26-Spark

Apache Spark：

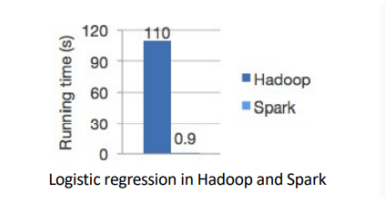
unified engine for large-scale data analytics

基本思想：

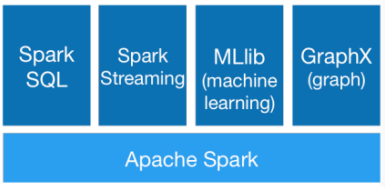
MapReduce 所有事情都在内存里做，速度快

可以集成分布式文件系统

- Outstanding performance
  - Run workloads 100x faster
  - Memory computing
- Components for various usage
  - Spark Core
  - Spark SQL
  - Spark Streaming
  - MLlib
  - GraphX



Logistic regression in Hadoop and Spark

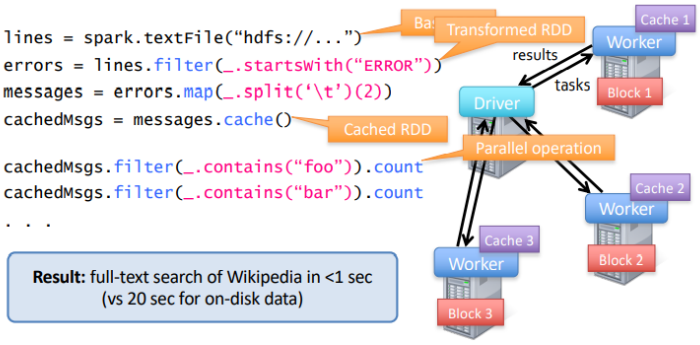


- 批处理数据/流处理数据(实际上是批处理一小段一小段时间的数据)
- 支持SQL型访问数据，如果加了其它配置也能通过driver统一读其他种类的数据库
- 支持大规模的数据科学（因为Spark是分布式的）
- 机器学习库、GraphX处理图数据

CANVAS提供代码：sgl(函数式编程)/python/java

E.G. Load error messages from a log into memory, then interactively search for various patterns

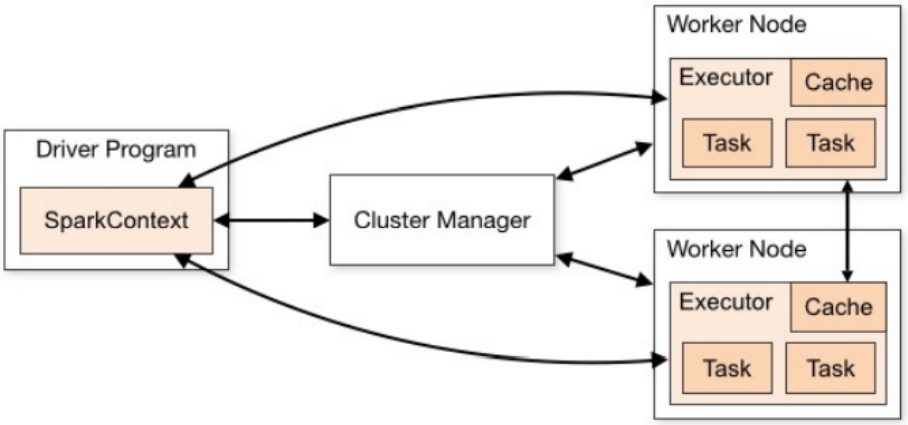
- line: 存读进来的数据（弹性分布式数据集），读进来之后不会改
- 把error开头的保留下来，存在errors里
- message很重要，一定要放在内存里，不参与垃圾回收（用cache来声明）
- count是action，要三台机器分别做好后给driver整合



Driver的作用：发代码，告知任务以及什么时候开始执行

真的可以cache住吗？要求不合理时（存不下），会自己压缩一下，不行再扔到硬盘里

Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in your main program (called the *driver program*).



同时可以设置k个reducer，但时实际上会起很多reducer实例

Worker Nodes 最好靠的近，是需要通过网络通讯的

Spark只是起监视作用，与作业无关

rdd启动时不需要从硬盘里读数据

Caching

- Spark also supports pulling data sets into a cluster-wide in-memory cache.
- This is very useful when data is accessed repeatedly, such as when querying a small "hot" dataset or when running an iterative algorithm like PageRank.
- As a simple example, let's mark our linesWithSpark dataset to be cached:

```
scala> linesWithSpark.cache()
res6: linesWithSpark.type = [value: string]

scala> linesWithSpark.count()
res7: Long = 19

scala> linesWithSpark.count()
res8: Long = 19
```

跑程序注意：

创SDK

跑的时候先编译，要把所有库写到build依赖文件里，再编译，一起打包，要把

spark-sql

RDD

- 什么是RDD？

RDD叫做**弹性分布式数据集，是Spark中最基本的数据抽象**，它代表一个不可变、可分区、里面的元素可并行计算的集合。RDD具有数据流模型的特点：自动容错、位置感知性调度和可伸缩性。RDD允许用户在执行多个查询时显式地将工作集**缓存在内存中**，后续的查询能够重用工作集，这极大地提升了查询速度。

- RDD的属性

（1）一组分片（Partition），即数据集的基本组成单位。对于RDD来说，每个分片都会被一个计算任务处理，并决定并行计算的粒度。用户可以在创建RDD时指定RDD的分片个数，如果没有指定，那么就会采用默认值。默认值就是程序所分配到的CPU Core的数目。

（2）一个计算每个分区的函数。Spark中RDD的计算是以分片为单位的，每个RDD都会实现compute函数以达到这个目的。compute函数会对迭代器进行复合，不需要保存每次计算的结果。

（3）RDD之间的依赖关系。RDD的每次转换都会生成一个新的RDD，所以RDD之间就会形成类似于流水线一样的前后依赖关系。在部分分区数据丢失时，Spark可以通过这个依赖关系重新计算丢失的分区数据，而不是对RDD的所有分区进行重新计算。

（4）一个Partitioner，即RDD的分片函数。当前Spark中实现了两种类型的分片函数，一个是基于哈希的HashPartitioner，另外一个是基于范围的RangePartitioner。只有对于于key-value的RDD，才会有Partitioner，非key-value的RDD的Parititioner的值是None。Partitioner函数不但决定了RDD本身的分片数量，也决定了parent RDD Shuffle输出时的分片数量。

（5）一个列表，存储存取每个Partition的优先位置（preferred location）。对于一个HDFS文件来说，这个列表保存的就是每个Partition所在的块的位置。按照“移动数据不如移动计算”的理念，Spark在进行任务调度的时候，会尽可能地将计算任务分配到其所要处理数据块的存储位置。

- Spark支持两个类型（算子）操作：**Transformation和Action**

transformation生成新的数据集，只依赖与前面的一块，可以在本地做，会生成新的RDD，如map、flatMap、error等操作，是lazy的

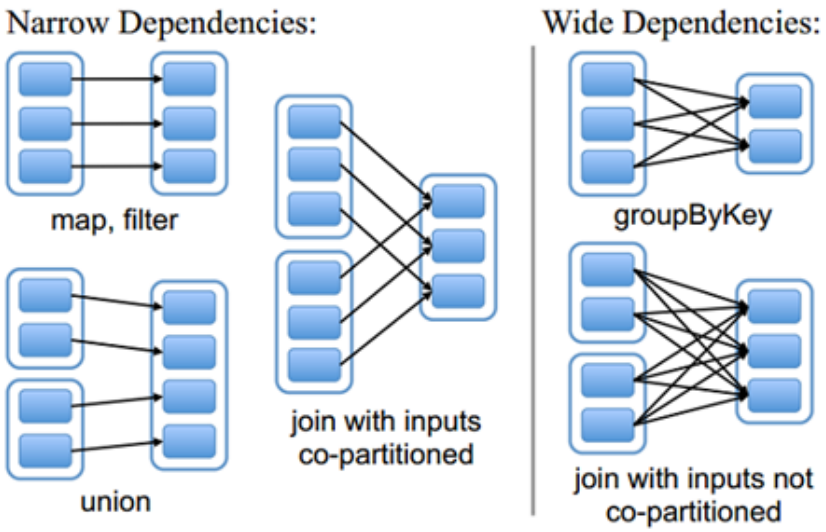
actions要依赖于前面多个块（可能在多个机器上），计算开销比较大，如reduce、count等操作，是非惰性的

- **RDD依赖关系的本质**

由于RDD是粗粒度的操作数据集，每个Transformation操作都会生成一个新的RDD，所以RDD之间就会形成类似流水线的前后依赖关系；RDD和它依赖的父RDD（s）的关系有两种不同的类型，即窄依赖（narrow dependency）和宽依赖（wide dependency）。如图所示显示了RDD之间的依赖关系。

窄依赖：父RDD的每个分区只被子RDD的一个分区使用，如map、union

宽依赖：父RDD的每个分区可能被多个子RDD分区使用，如shuffle(数据的重新分布)



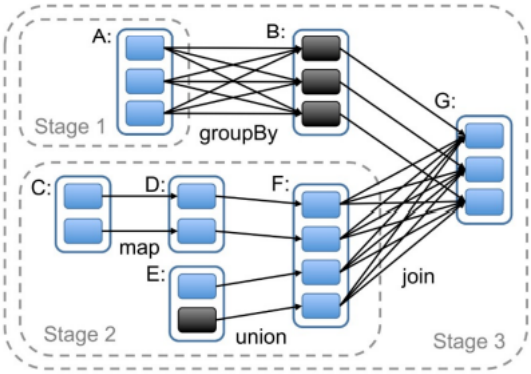
需要特别说明的是对join操作有两种情况：

（1）图中左半部分join：如果两个RDD在进行join操作时，一个RDD的partition仅仅和另一个RDD中已知个数的Partition进行join，那么这种类型的join操作就是**窄依赖**，例如图1中左半部分的join操作(join with inputs co-partitioned)；

（2）图中右半部分join：其它情况的join操作就是宽依赖,例如图1中右半部分的join操作(join with inputs not co-partitioned)，由于是需要父RDD的所有partition进行join的转换，这就涉及到了shuffle，因此这种类型的join操作也是**宽依赖**。

Stage

- 每个阶段stage内部尽可能多地包含一组具有窄依赖关系的transformations操作，以便将它们流水线并行化（pipeline）
- 边界有两种情况：一是宽依赖上的Shuffle操作；二是已缓存分区



变成stage，即将所有任务拆分成许多个stage，构成有向无环图，少占内存

A→B B→G F→G 是宽依赖，其余是窄依赖

为什么B→G是宽依赖？对于G来说是宽依赖，单看B→G其实是窄依赖

上图会一直维护住，如果G挂了重新跑一次这个图

一台机子里的Java对象序列化后打包传给另一个机子，另一个反序列化后重构这个Java对象

RDD 存储等级: 只存内存or存内存或硬盘、有无副本、是否序列化、off_heap？

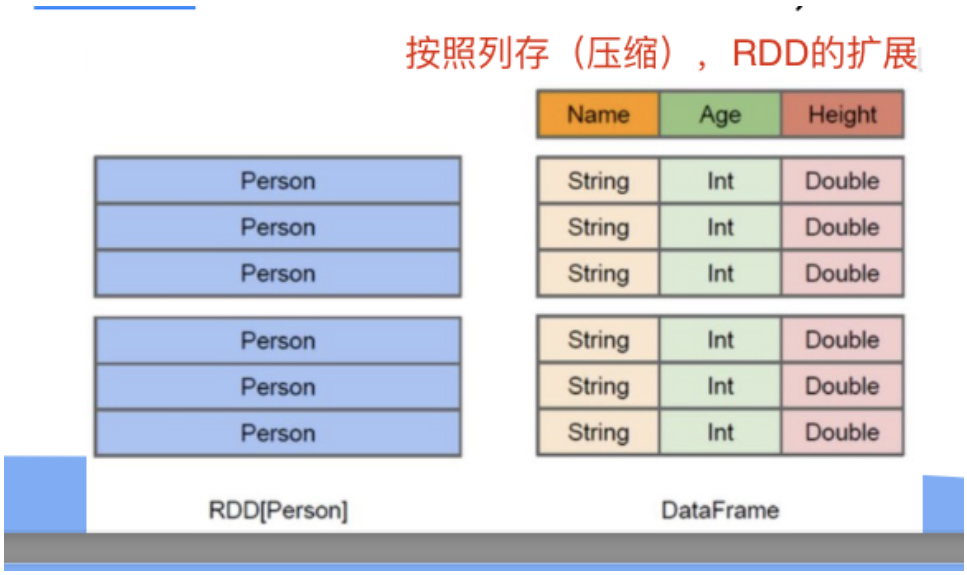
为了使heap要尽量保持小一点，on heap由进程管理（比较小），off heap由操作系统管理（比较大）

使用OFF_HEAP(使用JVM堆外内存)的优点：在内存有限时，可以减少频繁GC及不必要的内存消耗（减少内存的使用），提升程序性能。

SparkSQL is a Spark module for structured data processing

内存数据库，默认按列存，可以按行存

如何组成内存表？Spark提供从不同文件里读取数据，可以跨机



DataFrame

结构化流式数据处理，来的数据会存到内存里，看起来内存里的表会无限增长；实际上是按时间顺序将流式数据分成一小块一小块，每一块做批量处理

GraphX: 图可能会很大，节点(id、属性)、边(起点、终点、区分有向边)

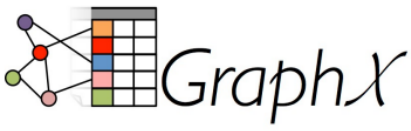
Machine Learning Library (MLlib) Guide

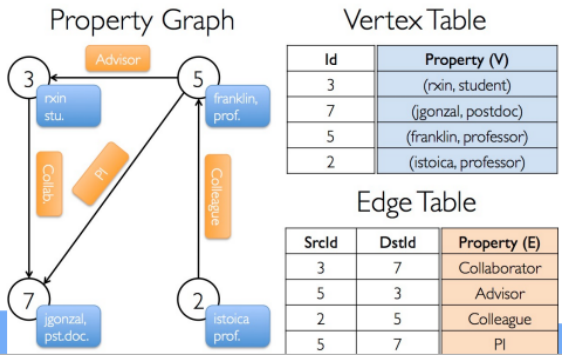
– MLlib is Spark's machine learning (ML) library. Its goal is to make practical machine learning scalable and easy. At a high level, it provides tools such as:

GraphX

– a new component in Spark for graphs and graph-parallel computation.

– At a high level, GraphX extends the Spark RDD by introducing a new Graph abstraction: a directed multigraph with properties attached to each vertex and edge.





Vertex Table

Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

69

这里2、3放在一个机器里，5、7放另一个机器里就不好