大数据编程模型和使用技巧

SPARK

陈一帅

yschen@bjtu.edu.cn

北京交通大学电子信息工程学院

内容

- 图计算模型
- Spark 简介
- 算法实例
- Spark SQL 和 DataFrame
- 编程实例

图执行模型

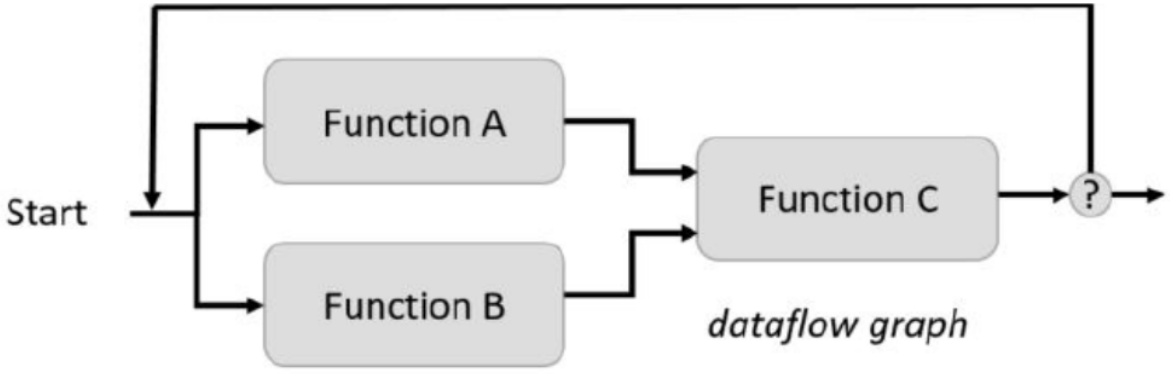
- 计算由有向图（通常为非循环图）的任务图表示
- 执行从图的源开始
- 当节点的所有父节点都已完成时，就安排该节点执行
- 图节点执行涉及一个或多个分布式节点的并行操作

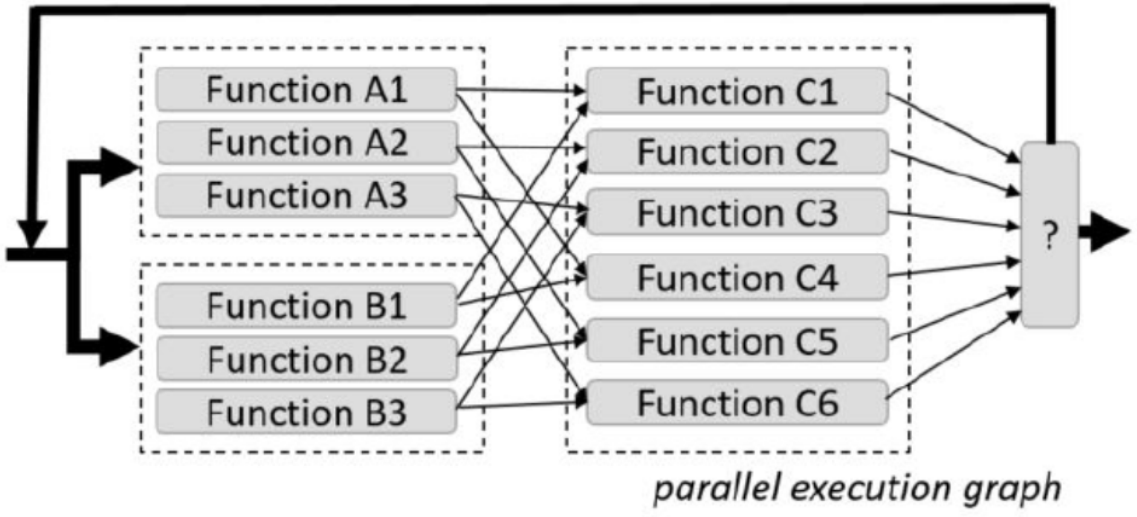
图执行模型

- 可以手动构造图形，也可以由编译器从程序中隐式或显式地构造图形
- 大数据工具
  - Spark，Apache Flink，Storm，Google Dataflow
- 机器学习工具
  - Google TensorFlow，Microsoft Cognitive Toolkit

图 Dataflow 执行

- 数据流图，编译为并行执行图





内容

- 图计算模型
- Spark 简介
- 算法实例
- Spark SQL 和 DataFrame
- 编程实例

Hadoop 的局限

- 线性数据流结构
- 基于磁盘
  - 程序从磁盘读取输入数据
  - 对输入数据进行映射（Map）功能
  - 简化（Reduce）映射结果
  - 将结果存储在磁盘上
- 速度慢

Spark

- 机器学习训练中需要大量迭代
- Hadoop
  - 在两个 MapReduce 作业之间重用数据，需要写入外部磁盘
  - 导致大量开销
- Spark
  - 在本地工作进程中的动态随机存取存储器（DRAM）上进行缓存，大大减少开销

Spark

- 基于内存
  - 比基于磁盘的 Hadoop 快得多
- 图执行模型
  - 允许 MapReduce 迭代以及更有效的数据重用
- 交互式操作界面
  - 类似 Matlab
- 可以在 YARN 和 Mesos 上运行，也可以在笔记本电脑和 Docker 容器中运行

伪分布式本地模式

- 在一台机器上运行
  - 每个 CPU 内核一个执行程序
- 不需要分布式存储，可使用本地文件系统
- 方便开发、测试

Spark Core

- 提供分布式任务调度、基本 I/O 功能
- 集群管理
  - 自己的独立调度程序，Hadoop YARN 或 Apache Mesos
- 存储接口
  - 支持 HDFS，MapR 文件系统（MapR-FS），Cassandra，OpenStack Swift，Amazon S3 等接口

Spark 模块

- Spark SQL 处理结构化数据
- Spark Streaming 处理实时数据流
- MLlib 包含常见的机器学习功能
- GraphX 用于处理网络图

分布式执行模型

- 开发人员编写驱动程序
  - 定义一个或多个 RDD，及其操作
  - 连接到一组 Worker
- Worker
  - 运行在集群服务器上
  - 可将 RDD 分区 存储在 RAM 中

Spark RDD

- Resilient Distributed Dataset
  - 弹性分布式数据集（RDD）
  - Spark 的核心数据结构
- 分布在计算机集群上的一个只读的数据集
  - 利用集群中的持久性数据块得以缓存、复制和分发
  - 可以使用 join 和各种 Map and Reduce 转换操作来创建新的 RDD

RDD 容错

- 一个RDD的lineage记录了它如何从其他稳定存储的数据集派生计算过来的
- 在某些数据丢失情况下，通过lineage可重建 RDD
- 但需要重新计算，因此耗费 CPU

性能 （2014 年）

- 对 100 TB 数据（1 万亿条记录）进行分类
- 前世界纪录
  - Yahoo!使用 2100 个节点的 Hadoop MapReduce 集群创造的 72 分钟
- Spark
  - 206 个 EC2 节点，23 分钟
  - 所有排序都在磁盘（HDFS）上进行，没有使用 Spark 的内存缓存

性能 （2014 年）

- 1 PB 数据（10 万亿条记录）排序
  - 190 台计算机
  - 不到 4 小时
- 比较
  - 基于 Hadoop MapReduce
  - 3,800 台计算机
  - 16 小时

RDD 操作

- 两种类型的操作
- Transformations 变换
  - 将 RDD 映射到新 RDD
  - Lazy 操作，只记录要执行的操作，并不真正执行
- Action 动作
  - 返回值给主程序
  - 通常是 read-eval-print 循环，例如 Jupyter
  - 启动真正的计算，以将值返回到程序或将数据写入外部存储

Transformation

- map
  - 一一映射
- reduce
  - 合并 value
  - 多个 到 1个
- reducebykey
  - 按 Key 合并 Value

Map：——映射

- 示例（scala）
  - https://sparkbyexamples.com/pyspark/pyspark-map-transformation/

```
rdd2=rdd.map(lambda x: (x,1))
```

Reduce：合并

- 示例（scala）
  - https://sparkbyexamples.com/apache-spark-rdd/spark-rdd-reduce-function-example/

```
listRdd.reduce(_ + _)
```

ReducebyKey：按“键”合“值”

- 示例（scala）
  - https://sparkbyexamples.com/apache-spark-rdd/spark-reducebykey-usage-with-examples/

```
val rdd2=rdd.reduceByKey(_ + _)
```

Action

- count
  - 计数（返回数据集中元素的数量）
- collect
  - 收集（返回元素本身）
  - 示例（Python）：https://sparkbyexamples.com/pyspark/pyspark-collect/
- save
  - 保存（将数据集输出到存储系统）

练习

- Spark By Examples
  - https://sparkbyexamples.com/
  - https://github.com/spark-examples
- Spark RDD Tutorial
  - https://sparkbyexamples.com/spark-rdd-tutorial/
- Spark with Python (PySpark) Tutorial For Beginners
  - https://sparkbyexamples.com/pyspark-tutorial/
- PySpark RDD Tutorial | Learn with Examples
  - https://sparkbyexamples.com/pyspark-rdd/

RDD 编程： Persistence 控制

- 在多次迭代的工作中，可能有必要将一些版本的 RDD 存起来，以减少故障恢复时间
- 用户可以调用 persist，带上一个 reliable（可靠）标志，来执行此操作

基于 RDD Partition 的并行

- 在每个分区上进行并行计算
- Partition 优化
  - 如果两个数据集将要通过 join 连接到一起，可将它们通过相同的类分区，对后面的 join 有帮助
  - 要 join 的每行的两个数据都在一个分区上
  - join操作在一个分区上可以完成
- 可以编写一个自定义分区程序类来进行分区

```
links = spark.textFile(…).map(…) .partitionBy(myPartFunc)
```

内容

- 图计算模型
- Spark 简介
- 算法实例
- Spark SQL 和 DataFrame
- 编程实例

Spark Euler 计算 Pi

- Euler 公式 $\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \cdots = \frac{\pi^2}{6}$
    - 双核 CPU，分为两个 partition，平行

```
n = 1000000
ar = np.arange(n)
dat = ar.parallelize(ar, 2)
sqrs = dat.map(lambda i: 1.0/(i+1)**2)
t0 = time.time()
x = sqrs.reduce(lambda a,b: a+b)
t1 = time.time()
print("x=%f"%x)
print("time=%f"%(t1-t0))
```

例：K-means 聚类

- k 类
- 函数1：寻找最近的类中心点
  - 输入：输入点 p；当前 k 个 类的中心点列表 kPoints
  - 输出：KPoints 中和 p 最近的点的 index

```
def closestPoint(p, kPoints):
    bestIndex = 0
    closest = float("+inf")
    for i in range(len(kPoints)):
        tempDist = np.sum((p - kPoints[i]) ** 2)
        if tempDist < closest:
            closest = tempDist
            bestIndex = i
    return bestIndex
```

例：K-means 聚类

- 归到 k 类
- 将 data 中的每个点 p 都映射为
  - (j, (p,1))
  - j = closestPoint(p, kPoints)
  - (p,1)是一个常见的 MapReduce 习惯用法，请掌握

```
data.map(lambda p:(closestPoint(p,kPoints),(p,1)))
```

例：聚类

- 求 k 类的中心点
  - 找出属于类 j 的所有的点，取它们坐标的均值
- 输入：
- 求均值
  - 用 reduceByKey
  - j 是 Key，x 是 p，y 是 1
  - 得到一个大小为 k 的数组

```
reduceByKey(lambda x,y:(x[0]+y[0],x[1]+y[1]))
```

例：聚类完整代码

```
tempDist = 1.0
while tempDist > convergeDist:
    newPoints = data \
            .map( lambda p: (closestPoint(p, kPoints), (p, 1))) \
            .reduceByKey(lambda x, y : (x[0] + y[0], x[1] + y[1])) \
            .map(lambda x : (x[0], x[1][0]/ x[1][1])) \
            .collect()

    tempDist = sum(np.sum((kPoints[i] - y) ** 2)  \
                    for (i, y) in newPoints)
    for (i, y) in newPoints:
        kPoints[i] = y
```

- reduceByKey 得到大小为 k 数组
  - 对每一个类 j，计算 —，得到属于它的所有点的均值
  - collect 它，作为新的 kPoints
  - 注：仅示例，这不是最好的 k-means 算法实现，Spark 机器学习库有更好的实现

内容

- 图计算模型
- Spark 简介
- 算法实例
- Spark SQL 和 DataFrame
- 编程实例

Spark SQL

- 对结构化数据，能够执行 SQL 命令
- Spark SQL 包括优化器，列存储和代码生成，可快速回答查询
- 可扩展到数千个工作节点

例：Spark SQL 文件输入

- 数据文件
  - 例：建筑温度

```
Date, Time, desired temp, actual temp, buildingID
3/23/2016, 11:45, 67, 54, headquarters
3/23/2016, 11:51, 67, 77, lab1
3/23/2016, 11:20, 67, 33, coldroom
```

例：Spark SQL 文件输入

- Spark SQL 文件读入
  - 将文件加载到 Spark 中，通过 textFile 应用于 Spark 上下文对象，创建文本文件 RDD
  - 过滤掉标题行，将其余行映射到类型化的元组，将文本文件 RDD 转换为元组 RDD

```
from pyspark.sql.types import *
hvacText = sc.textFile("/pathto/file/hvac.csv")
hvac = hvacText.map(lambda s: s.split(",")) \
               .filter(lambda s: s[0] != "Date") \
               .map(lambda s:(str(s[0]), str(s[1]),
                              int(s[2]), int(s[3]), str(s[4]) ))
sqlCtx = SQLContext(sc)
hvacSchema = StructType([StructField("date", StringType(), False),
                         StructField("time", StringType(), False),
                         StructField("targettemp", IntegerType(), False),
                         StructField("actualtemp", IntegerType(), False),
                         StructField("buildingID", StringType(), False)])
hvacDF = sqlCtx.createDataFrame(hvac, hvacSchema)
```

例：Spark SQL

- 使用 sql()方法执行 SQL 命令
- 结果返回为 DataFrame

```
x = sqlCtx.sql('SELECT buildingID from hvac')
```

例：Spark SQL

- 魔术运算符 %% sql_show
  - Jupyter 和 IPython 魔术运算符定义语言小型扩展
  - 能以自然方式输入 SQL 命令，结果打印为表格

```
%%sql_show
SELECT buildingID ,
       (targettemp - actualtemp) AS temp_diff ,
       date FROM hvac
WHERE date = "3/23/2016"
```

```
+------------+---------+---------+
|  buildingID|temp_diff|     date|
+------------+---------+---------+
|headquarters|       13|3/23/2016|
|        lab1|      -10|3/23/2016|
|    coldroom|       34|3/23/2016|
+------------+---------+---------+
```

Dataframe

- 针对结构化数据的一种数据抽象 API
- 例：按年龄统计人数

```
countsByAge=df.groupBy(“age”).count()
countsByAge.show()
```

- 以 JSON 格式将结果保存到 S3

```
countsByAge.write.format(“json”).save(“s3a://…”)
```

DataFrame 示例：文本搜索

- 创建一个只有一个名为“line”的列的 DataFrame

```
textFile=sc.textFile(“hdfs:// …”)
df=textFile.map(lambda r: Row(r)).toDF([“line”])
```

- 计数所有错误

```
err=df.filter(col(“line”).like(“%ERROR%”))
err.count()
```

DataFrame 示例

- 计数提及 MySQL 的错误

```
err.filter(col("line").like("%MySQL%")).count()
```

- 以字符串数组的形式获取 MySQL 错误

```
err.filter(col("line")
        .like("%MySQL%")).collect()
```

练习

- Spark DataFrame & Dataset Tutorial
  - https://sparkbyexamples.com/spark-dataframe-tutorial/
- Spark SQL tutorial
  - https://sparkbyexamples.com

内容

- 图计算模型
- Spark 简介
- 算法实例
- Spark SQL 和 DataFrame
- 编程实例

例：网站用户访问计数

- 需求
  - 统计特定用户列表中的用户访问Wikipedia的总数
- 流程
  - 加载 Wikipedia 访问日志
  - 把每一行，转换为数组：用户，访问次数
  - 过滤出需要查找的用户的数据
  - 汇总每一个用户的访问记录，计算总数

加载数据

- 从 S3 加载一小部分 Wikipedia 访问日志（从 2008 年到 2010 年）

```
rawdata = sc.textFile("s3://support.
    elasticmapreduce/bigdatademo/sample/wiki")
rawdata.count()
rawdata.getNumPartitions()
```

- 将 RDD 重新划分为 10 段，以便后面更好地利用 Spark 的并行性

```
rawdata = rawdata.repartition(10)
```

文本拆分为数组

- 通过分割空白字符将每行转换为一个数组

```
def parseline(line):
    return np.array([x for x in line.split(' ')])

data = rawdata.map(parseline)
```

过滤

- 过滤函数：检查row[1]是否在 namelist 中

```
def filter_fun(row, titles):
    for title in titles:
        if row[1].find(title) > -1:
            return True
    else:
        return False
```

- 过滤数据

```
fd=data.filter(lambda p:filter_fun(p,namelist))
```

统计

- 返回人名

```
def mapname(row, names):
    for name in names:
        if row[1].find(name) > -1:
            return name
    else:
        return 'huh?'
```

- 计数
  - Map：用（name，count）对替换每一行
  - Reduce by name: 加 count

```
rd=fd.map(lambda row:(
        mapname(row,namelist),int(row[2])))
     .reduceByKey(lambda v1, v2: v1+v2)
```

小结

- 图计算模型
- Spark 简介
- 算法实例
- Spark SQL 和 DataFrame
- 编程实例

练习：安装

- Windows安装
  - https://sparkbyexamples.com/spark/apache-spark-installation-on-windows/
  - https://bigdata-madesimple.com/guide-to-install-spark-and-use-pyspark-from-jupyter-in-windows/
  - https://blog.csdn.net/SunChao3555/article/details/84202769
- Linux
  - https://sparkbyexamples.com/spark/spark-installation-on-linux-ubuntu/
  - https://blog.csdn.net/weixin_42902669/article/details/103055046
  - https://cloud.tencent.com/developer/article/1614367
  - https://blog.csdn.net/hecongqing/article/details/102938435

练习：编程（Python）

- https://github.com/piotrszul/spark-tutorial

1、文本单词计数

- 0.1_Welcome.ipynb
- prince_by_machiavelli.txt 小王子
- RDD，DF，Word count

2、RDD基础练习

- 1.1_RDD-Basics.ipynb
- 字符串RDD
- saveAsTextFile，flatmap，filter number，map，reduce

3、DataFrame基础练习

- 2.1_StructuredData-Introduction.ipynb
- 示例数据 row
- DataFrame, Schema
- filter, sort, col, select, groupby, join
- limit, toPandas
- write.csv

4、输入输出练习

- 2.2_StructuredData-Formats.ipynb*
- read.csv: 气温，tweet
- 写csv，parquet存储

参考材料

- Spark 原理和实践
  - https://yishuai.github.io/spark
  - B站视频，PDF
  - 去年的材料，更详细的介绍
  - 华为实验环境可能不免费了，其它内容有效
- 张璇，Python机器学习入门指南（Sklearn）
  - https://yishuai.github.io/lab/zhangxuan-guide.docx
- Jupyter Notebook 介绍
  - https://www.bilibili.com/video/BV1WZ4y1g7Zt/