadoop和spark的相同点和不同点?

Hadoop底层使用MapReduce计算架构，只有map和reduce两种操作，表达能力比较欠缺，而且在MR过程中会重复的读写hdfs，造成大量的磁盘io读写操作，所以适合高时延环境下批处理计算的应用；

Spark是基于内存的分布式计算架构，提供更加丰富的数据集操作类型，主要分成转化操作和行动操作，包括map、reduce、filter、flatmap、groupbykey、reducebykey、union和join等，数据分析更加快速，所以适合低时延环境下计算的应用；

spark与hadoop最大的区别在于迭代式计算模型。基于mapreduce框架的Hadoop主要分为map和reduce两个阶段，两个阶段完了就结束了，所以在一个job里面能做的处理很有限；spark计算模型是基于内存的迭代式计算模型，可以分为n个阶段，根据用户编写的RDD算子和程序，在处理完一个阶段后可以继续往下处理很多个阶段，而不只是两个阶段。所以spark相较于mapreduce，计算模型更加灵活，可以提供更强大的功能。

但是spark也有劣势，由于spark基于内存进行计算，虽然开发容易，但是真正面对大数据的时候，在没有进行调优的轻局昂下，可能会出现各种各样的问题，比如OOM内存溢出等情况，导致spark程序可能无法运行起来，而mapreduce虽然运行缓慢，但是至少可以慢慢运行完。

5. RDD持久化原理?

spark非常重要的一个功能特性就是可以将RDD持久化在内存中。

调用cache()和persist()方法即可。cache()和persist()的区别在于，cache()是persist()的一种简化方式，cache()的底层就是调用persist()的无参版本persist(MEMORY_ONLY)，将数据持久化到内存中。

如果需要从内存中清除缓存，可以使用unpersist()方法。RDD持久化是可以手动选择不同的策略的。在调用persist()时传入对应的StorageLevel即可。

6. checkpoint检查点机制?

应用场景：当spark应用程序特别复杂，从初始的RDD开始到最后整个应用程序完成有很多的步骤，而且整个应用运行时间特别长，这种情况下就比较适合使用checkpoint功能。

原因：对于特别复杂的Spark应用，会出现某个反复使用的RDD，即使之前持久化过但由于节点的故障导致数据丢失了，没有容错机制，所以需要重新计算一次数据。

Checkpoint首先会调用SparkContext的setCheckPointDIR()方法，设置一个容错的文件系统的目录，比如说HDFS；然后对RDD调用checkpoint()方法。之后在RDD所处的job运行结束之后，会启动一个单独的job，来将checkpoint过的RDD数据写入之前设置的文件系统，进行高可用、容错的类持久化操作。

检查点机制是我们在spark streaming中用来保障容错性的主要机制，它可以使spark streaming阶段性的把应用数据存储到诸如HDFS等可靠存储系统中，以供恢复时使用。具体来说基于以下两个目的服务：

- 控制发生失败时需要重算的状态数。Spark streaming可以通过转化图的谱系图来重算状态，检查点机制则可以控制需要在转化图中回溯多远。
- 提供驱动器程序容错。如果流计算应用中的驱动器程序崩溃了，你可以重启驱动器程序并让驱动器程序从检查点恢复，这样spark streaming就可以读取之前运行的程序处理数据的进度，并从那里继续。

7. checkpoint和持久化机制的区别?

最主要的区别在于持久化只是将数据保存在BlockManager中，但是RDD的lineage(血缘关系，依赖关系)是不变的。但是checkpoint执行完之后，rdd已经没有之前所谓的依赖rdd了，而只有一个强行为其设置的checkpointRDD，checkpoint之后rdd的lineage就改变了。

持久化的数据丢失的可能性更大，因为节点的故障会导致磁盘、内存的数据丢失。但是checkpoint的数据通常是保存在高可用的文件系统中，比如HDFS中，所以数据丢失可能性比较低

8. RDD机制理解吗?

- rdd分布式弹性数据集，简单的理解成一种数据结构，是spark框架上的通用货币。所有算子都是基于rdd来执行的，不同的场景会有不同的rdd实现类，但是都可以进行互相转换。rdd执行过程中会形成dag图，然后形成lineage保证容错性等。从物理的角度来看rdd存储的是block和node之间的映射。
- RDD是spark提供的核心抽象，全称为弹性分布式数据集。
- RDD在逻辑上是一个hdfs文件，在抽象上是一种元素集合，包含了数据。它是被分区的，分为多个分区，每个分区分布在集群中的不同结点上，从而让RDD中的数据可以被并行操作（分布式数据集）
- 比如有个RDD有90W数据，3个partition，则每个分区上有30W数据。RDD通常通过Hadoop上的文件，即HDFS或者HIVE表来创建，还可以通过应用程序中的集合来创建；RDD最重要的特性就是容错性，可以自动从节点失败中恢复过来。即如果某个结点上的RDD partition因为节点故障，导致数据丢失，那么RDD可以通过自己的数据来源重新计算该partition。这一切对使用者都是透明的。
- RDD的数据默认存放在内存中，但是当内存资源不足时，spark会自动将RDD数据写入磁盘。比如某结点内存只能处理20W数据，那么这20W数据就会放入内存中计算，剩下10W放到磁盘中。RDD的弹性体现在于RDD上自动进行内存和磁盘之间权衡和切换的机制。

9. Spark streaming以及基本工作原理?

Spark streaming是spark core API的一种扩展，可以用于进行大规模、高吞吐量、容错的实时数据流的处理。

它支持从多种数据源读取数据，比如Kafka、Flume、Twitter和TCP Socket，并且能够使用算子比如map、reduce、join和window等来处理数据，处理后的数据可以保存到文件系统、数据库等存储中。

Spark streaming内部的基本工作原理是：接受实时输入数据流，然后将数据拆分成batch，比如每收集一秒的数据封装成一个batch，然后将每个batch交给spark的计算引擎进行处理，最后会生产处一个结果数据流，其中的数据也是一个一个的batch组成的。

10. DStream以及基本工作原理?

- DStream是spark streaming提供的一种高级抽象，代表了一个持续不断的数据流。
- DStream可以通过输入数据源来创建，比如Kafka、flume等，也可以通过其他DStream的高阶函数来创建，比如map、reduce、join和window等。
- DStream内部其实不断产生RDD，每个RDD包含了一个时间段的数据。
- Spark streaming一定是有一个输入的DStream接收数据，按照时间划分成一个一个的batch，并转化为一个RDD，RDD的数据是分散在各个子节点的partition中。

11. spark有哪些组件?

- master：管理集群和节点，不参与计算。
- worker：计算节点，进程本身不参与计算，和master汇报。
- Driver：运行程序的main方法，创建spark context对象。
- spark context：控制整个application的生命周期，包括dagsheduler和task scheduler等组件。
- client：用户提交程序的入口。

12. spark工作机制?

用户在client端提交作业后，会由Driver运行main方法并创建spark context上下文。执行add算子，形成dag图输入dagscheduler，按照add之间的依赖关系划分stage输入task scheduler。task scheduler会将stage划分为task set分发到各个节点的executor中执行。

13. 说下宽依赖和窄依赖

宽依赖：
本质就是shuffle。父RDD的每一个partition中的数据，都可能会传输一部分到下一个子RDD的每一个partition中，此时会出现父RDD和子RDD的partition之间具有交互错综复杂的关系，这种情况就叫做两个RDD之间是宽依赖。

窄依赖：
父RDD和子RDD的partition之间的对应关系是一对一的。

14. Spark主备切换机制原理知道吗?

Master实际上可以配置两个，Spark原生的standalone模式是支持Master主备切换的。当Active Master节点挂掉以后，我们可以将Standby Master切换为Active Master。

Spark Master主备切换可以基于两种机制，一种是基于文件系统的，一种是基于ZooKeeper的。

基于文件系统的主备切换机制，需要在Active Master挂掉之后手动切换到Standby Master上；

而基于Zookeeper的主备切换机制，可以实现自动切换Master。

15. spark解决了hadoop的哪些问题?

- MR：抽象层次低，需要使用手工代码来完成程序编写，使用上难以上手；
- Spark：Spark采用RDD计算模型，简单容易上手。
- MR：只提供map和reduce两个操作，表达能力欠缺；
- Spark：Spark采用更加丰富的算子模型，包括map、flatmap、groupbykey、reducebykey等；
- MR：一个job只能包含map和reduce两个阶段，复杂的任务需要包含很多个job，这些job之间的管理以来需要开发者自己进行管理；
- Spark：Spark中一个job可以包含多个转换操作，在调度时可以生成多个stage，而且如果多个map操作的分区不变，是可以放在同一个task里面去执行；
- MR：中间结果存放在hdfs中；
- Spark：Spark的中间结果一般存在内存中，只有当内存不够了，才会存入本地磁盘，而不是hdfs；
- MR：只有等到所有的map task执行完毕后才能执行reduce task；
- Spark：Spark中分区相同的转换构成流水线在一个task中执行，分区不同的需要进行shuffle操作，被划分成不同的stage需要等待前面的stage执行完才能执行。
- MR：只适合batch批处理，时延高，对于交互式处理和实时处理支持不够；
- Spark：Spark streaming可以将流拆成时间间隔的batch进行处理，实时计算。

16. 数据倾斜的产生和解决办法?

数据倾斜以为着某一个或者某几个partition的数据特别大，导致这几个partition上的计算需要耗费相当长的时间。

在spark中同一个应用程序划分成多个stage，这些stage之间是串行执行的，而一个stage里面的多个task是可以并行执行，task数目由partition数目决定，如果一个partition的数目特别大，那么导致这个task执行时间很长，导致接下来的stage无法执行，从而导致整个job执行变慢。

避免数据倾斜，一般是要选用合适的key，或者自己定义相关的partitioner，通过加盐或者哈希值来拆分这些key，从而将这些数据分散到不同的partition去执行。

如下算子会导致shuffle操作，是导致数据倾斜可能发生的关键点所在：groupByKey；reduceByKey；aggregaByKey；join；cogroup；

17. 你用sparksql处理的时候， 处理过程中用的dataframe还是直接写的sql？为什么?

这个问题的宗旨是问你spark sql 中dataframe和sql的区别，从执行原理、操作方便程度和自定义程度来分析这个问题。

18. 现场写一个笔试题

有hdfs文件，文件每行的格式为作品ID，用户id，用户性别。请用一个spark任务实现以下功能：统计每个作品对应的用户（去重后）的性别分布。输出格式如下：作品ID，男性用户数量，女性用户数量

答案：

```
sc.textfile() .flatmap(.split(","))//分割成作
    品ID，用户id，用户性别
.map(((_.1,_._2),1))//((作品id,用户性别),1)
.reduceByKey(_+_)//((作品id,用户性别),n)
.map(_._1._1,_._1._2,_._2)//(作品id,用户性别,n)
```

19. RDD中reduceBykey与groupByKey哪个性能好，为什么

reduceByKey：reduceByKey会在结果发送至reducer之前会对每个mapper在本地进行merge，有点类似于在MapReduce中的combiner。这样做的好处在于，在map端进行一次reduce之后，数据量会大幅度减小，从而减小传输，保证reduce端能够更快的进行结果计算。

groupByKey：groupByKey会对每一个RDD中的value值进行聚合形成一个序列(Iterator)，此操作发生在reduce端，所以势必会将所有的数据通过网络进行传输，造成不必要的浪费。同时如果数据量十分大，可能还会造成OutOfMemoryError。

所以在进行大量数据的reduce操作时候建议使用reduceByKey。不仅可以提高速度，还可以防止使用groupByKey造成的内存溢出问题。

20. Spark master HA主从切换过程不会影响到集群已有作业的运行，为什么

不会的。

因为程序在运行之前，已经申请过资源了，driver和Executors通讯，不需要和master进行通讯的。

21. spark master使用zookeeper进行ha，有哪些源数据保存到Zookeeper里面

spark通过这个参数spark.deploy.zookeeper.dir指定master元数据在zookeeper中保存的位置，包括Worker，Driver和Application以及Executors。standby节点要从zk中，获得元数据信息，恢复集群运行状态，才能对外继续提供服务，作业提交资源申请等，在恢复前是不能接受请求的。

```
1、在Master切换的过程中，所有的已经在运行的程序皆正常运行！
因为Spark Application在运行前就已经通过Cluster Manager获得了
计算资源，所以在运行时Job本身的
调度和处理和Master是没有任何关系。
2、在Master的切换过程中唯一的影响是不能提交新的Job：
一方面不能够提交新的应用程序给集群，
因为只有Active Master才能接受新的程序的提交请求；
另外一方面，已经运行的程序中也不能够因
Action操作触发新的Job的提交请求。
```

浪尖聊大数据
主要分享大数据框架，如spark，flink，kafka，hbase原理源码，同时会分享数据仓...
366篇原创内容
Official Account

收录于话题 #spark·10个 >
上一篇
spark shuffle的几种特殊情况
下一篇
数据本地性对 Spark 生产作业容错能力的负面影响