**Spark知识点整理**

徐雅婕、潘念然、赵亚然

1. **Spark概述**
2. **Spark的简介**

Spark是由美国加州伯克利大学的AMP实验室于2009年开发的类Hadoop MapReduce的通用并行框架，可用于构建大型的、低延迟的数据分析应用程序，于2013年加入Apache。

Apache Spark是专为大规模数据处理而设计的通用分析引擎。

Apache最重要的三大分布式计算系统开源项目：Hadoop、Spark、Storm。

Spark是为了支持分布式数据集上的迭代作业，但实际上它是对Hadoop的补充，可以在Hadoop文件系统中并行运行。

1. **Spark的特点**

* Spark与Hadoop的对比

|  |  |  |
| --- | --- | --- |
|  | Hadoop | Spark |
| 数据读取 | 从磁盘读取 | 从内存读取 |
| 延迟 | 批处理，延迟 | 低延迟 |
| 处理单元 | 进程 | 线程 |
| 执行机制 | 迭代执行机制 | 基于DAG的任务调度执行机制 |
| 适合的数据处理情景 | 批处理 | 流数据处理，数据查询和统计分析，图数据和机器学习 |

1. **支持的语言**

Scala(原生开发语言)、Python、Java、R语言

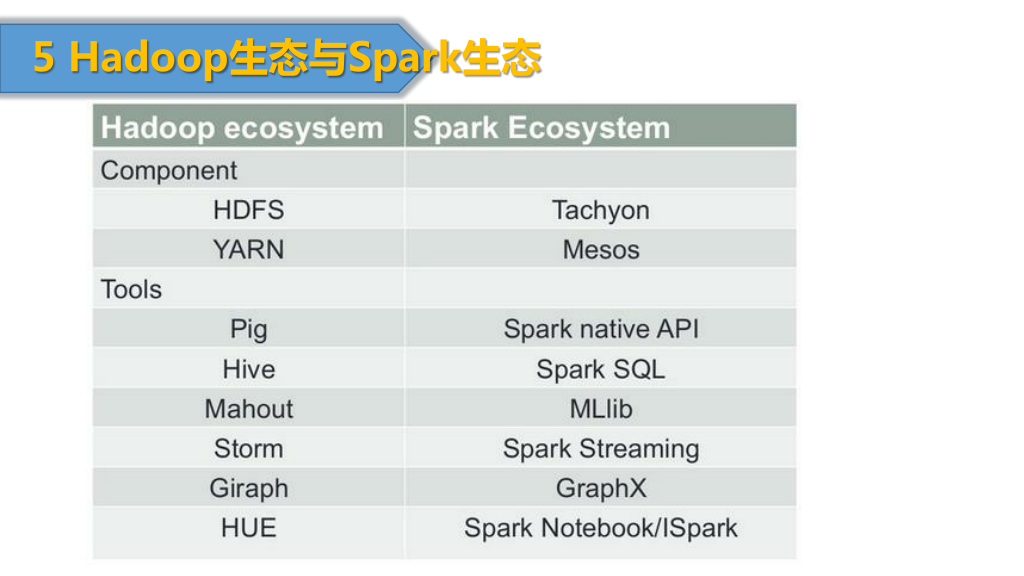
1. **MapReduce与Spark**

* Spark拥有MapReduce所具有的优点;又不同于MapReduce的是，其Job的中间输出结果可以保存在内存中，从而不再需要读写HDFS，因此Spark能更好地适用于数据挖掘与机器学习等需要迭代的MapReduce的算法。
* Hadoop MapReduce与Spark的对比

****

1. **Spark生态系统**

➢Hadoop生态与Spark生态

****

1. **Spark的运行机制**

**1、相关概念与术语***Application:*指用户编写的 Spark应用程序，其中包括一个 Driver功能的代码和分布在集群中多个节点上运行的Executor代码；  
*Driver:* SparkDriver中的（包含 Application的main函数）创建SparkContext对象，准备 Spark应用程序的运行环境， SparkContext负责与ClusterManager通信，进行资源申请、任务的分配和监控等；  
 *Executor:* Applicationworker行在节点上的一个进程，该进程负责运行一些ask，并且负责将数据存到内存或磁盘上每个Application都有各自独立的一批 Executor；  
*Cluster Manager:*指的是在集群上获取资源的外部服务，有三种类型；  
*Standalone:* SparkCluster原的 Manager；  
*Apache Mesos:*种通用 Cluster Manager也支持 Hadoop MapReduce的资源调度框架；  
*Hadoop Yarn:*yarn（Hadoop2）中的 ResourceManager；

*Worker:*集群中任何可以运行 Application代码的节点；  
*Task:*被送到某个 Executor上的工作单元，和 Hadoop Map Reduce中的 Map TaskReduce和 Task概念一样，是运行 Application的基本单位，多个Task组成一个 Stage，而Task的调度和管理 等是由Task Scheduler负责；  
*Job:*包含多个Task组成的并行计算，往往由 Spark Action触发生成，一个Application中往往会产生多个Job；  
*Stage:*每个Job会被拆分成多组Task，形成 TaskSet，被称为 Stage， Stage的划分和调度是由DAGScheduler来负责的， Stage可以分为非最终的 stage（ Shuffle Map Stage）和最终的Stage（ Result Stage）两种， Stage的边界就是发生 Shuffle的地方；  
*DAGScheduler:*根据ob构建 StageD基于dag Directed Acyclic Graph有向无环图），并提交 StageTASKScheduler给划分 Stage的依据是为RDD之间；的依赖的关系找出开销最小的调度方法；

*RDD:*是 Resillient Distributed Dataset（弹性分布式数据集）的简称，是 Spark的一个基本数据结构，表示不可变的分布式对象集，对象集中被划分为若干逻辑分区，每个分区投放到不同的节点并行计算。

**2、Spark的组件**

* Spark SQL可以完成我们传统的查询，统计，分析
* Spark Streaming可以完成我们的数据流程等相关任务
* Mlib（machine learning）它可以完成对大数据机器的数据算法来完成的相关任务
* GraphX(graph)可以完成大数据的图处理的相关任务

Apache Spark Core核心组件

1. **Spark运行架构**

* Spark架构

当有一个应用程序要请求Spark上的资源来进行计算的时候，首先我们要来创建一个Driver Program,其会为每个应用程序创建一个Spark Context ,Spark Context会向Cluster Manager请求计算资源，请求计算节点。Cluster Manager会向Work Node这些节点去询问，看看哪一个Worker Node可以完成这些任务。当它找到了相关的Work Node，可以支持目前应用的相关的任务的时候呢，Spark Context就会直接和执行的任务的Woke Node打交道来发取任务和获取当前的执行状态。在具体的Work Node里面，我们会有Executer负责专门的任务的执行。Executer会和我们刚才的Spark Context进行交互来传递我们要执行的任务来获取我们当前执行任务的状态等等。

* Job=多个stage;
* Stage=多个同种task
* Task分为ShuffleMapTask和ResultTask
* Dependency分为ShuffleDependency和NarrowDependency
* 集群资源管理器：在standalone模式中即为Master主节点，控制整个集群，监控Worker。在YARN模式中为资源管理器；
* 应用的任务控制节点：运行Application的main（）函数；
* 工作节点:负责运行作业任务的工作节点，控制工作节点
* 执行器：是为某个Application运行在worker node上的一个进程

**4、Spark运行原理**

其核心为RDD（Resilient Distributed Datased）

RDD是一个弹性分布式数据集，主要属性如下：

* 一组分片，数据集的基本组成单位
* 一个计算每个分片的函数
* 对parent RDD的依赖，这个依赖描述了RDD之间的lineage
* 对于key-value的RDD,还可以有一个partitioner,可选

1. 支持跨集群的分布式数据结构，只读的存储
2. 可以根据数据记录的key对结构进行分区
3. 提供粗粒度的操作，且这些操作都支持分区
4. 数据存储在内存中，从而提供低延迟性

RDD的转换和执行：

转换可以有很多个，执行只能有一个。

**5、Spark流程运行图**

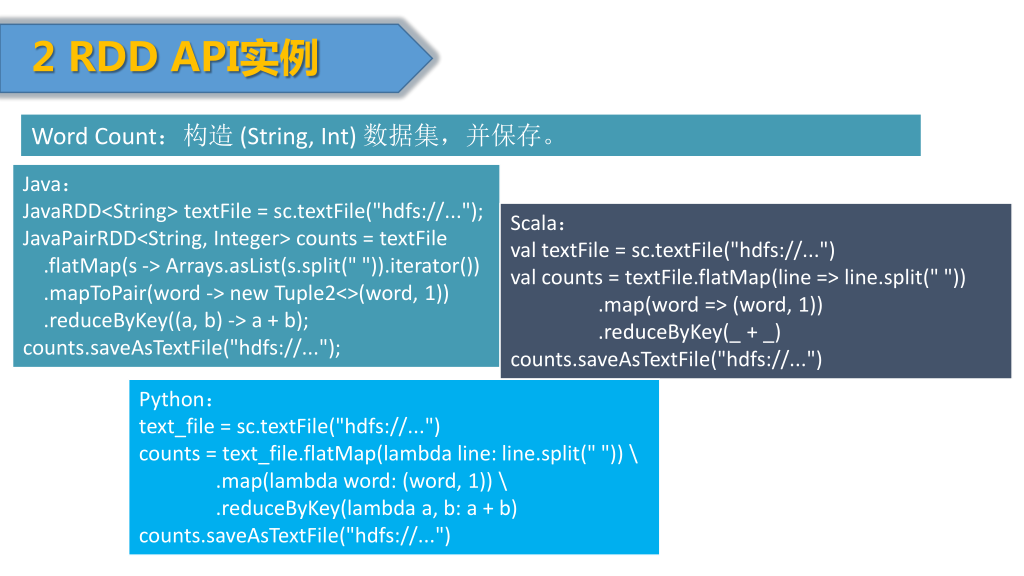
1. 构建Speak Application的运行环境，启动SparkContext, SparkContext向资源管理器申请运行Executor资源；
2. 资源管理器分配Executor资源并启动StandaloneExecutorBackend,Executor运行情况将随着心跳发送到资源管理器上
3. Spark Context构建成DAG图，将DAG图分解成stage,并把Taskset发送给Task Scheduler。Executor向Spark Context申请Task, Task Scheduler将Task发放给Executor运行，同时Spark Context将应用程序代码发放给Executor.
4. Task在Executor上运行，运行完毕后释放所有资源。
5. **Spark API实例**

**1、Spark的API**

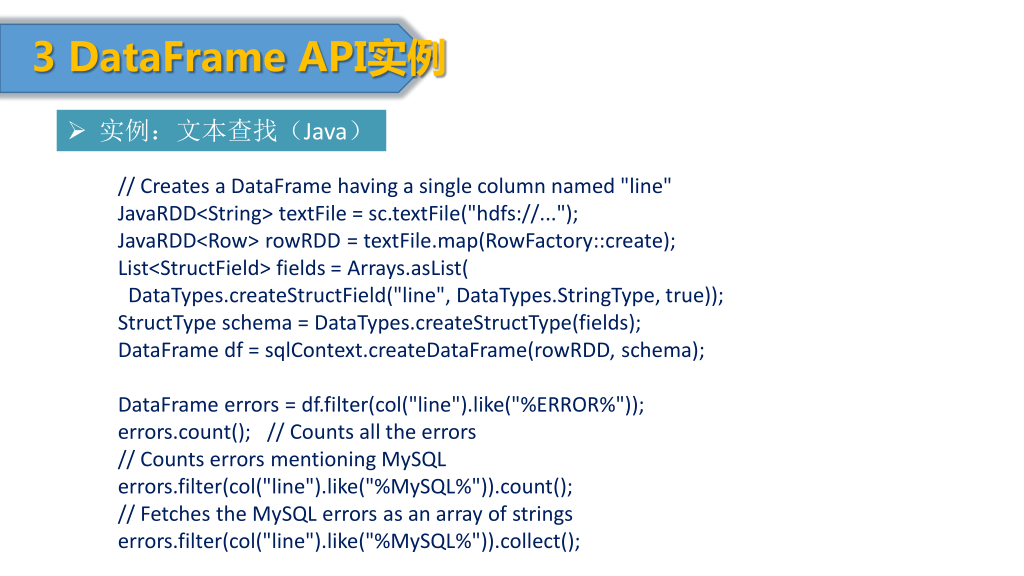
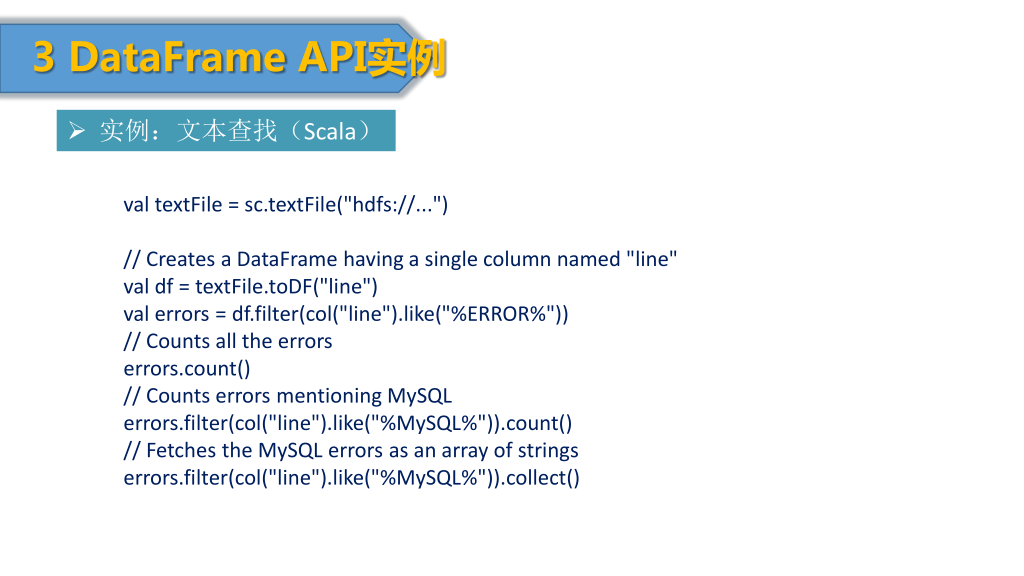
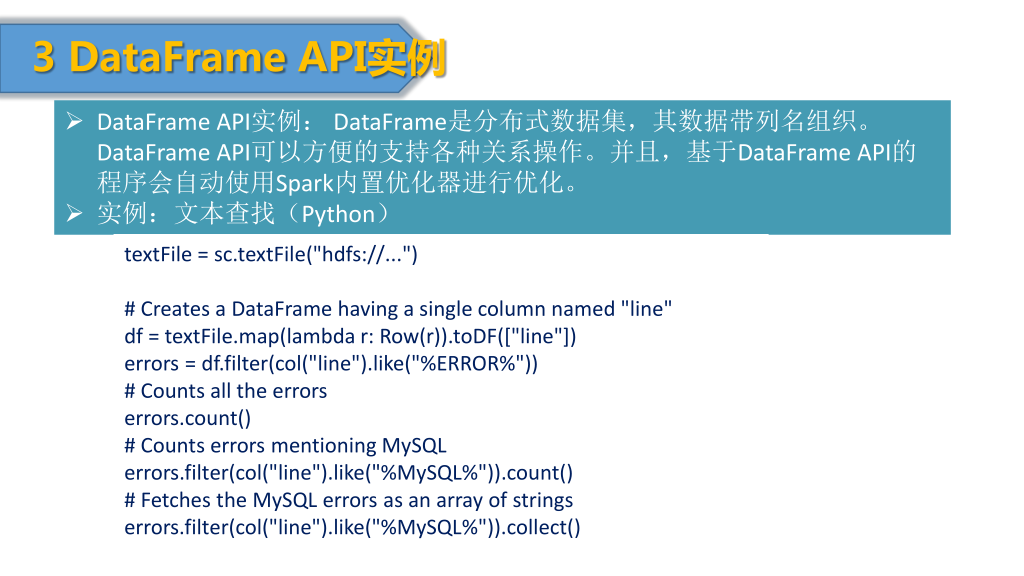
Spark是基于分布式数据集的概念的，可以包含任意的Java、Python对象。我们只需要基于这些外部数据构造数据集，然后对这些数据集进行并行操作。 Spark API的基础构建是RDD API，在RDD API之上，又提供了高层的API供使用，例如DataFrame API，机器学习API。这些更高层次的API提供了特定数据操作的方法。

1. **API实例**

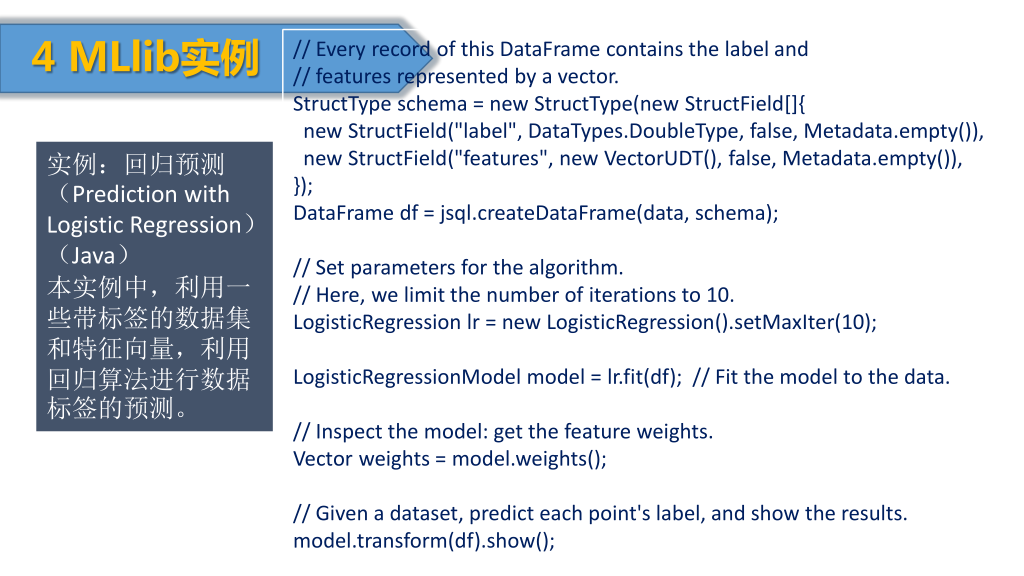
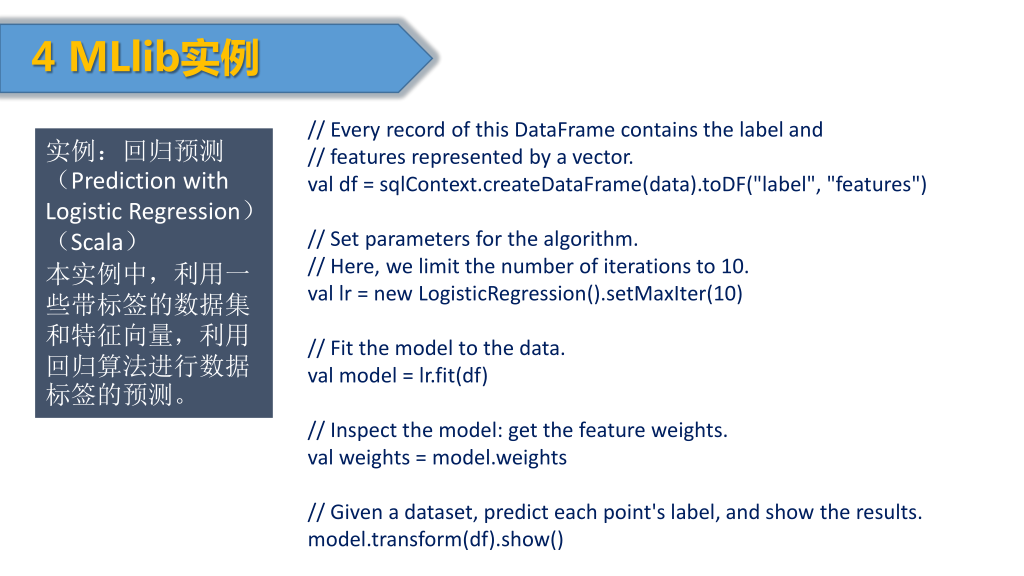
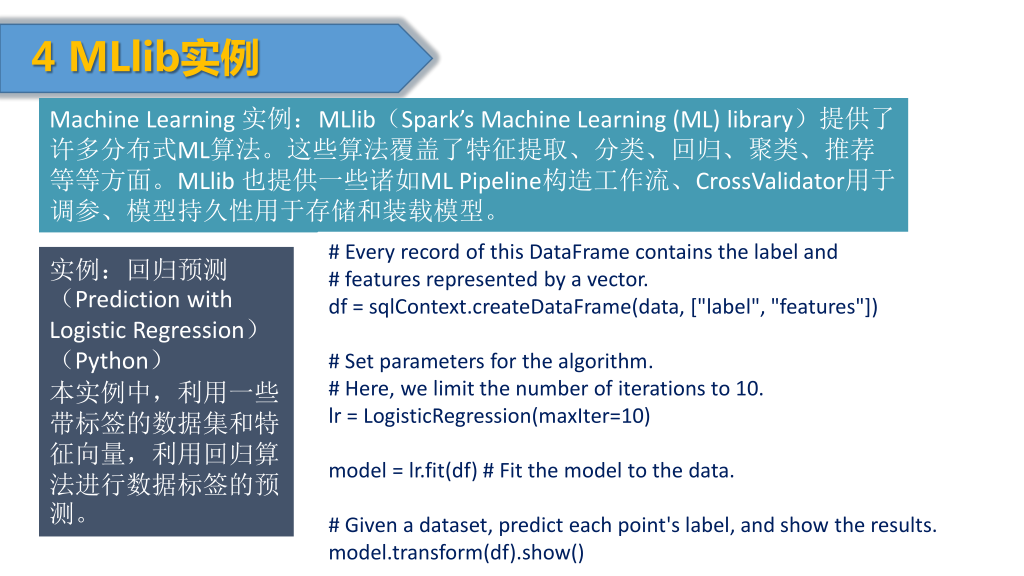
* RDD API实例

****

* DataFrame API实例

****

* MLlib实例

****

1. **Spark安装与运用**
2. **spark的安装**

推荐：把spark安装于Linux系统。具体安装Apache spark的步骤如下：

Step1：验证Java是否正确安装

$Java-version

若正确输出Java版本信息，则说明Java已经正确安装，否则需要安装Java，并设置相关路径。

Step2：验证Scala是否正确安装

$Scala-version

若正确输出Scala版本信息，则说明Scala正确安装，可跳过step3和step4，否则需要安装Scala。

Step3：下载Scala

下载最新版本的Scala，这里使用的是Scala-2.11.6

Step4：

解压tar文件，命令如下：

$ tar xvf scala-2.11.6.tgz

把解压后的文件移到/usr/local/scala(也可以是制定的其他文件夹位置)

$ su-

Password:

#cd/home/Hadoop/Downloads/

#mv scala-2.11.6/usr/local/scala

#exit

设置Scala路径：

$export PATH=$PATH:/usr/local/scala/bin

验证安装是否成功：

$scala-version

正确显示Scala版本信息，则说明安装成功

Step5：下载Apache Spark

到Apache spark 官网下载spark

Step6：安装spark

按解压，修改存放文件夹，设置环境变量几个步骤进行spark的设置，参考如下：

$tar xvf spark-1.3.1-bin-hadoop2.6.tgz

$su-

Password:

#cd/home/Hadoop/Downloads/

#mv spark-1.3.1-bin-hadoop2.6/usr/local/spark

#exit

设置环境变量：打开~/.bashrc文件，并且在文件后面加上如下路径：

Export PATH=$PATH:/usr/local/spark/bin

执行~/.bashrc file,使设置生效。

$ source~/.bashrc

Step 7:验证spark安装情况

$spark-shell

输出如下，则说明安装成功。

Spark assembly has been built with Hive ,including Datanucleus jars on classpath (这里省略部分输出)

Welcome to



Using Scala version 2.10.4(java HotSpot(TM)64-Bit Server VM,java1.7.0-710)

Type in expression to have them evaluated.

Spark context available as sc

Scala>

1. **scala程序开发环境与步骤**

spark shell 逐步允许可以马上看到程序运行结果，这对于程序的调试和编写具有很好的实时交互性。在所有的代码都写好时，还可以选择把代码存储成一个文件，然后打包成一个应用程序，提交给spark运行。以Scala为例，大数据运行程序利用spark进行数据处理程序开发环境可以构造如下。

常用开发环境：intelliJ IDEA、Eclipse等等

项目的组织：Maven、SBT(Simple Build Tool,SBT是SCALA平台上标准的项目 构建工具)、IVY(免费的Java依赖管理器)；

项目依赖项的添加；

项目的打包