大数据梅峰谷共享组

《Spark 面试 2000 题第 5 期试题》

参考答案

编写人　梅峰谷　　日期　2017 年 06 月 13 日

修改人　　日期　　年　月　日

版　本　V1.0

---------本节内容----------

1. 面试题 30 道

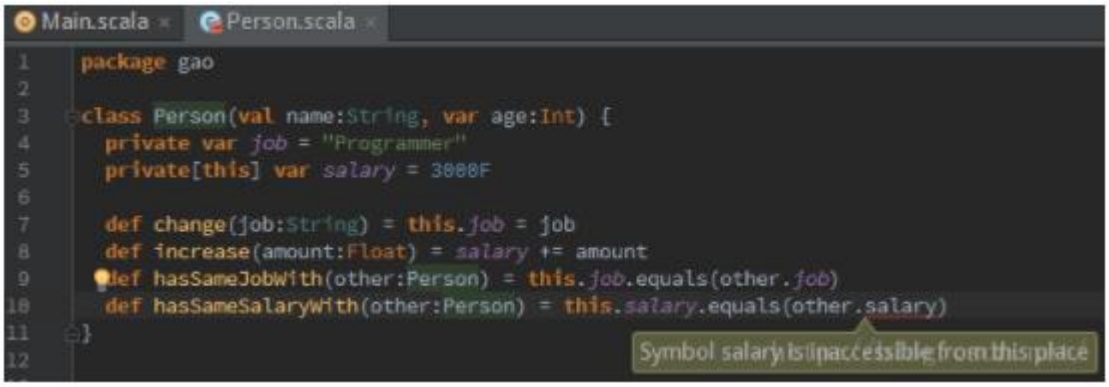
2. 参考资料

Spark RDD 是 Spark 的编程基础，掌握 RDD 以及 RDD 编程技巧是企业实际开发的必备技能，本篇整理 RDD 常见的问题，汇编成题，以加深对 RDD 及 RDD 编程的理解。先把题目列举出来，各位感兴趣的自己去做一遍把，下一遍再通过网盘的方式，把答案公布出来。

1. scala 中 private 与 private[this] 修饰符的区别？

1）private ，类私有的字段，Scala 会自动生成私有的 getter/setter 方法，通过对象实例可以调用如下面的 other.job；

2）private[this]，对象私有的字段，Scala 不生成 getter/setter 方法，所以只能在对象内部访问被修饰的字段，如下代码是不能编译通过的，因为没有生成 getter/setter 方法，所以不能通过这个方法调用。**private[this]比 private 要更加严格**，他将声明的变量只能在自己的同一个实例中可以被访问。



2. scala 中内部类和 java 中的内部类区别

1)scala 内部类：同样的类的内部类的不同实例，属于不同的类型

内部类纯属于对象的(属于外部类的实例本身)，比如构造内部类对象方法
scala:val test=new 外部类名.内部类构造方法
2）java 内部类：java 内部类是一个编译时的概念，一旦编译成功，就会成为完全不同的两类。对于一个名为 outer 的外部类和其内部定义的名为 inner 的内部类。编译完成后出现 outer.class 和 outer$inner.class 两类。所以内部类的成员变量/方法名可以和外部类的相同。

3.Spark 中 standalone 模式特点，有哪些优点和缺点？

特点：1)standalone 是 master/slave 架构，集群由 Master 与 Worker 节点组成，程序通过与 Master 节点交互申请资源，Worker 节点启动 Executor 运行；2）standalone 调度模式使用 FIFO 调度方式；3）无依赖任何其他资源管理系统，Master 负责管理集群资源
优点：1）部署简单；2）不依赖其他资源管理系统
缺点： 1）默认每个应用程序会独占所有可用节点的资源，当然可以通过 spark.cores.max 来决定一个应用可以申请的 CPU cores 个数；2）可能有单点故障，需要自己配置 master HA

4.FIFO 调度模式的基本原理、优点和缺点？

基本原理：按照先后顺序决定资源的使用，资源优先满足最先来的 job。第一个 job 优先获取所有可用的资源，接下来第二个 job 再获取剩余资源。以此类推，如果第一个 job 没有占用所有的资源，那么第二个 job 还可以继续获取剩余资源，这样多个 job 可以并行运行，如果第一个 job 很大，占用所有资源，则第二 job 就需要等待，等到第一个 job 释放所有资源。
优点和缺点：1)适合长作业，不适合短作业,2)适合 CPU 繁忙型作业（计算时间长，相当于长作业），不利于 IO 繁忙型作业（计算时间短，相当于短作业）

5.FAIR 调度模式的优点和缺点？

所有的任务拥有大致相当的优先级来共享集群资源，spark 多以轮训的方式为任务分配资源，不管长任务还是端任务都可以获得资源，并且获得不错的响应时间，对于短任务，不会像 FIFO 那样等待较长时间了，通过参数 spark.scheduler.mode 为 FAIR 指定.

6.CAPCACITY 调度模式的优点和缺点？

原理：

计算能力调度器支持多个队列，每个队列可配置一定的资源量，每个队列采用 FIFO 调度策略，为了防止同一个用户的作业独占队列中的资源，该调度器会对同一用户提交的作业所占资源量进行限定。调度时，首先按以下策略选择一个合适队列：计算每个队列中正在运行的任务数与其应该分得的计算资源之间的比值(即比较空闲的队列)，选择一个该比值最小的队列；然后按以下策略选择该队列中一个作业：按照作业优先级和提交时间顺序选择，同时考虑用户资源量限制和内存限制

优点：

1) 计算能力保证。支持多个队列，某个作业可被提交到某一个队列中。每个队列会配置一定比例的计算资源，且所有提交到队列中的作业共享该队列中的资源。
(2) 灵活性。空闲资源会被分配给那些未达到资源使用上限的队列，当某个未达到资源的队列需要资源时，一旦出现空闲资源资源，便会分配给他们。
(3) 支持优先级。队列支持作业优先级调度（默认是 FIFO）
(4) 多重租赁。综合考虑多种约束防止单个作业、用户或者队列独占队列或者集群中的资源。

(5) 基于资源的调度。 支持资源密集型作业，允许作业使用的资源量高于默认值，进而可容纳不同资源需求的作业。不过，当前仅支持内存资源的调度。

7.列举你了解的序列化方法，并谈谈序列化有什么好处？

1）序列化：将对象转换为字节流，本质也可以理解为将链表的非连续空间转为连续空间存储的数组，可以将数据进行流式传输或者块存储，反序列化就是将字节流转为对象。

kyro,Java 的 serialize 等

2）spark 中的序列化常见于

- 进程间通讯：不同节点的数据传输
- 数据持久化到磁盘

在 spark 中扮演非常重要的角色，序列化和反序列化的程度会影响到数据传输速度，甚至影响集群的传输效率，因此，高效的序列化方法有 2 点好处：a.提升数据传输速度，b.提升数据读写 IO 效率。

8.常见的数压缩方式，你们生产集群采用了什么压缩方式，提升了多少效率？

1)数据压缩，大片连续区域进行数据存储并且存储区域中数据重复性高的状况下，可以使用适当的压缩算法。数组，对象序列化后都可以使用压缩，数更紧凑，减少空间开销。常见的压缩方式有 snappy，LZO，gz 等

2)Hadoop 生产环境常用的是 snappy 压缩方式（使用压缩，实际上是 CPU 换 IO 吞吐量和磁盘空间，所以如果 CPU 利用率不高，不忙的情况下，可以大大提升集群处理效率）。snappy 压缩比一般 20%~30%之间，并且压缩和解压缩效率也非常高（参考数据如下）。

a.GZIP 的压缩率最高，但是其实 CPU 密集型的，对 CPU 的消耗比其他算法要多，压缩和解压速度也慢；b.LZO 的压缩率居中，比 GZIP 要低一些，但是压缩和解压速度明显要比 GZIP 快很多，其中解压速度快的更多；

c.Zippy/Snappy 的压缩率最低，而压缩和解压速度要稍微比 LZO 要快一些。

Algorithm	% remaining	Encoding	Decoding
GZIP	13.4%	21 MB/s	118 MB/s
LZO	20.5%	135 MB/s	410 MB/s
Zippy/Snappy	22.2%	172 MB/s	409 MB/s

提升了多少效率可以从 2 方面回答，1）数据存储节约多少存储，2）任务执行消耗时间节约了多少，可以举个实际例子展开描述。

9.简要描述 Spark 写数据的流程？

1）RDD 调用 compute 方法，进行指定分区的写入

2）CacheManager 中调用 BlockManager 判断数据是否已经写入，如果未写，则写入

3）BlockManager 中数据与其他节点同步

4）BlockManager 根据存储级别写入指定的存储层

5）BlockManager 向主节点汇报存储状态中

10.Spark 中 Lineage 的基本原理

这里应该是问你 Spark 的容错机制的原理：

1）Lineage（又称为 RDD 运算图或 RDD 依赖关系图）是 RDD 所有父 RDD 的 graph（图）。它是在 RDD 上执行 transformations 函数并创建 logical execution plan（逻辑执行计划）的结果，是 RDD 的逻辑执行计划，记录了 RDD 之间的依赖关系。

2） 使用 Lineage 实现 spark 的容错，本质上类似于数据库中重做日志，是容错机制的一种方式，不过这个重做日志粒度非常大，是对全局数据做同样的重做进行数据恢复。

11.使用 shell 和 scala 代码实现 WordCount？

这个题目即考察了你对 shell 的掌握，又考察了你对 scala 的了解，还考察了你动手写代码的能力，是比较好的一道题（实际开发中，有些代码是必须要背下来的，烂熟于心，劣等的程序员就是百度+copy，是不可取的）

---

scala 版本（spark）的 wordCount

```
val conf = new SparkConf() val sc = new SparkContext(conf) val line = sc.textFile("xxxx.txt") line.flatMap(_.split(" ")).map((_,1)).reduceByKey(_+_).collect().foreach(println) sc.stop()
```

12.请列举你碰到的 CPU 密集型的应用场景，你有做哪些优化？

1）CPU 密集型指的是系统的 硬盘/内存 效能 相对 CPU 的效能 要好很多，此时，系统运作，大部分的状况是 CPU Loading 100%，CPU 要读/写 I/O（硬盘/内存），I/O 在很短的时间就可以完成，而 CPU 还有许多运算要处理，CPU Loading 很高。->cpu 是瓶颈

I/O 密集型指的是系统的 CPU 效能相对硬盘/内存的效能要好很多，此时，系统运作，大部分的状况是 CPU 在等 I/O（硬盘/内存）的读/写，此时 CPU Loading 不高。->IO 是瓶颈

2）CPU 密集型主要特点是要进行大量的计算，常见应用场景有：图计算、大量的逻辑判断程序，机器学习等，Mahout 其实就是针对 CPU 密集的一个 apache 项目。

优化的点主要有，1）降低任务的并行执行，务越多，花在任务切换的时间就越多，CPU 执行任务的效率就越低，2）优化计算逻辑，减少计算逻辑的复杂度，3）尽量减少使用高强度压缩方式，对原始数据的压缩和解压缩会增加 CPU 的负担

13.Spark RDD 和 MR2 的区别

1）mr2 只有 2 个阶段，数据需要大量访问磁盘，数据来源相对单一 ，spark RDD ，可以无数个阶段进行迭代计算，数据来源非常丰富，数据落地介质也非常丰富 spark 计算基于内存，

2）mr2 需要频繁操作磁盘 IO 需要 大家明确的是如果是 SparkRDD 的话，你要知道每一种数据来源对应的是什么，RDD 从数据源加载数据,将数据放到不同的 partition 针对这些 partition 中的数据进行迭代式计算计算完成之后，落地到不同的介质当中

14.Spark 读取 hdfs 上的文件，然后 count 有多少行的操作，你可以说说过程吗。那这个 count 是在内存中，还是磁盘中计算的呢？

1）从任务执行的角度分析执行过程

driver 生成逻辑执行计划->driver 生成物理执行计划->driver 任务调度->executor 任务执行 。

• 四个阶段

逻辑执行计划-》成物理执行计划-》任务调度-》任务执行

• 四个对象

driver-》DAGScheduler-》TaskScheduler-》Executor

• 两种模式

任务解析、优化和提交单机模式-》任务执行分布式模式

2）计算过程发生在内存

15.Spark 和 Mapreduce 快？ 为什么快呢？ 快在哪里呢？

Spark 更加快的主要原因有几点：1）基于内存计算，减少低效的磁盘交互；2）高效的调度算法，基于 DAG;3)容错机制 Lingage，主要是 DAG 和 Lianage，及时 spark 不使用内存技术，也大大快于 mapreduce

16.Spark sql 又为什么比 hive 快呢？

计算引擎不一样，一个是 spark 计算模型，一个是 mapreudce 计算模型

17.RDD 的数据结构是怎么样的？

个 RDD 对象，包含如下 5 个核心属性。

1）一个分区列表，每个分区里是 RDD 的部分数据（或称数据块）。

2）一个依赖列表，存储依赖的其他 RDD。

3）一个名为 compute 的计算函数，用于计算 RDD 各分区的值。

4）分区器（可选），用于键/值类型的 RDD，比如某个 RDD 是按散列来分区。

5）计算各分区时优先的位置列表（可选），比如从 HDFS 上的文件生成 RDD 时，RDD 分区的位置优先选择数据所在的节点，这样可以避免数据移动带来的开销。

18.RDD 算子里操作一个外部 map 比如往里面 put 数据。然后算子外再遍历 map。会有什么问题吗。

频繁创建额外对象，容易 oom

19.hadoop 的生态呢。说说你的认识。

hadoop 生态主要分为三大类型，1）分布式文件系统，2）分布式计算引擎，3）周边工具

1）分布式系统：HDFS，hbase

2）分布式计算引擎：Spark，MapReduce

3）周边工具：如 zookeeper，pig，hive，oozie，sqoop，ranger，kafka 等

可以参考介绍 http://www.cnblogs.com/zhijianliutang/articles/5195045.html

20.jvm 怎么调优的，介绍你的 Spark JVM 调优经验？

参考梅峰谷博文 《Spark 应用经验与程序调优》设置合理的 JVM 部分

21.jvm 结构？堆里面几个区？

1）JVM 内存区域分为方法去、虚拟机栈、本地方法栈、堆、程序计数器

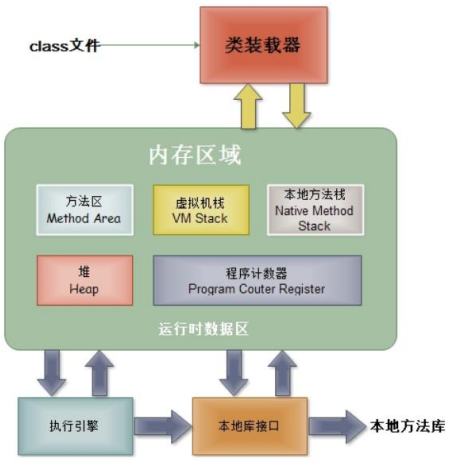
方法区：**也称"永久代" 、"非堆"**， 它用于存储虚拟机加载的类信息、常量、静态变量、是各个线程共享的内存区域

虚拟机栈：描述的是 Java 方法执行的内存模型：每个方法被执行的时候 都会创建一个"栈帧"用于存储局部变量表(包括参数)、操作栈、方法出口等信息

本地方法栈：与虚拟机栈基本类似，区别在于虚拟机栈为虚拟机执行的 java 方法服务，而本地方法栈则是为 Native 方法服务

堆：也叫做 java 堆、GC 堆是 java 虚拟机所管理的内存中最大的一块内存区域，也是被各个线程共享的内存区域，在 JVM 启动时创建。

程序计数器：是最小的一块内存区域，它的作用是当前线程所执行的字节码的行号指示器。



2）堆：a.JVM 中共享数据空间可以分成三个大区，新生代（Young Generation）、老年代（Old Generation）、永久代（Permanent Generation），其中 JVM 堆分为新生代和老年代，b.新生代可以划分为三个区，Eden 区（存放新生对象），两个幸存区（From Survivor 和 To Survivor）（存放每次垃圾回收后存活的对象）；c.永久代管理 class 文件、静态对象、属性等（JVM uses a separate region of memory, called the Permanent Generation (orPermGen for short), to hold internal representations of java classes. PermGen is also used to store more information ）；d.JVM 垃圾回收机制采用“分代收集”：新生代采用复制算法，老年代采用标记清理算法

备注:对虚拟机不了解的朋友，可以看梅峰谷分享的 JVM 虚拟机系列视频教程和书籍
链接：https://pan.baidu.com/s/1c1Rw79e 密码：6qof

22.怎么用 Spark 做数据清洗

spark 的 RDD 的转化

23.Spark 怎么整合 hive?

1).将 hive 的配置文件 hive-site.xml 复制到 Spark conf 目录下

2）根据 hive 的配置参数 hive.metastore.uris 的情况，采用不同的集成方式

a. jdbc 方式：hive.metastore.uris 没有给定配置值，为空(默认情况),SparkSQL 通过 hive 配置的 javax.jdo.option.XXX 相关配置值直接连接 metastore 数据库直接获取 hive 表元数据， 需要将连接数据库的驱动添加到 Spark 应用的 classpath 中

b.metastore 服务方式:hive.metastore.uris 给定了具体的参数值,SparkSQL 通过连接 hive 提供的 metastore 服务来获取 hive 表的元数据， 直接启动 hive 的 metastore 服务即可完成 SparkSQL 和 Hive 的集成:

3)使用 metastore 服务方式，对 hive-site.xml 进行配置

<property>

<name>hive.metastore.uris</name>

<value> trhift://mfg-hadoop:9083</value>
</property>
4).启动 hive service metastore 服务
bin/hive --service metastore &
5)启动 spark-sql 测试，执行 show databases 命令，检查是不是和 hive 的数据库一样的。

24.Spark 读取数据，是几个 Partition 呢？
答：从 2 方面介绍和回答，一是说下 partition 是什么玩意，二是说下 partition 如何建的
1）spark 中的 partion 是弹性分布式数据集 RDD 的最小单元，RDD 是由分布在各个节点上的 partion 组成的。partion 是指的 spark 在计算过程中，生成的数据在计算空间内最小单元，同一份数据（RDD）的 partion 大小不一，数量不定，是根据 application 里的算子和最初读入的数据分块数量决定的,这也是为什么叫“弹性分布式”数据集的原因之一。Partition 不会根据文件的偏移量来截取的（比如有 3 个 Partition，1 个是头多少 M 的数据，1 个是中间多少 M 的数据，1 个是尾部多少 M 的数据），而是从一个原文件这个大的集合里根据某种计算规则抽取符合的数据来形成一个 Partition 的
2） 如何创建分区，有两种情况，创建 RDD 时和通过转换操作得到新 RDD 时。对于前者，在调用 textFile 和 parallelize 方法时候手动指定分区个数即可。例如 sc.parallelize(Array(1, 2, 3, 5, 6), 2) 指定创建得到的 RDD 分区个数为 2。如果没有指定，partition 数等于 block 数；对于后者，直接调用 repartition 方法即可。实际上分区的个数是根据转换操作对应多个 RDD 之间的依赖关系来确定，窄依赖子 RDD 由父 RDD 分区个数决定，例如 map 操作，父 RDD 和子 RDD 分区个数一致；Shuffle 依赖则由分区器（Partitioner）决定，例如 groupByKey(new HashPartitioner(2)) 或者直接 groupByKey(2) 得到的新 RDD 分区个数等于 2。

25.hbase region 多大会分区，Spark 读取 hbase 数据是如何划分 partition 的？
答：region 超过了 hbase.hregion.max.filesize 这个参数配置的大小就会自动裂分，默认值是 1G。
默认情况下，hbase 有多少个 region，Spark 读取时就会有多少个 partition
26.画图，画 Spark 的工作模式，部署分布架构图
参考梅峰谷博文《【Spark 你妈喊你回家吃饭-11】Spark 基本概念和运行模式》
27.画图，画图讲解 Spark 工作流程。以及在集群上和各个角色的对应关系。
参考梅峰谷博文《【Spark 你妈喊你回家吃饭-13】Spark 计算引擎剖析》
28.Java 自带有哪几种线程池。
1)newCachedThreadPool
创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。这种类型的线程池特点是：

- 工作线程的创建数量几乎没有限制(其实也有限制的,数目为 Interger. MAX_VALUE), 这样可灵活的往线程池中添加线程。
- 如果长时间没有往线程池中提交任务，即如果工作线程空闲了指定的时间(默认为 1 分钟)，则该工作线程将自动终止。终止后，如果你又提交了新的任务，则线程池重新创建一个工作线程。
- 在使用 CachedThreadPool 时，一定要注意控制任务的数量，否则，由于大量线程同时运行，很有会造成系统瘫痪。

2)newFixedThreadPool
创建一个指定工作线程数量的线程池。每当提交一个任务就创建一个工作线程，如果工作线程数量达到线程池初始的最大数，则将提交的任务存入到池队列中。FixedThreadPool 是一个典型且优秀的线程池，它具有线程池提高程序效率和节省创建线程时所耗的开销的优点。但是，在线程池空闲时，即线程池中没有可运行任务时，它不会释放工作线程，还会占用一定的系统资源。
3)newSingleThreadExecutor
创建一个单线程化的 Executor，即只创建唯一的工作者线程来执行任务，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行。如果这个线程异常结束，会有另一个取代它，保证顺序执行。单工作线程最大的特点是可保证顺序地执行各个任务，并且在任意给定的时间不会有多个线程是活动的。
4)newScheduleThreadPool
创建一个定长的线程池，而且支持定时的以及周期性的任务执行，支持定时及周期性任务执行。延迟 3 秒执行.

29.画图，讲讲 shuffle 的过程。那你怎么在编程的时候注意避免这些性能问题

参考梅峰谷博文《【Spark 你妈喊你回家吃饭-13】Spark 计算引擎剖析》 Spark Shuffle 解析部分

30.BlockManager 怎么管理硬盘和内存的

参考王家林的视频教程《38-BlockManager 架构原理、运行流程图和源码解密》

2. 参考资料

1. http://blog.csdn.net/kwu_ganymede/article/details/51299115 JVM 调优经验
2. http://blog.csdn.net/lxhandlbb/article/details/52987928 JVM 调优之原理概述 以及降低 cache 操作的内存占比
3. http://blog.csdn.net/smithdoudou88/article/details/43152233 JVM 结构图（JVM 几个区整体上描述）
4. http://www.cnblogs.com/aaron911/p/6213808.html Java 几种常用的线程池比较
5. http://mt.sohu.com/it/d20170406/132359011_470008.shtmlSpark 自己的分布式存储系统 BlockManager 全解析

本文由梅峰谷原创，转载请注明来自大数据梅峰谷更多技术文章请关注微信公众号，答案在下一期通过百度网盘的方式公布出来，

更多面试题，请扫面上面的二维码