1. 数据预处理

数据来源

movieLens 数据集

links.txt

```
1,0114709,862
2,0113497,8844
3,0113228,15602
4,0114885,31357
5,0113041,11862
```

movies.txt

```
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
5,Father of the Bride Part II (1995),Comedy
```

ratings.txt

```
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
5,Father of the Bride Part II (1995),Comedy
```

数据处理

为了实现高效的数据读取与处理，这里我们不是直接读取文件，而是实现一个 ETTL 类，将数据存在数据库中，实现快速读取。

构建 case class

```
case class Links(movieId:Int,imdbId:Int,tmdbId:Int)
case class Movies(movieId:Int,title:String,genres:String)
case class Ratings(userId:Int,movieId:Int,rating:Double,timestamp:Int)
case class Result(userId:Int,movieId:Int,rating:Double)
case class Tags(userId:Int,movieId:Int,tag:String,timestamp:Int)
```

数据清洗

```scala
val localClusterURL = "local[2]"
val clusterMasterURL = "spark://master:7077"
val conf = new SparkConf().setAppName("ETL").setMaster(clusterMasterURL)
val sc = new SparkContext(conf)

//设置RDD的partition的数量一般以集群分配给应用的CPU核数的整数倍为宜。
//至少有多少个 partition,
val minPartitions = 8
//    通过case class来定义Links的数据结构，数据的schema，适用于schama已知的数据
//也可以通过StructType的方式，适用于schema未知的数据
//    http://spark.apache.org/docs/1.6.2/sql-programming-
guide.html#programmatically-specifying-the-schema
val links = sc.textFile("data/links.txt", minPartitions) //driver
.filter { !_.endsWith(",") } //executor
.map(_.split(",")) //executor
//movieId:Int,imdbId:Int,tmdbId:Int
.map(x => Links(x(0).trim.toInt, x(1).trim.toInt, x(2).trim().toInt))
//executor
.toDF()//将 Links 转成 DataFrame

val movies = sc.textFile("data/movies.txt", minPartitions)
.filter { !_.endsWith(",") }
.map(_.split(","))
.map(x => Movies(x(0).trim().toInt, x(1).trim(), x(2).trim()))
.toDF()

val ratings = sc.textFile("data/ratings.txt", minPartitions).filter {
!_.endsWith(",") }
.map(_.split(","))
.map(x => Ratings(x(0).trim().toInt, x(1).trim().toInt,
x(2).trim().toDouble, x(3).trim().toInt))
.toDF()

val tags = sc.textFile("data/tags.txt", minPartitions)
.filter { !_.endsWith(",") }
.map(x=>rebuild(x))
.map(_.split(","))
.map(x => Tags(x(0).trim().toInt, x(1).trim().toInt, x(2).trim(),
x(3).trim().toInt))
.toDF()

private def rebuild(input:String):String = {
    // 29,95875,"Philip K. Dick",1378487045
    val a = input.split(",") // Array(29,95875,Philip K. Dick,1378487045)
    val head = a.take(2).mkString(",") // 29,95875
    val tail = a.takeRight(1).mkString // 1378487045
    //去掉 中间字符 的引号
    val b = a.drop(2).dropRight(1).mkString.replace("\"", "")  // Philip K.
Dick
```

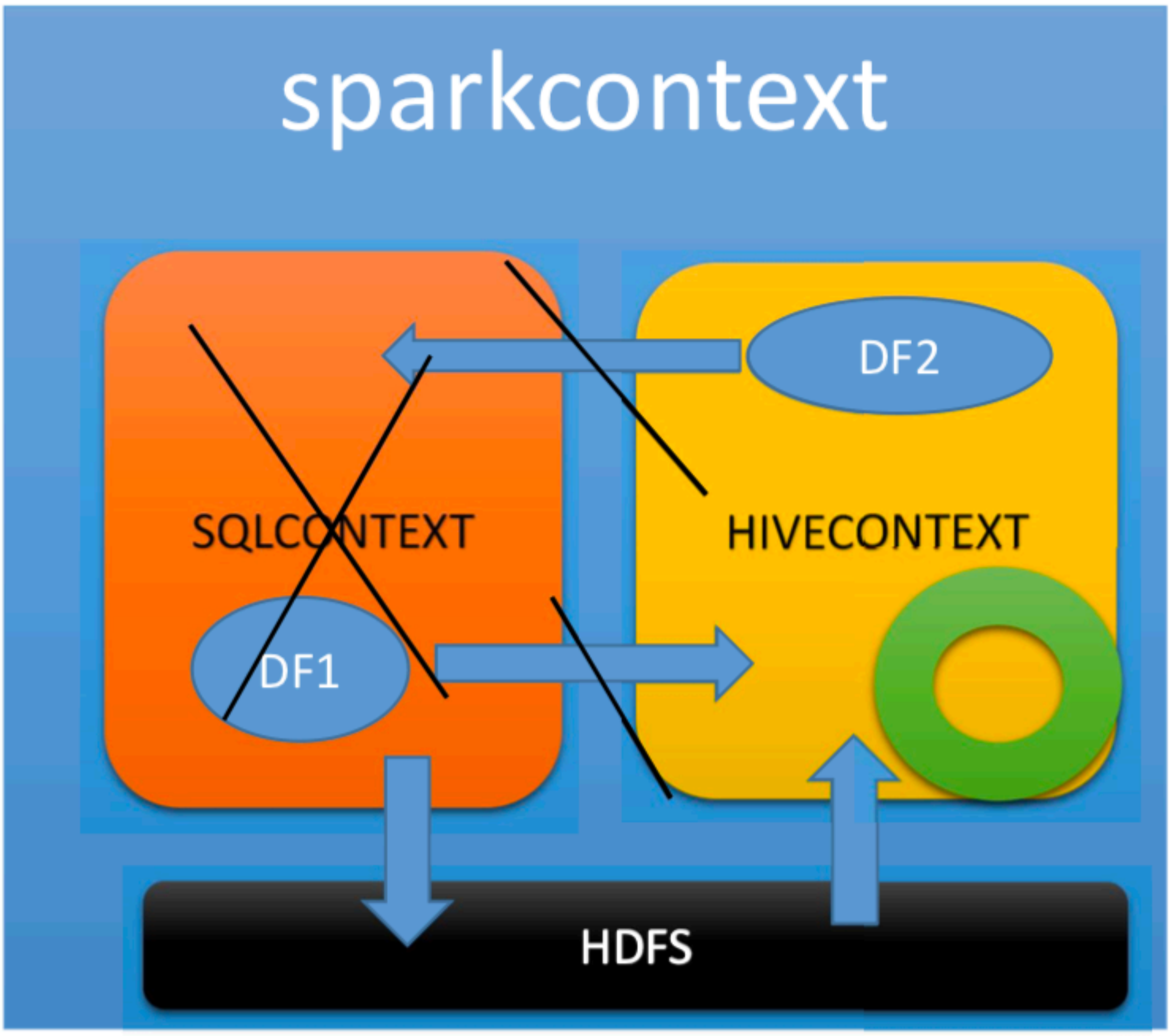
```
    val output = head + "," + b + "," + tail // 29,95875,Philip K.
Dick,1378487045
    output
}
```

将数据写入 HDFS，将表存入 Hive

在 SparkContext 中，SQLContext 创建的表是临时的，当i、程序结束，SQLContext 变量就不存在。

SQLContext 和 HiveContext 中的 DataFrame 是不能直接通信，互相读取数据的，只能通过 HDFS ，利用 SQLContext 将 DF 数据写入 HDFS 中，通过 HiveContext 从 HDFS 中将 DF 数据读到 Hive 中，达到数据存储的目的。

优点：将数据存到数据仓库，即使 APP 端数据已经关闭，仍然可以从数据仓库中读取数据。



```
val sqlContext = new SQLContext(sc)
val hc = new HiveContext(sc)

//在SQLCONTEXT下只能构建临时表
//links.registerTempTable("links0629")
//通过数据写入到HDFS，将表存到hive中
```

```
//links
//Overwrite：以前有则覆盖，否则写入
links.write.mode(SaveMode.Overwrite).parquet("/tmp/links")
hc.sql("drop table if exists links")
//创建好表以后，将数据写入到表中
hc.sql("create table if not exists links(movieId int,imdbId int,tmdbId int)
stored as parquet")
hc.sql("load data inpath '/tmp/links' overwrite into table links")

//movies
movies.write.mode(SaveMode.Overwrite).parquet("/tmp/movies")
hc.sql("drop table if exists movies")
hc.sql("create table if not exists movies(movieId int,title string,genres
string) stored as parquet")
hc.sql("load data inpath '/tmp/movies' overwrite into table movies")

//ratings
ratings.write.mode(SaveMode.Overwrite).parquet("/tmp/ratings")
hc.sql("drop table if exists ratings")
hc.sql("create table if not exists ratings(userId int,movieId int,rating
double,timestamp int) stored as parquet")
hc.sql("load data inpath '/tmp/ratings' overwrite into table ratings")

//tags
tags.write.mode(SaveMode.Overwrite).parquet("/tmp/tags")
hc.sql("drop table if exists tags")
hc.sql("create table if not exists tags(userId int,movieId int,tag
string,timestamp int) stored as parquet")
hc.sql("load data inpath '/tmp/tags' overwrite into table tags")
```

查看 hadoop 中文件

```
hadoop fs -ls /user/hive/warehouse

spark-submit --class
```