《Spark编程基础》

教材官网： http://dblab.xmu.edu.cn/post/spark/

温馨提示：编辑幻灯片母版，可以修改每页PPT的厦大校徽和底部文字

第6章 Spark SQL

（PPT版本号：2018年春季学期）

林子雨

厦门大学计算机科学系

E-mail: ziyulin@xmu.edu.cn

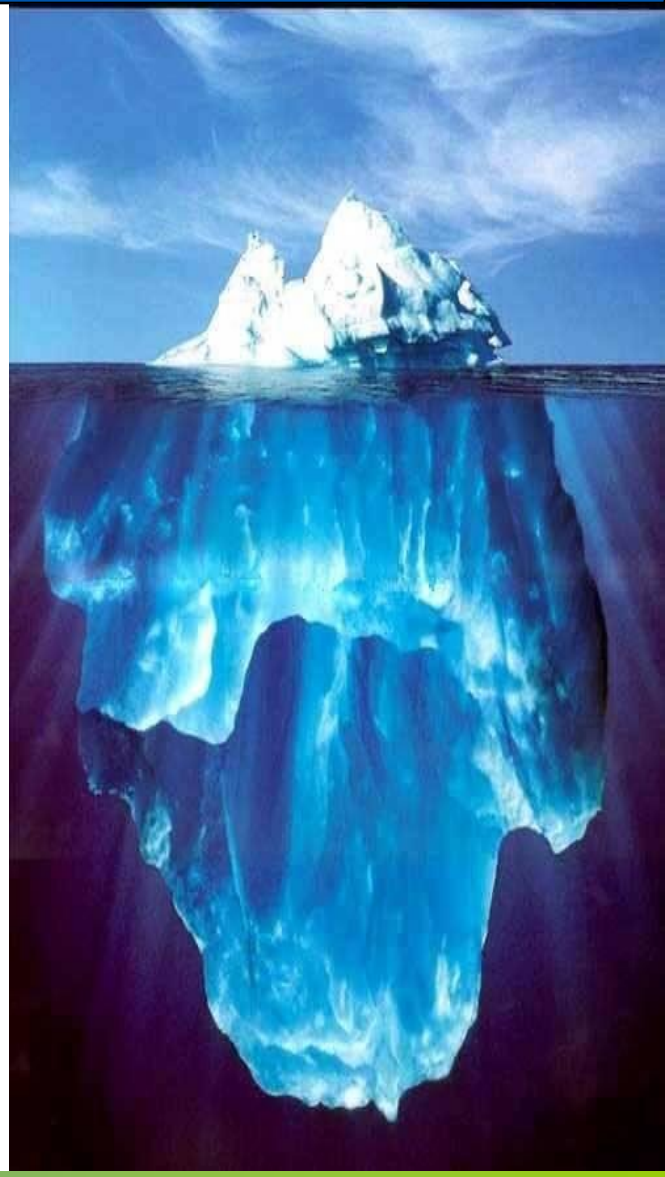
主页：http://www.cs.xmu.edu.cn/linziyu

扫一扫访问教材官网

课程配套授课视频

课程在线视频地址：http://dblab.xmu.edu.cn/post/10482/

提纲

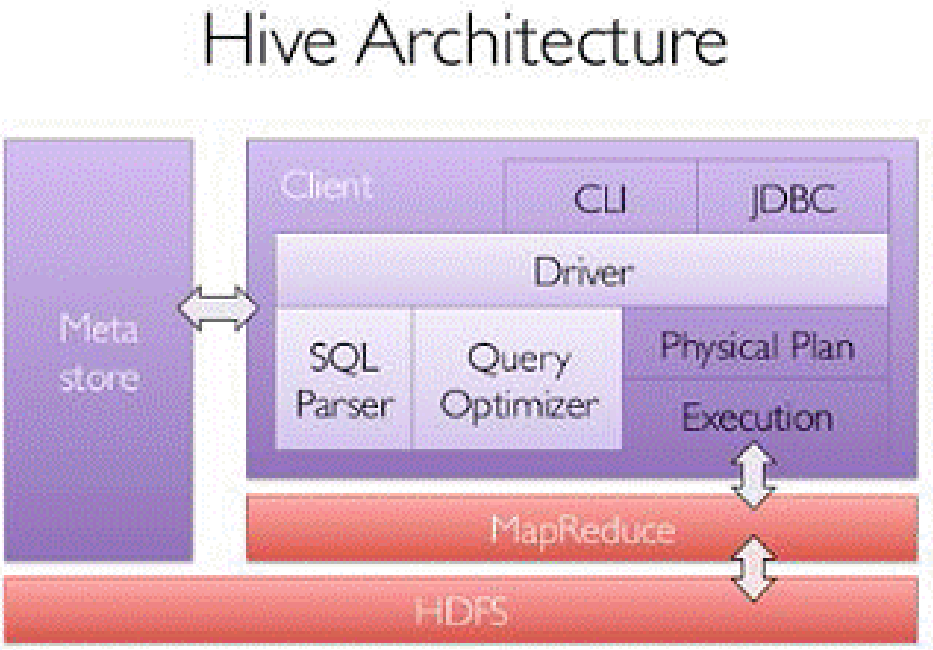


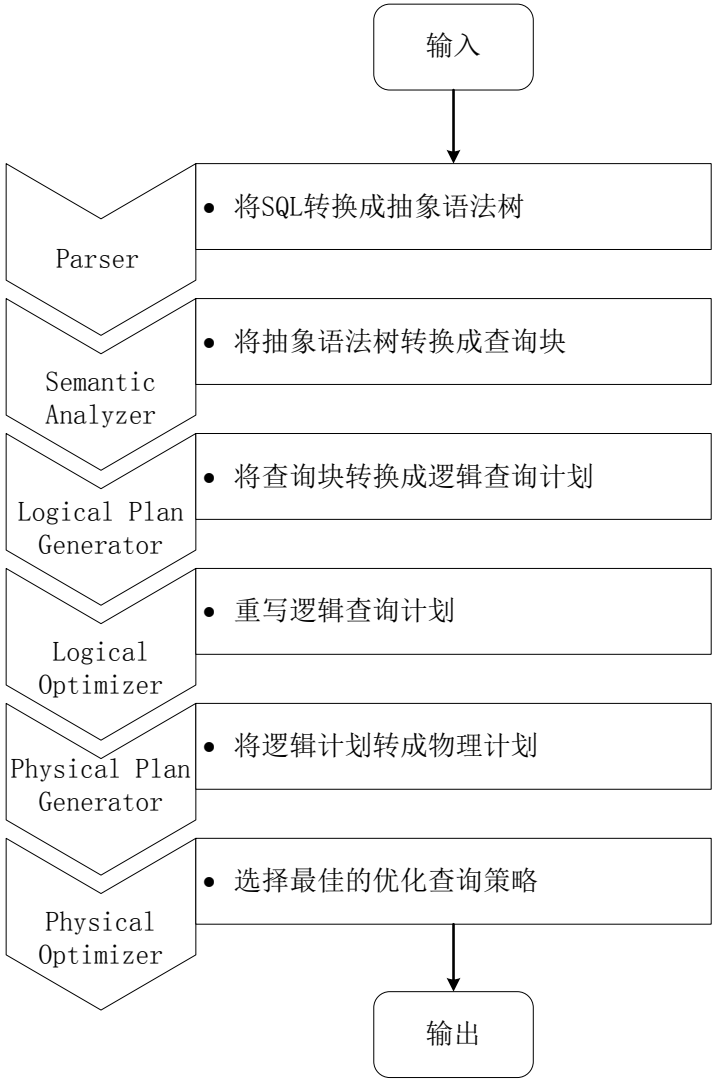
6.1 Spark SQL简介

6.1.1 从Shark说起
6.1.2 Spark SQL设计
6.1.3 为什么推出Spark SQL

6.1.1 从Shark说起

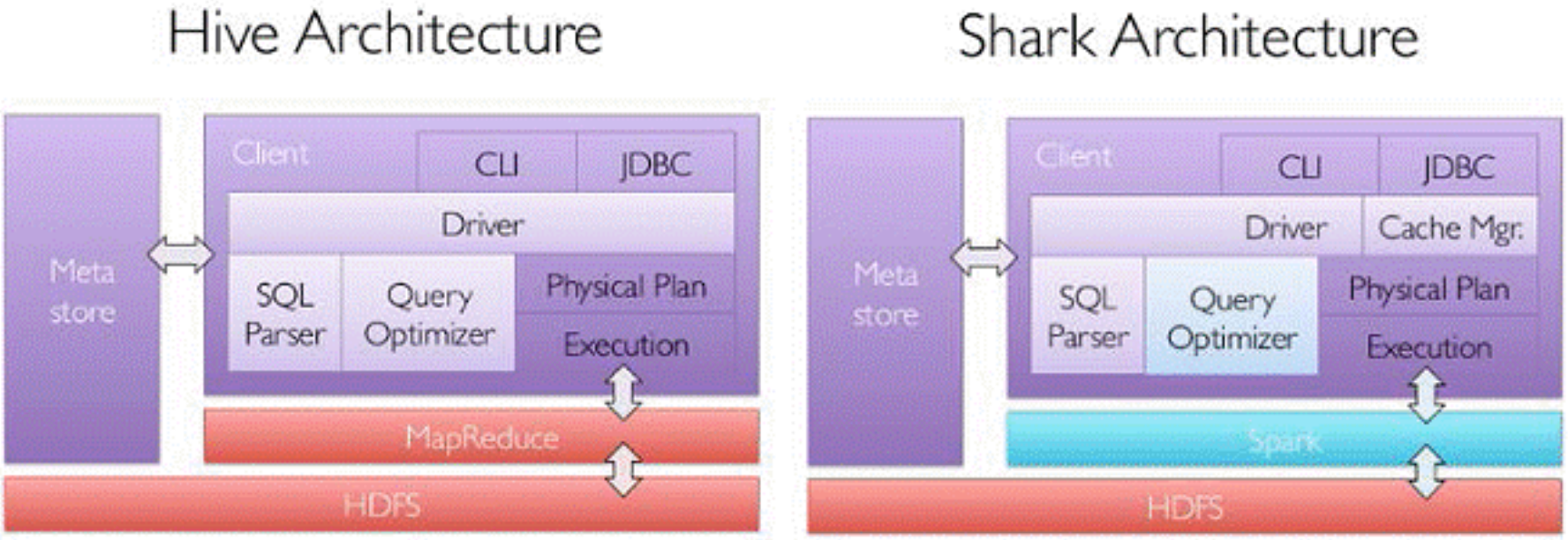
Hive: SQL-on-Hadoop





Hive中SQL查询的MapReduce作业转化过程

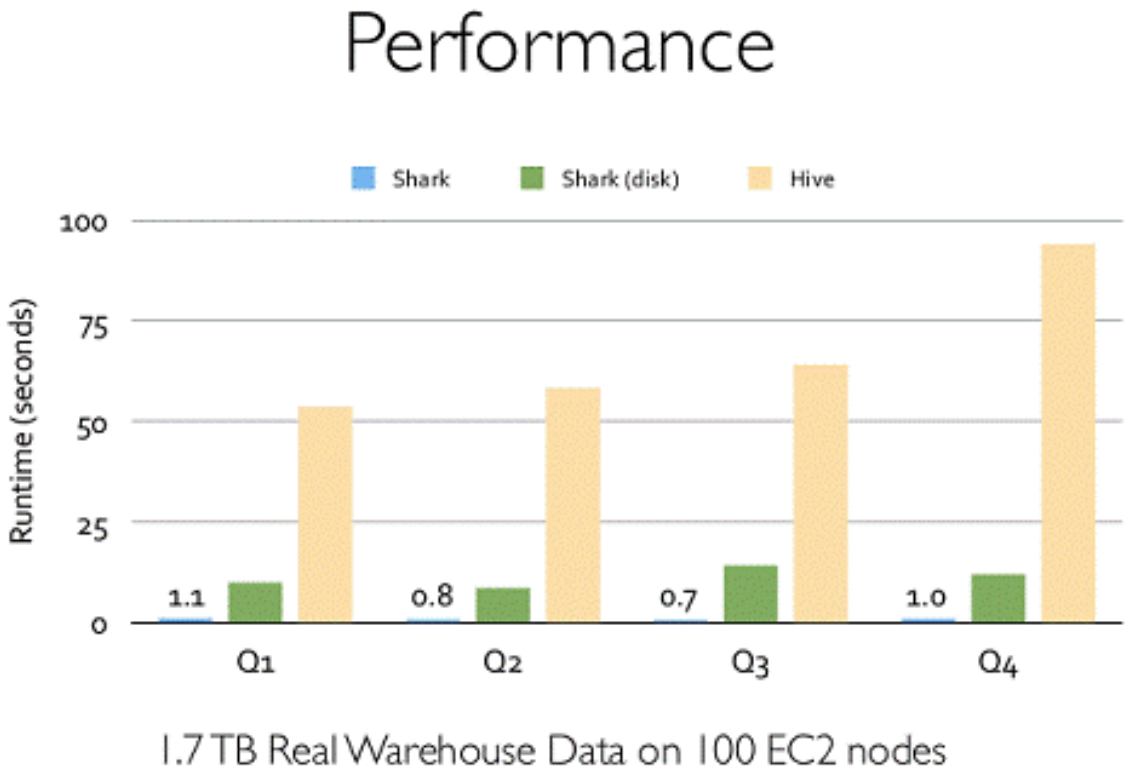
6.1.1 从Shark说起



•Shark即Hive on Spark，为了实现与Hive兼容，Shark在HiveQL方面重用了Hive中HiveQL的解析、逻辑执行计划翻译、执行计划优化等逻辑，可以近似认为仅将物理执行计划从MapReduce作业替换成了Spark作业，通过Hive的HiveQL解析，把HiveQL翻译成Spark上的RDD操作

6.1.1 从Shark说起

Shark的出现，使得SQL-on-Hadoop的性能比Hive有了10-100倍的提高



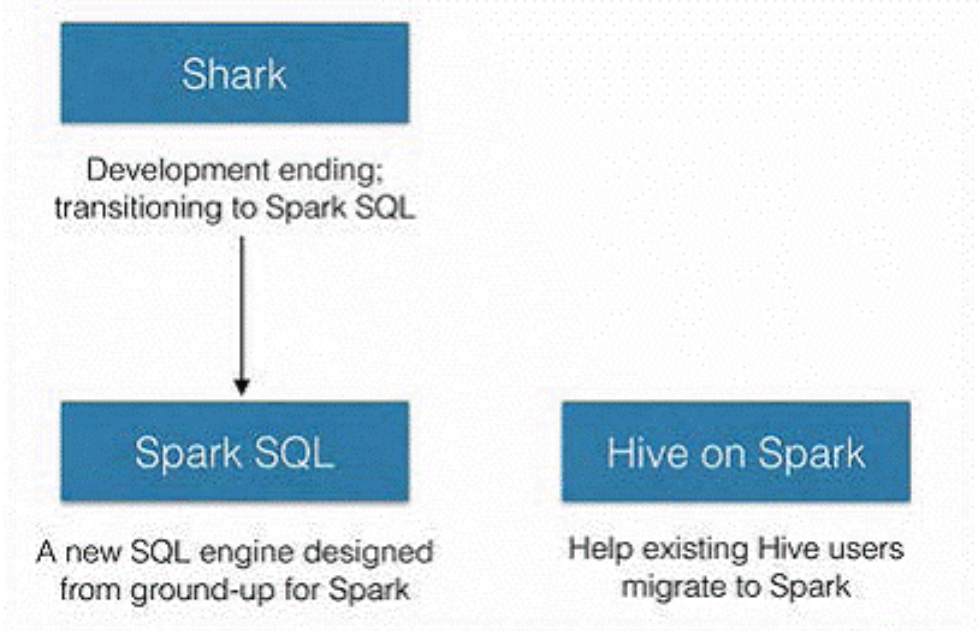
6.1.1 从Shark说起

Shark的设计导致了两个问题：

- 一是执行计划优化完全依赖于Hive，不方便添加新的优化策略
- 二是因为Spark是线程级并行，而MapReduce是进程级并行，因此，Spark在兼容Hive的实现上存在线程安全问题，导致Shark不得不使用另外一套独立维护的打了补丁的Hive源码分支

6.1.1 从Shark说起

2014年6月1日Shark项目和Spark SQL项目的主持人Reynold Xin宣布：停止对Shark的开发，团队将所有资源放在Spark SQL项目上，至此，Shark的发展画上了句话，但也因此发展出两个直线：Spark SQL和Hive on Spark



- Spark SQL作为Spark生态的一员继续发展，而不再受限于Hive，只是兼容Hive
- Hive on Spark是一个Hive的发展计划，该计划将Spark作为Hive的底层引擎之一，也就是说，Hive将不再受限于一个引擎，可以采用Map-Reduce、Tez、Spark等引擎

6.1.2 Spark SQL设计

Spark SQL在Hive兼容层面仅依赖HiveQL解析、Hive元数据，也就是说，从HQL被解析成抽象语法树（AST）起，就全部由Spark SQL接管了。Spark SQL执行计划生成和优化都由Catalyst（函数式关系查询优化框架）负责

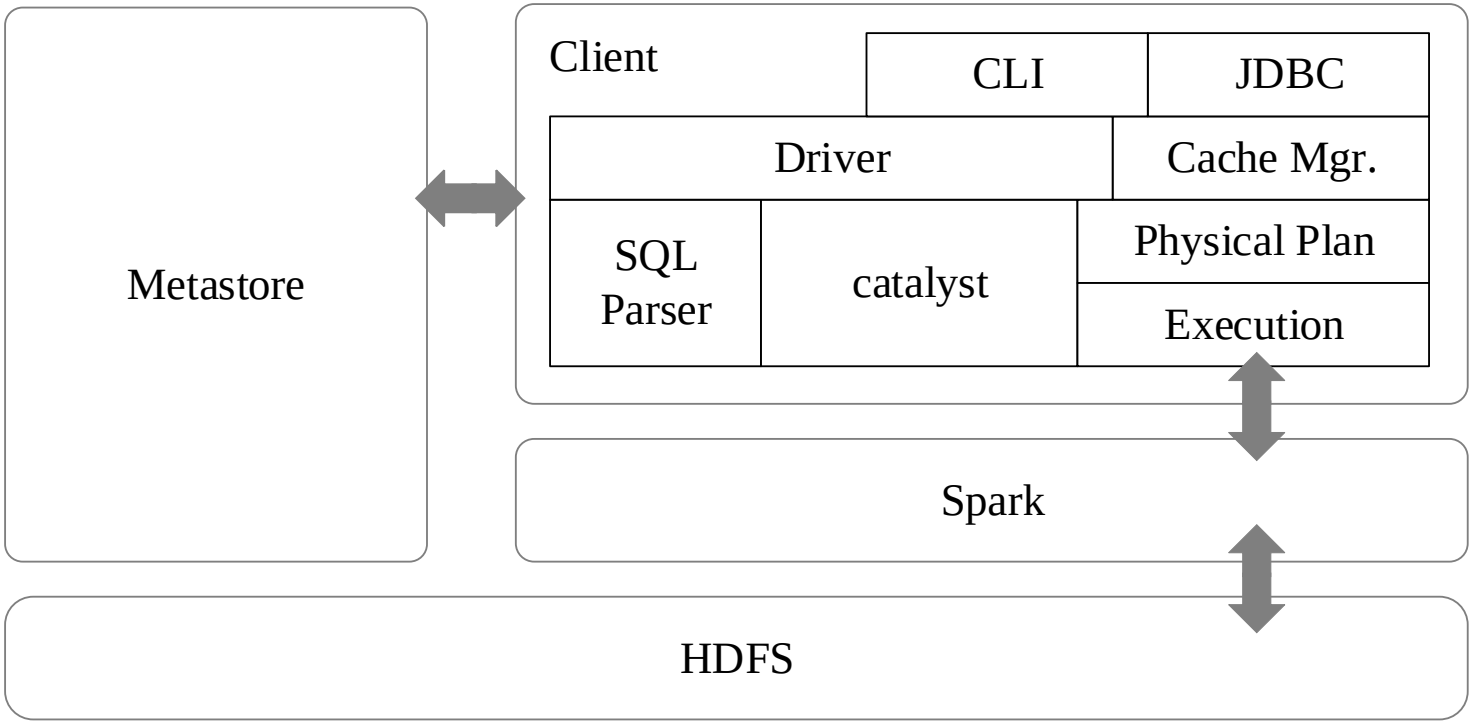


图 Spark SQL架构

6.1.2 Spark SQL设计

•Spark SQL增加了DataFrame（即带有Schema信息的RDD），使用户可以在Spark SQL中执行SQL语句，数据既可以来自RDD，也可以是Hive、HDFS、Cassandra等外部数据源，还可以是JSON格式的数据
•Spark SQL目前支持Scala、Java、Python三种语言，支持SQL-92规范

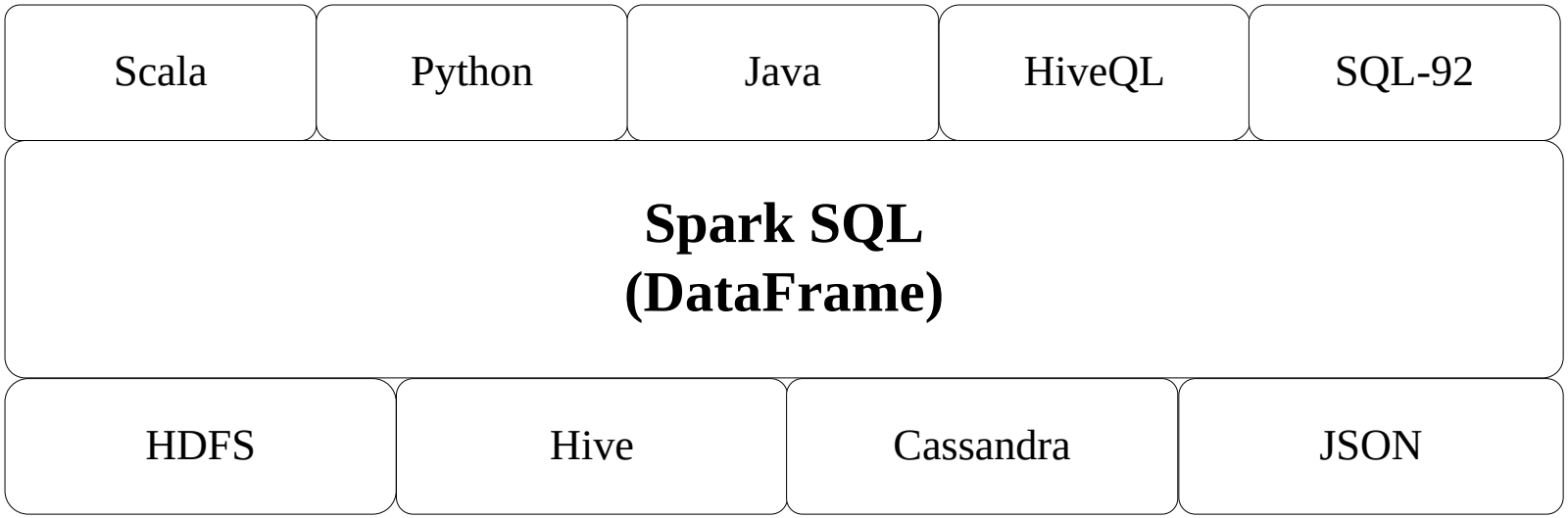


图　Spark SQL支持的数据格式和编程语言

6.1.3 为什么推出Spark SQL

Spark SQL: Relational Data Processing in Spark

Michael Armbrust[†], Reynold S. Xin[†], Cheng Lian[†], Yin Huai[†], Davies Liu[†], Joseph K. Bradley[†], Xiangrui Meng[†], Tomer Kaftan[†], Michael J. Franklin[†‡], Ali Ghodsi[†], Matei Zaharia[†*]

[†]Databricks Inc. [*]MIT CSAIL [‡]AMPLab, UC Berkeley

While the popularity of relational systems shows that users often prefer writing declarative queries, the relational approach is insufficient for many big data applications. First, users want to perform ETL to and from various data sources that might be semi- or unstructured, requiring custom code. Second, users want to perform advanced analytics, such as machine learning and graph processing, that are challenging to express in relational systems. In practice, we have observed that most data pipelines would ideally be expressed with a combination of both relational queries and complex procedural algorithms. Unfortunately, these two classes of systems—relational and procedural—have until now remained largely disjoint,

6.1.3 为什么推出Spark SQL

Spark SQL: Relational Data Processing in Spark

Michael Armbrust[†], Reynold S. Xin[†], Cheng Lian[†], Yin Huai[†], Davies Liu[†], Joseph K. Bradley[†], Xiangrui Meng[†], Tomer Kaftan[†], Michael J. Franklin[†‡], Ali Ghodsi[†], Matei Zaharia[†*]

[†]Databricks Inc. [*]MIT CSAIL [‡]AMPLab, UC Berkeley

Spark SQL bridges the gap between the two models through two contributions. First, Spark SQL provides a *DataFrame API* that can perform relational operations on both external data sources and Spark's built-in distributed collections. This API is similar to the widely used data frame concept in R [32], but evaluates operations lazily so that it can perform relational optimizations. Second, to support the wide range of data sources and algorithms in big data, Spark SQL introduces a novel extensible optimizer called *Catalyst*. Catalyst makes it easy to add data sources, optimization rules, and

6.1.3 为什么推出Spark SQL

- 关系数据库已经很流行
- 关系数据库在大数据时代已经不能满足要求
  - 首先，用户需要从不同数据源执行各种操作，包括结构化、半结构化和非结构化数据
  - 其次，用户需要执行高级分析，比如机器学习和图像处理
- 在实际大数据应用中，经常需要融合关系查询和复杂分析算法（比如机器学习或图像处理），但是，缺少这样的系统

Spark SQL填补了这个鸿沟：

- 首先，可以提供DataFrame API，可以对内部和外部各种数据源执行各种关系型操作
- 其次，可以支持大数据中的大量数据源和数据分析算法

Spark SQL可以融合：传统关系数据库的结构化数据管理能力和机器学习算法的数据处理能力

6.2 DataFrame概述

•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

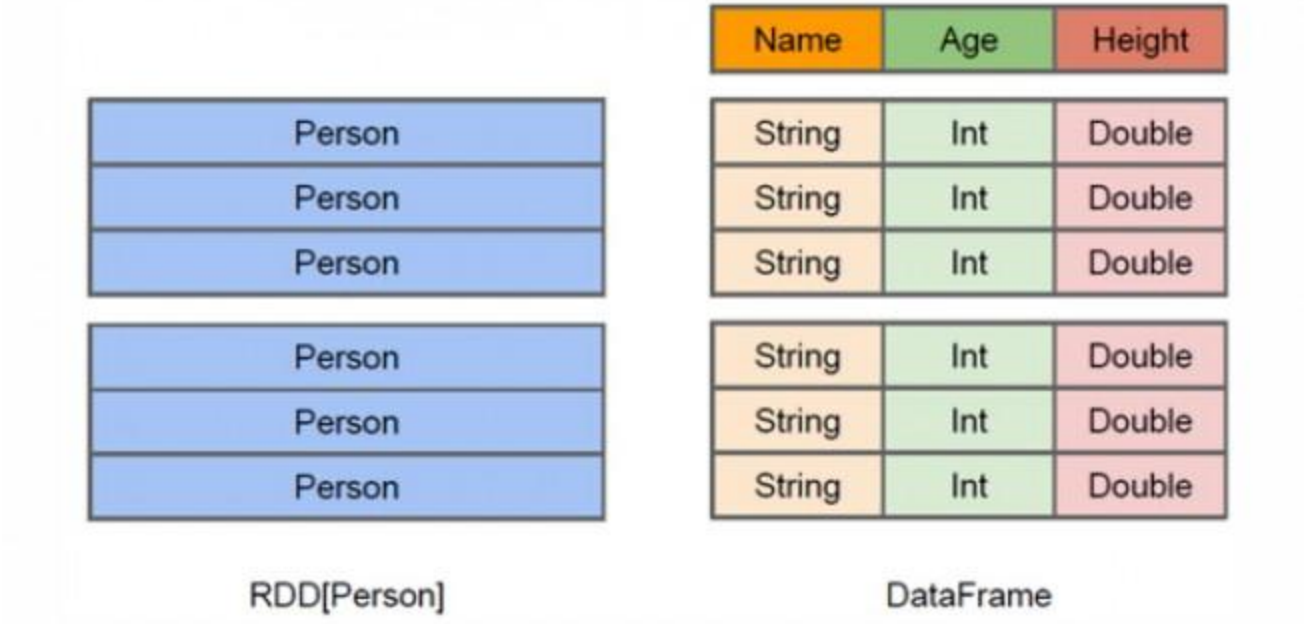


图 DataFrame与RDD的区别

•RDD是分布式的 Java对象的集合，但是，对象内部结构对于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，提供了详细的结构信息

6.3 DataFrame的创建

•从Spark2.0以上版本开始，Spark使用全新的SparkSession接口替代Spark1.6中的SQLContext及HiveContext接口来实现其对数据加载、转换、处理等功能。SparkSession实现了SQLContext及HiveContext所有功能

•SparkSession支持从不同的数据源加载数据，并把数据转换成DataFrame，并且支持把DataFrame转换成SQLContext自身中的表，然后使用SQL语句来操作数据。SparkSession亦提供了HiveQL以及其他依赖于Hive的功能的支持

可以通过如下语句创建一个SparkSession对象：

```
scala> import org.apache.spark.sql.SparkSession
scala> val spark=SparkSession.builder().getOrCreate()
```

6.3 DataFrame的创建

在创建DataFrame之前，为了支持RDD转换为DataFrame及后续的SQL操作，需要通过import语句（即import spark.implicits._）导入相应的包，启用隐式转换。

在创建DataFrame时，可以使用spark.read操作，从不同类型的文件中加载数据创建DataFrame，例如：

- spark.read.json("people.json")：读取people.json文件创建DataFrame；在读取本地文件或HDFS文件时，要注意给出正确的文件路径；
- spark.read.parquet("people.parquet")：读取people.parquet文件创建DataFrame；
- spark.read.csv("people.csv")：读取people.csv文件创建DataFrame。

6.3 DataFrame的创建

一个实例

•在“/usr/local/spark/examples/src/main/resources/”这个目录下，这个目录下有两个样例数据people.json和people.txt。
people.json文件的内容如下：

```
{"name":"Michael"}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
```

people.txt文件的内容如下：

```
Michael, 29
Andy, 30
Justin, 19
```

6.3 DataFrame的创建

```
scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession

scala> val spark=SparkSession.builder().getOrCreate()
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@2bdab835

//使支持RDDs转换为DataFrames及后续sql操作
scala> import spark.implicits._
import spark.implicits._

scala> val df = spark.read.json("file:///usr/local/spark/examples/src/main/resources/people.json")
df: org.apache.spark.sql.DataFrame = [age: bigint, name: string]

scala> df.show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

6.4 DataFrame的保存

可以使用spark.write操作，把一个DataFrame保存成不同格式的文件，例如，把一个名称为df的DataFrame保存到不同格式文件中，方法如下：

- df.write.json("people.json“)
- df.write.parquet("people.parquet“)
- df.write.csv("people.csv")

下面从示例文件people.json中创建一个DataFrame，然后保存成csv格式文件，代码如下：

```
scala> val peopleDF = spark.read.format("json").
| load("file:///usr/local/spark/examples/src/main/resources/people.json")
scala> peopleDF.select("name", "age").write.format("csv").
| save("file:///usr/local/spark/mycode/sql/newpeople.csv")
```

6.5 DataFrame的常用操作

可以执行一些常用的DataFrame操作

```
// 打印模式信息
scala> df.printSchema()
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)

// 选择多列
scala> df.select(df("name"),df("age")+1).show()
+-------+---------+
|   name|(age + 1)|
+-------+---------+
|Michael|     null|
|   Andy|       31|
| Justin|       20|
+-------+---------+

// 条件过滤
scala> df.filter(df("age") > 20 ).show()
+---+----+
|age|name|
+---+----+
| 30|Andy|
+---+----+
```

6.5 DataFrame的常用操作

```
// 分组聚合
scala> df.groupBy("age").count().show()
+----+-----+
| age|count|
+----+-----+
|  19|    1|
|null|    1|
|  30|    1|
+----+-----+

// 排序
scala> df.sort(df("age").desc).show()
+----+-------+
| age|   name|
+----+-------+
|  30|   Andy|
|  19| Justin|
|null|Michael|
+----+-------+
```

6.5 DataFrame的常用操作

```
//多列排序
scala> df.sort(df("age").desc, df("name").asc).show()
+----+-------+
| age|   name|
+----+-------+
|  30|   Andy|
|  19| Justin|
|null|Michael|
+----+-------+

//对列进行重命名
scala> df.select(df("name").as("username"),df("age")).show()
+--------+----+
|username| age|
+--------+----+
| Michael|null|
|    Andy|  30|
|  Justin|  19|
+--------+----+
```

6.6 从RDD转换得到DataFrame

6.6.1 利用反射机制推断RDD模式
6.6.2 使用编程方式定义RDD模式

6.6.1 利用反射机制推断RDD模式

在“/usr/local/spark/examples/src/main/resources/”目录下，有个Spark安装时自带的样例数据people.txt，其内容如下：

```
Michael, 29
Andy, 30
Justin, 19
```

现在要把people.txt加载到内存中生成一个DataFrame，并查询其中的数据

6.6.1 利用反射机制推断RDD模式

在利用反射机制推断RDD模式时，需要首先定义一个case class，因为，只有case class才能被Spark隐式地转换为DataFrame

```
scala> import org.apache.spark.sql.catalyst.encoders.ExpressionEncoder
import org.apache.spark.sql.catalyst.encoders.ExpressionEncoder
scala> import org.apache.spark.sql.Encoder
import org.apache.spark.sql.Encoder
scala> import spark.implicits._  //导入包，支持把一个RDD隐式转换为一个DataFrame
import spark.implicits._
```

剩余代码见下一页

6.6.1 利用反射机制推断RDD模式

```
scala> case class Person(name: String, age: Long)  //定义一个case class
defined class Person
scala> val peopleDF = spark.sparkContext.
|
textFile("file:///usr/local/spark/examples/src/main/resources/people.txt").
| map(_.split(",")).
| map(attributes => Person(attributes(0),
attributes(1).trim.toInt)).toDF()
peopleDF: org.apache.spark.sql.DataFrame = [name: string, age: bigint]
```

剩余代码见下一页

6.6.1 利用反射机制推断RDD模式

```
scala> peopleDF.createOrReplaceTempView("people") //必须注册为临时表才能供下面的查询使用
scala> val personsRDD = spark.sql("select name,age from people where age > 20")
//最终生成一个DataFrame，下面是系统执行返回的信息
personsRDD: org.apache.spark.sql.DataFrame = [name: string, age: bigint]
scala> personsRDD.map(t => "Name: "+t(0)+ ","+"Age: "+t(1)).show()
//DataFrame中的每个元素都是一行记录，包含name和age两个字段，分别用t(0)和t(1)来获取值
//下面是系统执行返回的信息
+-----------------+
| value|
+-----------------+
|Name:Michael,Age:29|
| Name:Andy,Age:30|
+-----------------+
```

6.6.2 使用编程方式定义RDD模式

当无法提前定义case class时，就需要采用编程方式定义RDD模式。

比如，现在需要通过编程方式把people.txt加载进来生成DataFrame，并完成SQL查询。

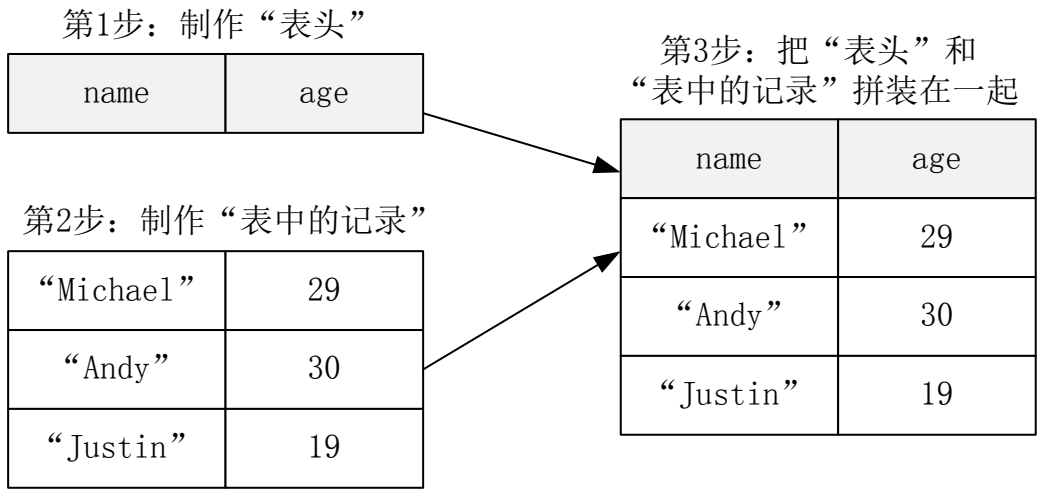


图 通过编程方式定义RDD模式的实现过程

6.6.2 使用编程方式定义RDD模式

```
scala> import org.apache.spark.sql.types._
import org.apache.spark.sql.types._
scala> import org.apache.spark.sql.Row
import org.apache.spark.sql.Row
//生成字段
scala> val fields = Array(StructField("name",StringType,true),
StructField("age",IntegerType,true))
fields: Array[org.apache.spark.sql.types.StructField] =
Array(StructField(name,StringType,true),
StructField(age,IntegerType,true))
scala> val schema = StructType(fields)
schema: org.apache.spark.sql.types.StructType =
StructType(StructField(name,StringType,true), StructField(age,
IntegerType,true))
//从上面信息可以看出，schema描述了模式信息，模式中包含name和age
两个字段
//shcema就是“表头”
```

剩余代码见下一页

6.6.2 使用编程方式定义RDD模式

```
// 下面加载文件生成RDD
scala> val peopleRDD = spark.sparkContext.
| textFile("file:///usr/local/spark/examples/src/main/resources/people.txt")
peopleRDD: org.apache.spark.rdd.RDD[String] =
file:///usr/local/spark/examples/src/main/resources/people.txt
MapPartitionsRDD[1] at textFile at <console>:26
//对peopleRDD 这个RDD中的每一行元素都进行解析
scala> val rowRDD = peopleRDD.map(_.split(",")).
|  map(attributes => Row(attributes(0), attributes(1).trim.toInt))
rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] =
MapPartitionsRDD[3] at map at <console>:29
//上面得到的rowRDD就是“表中的记录”
```

剩余代码见下一页

6.6.2 使用编程方式定义RDD模式

```
//下面把“表头”和“表中的记录”拼装起来
 scala> val peopleDF = spark.createDataFrame(rowRDD, schema)
peopleDF: org.apache.spark.sql.DataFrame = [name: string, age: int]
 //必须注册为临时表才能供下面查询使用
scala> peopleDF.createOrReplaceTempView("people")
 scala> val results = spark.sql("SELECT name,age FROM people")
results: org.apache.spark.sql.DataFrame = [name: string, age: int]
scala> results.
|  map(attributes => "name: " + attributes(0)+","+"age:"+attributes(1)).
|  show()
+-------------------+
| value|
+-------------------+
|name: Michael,age:29|
| name: Andy,age:30|
| name: Justin,age:19|
+-------------------+
```

6.7 使用Spark SQL读写数据库

Spark SQL可以支持Parquet、JSON、Hive等数据源，并且可以通过JDBC连接外部数据源

6.7.1 通过JDBC连接数据库
6.7.2 连接Hive读写数据

6.7.1 通过JDBC连接数据库

1. 准备工作
2. 读取MySQL数据库中的数据
3. 向MySQL数据库写入数据

6.7.1 通过JDBC连接数据库

1. 准备工作

•请参考厦门大学数据库实验室博客教程《 Ubuntu安装MySQL 》，在Linux系统中安装好MySQL数据库
教程地址： http://dblab.xmu.edu.cn/blog/install-mysql/

高校大数据课程
公 共 服 务 平 台

平台每年访问量超过100万次

6.7.1 通过JDBC连接数据库

•在Linux中启动MySQL数据库

```
$ service mysql start
$ mysql -u root -p
#屏幕会提示你输入密码
```

输入下面SQL语句完成数据库和表的创建：

```
mysql> create database spark;
mysql> use spark;
mysql> create table student (id int(4), name char(20), gender char(4), age int(4));
mysql> insert into student values(1,'Xueqian','F',23);
mysql> insert into student values(2,'Weiliang','M',24);
mysql> select * from student;
```

6.7.1 通过JDBC连接数据库

- 下载MySQL的JDBC驱动程序，比如mysql-connector-java-5.1.40.tar.gz
- 把该驱动程序拷贝到spark的安装目录”/usr/local/spark/jars”下
- 启动一个spark-shell，启动Spark Shell时，必须指定mysql连接驱动jar包

```
$ cd /usr/local/spark
$ ./bin/spark-shell  \
--jars /usr/local/spark/jars/mysql-connector-java-5.1.40/mysql-connector-java-5.1.40-bin.jar \
--driver-class-path /usr/local/spark/jars/mysql-connector-java-5.1.40/mysql-connector-java-5.1.40-bin.jar
```

6.7.1 通过JDBC连接数据库

2. 读取MySQL数据库中的数据

执行以下命令连接数据库，读取数据，并显示：

```
scala> val jdbcDF = spark.read.format("jdbc").
| option("url","jdbc:mysql://localhost:3306/spark").
| option("driver","com.mysql.jdbc.Driver").
| option("dbtable", "student").
| option("user", "root").
| option("password", "hadoop").
| load()
scala> jdbcDF.show()
+---+--------+------+---+
| id| name|gender|age|
+---+--------+------+---+
| 1| Xueqian| F| 23|
| 2|Weiliang| M| 24|
+---+--------+------+---+
```

6.7.1 通过JDBC连接数据库

3. 向MySQL数据库写入数据

在MySQL数据库中创建了一个名称为spark的数据库，并创建了一个名称为student的表

创建后，查看一下数据库内容：

```
mysql>  use spark;
Database changed

mysql> select * from student;
//上面命令执行后返回下面结果
+------+----------+--------+------+
| id   | name     | gender | age  |
+------+----------+--------+------+
|    1 | Xueqian  | F      |   23 |
|    2 | Weiliang | M      |   24 |
+------+----------+--------+------+
2 rows in set (0.00 sec)
```

6.7.1 通过JDBC连接数据库

现在开始在spark-shell中编写程序，往spark.student表中插入两条记录

```
import java.util.Properties
import org.apache.spark.sql.types._
import org.apache.spark.sql.Row

//下面我们设置两条数据表示两个学生信息
val studentRDD = spark.sparkContext.parallelize(Array("3 Rongcheng M 26","4
Guanhua M 27")).map(_.split(" "))

//下面要设置模式信息
val schema = StructType(List(StructField("id", IntegerType,
true),StructField("name", StringType, true),StructField("gender", StringType,
true),StructField("age", IntegerType, true)))
```

剩余代码见下一页

6.7.1 通过JDBC连接数据库

```
//下面创建Row对象，每个Row对象都是rowRDD中的一行
val rowRDD = studentRDD.map(p => Row(p(0).toInt, p(1).trim, p(2).trim, p(3).toInt))

//建立起Row对象和模式之间的对应关系，也就是把数据和模式对应起来
val studentDF = spark.createDataFrame(rowRDD, schema)

//下面创建一个prop变量用来保存JDBC连接参数
val prop = new Properties()
prop.put("user", "root") //表示用户名是root
prop.put("password", "hadoop") //表示密码是hadoop
prop.put("driver","com.mysql.jdbc.Driver") //表示驱动程序是com.mysql.jdbc.Driver

//下面就可以连接数据库，采用append模式，表示追加记录到数据库spark的student表中
studentDF.write.mode("append").jdbc("jdbc:mysql://localhost:3306/spark",
"spark.student", prop)
```

6.7.1 通过JDBC连接数据库

可以看一下效果，看看MySQL数据库中的spark.student表发生了什么变化

```
mysql> select * from student;
+------+----------+--------+------+
| id | name | gender | age |
+------+----------+--------+------+
| 1 | Xueqian | F | 23 |
| 2 | Weiliang | M | 24 |
| 3 | Rongcheng | M | 26 |
| 4 | Guanhua | M | 27 |
+------+----------+--------+------+
4 rows in set (0.00 sec)
```

6.7.2 连接Hive读写数据

1. 准备工作
2. 在Hive中创建数据库和表
3. 连接Hive读写数据

6.7.2 连接Hive读写数据

1.准备工作

数据仓库（Data Warehouse）是一个面向主题的（Subject Oriented）、集成的（Integrated）、相对稳定的（Non-Volatile）、反映历史变化（Time Variant）的数据集合，用于支持管理决策。

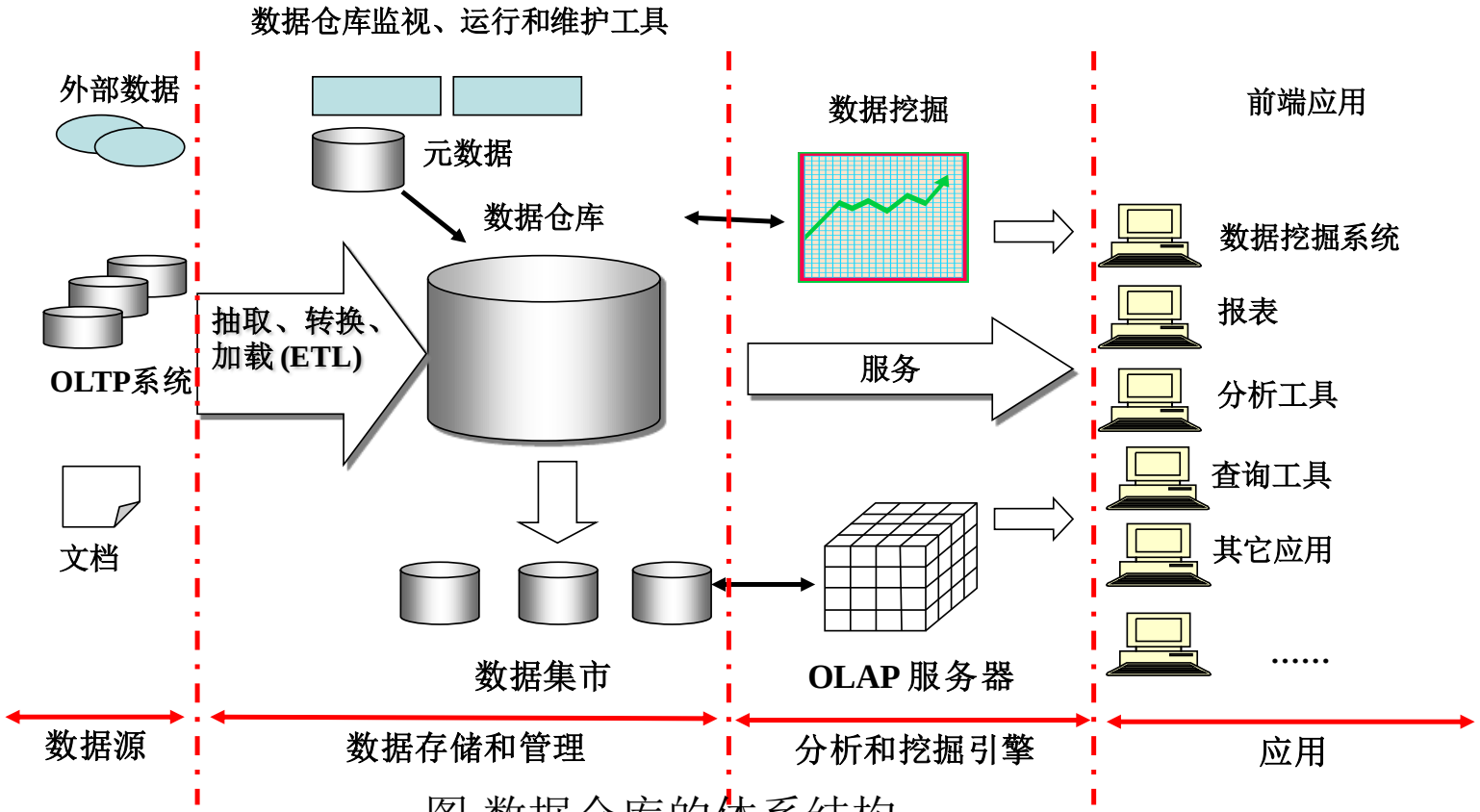


图 数据仓库的体系结构

6.7.2 连接Hive读写数据

- Hive是一个构建于Hadoop顶层的数据仓库工具
- 支持大规模数据存储、分析，具有良好的可扩展性

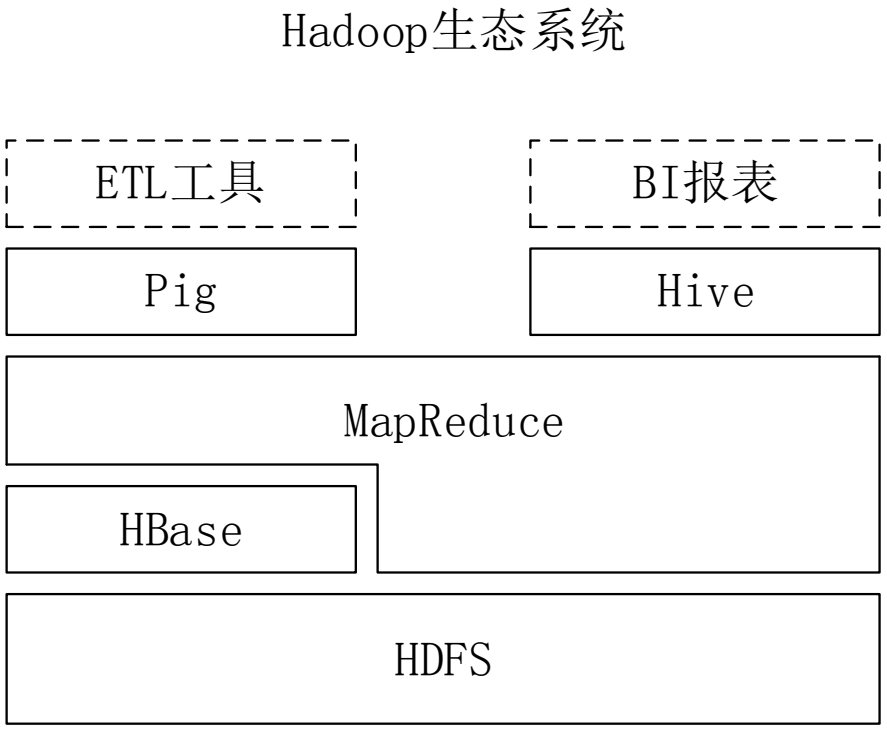
- 某种程度上可以看作是用户编程接口，本身不存储和处理数据
- 依赖分布式文件系统HDFS存储数据
- 依赖分布式并行计算模型MapReduce处理数据

- 定义了简单的类似SQL 的查询语言——HiveQL
- 用户可以通过编写的HiveQL语句运行MapReduce任务

- 可以很容易把原来构建在关系数据库上的数据仓库应用程序移植到Hadoop平台上
- 是一个可以提供有效、合理、直观组织和使用数据的分析工具

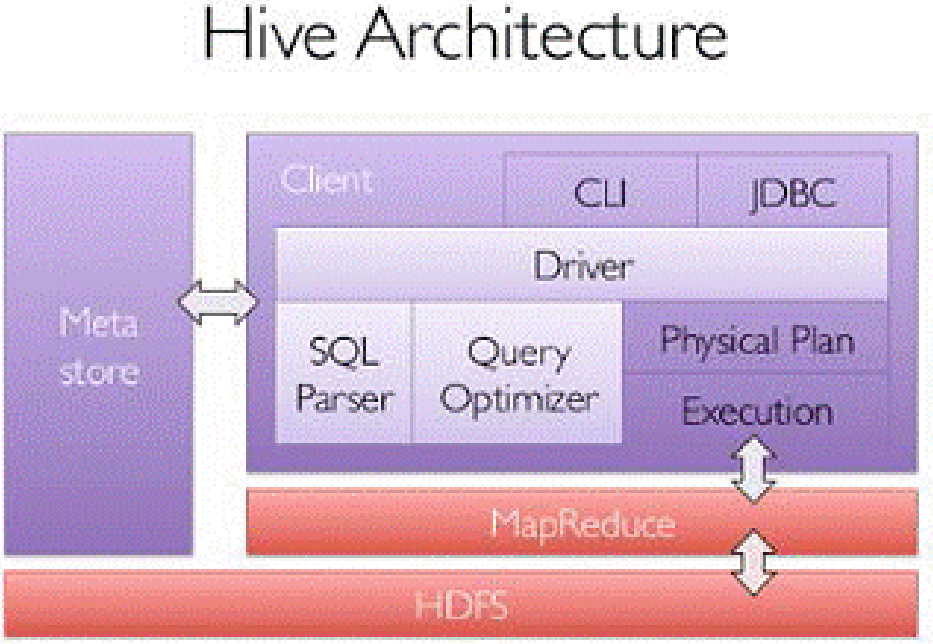
6.7.2 连接Hive读写数据

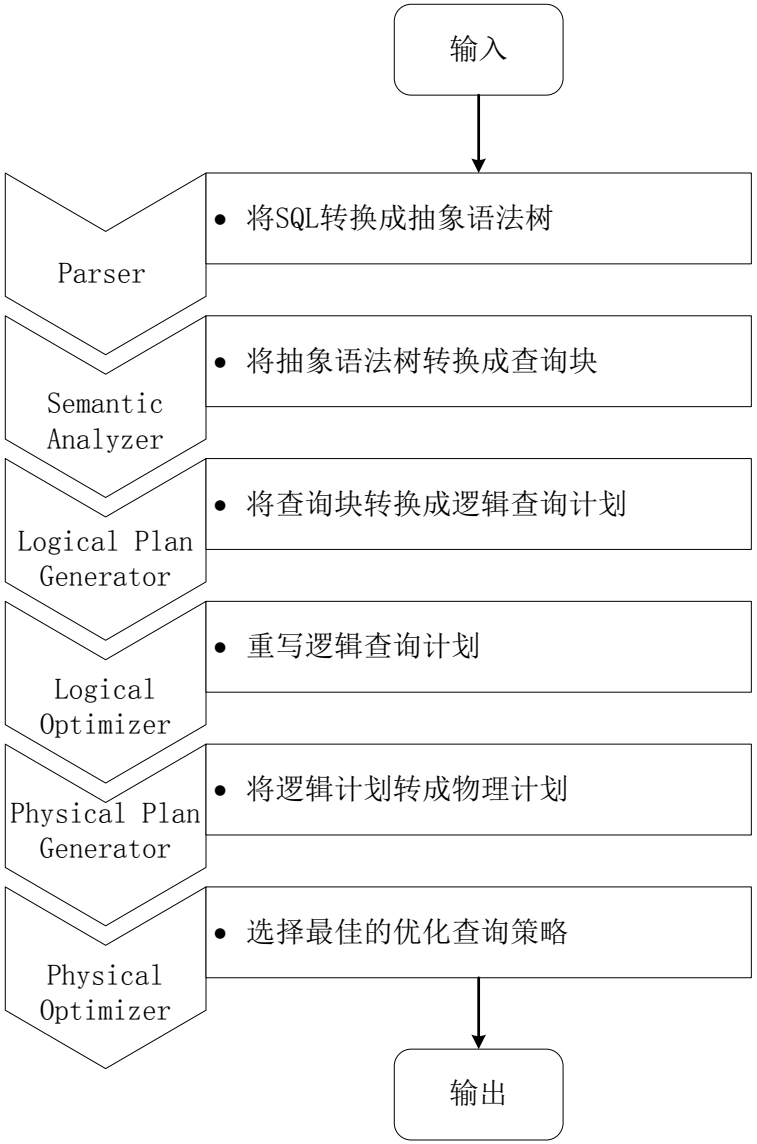
- **Hive依赖于HDFS 存储数据**
- **Hive依赖于MapReduce 处理数据**



Hive与Hadoop生态系统中其他组件的关系

6.7.2 连接Hive读写数据





6.7.2 连接Hive读写数据

•Hive的安装，请参考厦门大学数据库实验室建设的高校大数据课程公共服务平台上的技术博客：

•《Ubuntu安装hive，并配置mysql作为元数据库》

http://dblab.xmu.edu.cn/blog/install-hive/

6.7.2 连接Hive读写数据

•为了让Spark能够访问Hive，必须为Spark添加Hive支持
•Spark官方提供的预编译版本，通常是不包含Hive支持的，需要采用源码编译，编译得到一个包含Hive支持的Spark版本

（1）测试一下自己电脑上已经安装的Spark版本是否支持Hive

启动进入了spark-shell，如果不支持Hive，会显示如下信息：

```
scala> import org.apache.spark.sql.hive.HiveContext
<console>:25: error: object hive is not a member of package org.apache.spark.sql
         import org.apache.spark.sql.hive.HiveContext
                                          ^
```

如果你当前电脑上的Spark版本包含Hive支持，那么应该显示下面的正确信息：

```
scala> import org.apache.spark.sql.hive.HiveContext
import org.apache.spark.sql.hive.HiveContext
```

6.7.2 连接Hive读写数据

（2）采用源码编译方法得到支持Hive的Spark版本

到Spark官网下载源码
http://spark.apache.org/downloads.html

6.7.2 连接Hive读写数据

解压文件

```
$ cd /home/hadoop/下载 //spark-2.1.0.tgz就在这个目录下面
$ ls #可以看到刚才下载的spark-2.1.0.tgz文件
$ sudo tar -zxf ./spark-2.1.0.tgz -C /home/hadoop/
$ cd /home/hadoop
$ ls #这时可以看到解压得到的文件夹spark-2.1.0
```

在编译时，需要给出电脑上之前已经安装好的Hadoop的版本

```
$ hadoop version
```

运行编译命令，对Spark源码进行编译

```
$ cd /home/hadoop/spark-2.1.0
$ ./dev/make-distribution.sh —tgz —name h27hive -Pyarn -Phadoop-2.7 -
Dhadoop.version=2.7.1 -Phive -Phive-thriftserver -DskipTests
```

编译成功后会得到文件名“spark-2.1.0-bin-h27hive.tgz”，这个就是包含Hive支持的Spark安装文件

6.7.2 连接Hive读写数据

（3）安装支持Hive的Spark版本

Spark的安装详细过程，请参考厦门大学数据库实验室建设的高校大数据课程公共服务平台上的技术博客：

《Spark2.1.0入门：Spark的安装和使用》

博客地址：http://dblab.xmu.edu.cn/blog/1307-2/

启动进入了spark-shell，由于已经可以支持Hive，会显示如下信息：

```
scala> import org.apache.spark.sql.hive.HiveContext
import org.apache.spark.sql.hive.HiveContext
```

6.7.2 连接Hive读写数据

2.在Hive中创建数据库和表

假设已经完成了Hive的安装，并且使用的是MySQL数据库来存放Hive的元数据

需要借助于MySQL保存Hive的元数据，首先启动MySQL数据库：

```
$ service mysql start
```

由于Hive是基于Hadoop的数据仓库，使用HiveQL语言撰写的查询语句，最终都会被Hive自动解析成MapReduce任务由Hadoop去具体执行，因此，需要启动Hadoop，然后再启动Hive

启动Hadoop：

```
$ cd /usr/local/hadoop
$ ./sbin/start-all.sh
```

Hadoop启动成功以后，可以再启动Hive：

```
$ cd /usr/local/hive
$ ./bin/hive
```

6.7.2 连接Hive读写数据

进入Hive，新建一个数据库sparktest，并在这个数据库下面创建一个表student，并录入两条数据

```
hive> create database if not exists sparktest;//创建数据库sparktest
hive> show databases; //显示一下是否创建出了sparktest数据库
//下面在数据库sparktest中创建一个表student
hive> create table if not exists sparktest.student(
> id int,
> name string,
> gender string,
> age int);
hive> use sparktest; //切换到sparktest
hive> show tables; //显示sparktest数据库下面有哪些表
hive> insert into student values(1,'Xueqian','F',23); //插入一条记录
hive> insert into student values(2,'Weiliang','M',24); //再插入一条记录
hive> select * from student; //显示student表中的记录
```

6.7.2 连接Hive读写数据

3.连接Hive读写数据

需要修改“/usr/local/sparkwithhive/conf/spark-env.sh”这个配置文件：

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export CLASSPATH=$CLASSPATH:/usr/local/hive/lib
export SCALA_HOME=/usr/local/scala
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export HIVE_CONF_DIR=/usr/local/hive/conf
export SPARK_CLASSPATH=$SPARK_CLASSPATH:/usr/local/hive/lib/mysql-connector-java-5.1.40-bin.jar
```

6.7.2 连接Hive读写数据

请在spark-shell（包含Hive支持）中执行以下命令从Hive中读取数据：

```
Scala> import org.apache.spark.sql.Row
Scala> import org.apache.spark.sql.SparkSession
Scala> case class Record(key: Int, value: String)
// warehouseLocation points to the default location for
managed databases and tables
Scala> val warehouseLocation = "spark-warehouse”
Scala> val spark = SparkSession.builder().appName("Spark
Hive Example").config("spark.sql.warehouse.dir",
warehouseLocation).enableHiveSupport().getOrCreate()
Scala> import spark.implicits._
Scala> import spark.sql
```

剩余代码见下一页

6.7.2 连接Hive读写数据

```
//下面是运行结果
scala> sql("SELECT * FROM sparktest.student").show()
+---+--------+------+---+
| id| name|gender|age|
+---+--------+------+---+
| 1| Xueqian| F| 23|
| 2|Weiliang| M| 24|
+---+--------+------+---+
```

6.7.2 连接Hive读写数据

编写程序向Hive数据库的sparktest.student表中插入两条数据
在插入数据之前，先查看一下已有的2条数据

```
hive> use sparktest;
OK
Time taken: 0.016 seconds

hive> select * from student;
OK
1    Xueqian F    23
2    Weiliang     M    24
Time taken: 0.05 seconds, Fetched: 2 row(s)
```

6.7.2 连接Hive读写数据

编写程序向Hive数据库的sparktest.student表中插入两条数据：

```
scala> import java.util.Properties
scala> import org.apache.spark.sql.types._
scala> import org.apache.spark.sql.Row
//下面我们设置两条数据表示两个学生信息
scala> val studentRDD =
spark.sparkContext.parallelize(Array("3 Rongcheng M
26","4 Guanhua M 27")).map(_.split(" "))
```

剩余代码见下一页

6.7.2 连接Hive读写数据

```
//下面要设置模式信息
scala> val schema = StructType(List(StructField("id",
IntegerType, true),StructField("name", StringType,
true),StructField("gender", StringType,
true),StructField("age", IntegerType, true)))
//下面创建Row对象，每个Row对象都是rowRDD中的一行
scala> val rowRDD = studentRDD.map(p => Row(p(0).toInt,
p(1).trim, p(2).trim, p(3).toInt))
//建立起Row对象和模式之间的对应关系，也就是把数据和模式对应起来
scala> val studentDF = spark.createDataFrame(rowRDD,
schema)
```

剩余代码见下一页

6.7.2 连接Hive读写数据

```
//查看studentDF
scala> studentDF.show()
+---+---------+------+---+
| id| name|gender|age|
+---+---------+------+---+
| 3|Rongcheng| M| 26|
| 4| Guanhua| M| 27|
+---+---------+------+---+
//下面注册临时表
scala> studentDF.registerTempTable("tempTable")

scala> sql("insert into sparktest.student select * from
tempTable")
```

6.7.2 连接Hive读写数据

输入以下命令查看Hive数据库内容的变化：

```
hive> select * from student;
OK
1    Xueqian F    23
2    Weiliang     M    24
3    Rongcheng    M    26
4    Guanhua M    27
Time taken: 0.049 seconds, Fetched: 4 row(s)
```

可以看到，插入数据操作执行成功了!

附录A：主讲教师林子雨简介

主讲教师：林子雨

单位：厦门大学计算机科学系
E-mail: ziyulin@xmu.edu.cn
个人网页：http://www.cs.xmu.edu.cn/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn

扫一扫访问个人主页

林子雨，男，1978年出生，博士（毕业于北京大学），现为厦门大学计算机科学系助理教授（讲师），曾任厦门大学信息科学与技术学院院长助理、晋江市发展和改革局副局长。中国计算机学会数据库专业委员会委员，中国计算机学会信息系统专业委员会委员。国内高校首个“数字教师”提出者和建设者，厦门大学数据库实验室负责人，厦门大学云计算与大数据研究中心主要建设者和骨干成员，2013年度和2017年度厦门大学教学类奖教金获得者，荣获2017年福建省精品在线开放课程和2017年厦门大学高等教育成果二等奖。主要研究方向为数据库、数据仓库、数据挖掘、大数据、云计算和物联网，并以第一作者身份在《软件学报》《计算机学报》和《计算机研究与发展》等国家重点期刊以及国际学术会议上发表多篇学术论文。作为项目负责人主持的科研项目包括1项国家自然科学青年基金项目(No.61303004)、1项福建省自然科学青年基金项目(No.2013J05099)和1项中央高校基本科研业务费项目(No.2011121049)，主持的教改课题包括1项2016年福建省教改课题和1项2016年教育部产学协作育人项目，同时，作为课题负责人完成了国家发改委城市信息化重大课题、国家物联网重大应用示范工程区域试点泉州市工作方案、2015泉州市互联网经济调研等课题。中国高校首个“数字教师”提出者和建设者，2009年至今，“数字教师”大平台累计向网络免费发布超过500万字高价值的研究和教学资料，累计网络访问量超过500万次。打造了中国高校大数据教学知名品牌，编著出版了中国高校第一本系统介绍大数据知识的专业教材《大数据技术原理与应用》，并成为京东、当当网等网店畅销书籍；建设了国内高校首个大数据课程公共服务平台，为教师教学和学生学习大数据课程提供全方位、一站式服务，年访问量超过100万次。

附录B：大数据学习路线图

大数据学习路线图访问地址：http://dblab.xmu.edu.cn/post/10164/

附录C：《大数据技术原理与应用》教材

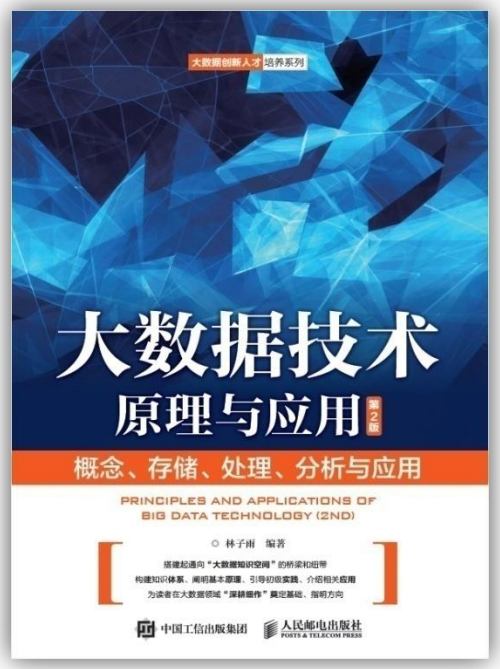
《大数据技术原理与应用——概念、存储、处理、分析与应用（第2版）》，由厦门大学计算机科学系林子雨博士编著，是国内高校第一本系统介绍大数据知识的专业教材。人民邮电出版社 ISBN:978-7-115-44330-4 定价：49.80元

全书共有15章，系统地论述了大数据的基本概念、大数据处理架构Hadoop、分布式文件系统HDFS、分布式数据 库HBase、NoSQL数据库、云数据库、分布式并行编程模型MapReduce、Spark、流计算、图计算、数据可视化以及大数据在互联网、生物医学和物流等各个领域的应用。在Hadoop、HDFS、HBase和MapReduce等重要章节，安排了入门级的实践操作，让读者更好地学习和掌握大数据关键技术。

本书可以作为高等院校计算机专业、信息管理等相关专业的大数据课程教材，也可供相关技术人员参考、学习、培训之用。

欢迎访问《大数据技术原理与应用——概念、存储、处理、分析与应用》教材官方网站：
http://dblab.xmu.edu.cn/post/bigdata

扫一扫访问教材官网



附录D：《大数据基础编程、实验和案例教程》

本书是与《大数据技术原理与应用（第2版）》教材配套的唯一指定实验指导书

大数据教材

1+1黄金组合

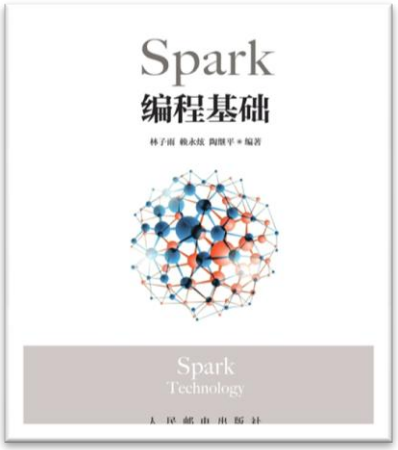
厦门大学林子雨编著

配套实验指导书

清华大学出版社 ISBN:978-7-302-47209-4 定价：59元

- 步步引导，循序渐进，详尽的安装指南为顺利搭建大数据实验环境铺平道路
- 深入浅出，去粗取精，丰富的代码实例帮助快速掌握大数据基础编程方法
- 精心设计，巧妙融合，五套大数据实验题目促进理论与编程知识的消化和吸收
- 结合理论，联系实际，大数据课程综合实验案例精彩呈现大数据分析全流程

附录E：《Spark编程基础》



《Spark编程基础》

厦门大学 林子雨，赖永炫，陶继平 编著

披荆斩棘，在大数据丛林中开辟学习捷径
填沟削坎，为快速学习Spark技术铺平道路
深入浅出，有效降低Spark技术学习门槛
资源全面，构建全方位一站式在线服务体系

人民邮电出版社出版发行，ISBN:978-7-115-47598-5
教材官网：http://dblab.xmu.edu.cn/post/spark/

本书以Scala作为开发Spark应用程序的编程语言，系统介绍了Spark编程的基础知识。全书共8章，内容包括大数据技术概述、Scala语言基础、Spark的设计与运行原理、Spark环境搭建和使用方法、RDD编程、Spark SQL、Spark Streaming、Spark MLlib等。本书每个章节都安排了入门级的编程实践操作，以便读者更好地学习和掌握Spark编程方法。本书官网免费提供了全套的在线教学资源，包括讲义PPT、习题、源代码、软件、数据集、授课视频、上机实验指南等。

附录F：高校大数据课程公共服务平台

高校大数据课程

公 共 服 务 平 台

http://dblab.xmu.edu.cn/post/bigdata-teaching-platform/

扫一扫访问平台主页

扫一扫观看3分钟FLASH动画宣传片


Thank You!