1. 讲师介绍

讲师：马中华

1、前Oracle数据开发技术组Leader
2、前动批网大数据运营平台负责人
3、2018阿里杭州云栖大会特邀NLP专场演讲嘉宾

NX 奈学教育

Hive底层执行引擎深度剖析

一次课让你成为Hive真正顶尖高手

课程大纲

（1）Hive架构设计

（2）HQL转换成MapReduce底层核心逻辑

（3）HQL转换MapReduce六大核心步骤原理

马中华

前Oracle数据开发技术组负责人

历史受训学员2000+

2020年6月19日晚20：00准时开播

找我！获取 免费 听课权限

2. 课程介绍

2.1. 课程主题

Hive到底是怎么把HQL转换成MapReduce的?

```
Hive的HQL编译过程
```

详解介绍:

```
（1）Hive架构设计
（2）HQL转换成MapReduce的底层核心逻辑
（3）HQL转换MapReduce六大核心步骤原理
```

2.2. 开课目的

FaceBook 的工程师早期在使用开源 Hadoop 进行海量数据分析的时候，发现直接编写 MapReduce 比较低效，遂研发了一个Hadoop的SQL客户端来管理存储在 Hadoop 中的结构化数据，从而提高开发效率，而且也降低了入门大数据开发和分析的门槛。

未来开发的趋势:

```
1、Web UI 平台
2、SQL 开发 / 拖拽开发
3、流式计算处理引擎
```

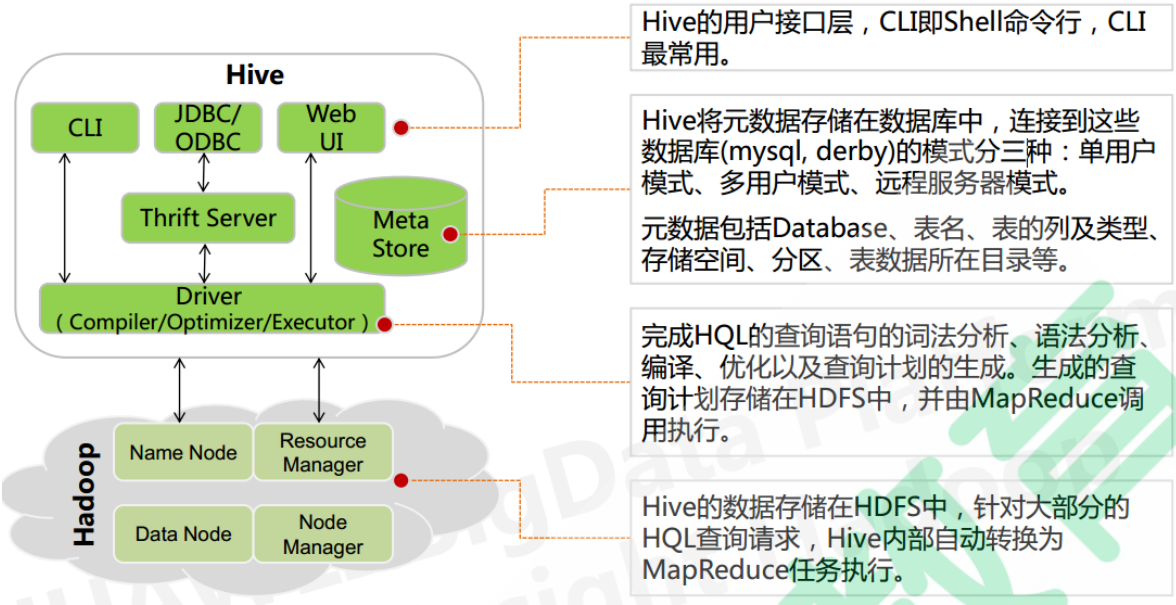
一句话总结就是：将来的大数据处理，都是平台化的管理方式，底层运行的是高效的流式计算引擎，用户的不是直接编写流式计算引擎的应用代码，而是SQL语句。

3. Hive的概念

Hive依赖于HDFS存储数据，Hive将HQL转换成MapReduce执行，所以说Hive是基于Hadoop的一个数据仓库工具，实质就是一款基于HDFS的MapReduce计算框架，对存储在HDFS中的数据进行分析和管理。

```
1、Hive由Facebook实现并开源
2、Hive是基于Hadoop的一个数据仓库工具
3、Hive存储的数据其实底层存储在HDFS上
4、Hive将HDFS上的结构化的数据映射为一张数据库表，类似于excel或者msyql的表
5、Hive提供HQL(Hive SQL)查询功能
6、Hive的本质是将SQL语句转换为MapReduce任务运行，使不熟悉MapReduce的用户很方便地利用HQL处理和计算HDFS上的结构化的数据，适用于离线的批量数据计算
7、Hive使用户可以极大简化分布式计算程序的编写，而将精力集中于业务逻辑
```

4. Hive的架构设计



5. Hive的支持的语法

支持的语法：

```
1、select * from db.table1
2、select count(distinct uid) from db.table1
3、支持select、union all、join（left、right、full join）、like、where、having、各种聚合函数、支持json解析
4、UDF（User Defined Function）/ UDAF/UDTF
5、不支持update和delete
6、hive虽然支持in/exists（老版本是不支持的），但是hive推荐使用semi join的方式来代替实现，而且效率更高。
7、支持case … when …
```

不支持的语法：

```
1、支持等值链接，不支持非等值链接
2、支持and多条件过滤，不支持or多条件过滤
3、不支持update和delete
```

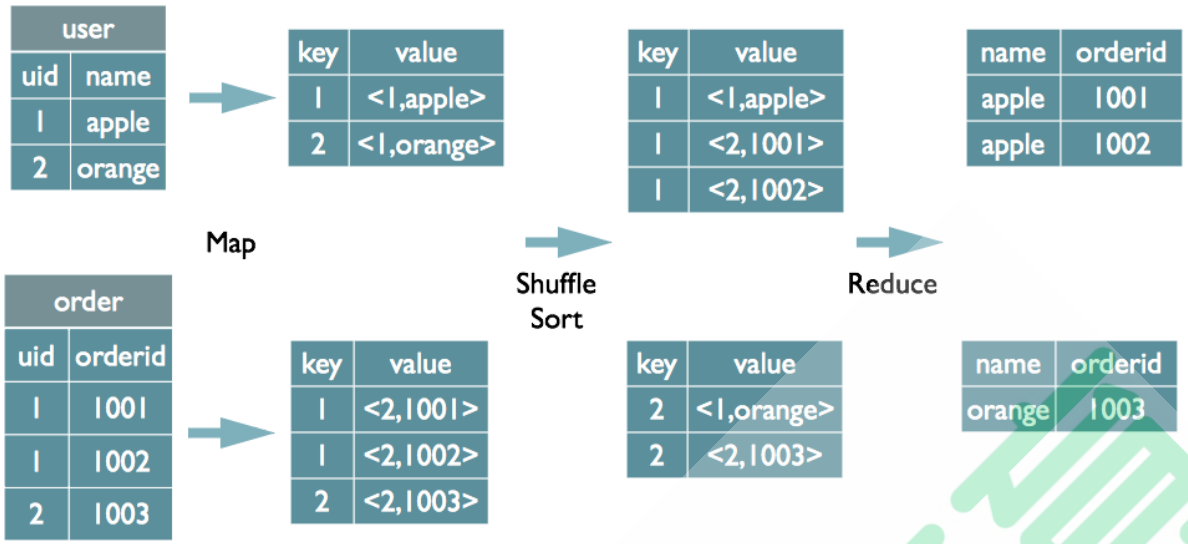
6. MapReduce实现基本SQL操作的原理

6.1. join 实现

代码：

```
select u.name, o.orderid from order o join user u on o.uid = u.uid;
```

实现思路图：

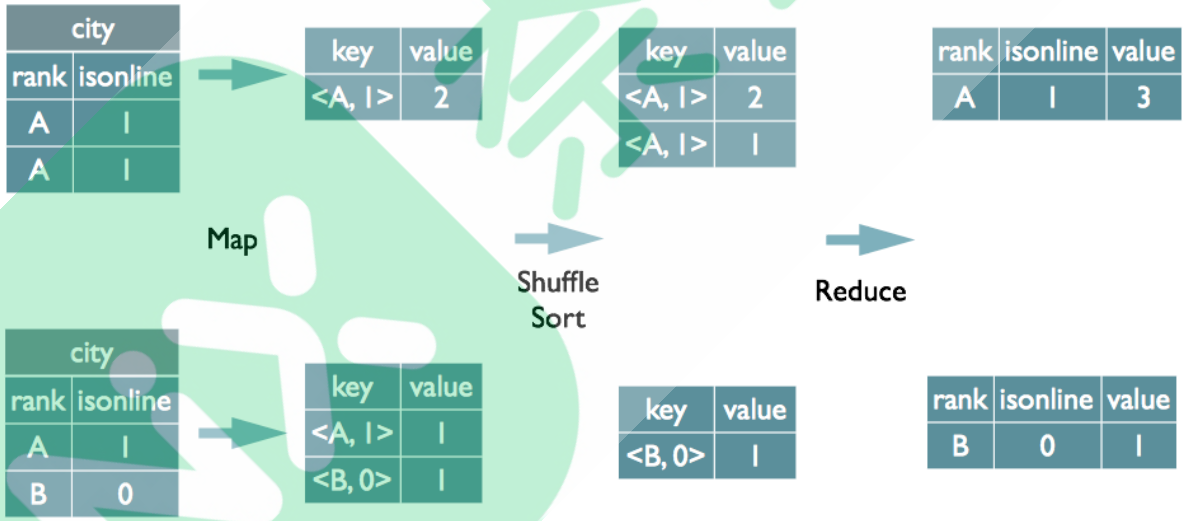


6.2. group by 实现

代码：

```
select rank, isonline, count(*) from city group by rank, isonline;
```

实现思路图：

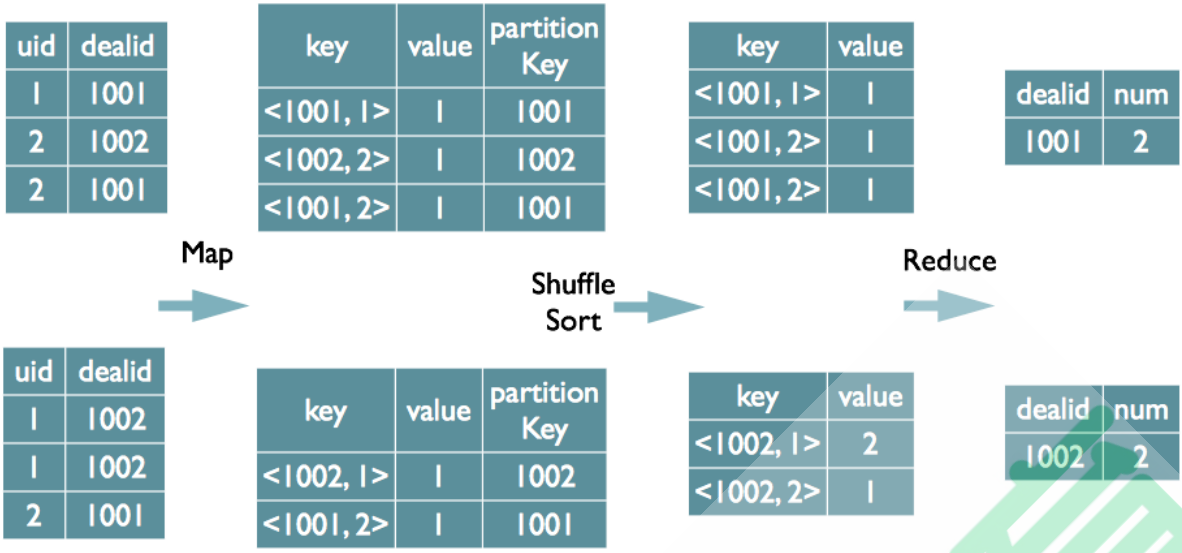


6.3. distinct 实现

代码：

```
select dealid, count(distinct uid) num from order group by dealid;
```

实现思路图：



7. Hive的HQL怎么转换成MapReduce?

了解了MapReduce实现SQL基本操作之后，我们来看看Hive是如何将SQL转化为MapReduce任务的，整个编译过程分为六个阶段：

```
1、Antlr定义SQL的语法规则，完成SQL词法，语法解析，将SQL转化为抽象语法树AST Tree
2、遍历AST Tree，抽象出查询的基本组成单元QueryBlock
3、遍历QueryBlock，翻译为执行操作树OperatorTree
4、逻辑层优化器进行OperatorTree变换，合并不必要的ReduceSinkOperator，减少shuffle数据量
5、遍历OperatorTree，翻译为MapReduce任务
6、物理层优化器进行MapReduce任务的变换，生成最终的执行计划
```

8. 详细过程

8.1. 第一阶段：SQL词法，语法解析

8.1.1. antlr介绍

Hive 使用 Antlr 实现 SQL 的词法和语法解析。Antlr 是一种语言识别的工具，可以用来构造领域语言。Antlr 完成了词法分析、语法分析、语义分析、中间代码生成的过程。

Antlr的工作方式：编写一个语法文件，构造特定规则的语法，定义定义语法和词法规则完成最终的替换，生成代码。

Hive的语法规则和词法规则，都是定义在类似于 xxx.g 的文件中。其中：

0.10x版本以前：

```
一个统一的语法和词法文件：Hive.g
```

0.11版本往后：

```
定义词法规则：HiveLexer.g
定义语法规则：SelectClauseParser.g，FromClauseParser.g，IdentifiersParser.g，
HiveParser.g
```

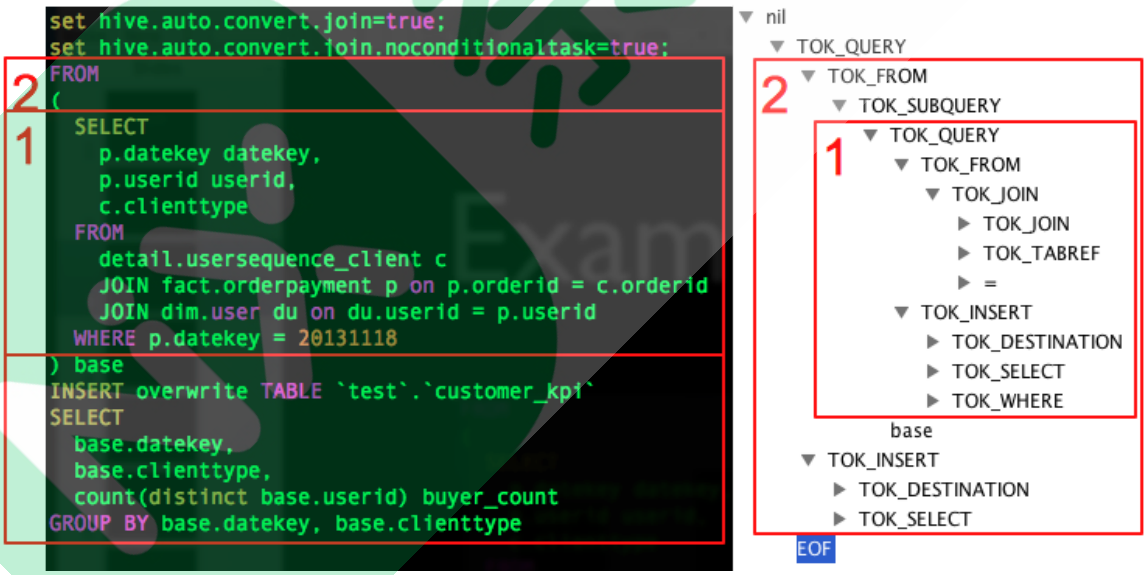
Hive 接收到用户编写的 HQL，就通过 antlr 进行解析生成代码。

8.1.2. 第一阶段：SQL生成抽象语法树AST Tree

Antlr 对 Hive SQL 解析的代码如下，HiveLexerX，HiveParser 分别是 Antlr 对语法文件 Hive.g 编译后自动生成的词法解析和语法解析类，在这两个类中进行复杂的解析。

代码跳转关系：

```
CliDriver.main()
CliDriver.run()
CliDriver.executeDriver()
CliDriver.processLine()
CliDriver.processCmd()
CliDriver.processLocalCmd()          # 完整的执行，输出过程
Driver.run()
Driver.runInternal()                 # 编译和执行
Driver.compileInternal()             # 编译
Driver.compile()         # 编译：SQL -> AST -> ResovleTree -> OperatorTree ->
TaskTree
ParseUtils.parse()
ParseDriver.parse()
```



8.1.3. 第二阶段：SQL基本组成单元QueryBlock

AST Tree 仍然非常复杂，不够结构化，不方便直接翻译为MapReduce程序，AST Tree 转化为 QueryBlock 就是将SQL 进一部抽象和结构化。

QueryBlock 是一条 SQL 最基本的组成单元，包括三个部分：输入源，计算过程，输出。QueryBlock 可理解子查询

详见源代码：

```
org.apache.hadoop.hive.ql.parse.QB
```

AST Tree 生成 QueryBlock 的过程是一个递归的过程，先序遍历 AST Tree，遇到不同的 Token 节点，保存到相应的属性中，主要包含以下几个过程：

```
TOK_QUERY => 创建QB对象，循环递归子节点
TOK_FROM => 将表名语法部分保存到QB对象的aliasToTabs等属性中
TOK_INSERT => 循环递归子节点
TOK_DESTINATION => 将输出目标的语法部分保存在QBParseInfo对象的nameToDest属性中
TOK_SELECT => 分别将查询表达式的语法部分保存在destToSelExpr、destToAggregationExprs、
destToDistinctFuncExprs三个属性中
TOK_WHERE => 将where部分的语法保存在QBParseInfo对象的destToWhereExpr属性中
```

8.1.4. 第三阶段：逻辑操作符Operator

Hive最终生成的MapReduce任务，Map阶段和Reduce阶段均由OperatorTree组成。逻辑操作符，就是在Map阶段或者Reduce阶段完成单一特定的操作。

基本的操作符包括TableScanOperator，SelectOperator，FilterOperator，JoinOperator，GroupByOperator，ReduceSinkOperator

由于Join/GroupBy/OrderBy均需要在Reduce阶段完成，所以在生成相应操作的Operator之前都会先生成一个ReduceSinkOperator，将字段组合并序列化为Reduce Key/value, Partition Key

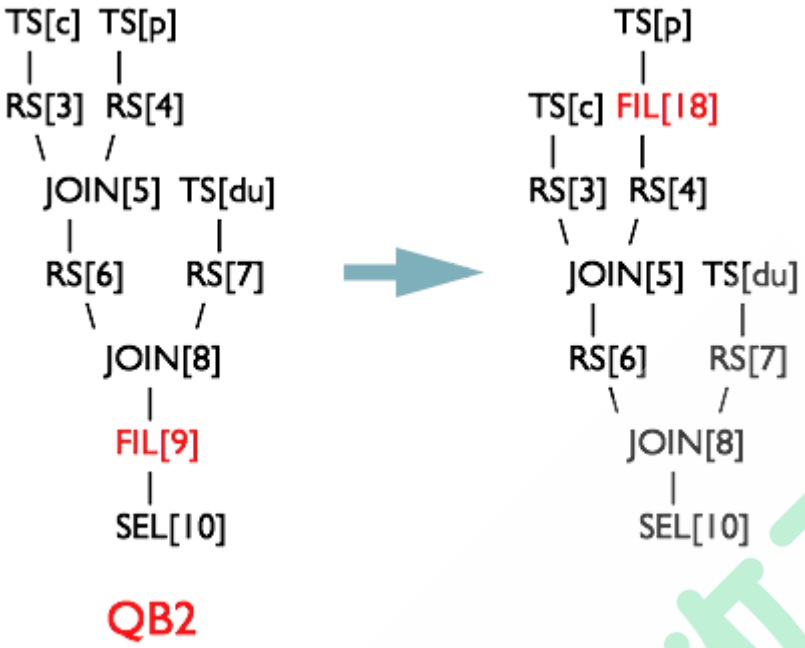
8.1.5. 第四阶段：逻辑层优化器

大部分逻辑层优化器通过变换OperatorTree，合并操作符，达到减少MapReduce Job，减少shuffle数据量的目的。

名称	作用
② SimpleFetchOptimizer	优化没有GroupBy表达式的聚合查询
② MapJoinProcessor	MapJoin，需要SQL中提供hint，0.11版本已不用
② BucketMapJoinOptimizer	BucketMapJoin
② GroupByOptimizer	Map端聚合
① ReduceSinkDeDuplication	合并线性的OperatorTree中partition/sort key相同的reduce
① PredicatePushDown	谓词前置/谓词下推
① CorrelationOptimizer	利用查询中的相关性，合并有相关性的Job，HIVE-2206
② ColumnPruner	字段剪枝

表格中①的优化器均是一个Job干尽可能多的事情/合并。②的都是减少shuffle数据量，甚至不做Reduce。

PredicatePushDown优化器：



8.1.6. 第五阶段：OperatorTree生成MapReduce Job的过程

OperatorTree 转化为 MapReduce Job 的过程分为下面几个阶段：

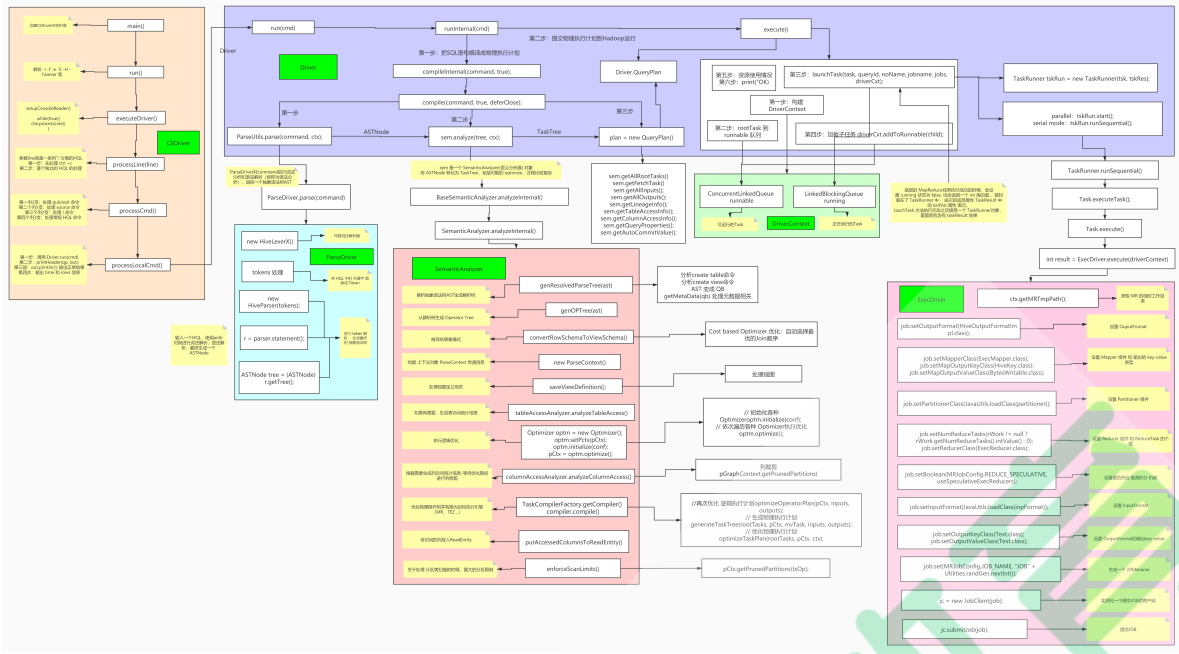
```
1、对输出表生成MoveTask
2、从OperatorTree的其中一个根节点向下深度优先遍历
3、ReduceSinkOperator标示Map/Reduce的界限，多个Job间的界限
4、遍历其他根节点，遇过碰到JoinOperator合并MapReduceTask
5、生成StatTask更新元数据
6、剪断Map与Reduce间的Operator的关系
```

8.1.7. 第六阶段：物理层优化器

这里不详细介绍每个优化器的原理，单独介绍一下MapJoin的优化器

名称	作用
Vectorizer	HIVE-4160，将在0.13中发布
SortMergeJoinResolver	与bucket配合，类似于归并排序
SamplingOptimizer	并行order by优化器，在0.12中发布
CommonJoinResolver + MapJoinResolver	MapJoin优化器

9. Hive的HQL编译源码解读



10. Hive SQL编译过程的设计

从上述整个SQL编译的过程，可以看出编译过程的设计有几个优点值得学习和借鉴

1. 使用Antlr开源软件定义语法规则，大大简化了词法和语法的编译解析过程，仅仅需要维护一份语法文件即可。

2. 整体思路很清晰，分阶段的设计使整个编译过程代码容易维护，使得后续各种优化器方便的以可插拔的方式开关，譬如Hive 0.13最新的特性Vectorization和对Tez引擎的支持都是可插拔的。

3. 每个Operator只完成单一的功能，简化了整个MapReduce程序。