技术文档

技术可行性分析

技术验证

项目计划

服务器公网ip 119.23.212.82

# 环境变量

　　这里讲两句题外话，各位看官不关心的话可以跳过这里，不影响后续的安装步骤。  
　　在软件安装的时候，相信各位没少遇到过环境变量和系统变量，所以这里就来扒一扒令人头疼的**PATH**, **CLASSPATH**和**JAVA\_HOME**等参数的具体含义。

### 1.3.1 环境变量、系统变量和用户变量

* 环境变量包括系统变量和用户变量
* 系统变量的设置针对该操作系统下的所有用户起作用；
* 用户变量的设置只针对当前用户起作用

如果对这些概念还不是特别熟悉的，建议先看完下面几个点之后，再回过头来看这三句话。

### 1.3.2 PATH

　　也就是上一步设置的系统变量，告诉操作系统去哪里找到Java.exe的执行路径，当你在命令行窗口冷不丁的敲上如下命令的时候，

java -version

* 1

操作系统首先会一惊，What the hell does “java” mean? 不过吐槽归吐槽，活还是得干，于是悠悠的记起来了盖茨爸爸说过的三句话：

1. 当你看不懂命令行窗口中的一个命令的时候，你首先去你所在的当前目录下找找，是否有这个命令的.exe程序？如果有，那就用它来启动执行；
2. 如果没有，千万别放弃，记得要去**Path**系统变量下的那些目录下去找一找，如果找到了，启动并执行命令；
3. 如果上面两个地方依然还没找到，那你就撒个娇，报个错好了。

所以我们将JDK安装目录下的bin文件夹添加到**Path**系统变量的目的也就在这里，告诉操作系统：如果在当前目录下找不到java.exe，就去**Path**系统变量里的那些路径下挨个找一找，直到找到java.exe为止。那为什么要设置bin文件夹，而不是JDK安装的根目录呢？原因就在于根目录下没有java.exe啊，只有bin文件夹下才有啊喂……

如果只是在命令行窗口下运行一下java的命令，那其实也可以不设置系统变量，只是每次在命令行窗口运行java的命令时，都必须带上一长串路径名，来直接指定java.exe的位置，如下所示。

C:\Users\weizierxu>F:\Program Files\Java\jdk1.8.0\_101\bin\java.exe -version

'F:\Program' 不是内部或外部命令，也不是可运行的程序

或批处理文件。

* 1
* 2
* 3

注意：这里报错的原因并不是说直接指定java.exe的路径名这种方式有问题，而是命令行下无法解析带有空格的路径名，所以需要用到双引号，­­如下：

C:\Users\weizierxu>"F:\Program Files"\Java\jdk1.8.0\_101\bin\java.exe -version

java version "1.8.0\_101"

Java(TM) SE Runtime Environment (build 1.8.0\_101-b13)

Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)

* 1
* 2
* 3
* 4

### 1.3.3 CLASSPATH

　　CLASSPATH是在Java执行一个已经编译好的class文件时，告诉Java去哪些目录下找到这个class文件，比如你的程序里用到某个Jar包（Jar包里的都是已经编译好的class文件），那么在执行的时候，Java需要找到这个Jar包才行，去哪找呢？从CLASSPATH指定的目录下，从左至右开始寻找（用分号区分开的那些路径名），直到找到你指定名字的class文件，如果找不到就会报错。这里做一个实验，就能明白具体是什么意思了。  
　　首先，我在F:\Program Files\Java目录下，利用Windows自带的记事本写了一个类似于Hello World的程序，保存为testClassPath.java文件（注意后缀名得改成java），内容如下：

public class testClassPath{

public static void main(String[] args){

System.out.println("Hello, this is a test on CLASSPATH!");

}

}

* 1
* 2
* 3
* 4
* 5

然后，我将cmd的当前目录切换到（通过cd命令）F:\Program Files\Java目录下，然后用javac命令来对这个.java文件进行编译，如下图所示：  
  
  
从上图中可以看到，javac命令可以正常使用（没有任何输出的就表明正确编译了），这是因为执行该命令的javac.exe同样存在于JDK安装路径下的bin目录中，而这个目录我们已经添加到**Path**系统变量中去了，所以cmd能够认识这个命令。这个时候可以看到F:\Program Files\Java目录下多了一个testClassPath.class文件。不过运行这个class文件的时候，报错了。这个时候，CLASSPATH就派上用场了，和**1.2**节中对**Path**系统变量设置的方法一样，这里在CLASSPATH（如果系统变量的列表中没有CLASSPATH这个选项，那么点击**新建**，然后添加路径即可）中最后面添加上;.，英文的分号表示和前面已有的路径分割开，后面的小点.表示当前目录的意思。  
这个时候记得要另起一个新的cmd窗口，然后利用cd命令切换到testClassPath.class所在目录，然后再去执行，便可以成功得到结果了。

F:\Program Files\Java>java testClassPath

Hello, this is a test on CLASSPATH!

* 1
* 2

　　因此，和**Path**变量不同的是，Java在执行某个class文件的时候，并不会有默认的先从当前目录找这个文件，而是只去**CLASSPATH**指定的目录下找这个class文件，如果**CLASSPATH**指定的目录下有这个class文件，则开始执行，如果没有则报错（这里有去当前目录下找这个class文件，是因为当前路径通过.的方式，已经添加到了**CLASSPATH**系统变量中）。  
　　上面讲的指定**CLASSPATH**系统变量的方法，都是直接写死在系统变量中的，为了避免造成干扰（比如多个同名class文件存在于多个路径中，这些路径都有添加到**CLASSPATH**系统变量中，由于在找class文件的时候，是从左往右扫描**CLASSPATH**系统变量中的路径的，所以在利用java testClassPath方法执行的时候，运行的便是位置在**CLASSPATH**系统变量中最左边的那个路径中，对应的class文件，而这显然不是我们想要的结果），因此在诸如Eclipse等等这些IDE中，并不需要人为手动设定**CLASSPATH**系统变量，而是只设定当前程序的特定的**CLASSPATH**系统变量，这样便不会影响到其他程序的运行了。

### 1.3.4 JAVA\_HOME

**JAVA\_HOME**并不是Java本身所需要的参数，而是其他的一些第三方工具需要这个参数来配置它们自己的参数，它存在的意义无非是告诉那些软件，我的JDK安装在这个目录下，你如果要用到我的Java程序的话，直接来我这个目录下找就好了，而JAVA\_HOME就是JDK的安装路径名。比如我的JDK安装在F:\Program Files\Java\jdk1.8.0\_101目录下（注意该目录下的bin目录，就是在**1.3.2**节里**Path**系统变量中要添加的值），那么**JAVA\_HOME**里要添加的值便是F:\Program Files\Java\jdk1.8.0\_101，以后碰到类似HOME的系统变量，都是软件的安装目录。

# 大数据技术生态

作者：有点文  
链接：https://www.zhihu.com/question/27974418/answer/156227565  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

**学习很重要的是能将纷繁复杂的信息进行归类和抽象。**  
对应到大数据技术体系，虽然各种技术百花齐放，层出不穷，但大数据技术本质上无非解决4个核心问题。

1. **存储，**海量的数据怎样有效的存储？主要包括hdfs、Kafka；
2. **计算，**海量的数据怎样快速计算？主要包括MapReduce、Spark、Flink等；
3. **查询，**海量数据怎样快速查询？主要为Nosql和Olap，Nosql主要包括Hbase、 Cassandra 等，其中olap包括kylin、impla等，其中Nosql主要解决随机查询，Olap技术主要解决关联查询；
4. **挖掘，**海量数据怎样挖掘出隐藏的知识？也就是当前火热的机器学习和深度学习等技术，包括TensorFlow、caffe、mahout等；

**大数据技术生态其实是一个江湖....**

在一个夜黑风高的晚上，江湖第一大帮会Google三本阵法修炼秘籍流出，大数据技术江湖从此纷争四起、永无宁日...

这三本秘籍分别为：

* 《Google file system》：论述了怎样借助普通机器有效的存储海量的大数据；
* 《Google MapReduce》：论述了怎样快速计算海量的数据；
* 《Google BigTable》：论述了怎样实现海量数据的快速查询；

**以上三篇论文秘籍是大数据入门的最好文章，通俗易懂，先看此三篇再看其它技术；**

在Google三大秘籍流出之后，江湖上，致力于武学开放的apache根据这三本秘籍分别研究出了对应的武学巨著《hadoop》，并开放给各大门派研习，Hadoop包括三大部分，分别是hdfs、MapReduce和hbase：  
hdfs解决大数据的存储问题。  
mapreduce解决大数据的计算问题。  
hbase解决大数据量的查询问题。

之后，在各大门派的支持下，Hadoop不断衍生和进化各种分支流派，其中最激烈的当属计算技术，其次是查询技术。存储技术基本无太多变化，hdfs一统天下。

以下为大概的演进：

1，传统数据仓库派说你mapreduce修炼太复杂，老子不会编程，老子以前用sql吃遍天下，为了将这拨人收入门下，并降低大数据修炼难度，遂出了hive，pig、impla等SQL ON Hadoop的简易修炼秘籍；

2，伯克利派说你MapReduce只重招数，内力无法施展，且不同的场景需要修炼不同的技术，太过复杂，于是推出基于**内力（内存）的《Spark》**，意图解决所有大数据计算问题。

3，流式计算相关门派说你hadoop只能憋大招（批量计算），太麻烦，于是出了SparkStreaming、Storm，S4等流式计算技术，能够实现数据一来就即时计算。

4，apache看各大门派纷争四起，推出flink，想一统流计算和批量计算的修炼；

# 消息队列

<https://mp.weixin.qq.com/s?__biz=MzI4Njg5MDA5NA==&mid=2247485080&idx=1&sn=f223feb9256727bde4387d918519766b&chksm=ebd74799dca0ce8fa46223a33042a79fc16ae6ac246cb8f07e63a4a2bdce33d8c6dc74e8bd20&token=1755043505&lang=zh_CN#rd>什么是消息队列？

# Kafka和spark百科

Kafka是由[Apache软件基金会](https://baike.baidu.com/item/Apache%E8%BD%AF%E4%BB%B6%E5%9F%BA%E9%87%91%E4%BC%9A)开发的一个开源流处理平台，由[Scala](https://baike.baidu.com/item/Scala)和[Java](https://baike.baidu.com/item/Java/85979)编写。Kafka是一种高吞吐量的[分布式](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F/19276232)发布订阅消息系统，它可以处理消费者在网站中的所有动作流数据。 这种动作（网页浏览，搜索和其他用户的行动）是在现代网络上的许多社会功能的一个关键因素。 这些数据通常是由于吞吐量的要求而通过处理日志和日志聚合来解决。 对于像[Hadoop](https://baike.baidu.com/item/Hadoop)一样的[日志](https://baike.baidu.com/item/%E6%97%A5%E5%BF%97/2769135)数据和离线分析系统，但又要求实时处理的限制，这是一个可行的解决方案。Kafka的目的是通过[Hadoop](https://baike.baidu.com/item/Hadoop)的并行加载机制来统一线上和离线的消息处理，也是为了通过[集群](https://baike.baidu.com/item/%E9%9B%86%E7%BE%A4/5486962)来提供实时的消息。

Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎

# Kafka的几个概念

<https://zhuanlan.zhihu.com/p/95215691>

java3y

broker topic（逻辑上partition的集合） partition（在不同的broker里面）

# kafka安装教程

<https://www.cnblogs.com/chejiangyi/p/6038932.html>卡法卡安装教程

kafka启动.\bin\windows\kafka-server-start.bat .\config\server.properties

# 环境搭建demo

**4. 创建topics**

4.1 打开cmd 并进入G:\kafka\_2.11-0.10.0.1\bin\windows

4.2 创建一个topic：

kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test

**5. 打开一个Producer：**

cd /d G:\kafka\_2.11-0.10.0.1\bin\windows

kafka-console-producer.bat --broker-list localhost:9092 --topic test

**6. 打开一个Consumer:**

cd /d G:\kafka\_2.11-0.10.0.1\bin\windows

kafka-console-consumer.bat --zookeeper localhost:2181 --topic test

然后就可以在Producer控制台窗口输入消息了。在消息输入过后，很快Consumer窗口就会显示出Producer发送的消息：

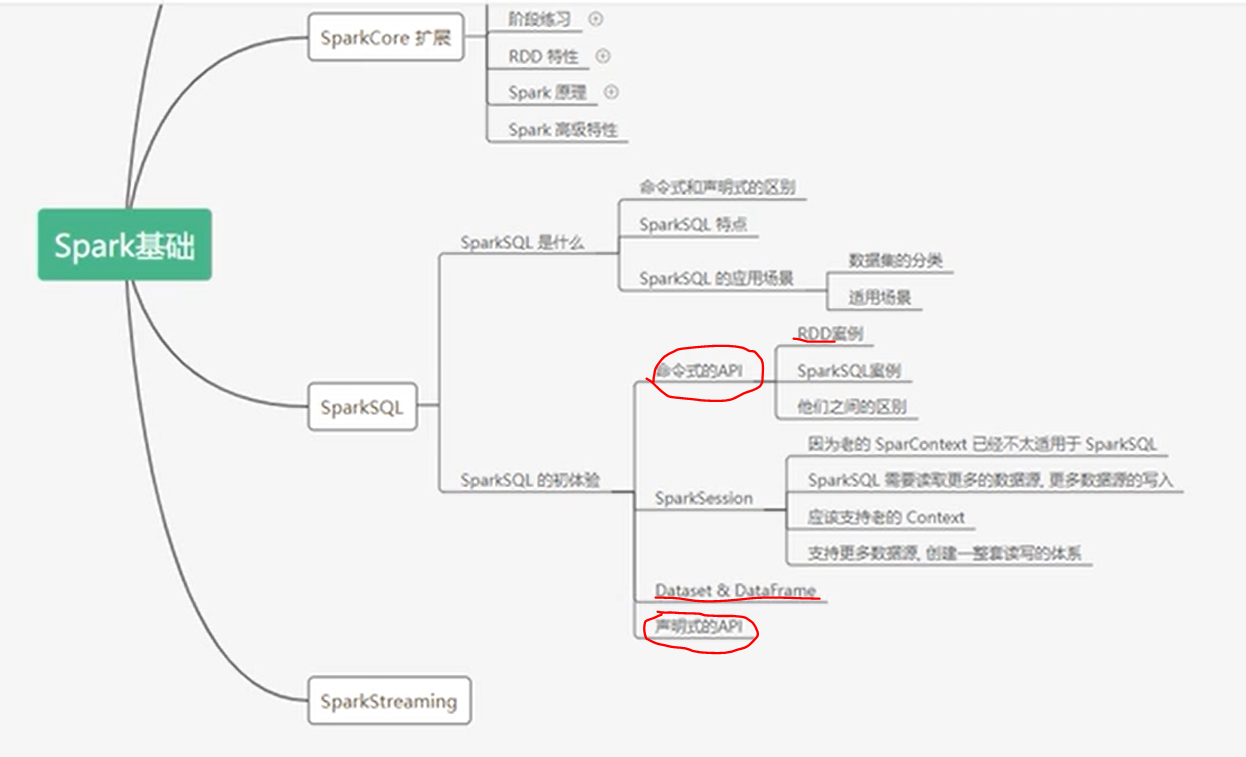
# Spark

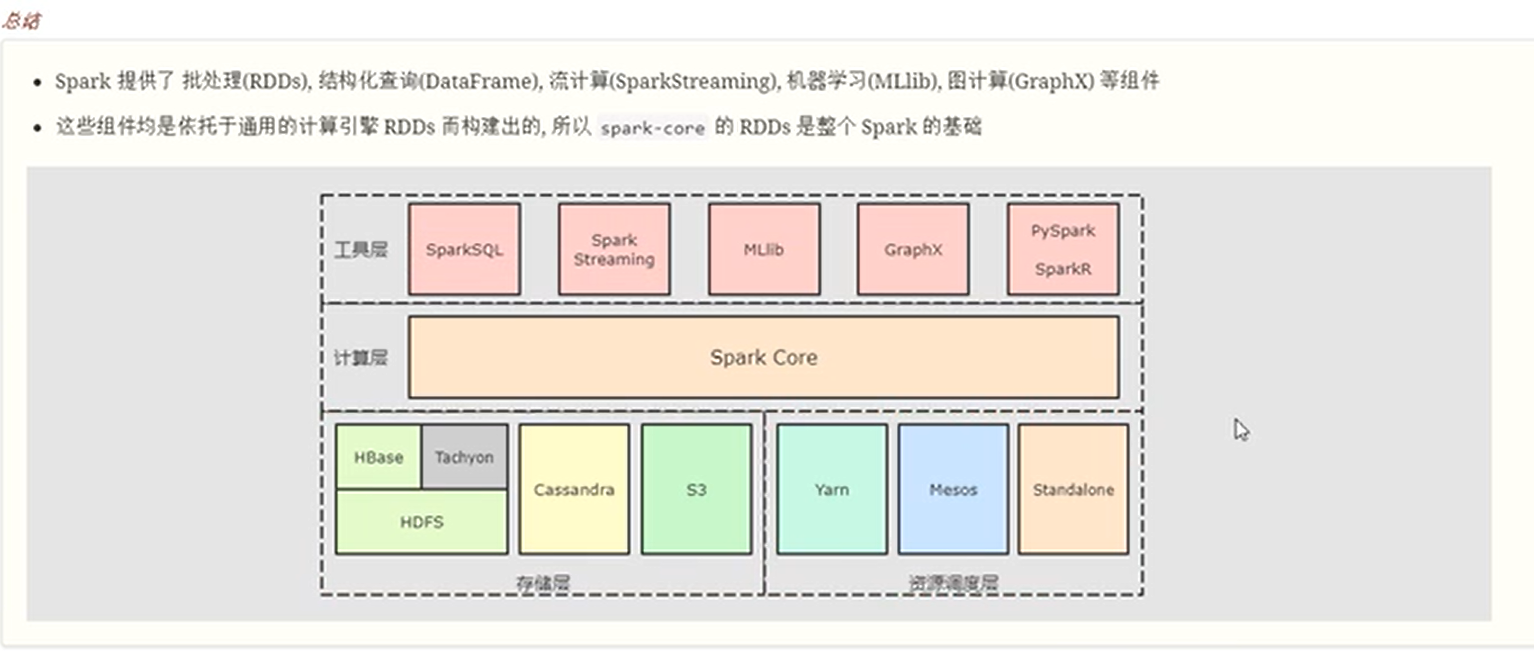
多用途的计算框架

Spark安装教程<https://www.cnblogs.com/xuliangxing/p/7279662.html>

PyCharm配置spark环境 https://www.jb51.net/article/169364.htm

法号阿兴



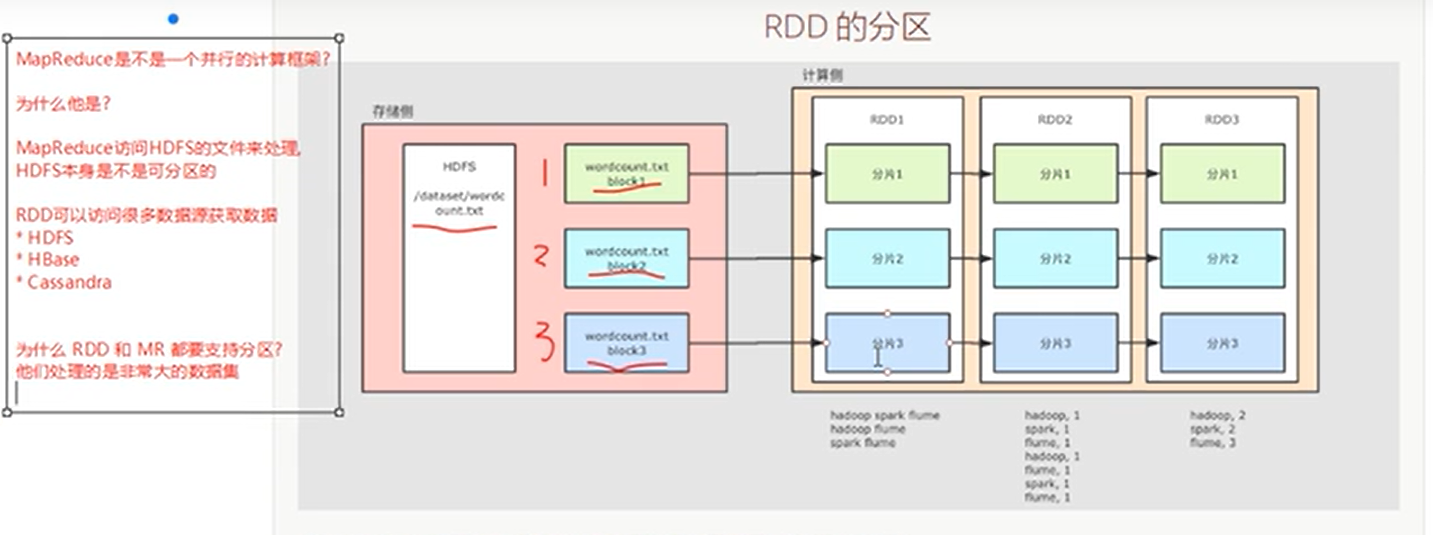




模块组成：Spark Core(RDD非结构化), SQL(DF/DataSet), Structured Streaming, MLlib/ML等。

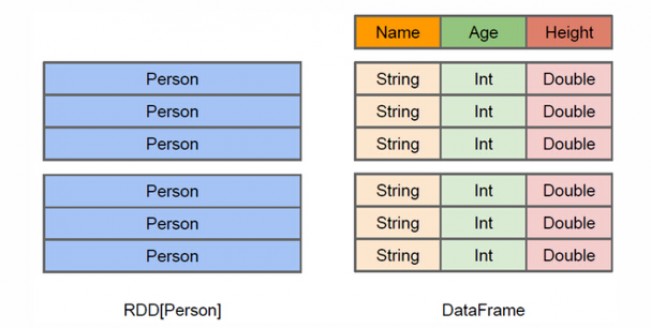
# RDD

Rdd是一个数据集，rdd是一个编程模型。Rdd可以分区



# DataFrame与RDD的区别

DataFrame的推出，让Spark具备了处理大规模结构化数据的能力，不仅比原有的RDD转化方式更加简单易用，而且获得了更高的计算性能。Spark能够轻松实现从MySQL到DataFrame的转化，并且支持SQL查询。

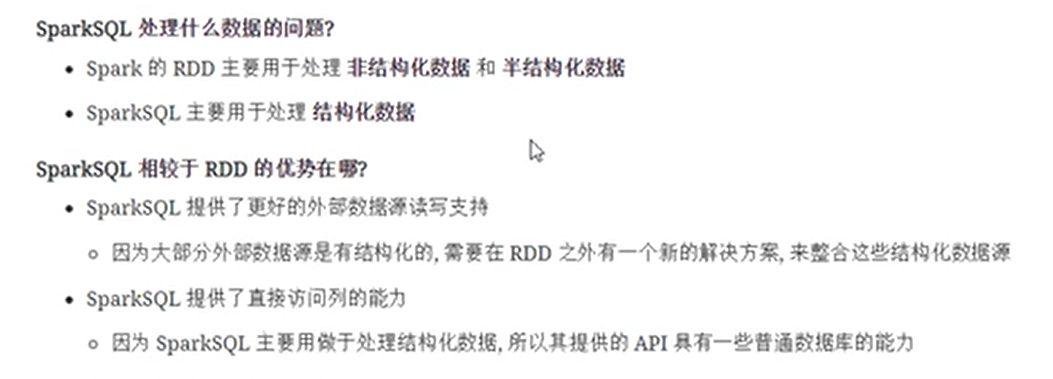
  
图 DataFrame与RDD的区别  
从上面的图中可以看出DataFrame和RDD的区别。RDD是分布式的 Java对象的集合，比如，RDD[Person]是以Person为类型参数，但是，Person类的内部结构对于RDD而言却是不可知的。DataFrame是一种以RDD为基础的分布式数据集，也就是分布式的Row对象的集合（每个Row对象代表一行记录），提供了详细的结构信息，也就是我们经常说的模式（schema），Spark SQL可以清楚地知道该数据集中包含哪些列、每列的名称和类型。  
和RDD一样，DataFrame的各种变换操作也采用惰性机制，只是记录了各种转换的逻辑转换路线图（是一个DAG图），不会发生真正的计算，这个DAG图相当于一个逻辑查询计划，最终，会被翻译成物理查询计划，生成RDD DAG，按照之前介绍的RDD DAG的执行方式去完成最终的计算得到结果

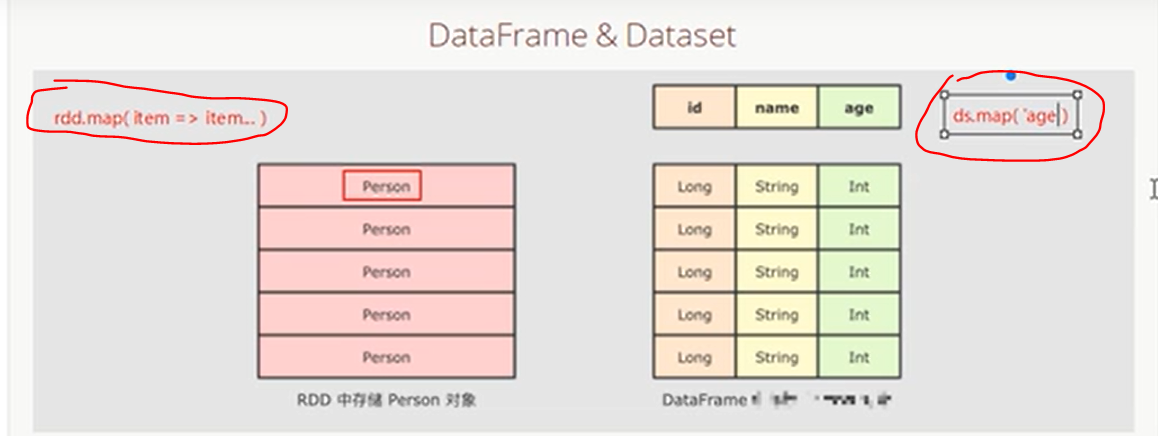
# [SparkContext、SparkConf和SparkSession的初始化](https://www.cnblogs.com/Forever-Road/p/7351245.html)

https://www.cnblogs.com/Forever-Road/p/7351245.html

# Spark SQL

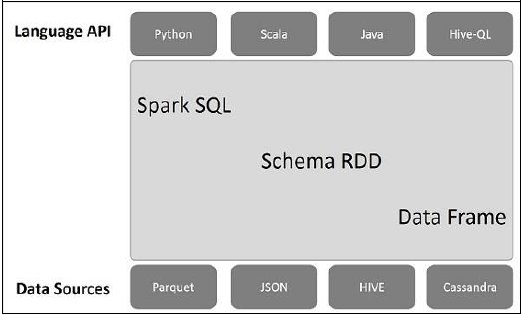
Spark SQL是应用于Spark的一个组件





## Spark SQL架构

下图说明了Spark SQL的体系结构



此架构包含三个层，即Language API，Schema RDD和数据源。

## 语言API

Spark与不同的语言和Spark SQL兼容。 它也是由这些语言支持的API（python，scala，java，HiveQL）。

## 模式RDD

Spark Core是使用称为RDD的特殊数据结构设计的。 通常，Spark SQL适用于模式，表和记录。 因此，我们可以使用Schema RDD作为临时表。 我们可以将此Schema RDD称为数据帧。

## 数据源

通常spark-core的数据源是文本文件，Avro文件等。但是，Spark SQL的数据源不同。 这些是Parquet文件，JSON文档，HIVE表和Cassandra数据库。

# Spark SQL 数据帧

DataFrame是一个分布式数据集合，它被 组织成 命名列。

DataFrame可以从不同来源的数组构造

## SQLContext

这是一个类，用于初始化**Spark SQL**的功能

初始化SQLContext类对象需要SparkContext类对象(sc)

先**SparkContext** 类对象（初始化）---（放入）-–>后**SQLContext**类对象（初始化）

以下命令用于通过spark-shell初始化SparkContext。

$ spark-shell

默认情况下，SparkContext对象在spark-shell启动时用namesc初始化。  
使用以下命令创建SQLContext。

scala> val **sqlcontext** = new org.apache.spark.sql.SQLContext(sc)

## ****DataFrame操作****

DataFrame为结构化数据操作提供了一个领域特定的语言。 这里，我们包括使用DataFrames的结构化数据处理的一些基本示例。

按照以下步骤执行DataFrame操作

**Spark SQL是应用于Spark的一个组件, DataFrame是sparkSQL中的一个模块**

**Spark为结构化数据处理引入了一个称为Spark SQL的编程模块。它提供了一个称为DataFrame的编程抽象，并且可以充当分布式SQL查询引擎**

### ****阅读JSON文档****

首先，我们要读取JSON文档。 基于此，生成名为(dfs)的DataFrame。

使用以下命令读取JSON文档namedemployee.json。 数据显示为带有字段id，name和age的表。

scala> val dfs = **sqlContext**.read.json("employee.json")

**输出：**字段名称从employee.json自动获取。

dfs: org.apache.spark.sql.DataFrame = [age: string, id: string, name: string]

### 显示数据

如果你想看到的数据框的数据，然后使用以下命令。

scala> dfs.show()

**输出：**您可以以表格格式查看员工数据。

<console>:22, took 0.052610 s

+----+------+--------+

|age | id | name |

+----+------+--------+

| 25 | 1201 | satish |

| 28 | 1202 | krishna|

| 39 | 1203 | amith |

| 23 | 1204 | javed |

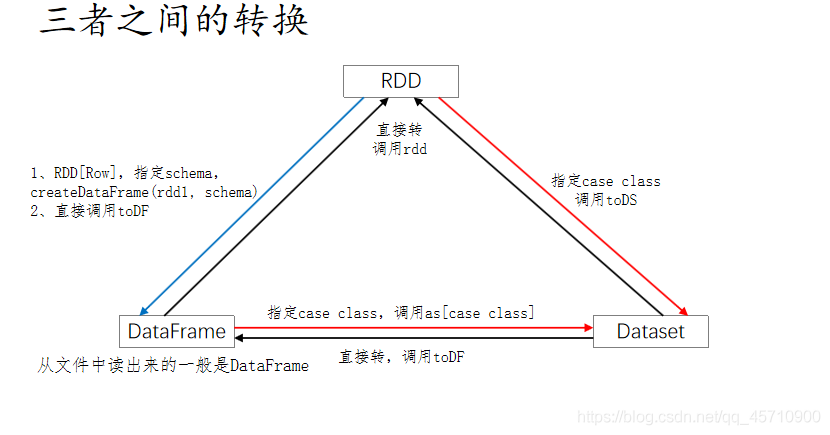
| 23 | 1205 | prudvi |

+----+------+--------+

### 使用方法printSchema

Spark sparkSQL -- dataframe&dataset

## 相互转换RDD & DataFrame & DataSet



RDD、DataFrame、Dataset都是spark平台下的分布式弹性数据集，为处理超大型数据提供便利

examples/src/main/python/sql/basic.py

# 大数据项目8.19汇报

学习了kafka的基础知识并理解基本概念

学习spark基础知识.

已在windows下装好Kafka，已实现在Python环境下创建producer，并向kafka写入简单的数据，能创建consumer且能读到数据。

已在windows下装好spark，已实现在Python环境下将简单的JSON格式数据写入Spark DataFrame。

接下来要攻克的难题：

继续学习kafka与spark知识。

如何使用Spark消费Kafka里面的数据？

Spark使用什么方法才能瞬时读取大量的数据？

Consumer程序如若挂掉或者重启，是否会引起重复读？

<https://blog.csdn.net/songhao8080/article/details/103670122>

# Pyspark 从kafka 读取数据 rdd，转成 DataFrame

# Streamset官方文档

https://streamsets.com/documentation/datacollector/latest/help/datacollector/UserGuide/Processors/Jython.html?hl=jython%2Cevaluator

# StreamSet知识点

streamsets data collector中的各种阶段的处理数据组件：

           origins (extract):数据来源，数据从不同的数据源抽取。（一个pipeline中只能有一个数据来源）

           processor(transform):数据转化，将抽取来的数据进行过滤，清洗。

           destination(load):数据存储，将数据处理完后存入目标系统或者转入另一个pipeline进行再次处理。

           executor：由处理数据组件的事件触发executor,执行相应任务。例如：某个组件处理失败，发送邮件通知。

# 日志日志日志

### org.apache.spark.SparkException: Only one SparkContext may be running in this JVM (see SPARK-2243). To ignore this error, set spark.driver.allowMultipleContexts = true.

解决方法：sparkConf.set(“spark.driver.allowMultipleContexts”,“true”);

# 适当用英文搜索

# [jupyter](http://www.baidu.com/link?url=tTPmZjDL7wFPGEc5lJTmCSBVp5oCJ8MFXTIUSGNTzu-ehZHAUiAtSxSGjvnABepHylPQYB2b9AJ015pw3WXRy2kktLm3-qgIdEeDh5bZ-ju)

 就批特儿。音标:/ˈdʒuːpɪtə(r)/

<https://blog.csdn.net/Demonslzh/article/details/107547626>

# 常见报错解决

1、如果是在findspark.init()报错的话那么一般是没有设置SPARK\_HOME环境变量，记得正确配置。  
2、Py4JError:org.apache.spark.api.python.PythonUtils.isEncryptionEnabled does not exist in the JVM这个问题困扰了我很长时间，如果在jdk、spark、Hadoop都已正确配置的前提下（正确配置的前提是可以在命令行中启动spark-shell），那么问题就出在pyspark的版本问题上。我的解决方案是回退pyspark的版本到2.3.2  
分别输入以下命令即可

pip uninstall pyspark

pip install pyspark==2.3.2

# json格式转变

<http://c.runoob.com/front-end/53>

# json扁平化 normalize

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.json_normalize.html>

# 思绪整理

jupyter 平台 kafka环境可以 spark环境可以 难题：json数据不能转成dataframe ，因为json是嵌套的

json->dataframe

json->rdd

spark.sparkContext.parallelize ？ 转成rdd

df = spark.createDataFrame(a)

df = spark.read.json(a) 这两种有啥区别？

# Hbase

## 访问接口

1. Native Java API，最常规和高效的访问方式，适合Hadoop MapReduce Job并行[批处理](https://baike.baidu.com/item/%E6%89%B9%E5%A4%84%E7%90%86)HBase表数据

2. HBase Shell，HBase的命令行工具，最简单的接口，适合HBase管理使用

3. Thrift Gateway，利用Thrift序列化技术，支持C++，PHP，***Python等多种语言，***适合其他异构系统在线访问HBase表数据

4. REST Gateway，支持REST 风格的Http API访问HBase, 解除了语言限制

5. Pig，可以使用Pig Latin流式编程语言来操作HBase中的数据，和Hive类似，本质最终也是编译成MapReduce Job来处理HBase表数据，适合做数据统计

6. Hive，当前Hive的[Release版本](https://baike.baidu.com/item/Release%E7%89%88%E6%9C%AC)尚没有加入对HBase的支持，但在下一个版本Hive 0.7.0中将会支持HBase，可以使用类似SQL语言来访问HBase

# 群集技术

就像[冗余](https://baike.baidu.com/item/%E5%86%97%E4%BD%99)部件可以使你免于硬件[故障](https://baike.baidu.com/item/%E6%95%85%E9%9A%9C/9850042)一样，群集技术则可以使你免于整个系统的[瘫痪](https://baike.baidu.com/item/%E7%98%AB%E7%97%AA/9947450)以及操作系统和应用层次的故障。一台[服务器集群](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8%E9%9B%86%E7%BE%A4/5309783)包含多台拥有共享[数据存储](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AD%98%E5%82%A8)空间的服务器，各服务器之间通过内部局域网进行互相连接；当其中一台服务器发生故障时，它所运行的应用程序将被与之相连的服务器自动接管；在大多数情况下，集群中所有的计算机都拥有一个共同的名称，[集群系统](https://baike.baidu.com/item/%E9%9B%86%E7%BE%A4%E7%B3%BB%E7%BB%9F/6754805)内任意一台服务器都可被所有的网络用户所使用。

# Hbase定位

在 Hbase 中，Row-key 加上 CF 加上 Qulifier 再加上一个时间戳才可以定位到一个单元格数据（Hbase 中每个单元格默认有 3 个时间戳的版本数据）。

# HBase 深入浅出

<https://developer.ibm.com/zh/articles/ba-cn-bigdata-hbase/>

**HBase的表结构（https://blog.csdn.net/weixin\_30912051/article/details/94929687?utm\_medium=distribute.pc\_relevant.none-task-blog-title-2&spm=1001.2101.3001.4242）**

建表时要指定的是：表名、列族

建表语句

create 'user\_info', 'base\_info', 'ext\_info'

意思是新建一个表，名称是user\_info，包含两个列族base\_info和ext\_info

**列族** 是列的集合，一个列族中包含多个列

这时的表结构：

| **row key** | **base\_info** | **ext\_info** |
| --- | --- | --- |
| ... | ... | ... |

**row key** 是行键，每一行的ID，这个字段是自动创建的，建表时不需要指定

插入一条用户数据：name为‘a’，tel为‘123’

插入语句

put 'user\_info', 'row1', 'base\_info:name', 'a'

put 'user\_info', 'row1', 'base\_info:tel', '123'

意思是向user\_info表中**行健**为row1的base\_info列族中添加一项数据 name:a，接着又添加一项数据tel:123

name和tel就是具体字段，属于base\_info这个列族

| **row key** | **base\_info** | **ext\_info** |
| --- | --- | --- |
| row1 | name:a, tel:123 |  |

这时的表结构：

再插入一条数据：name为‘b’，addr为‘beijing’

put 'user\_info', 'row2', 'base\_info:name', 'b'

put 'user\_info', 'row2', 'ext\_info:addr', 'bj'

这时的表结构：

| **row key** | **base\_info** | **ext\_info** |
| --- | --- | --- |
| row1 | name:a, tel:123 |  |
| row2 | name:b | addr:bj |

HBase表中还有一个重要概念：**版本**，每个字段的值都有版本信息（通过时间戳指定）

例如 base\_info:name，每次修改时都会保留之前的值，就是说可以取到他的旧值

| **row key** | **base\_info** | **ext\_info** |
| --- | --- | --- |
| row1 | name:a, tel:123 |  |
| row2 | name:c(v2)[name:b(v1)] | addr:bj |

# Dataframe的select查询

无法查询嵌套的dataframe

http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame

# Kudu知识点

Kudu 的特点正如它的 slogan 所说，Fast Analytics on Fast Data。