开放数据处理服务ODPS

MapReduce

阿里云
aliyun.com

MapReduce

概要

MapReduce

ODPS提供了MapReduce编程接口。用户可以使用MapReduce提供的接口(Java API)编写MapReduce程序处理ODPS的中的数据。 本章节只对MapReduce SDK的使用方法作简单的介绍，关于MapReduce SDK的详细说明请参考官方提供的Java Doc。

应用场景

MapReduce最早是由Google提出的分布式数据处理模型，随后受到了业内的广泛关注，并被大量应用到各种商业场景中。比如：

- 搜索：网页爬取、倒排索引、PageRank。
- Web访问日志分析：分析和挖掘用户在web上的访问、购物行为特征，实现个性化推荐；分析用户访问行为。
- 文本统计分析：比如莫言小说的WordCount、词频TFIDF分析；学术论文、专利文献的引用分析和统计；维基百科数据分析等。
- 海量数据挖掘：非结构化数据、时空数据、图像数据的挖掘。
- 机器学习：监督学习、无监督学习、分类算法如决策树、SVM等。
- 自然语言处理：基于大数据的训练和预测；基于语料库构建单词同现矩阵，频繁项集数据挖掘、重复文档检测等。
- 广告推荐：用户点击（CTR)和购买行为（CVR）预测。

处理流程

MapReduce处理数据过程主要分成2个阶段：Map阶段和Reduce阶段。首先执行Map阶段，再执行Reduce阶段。Map和Reduce的处理逻辑由用户自定义实现，但要符合MapReduce框架的约定。

- 在正式执行Map前，需要将输入数据进行”分片”。所谓分片，就是将输入数据切分为大小相等的数据块，每一块作为单个Map Worker的输入被处理，以便于多个Map Worker同时工作。
- 分片完毕后，多个Map Worker就可以同时工作了。每个Map Worker在读入各自的数据后，进行计算处理，最终输出给Reduce。Map Worker在输出数据时，需要为每一条输出数据指定一个Key。这个Key值决定了这条数据将会被发送给哪一个Reduce Worker。Key值和Reduce Worker是多对一的关系，具有相同Key的数据会被发送给同一个Reduce Worker，单个Reduce Worker有可能会接收到多个Key值的数据。
- 在进入Reduce阶段之前，MapReduce框架会对数据按照Key值排序，使得具有相同Key的数据彼此相邻。如果用户指定了”合并操作”(Combiner)，框架会调用Combiner，将具有相同Key的数据进行聚合。Combiner的逻辑可以由用户自定义实现。与经典的MapReduce框架协议不同，在ODPS中，Combiner的输入、输出的参数必须与Reduce保持一致。这部分的处理通常也叫做”洗牌

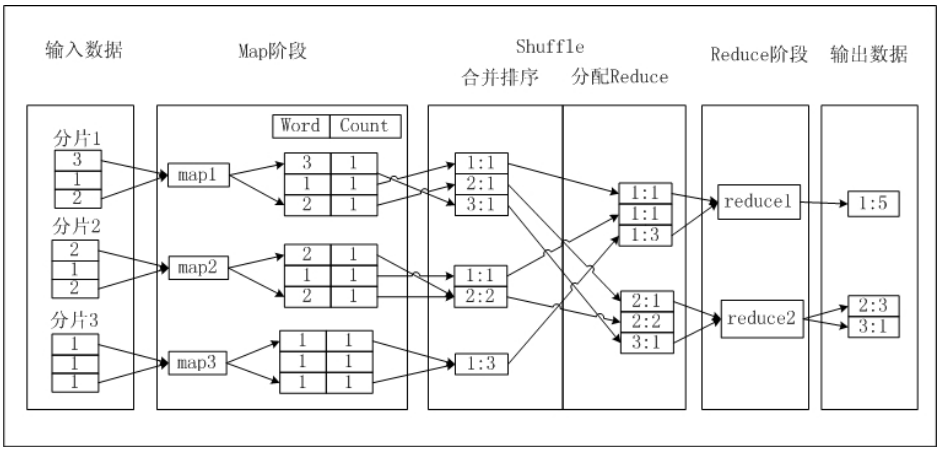
”(Shuffle)。

- 接下来进入Reduce阶段。相同的Key的数据会到达同一个Reduce Worker。同一个Reduce Worker会接收来自多个Map Worker的数据。每个Reduce Worker会对Key相同的多个数据进行Reduce操作。最后，一个Key的多条数据经过Reduce的作用后，将变成了一个值。

备注:

- 上文仅是对MapReduce框架做简单介绍，更多相关信息请查阅其他资料。

下面将以WordCount为例，解释ODPS MapReduce各个阶段的概念。 假设存在一个文本a.txt，文本内每行是一个数字，我们要统计每个数字出现的次数。文本内的数字称为Word，数字出现的次数称为Count。如果ODPS Mapreduce完成这一功能，需要经历下图描述的几个步骤：



首先对文本进行分片，将每片内的数据作为单个Map Worker的输入；

- Map处理输入，每获取一个数字，将数字的Count设置为1，并将此<Word, Count>对输出，此时以Word作为输出数据的Key；
- 在Shuffle阶段前期，首先对每个Map Worker的输出，按照Key值，即Word值排序。排序后进行Combine操作，即将Key值(Word值)相同的Count累加， 构成一个新的<Word, Count>对。此过程被称为合并排序；
- 在Shuffle阶段后期，数据被发送到Reduce端。Reduce Worker收到数据后依赖Key值再次对数据排序；
- 每个Reduce Worker对数据进行处理时，采用与Combiner相同的逻辑，将Key值(Word值)相同的Count累加，得到输出结果；

备注：

- 由于ODPS的所有数据都被存放在表中，因此ODPS MapReduce的输入、输出只能是表，不允许用户自定义输出格式，不提供类似文件系统的接口。

扩展MapReduce

传统的MapReduce模型要求每一轮MapReduce操作之后，数据必须落地到分布式文件系统上（比如HDFS或ODPS表）。 而一般的MapReduce应用通常由多个MapReduce作业组成，每个作业结束之后需要写入磁盘，接下去的Map任务很多情况下只是读一遍数据， 为后续的Shuffle阶段做准备，这样其实造成了冗余的IO操作。

ODPS的计算调度逻辑可以支持更复杂编程模型， 针对上面的那种情况，可以在Reduce后面直接执行下一次的Reduce操作，而不需要中间插入一个Map操作。 基于此，ODPS提供了扩展的MapReduce模型，即可以支持Map后连接任意多个Reduce操作，比如Map-Reduce-Reduce。

Hadoop Chain Mappper/Reducer也支持类似的串行化Map或Reduce操作，但和ODPS的扩展MapReduce(MR^2)模型有本质的区别， 因为Chain Mapper/Reducer还是基于传统的MapReduce模型，只是可以在原有的Mapper或Reducer后面在增加一个或多个Mapper操作(不允许增加Reducer)。 这带来的好处是用户可以复用之前的Mapper业务逻辑，可以把一个Map或Reduce拆成多个Mapper阶段，但本质上并没有改变底层的调度和IO模型。

功能介绍

运行命令

ODPS客户端提供一个jar命令用于运行MapReduce作业，具体语法：

```
Usage: jar [<GENERIC_OPTIONS>] <MAIN_CLASS> [ARGS]
    -conf <configuration_file>        Specify an application configuration file
    -classpath <local_file_list>      classpaths used to run mainClass
    -D <name>=<value>                 Property value pair, which will be used to run mainClass
    -l                                Run job in local mode
    -resources <resource_name_list>   file/table resources used in mapper or reducer, seperate by comma
For example:
    jar -conf /home/admin/myconf -resources a.txt,example.jar -classpath ../lib/example.jar:./other_lib.jar -
Djava.library.path=./native -Xmx512M mycompany.WordCount -m 10 -r 10 in out;
```

其中<GENERIC_OPTIONS>包括(均为可选参数)：

- -conf <configuration file> ：指定JobConf配置文件；
- -classpath <local_file_list>: 本地执行时的classpath，主要用于指定main函数所在的jar包的本地路径（包含相对路径和绝对路径）。大多数情况下， 用户更习惯于将main函数与Map/Reduce函数编写在一个包中，例如：WordCount代码示例 ，因此，在执行示例程序时， -resources及-classpath的参数中都出现了mapreduce-examples.jar，但二者意义不同，-resources引用的是Map/Reduce函数，运行于分布式环境中， 而-classpath引用的是main函数，运行于本地，指定的jar包路径也是本地文件路径。包名之间使用系统默认的文件分割符作分割(通常情况下， windows系统是分号";",linux系统是冒号":",如果用户是在云端使用mr任务，jar包之间的分隔是逗号",",。)；
- -D <prop_name>=<prop_value>：本地执行时，<mainClass>的java属性，可以定义多个；
- -l：以本地模式执行MapReduce作业，主要用于程序调试；
- -resources <resource_name_list>: MapReduce作业运行时使用的资源声明。一般情况下，resource_name_list中需要指定Map/Reduce函数所在的资源名称。 请特别注意，如果用户在Map/Reduce函数中读取了其他ODPS资源，那么，这些资源名称也需要被添加到

resource_name_list中。资源之间使用逗号分隔，使用跨项目空间使用资源时， 需要前面加上：PROJECT/resources/，示例：-resources otherproject/resources/resfile。有关于如何在Map/Reduce函数中读取资源的示例请查看 资源使用示例 ； 用户可以通过-conf选项指定JobConf配置文件。该文件可以影响SDK中JobConf的设置。关于JobConf的介绍请参考MapReduce核心接口的介绍。下面将给出一个JobConf配置文件的示例：

```
<configuration>
  <property>
    <name>import.filename</name>
    <value>resource.txt</value>
  </property>
</configuration>
```

在上述示例中，通过JobConf配置文件定义一个名为import.filename的变量，该变量的值为resource.txt。 用户可以在MapReduce程序中通过JobConf接口获取该变量的值。用户通过SDK中JobConf接口可以达到相同的目的。具体使用方式可以参考 资源使用示例 。

示例：

```
jar -resources mapreduce-examples.jar -classpath mapreduce-examples.jar
   org.alidata.odps.mr.examples.WordCount wc_in wc_out

add file data/src.txt
jar -resources src.txt,mapreduce-examples.jar -classpath mapreduce-examples.jar
   org.alidata.odps.mr.examples.WordCount wc_in wc_out

add file data/a.txt
add table wc_in as test_table
add jar work.jar
jar -conf odps-mapred.xml -resources a.txt,test_table,work.jar
   -classpath work.jar:otherlib.jar
   -D import.filename=resource.txt org.alidata.odps.mr.examples.WordCount args ...
```

基本概念

Map/Reduce

Map及Reduce分别支持对应的map/reduce方法，setup及cleanup方法。setup方法在map/reduce方法之前调用，每个Worker调用且仅调用一次。 cleanup方法在map/reduce方法之后调用，每个Worker调用且仅调用一次。

备注：

- 详细使用示例请参考 示例程序 。

排序

支持将map输出的key record中的某几列作为排序(Sort)列，不支持用户自定义的比较器(comparator)。用户

可以在排序列中选择某几列作为Group列， 不支持用户自定义的Group比较器。Sort列一般用来对用户数据进行排序，而Group列一般用来进行二次排序。

备注:

- 详细使用示例请参考 二次排序源代码 。

哈希

支持设置哈希(partition)列及用户自定义哈希函数(partitioner)。哈希列的使用优先级高于自定义哈希函数。哈希函数用于将map端的输出数据按照哈希逻辑分配到不同的Reduce Worker上。

归并

归并(Combiner)函数将Shuffle阶段相邻的Record进行归并。用户可以根据不同的业务逻辑选择是否使用归并函数。 归并函数是MapReduce计算框架的一种优化，通常情况下Combiner的逻辑与reduce相同。当map输出数据后， 框架会在map端对相同key值的数据进行本地的归并操作。

备注:

- 详细使用示例请参考 WordCount代码示例 。

输入与输出

- ODPS MapReduce的输入、输出支持ODPS内置类型的Bigint，Double，String，Datetime及Boolean类型，不支持用户自定义类型。
- 接受多表输入，且输入表的Schema可以不相同。在map函数中用户可以获取当前Record对应的Table信息；
- 输入可以为空，不支持视图(View)作为输入；
- Reduce接受多路输出，可以输出到不同表，或者同一张表的不同分区。不同输出的Schema可以不同。不同输出间通过label进行区分， 默认输出不必加lablel。但目前不接受没有输出的情况；

备注:

- 详细输入输出使用示例请参考 示例程序 。

资源读取

允许用户在map/reduce中读取ODPS资源。map/reduce的任意Worker都会将资源加载到内存中供用户代码使用。

备注：

- 详细使用示例请参考 资源使用示例 。

本地运行

用户通过在jar命令中设置–local参数，在本地模拟MapReduce的运行过程，从而进行本地调试。本地运行时，客户端会从ODPS中下载本地调试所需要的输入表的元信息、数据，所需要的资源以及输出表的元信息，并将这些信息保存到一个名为warehouse的本地目录中。 在程序运行结束后，会将计算结果输出到warehouse目录内的一个文件中。如果本地的warehouse目录下已经下载了输入表及被引用的资源，在下一次运行时， 会直接引用warehouse下的数据及文件，而不会重复下载。

在本地运行过程中，仍然会启动多个Map及Reduce进程处理数据，但这些进程不是并发运行，而是依次串行运行。此外这个模拟运行过程与真正的分布式运行有如下差别：

- 输入表行数限制：目前，最多只会下载100行数据；
- 资源的使用：在分布式环境中，ODPS会限制引用资源的大小，详情请参考 应用限制 。但在本地运行环境中，不会有资源大小的限制；
- 安全限制：ODPS MapReduce及UDF程序在分布式环境中运行时受到 Java沙箱 的限制。但在本地运行时，则没有此限制；

下面将给出一个简单的本地本地运行示例：

```
odps:my_project> jar -l com.aliyun.odps.mapred.example.WordCount wc_in wc_out
Summary:
counters: 10
   map-reduce framework
         combine_input_groups=2
         combine_output_records=2
         map_input_bytes=4
         map_input_records=1
         map_output_records=2
         map_output_[wc_out]_bytes=0
         map_output_[wc_out]_records=0
         reduce_input_groups=2
         reduce_output_[wc_out]_bytes=8
         reduce_output_[wc_out]_records=2

OK
```

备注:

- 关于WordCount示例的介绍请参考 WordCount代码示例 。

如果用户是第一次运行本地调试命令，命令成功结束后，会在当前路径下看到一个名为warehouse的路径。warehouse的目录结构如下所示：

```
<warehouse>
  |___my_project(项目空间目录)
        |___ <__tables__>
        |      |__wc_in(表数据目录)
        |      |      |___ data(文件)
```

```
|     |     |
|     |     |___ <__schema__> (文件)
|     |__wc_out ( 表数据目录 )
|           |___ data(文件)
|           |
|           |___ <__schema__> (文件)
|
|___ <__resources__>
      |
      |___table_resource_name (表资源)
      |       |___<__ref__>
      |
      |___ file_resource_name ( 文件资源 )
```

my*project的同级目录表示项目空间。wcin及wc_out表示数据表，用户在jar命令中读写的表文件数据会被下载到这级目录下。 __schema*文件中的内容表示表的元信息，其文件格式定义为：

```
project=local_project_name
table=local_table_name
columns=col1_name:col1_type,col2_name:col2_type
partitions=p1:STRING,p2:BIGINT    -- 本示例中不需要此字
```

其中，列名与列类型使用冒号”:”分隔，列与列之间使用逗号”,”分隔。**schema**文件的最前面需要声明Project名字及Table名字， 即project*name.tablename，使用逗号与列的定义做分隔。data文件表示表的数据。列的数量及数据必须与__schema*文件的定义相符， 不能多列或者少列，列之间使用逗号分隔。

wc*in的_schema*文件内容：

```
my_project.wc_in,key:STRING,value:STRING
```

data文件内容示例：

```
0,2
```

客户端会从ODPS中下载表的元信息及部分数据内容并保存到上述两个文件中。如果再次运行这个示例，将直接使用wc_in目录下的数据， 不会再次下载。需要特殊声明的是，从ODPS中下载数据的功能只在MapReduce的本地运行模式下才支持，在 Eclipse开发插件 中进行本地调试时， 不支持将ODPS的数据下载到本地。

wc*out的_schema*文件内容：

```
my_project.wc_out,key:STRING,cnt:BIGINT
```

data文件内容：

```
0,1
2,1
```

客户端会从ODPS现在wc*out表的元信息，并保存到_schema*文件中。而data文件的内容是在本地运行后，生成的结果数据文件。

Note:

- 用户也可以自行编辑**schema**及data文件，而后将这两个文件放置在对应的表目录下。在本地运行时，客户端检测到表目录已经存在， 则不会从ODPS中下载这个表的信息。本地的表目录可以是ODPS中不存在的表。

SDK介绍

在本小节，我们仅会对较为常用的MapReduce核心接口做简短介绍。使用Maven的用户可以从Maven库中搜索"odps-sdk-mapred"获取不同版本的Java SDK，相关配置信息：

```
<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>odps-sdk-mapred</artifactId>
  <version>0.19.3-public</version>
</dependency>
```

主要接口	描述
MapperBase	用户自定义的Map函数需要继承自此类。处理输入表的记录对 象，加工处理成键值对集合输出到Reduce阶段，或者不经过 Reduce阶段直接输出结果记录到结果表。不经过Reduce阶段而 直接输出计算结果的作业，也可称之为Map-Only作业。
ReducerBase	用户自定义的Reduce函数需要继承自此类。对与一个键(Key) 关联的一组数值集(Values)进行归约计算。
TaskContext	是MapperBase及ReducerBase多个成员函数的输入参数之一。 含有任务运行的上下文信息。
JobClient	用于提交和管理作业，提交方式包括阻塞(同步)方式及非阻塞 (异步)方式。
RunningJob	作业运行时对象，用于跟踪运行中的MapReduce作业实例。
JobConf	描述一个MapReduce任务的配置，通常在主程序(main函数)中 定义JobConf对象，然后通过JobClient提交作业给ODPS服务。

MapperBase

主要函数接口：

主要接口	描述
void cleanup(TaskContext context)	在Map阶段结束时，map方法之后调用。

void map(long key, Record record, TaskContext context)	map方法，处理输入表的记录。
void setup(TaskContext context)	在Map阶段开始时，map方法之前调用。

ReducerBase

主要函数接口：

主要接口	描述
void cleanup(TaskContext context)	在Reduce阶段结束时，reduce方法之后调用。
void reduce(Record key, Iterator<Record > values, TaskContext context)	reduce方法，处理输入表的记录。
void setup(TaskContext context)	在Reduce阶段开始时，reduce方法之前调用。

TaskContext

主要函数接口：

主要接口	描述
TableInfo[] getOutputTableInfo()	获取输出的表信息
Record createOutputRecord()	创建默认输出表的记录对象
Record createOutputRecord(String label)	创建给定label输出表的记录对象
Record createMapOutputKeyRecord()	创建Map输出Key的记录对象
Record createMapOutputValueRecord()	创建Map输出Value的记录对象
void write(Record record)	写记录到默认输出，用于Reduce端写出数据， 可以在Reduce端多次调用。
void write(Record record, String label)	写记录到给定标签输出，用于Reduce端写出数据。可以在 Reduce端多次调用。
void write(Record key, Record value)	Map写记录到中间结果，可以在Map函数中多次调用。 可以在Map端多次调用。
BufferedInputStream readResourceFileAsStream(String resourceName)	读取文件类型资源
Iterator<Record > readResourceTable(String resourceName)	读取表类型资源
Counter getCounter(Enum<? > name)	获取给定名称的Counter对象
Counter getCounter(String group, String name)	获取给定组名和名称的Counter对象
void progress()	向MapReduce框架报告心跳信息。 如果用户方法处理时间 很长，且中间没有调用框架，可以调用这个方法避免task 超时，框架默认600秒超时。

备注：ODPS的TaskContext接口中提供了progress功能，但此功能是防止Worker长时间运行未结束，被框架误认为超时而被杀的情况出现。 这个接口更类似于向框架发送心跳信息，并不是用来汇报Worker进度。ODPS MapReduce默认Worker超时时间为10分钟(系统默认配置，不受用户控制)， 如>果超过10分钟，Worker仍然没有向框架发送心跳(调用progress接口)，框架会强制停止该Worker，MapReduce任务失败退出。因此， 建议用户在Mapper/Reducer函数中，定期调用progress接口，防止框架认为Worker超时，误杀任务。

JobConf

主要函数接口：

主要接口	描述
void setResources(String resourceNames)	声明本作业使用的资源。只有声明的资源才能在运行 Mapper/Reducer时通过TaskContext对象读取。
void setMapOutputKeySchema(Column[] schema)	设置Mapper输出到Reducer的Key属性
void setMapOutputValueSchema(Column[] schema)	设置Mapper输出到Reducer的Value属性
void setOutputKeySortColumns(String[] cols)	设置Mapper输出到Reducer的Key排序列
void setOutputGroupingColumns(String[] cols)	设置Key分组列
void setMapperClass(Class<? extends Mapper > theClass)	设置作业的Mapper函数
void setPartitionColumns(String[] cols)	设置作业指定的分区列. 默认是Mapper输出Key的所有列
void setReducerClass(Class<? extends Reducer > theClass)	设置作业的Reducer
void setCombinerClass(Class<? extends Reducer > theClass)	设置作业的combiner。在Map端运行，作用类似于单个Map 对本地的相同Key值做Reduce
void setSplitSize(long size)	设置输入分片大小，单位 MB，默认640
void setNumReduceTasks(int n)	设置Reducer任务数，默认为Mapper任务数的1/4
void setMemoryForMapTask(int mem)	设置Mapper任务中单个Worker的内存大小，单位：MB， 默认值2048。
void setMemoryForReduceTask(int mem)	设置Reducer任务中单个Worker的内存大小，单位：MB， 默认值 2048。
void setOutputSchema(Column[] schema, String label)	设置指定label的输出属性。多路输出时，每一路输出对 应一个label。

备注:

- 通常情况下，GroupingColumns包含在KeySortColumns中，KeySortColumns要包含在Key中。

JobClient

主要函数接口：

主要接口	描述
static RunningJob runJob(JobConf job)	阻塞(同步)方式提交MapReduce作业后立即返回
static RunningJob submitJob(JobConf job)	非阻塞(异步)方式提交MapReduce作业后立即返回

RunningJob

主要函数接口：

主要接口	描述
String getInstanceID()	获取作业运行实例ID，用于查看运行日志和作业管理。
boolean isComplete()	查询作业是否结束。
boolean isSuccessful()	查询作业实例是否运行成功。
void waitForCompletion()	等待直至作业实例结束。一般使用于异步方式提交的作业。
JobStatus getJobStatus()	查询作业实例运行状态。
void killJob()	结束此作业。
Counters getCounters()	获取Conter信息。

InputUtils

主要函数接口：

主要接口	描述
static void addTable(TableInfo table, JobConf conf)	添加表table到任务输入，可以被调用多次 ，新加入的表以append方式添加到输入队列中。
static void setTables(TableInfo [] tables, JobConf conf)	添加多张表到任务输入中。

OutputUtils

主要函数接口：

主要接口	描述

static void addTable(TableInfo table, JobConf conf)	添加表table到任务输出，可以被调用多次 ，新加入的表以append方式添加到输出队 列中。
static void setTables(TableInfo [] tables, JobConf conf)	添加多张表到任务输出中。

Pipeline

Pipeline是MR^{2}的主体类。可以通过Pipeline.builder构建一个Pipeline。Pipeline的主要接口如下：

```
public Builder addMapper(Class<? extends Mapper> mapper)
public Builder addMapper(Class<? extends Mapper> mapper,
    Column[] keySchema, Column[] valueSchema, String[] sortCols,
    SortOrder[] order, String[] partCols,
    Class<? extends Partitioner> theClass, String[] groupCols)
public Builder addReducer(Class<? extends Reducer> reducer)
public Builder addReducer(Class<? extends Reducer> reducer,
    Column[] keySchema, Column[] valueSchema, String[] sortCols,
    SortOrder[] order, String[] partCols,
    Class<? extends Partitioner> theClass, String[] groupCols)
public Builder setOutputKeySchema(Column[] keySchema)
public Builder setOutputValueSchema(Column[] valueSchema)
public Builder setOutputKeySortColumns(String[] sortCols)
public Builder setOutputKeySortOrder(SortOrder[] order)
public Builder setPartitionColumns(String[] partCols)
public Builder setPartitionerClass(Class<? extends Partitioner> theClass)
public Builder setOutputGroupingColumns(String[] cols)
```

使用示例：

```
Job job = new Job();

Pipeline pipeline = Pipeline.builder()
 .addMapper(TokenizerMapper.class)
```

```
.setOutputKeySchema(
    new Column[] { new Column("word", OdpsType.STRING) })
.setOutputValueSchema(
    new Column[] { new Column("count", OdpsType.BIGINT) })
.addReducer(SumReducer.class)
.setOutputKeySchema(
    new Column[] { new Column("count", OdpsType.BIGINT) })
.setOutputValueSchema(
    new Column[] { new Column("word", OdpsType.STRING),
    new Column("count", OdpsType.BIGINT) })
.addReducer(IdentityReducer.class).createPipeline();

job.setPipeline(pipeline);
job.addInput(...)
job.addOutput(...)
job.submit();
```

如上所示，用户可以在main函数中构建一个Map后连续接两个Reduce的MapReduce任务。如果用户比较熟悉MapReduce的基础功能，可以轻松的使用MR^2 。

我们也建议用户在使用MR^{2}功能之前，先了解MapReduce的基础用法，即通过 JobConf 完成MapReduce任务的配置。

当然，JobConf 仅能够配置Map后接单Reduce的MapReduce任务。

数据类型

MapReduce支持的数据类型有：bigint, string, double, boolean以及datetime类型。ODPS数据类型与Java类型的对应关系如下：

ODPS SQL Type	Bigint	String	Double	Boolean	Datetime
Java Type	Long	String	Double	Boolean	Date

应用限制

目前，ODPS的使用限制包括：

- ODPS表string列内容长度不允许超过2MB。
- 单个任务引用的资源数量不超过512个，分区表按照一个单位计算。
- 单个任务引用的资源总计字节数大小不超过64MB。
- 单个任务的输入路数不能超过128，单个任务的输出路数不能超过128路。
- 单个任务中自定义Counter的数量不能超过64。
- 单个Map或Reduce Worker占用memory默认为2048MB，范围[256MB, 12GB]。
- 单个Map或Reduce Worker重复读一个资源次数限制 <=64次。
- 本地运行模式下，Map Worker个数不能超过100；Reduce Worker个数不能超过100；默认一路输入下载记录数100。

示例程序

WordCount示例

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

import java.io.IOException;
import java.util.Iterator;

import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;

public class WordCount {

  public static class TokenizerMapper extends MapperBase {
    private Record word;
    private Record one;

    @Override
    public void setup(TaskContext context) throws IOException {
      word = context.createMapOutputKeyRecord();
      one = context.createMapOutputValueRecord();
      one.set(new Object[] { 1L });
      System.out.println("TaskID:" + context.getTaskID().toString());
    }

    @Override
    public void map(long recordNum, Record record, TaskContext context)
        throws IOException {
```

```
    for (int i = 0; i < record.getColumnCount(); i++) {
      word.set(new Object[] { record.get(i).toString() });
      context.write(word, one);
    }
  }
}

/**
 * A combiner class that combines map output by sum them.
 **/
public static class SumCombiner extends ReducerBase {
  private Record count;

  @Override
  public void setup(TaskContext context) throws IOException {
    count = context.createMapOutputValueRecord();
  }

  @Override
  public void reduce(Record key, Iterator<Record> values, TaskContext context)
      throws IOException {
    long c = 0;
    while (values.hasNext()) {
      Record val = values.next();
      c += (Long) val.get(0);
    }
    count.set(0, c);
    context.write(key, count);
  }
}

/**
 * A reducer class that just emits the sum of the input values.
 **/
public static class SumReducer extends ReducerBase {
  private Record result = null;

  @Override
  public void setup(TaskContext context) throws IOException {
    result = context.createOutputRecord();
  }

  @Override
  public void reduce(Record key, Iterator<Record> values, TaskContext context)
      throws IOException {
    long count = 0;
    while (values.hasNext()) {
      Record val = values.next();
      count += (Long) val.get(0);
    }
    result.set(0, key.get(0));
    result.set(1, count);
    context.write(result);
  }
}
```

```
  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: WordCount <in_table> <out_table>");
      System.exit(2);
    }

    JobConf job = new JobConf();

    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(SumCombiner.class);
    job.setReducerClass(SumReducer.class);

    job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
    job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));

    InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
    OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);

    JobClient.runJob(job);
  }

}
```

MapOnly示例

对于MapOnly的作业，Map直接将 < Key, Value > 信息输出到ODPS的表中。用户只需要指定输出表即可，不再需要指定Map输出的Key/Value元信息。

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

import java.io.IOException;

import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.SchemaUtils;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.data.TableInfo;

public class MapOnly {

  public static class MapperClass extends MapperBase {
    @Override
    public void setup(TaskContext context) throws IOException {
      boolean is = context.getJobConf().getBoolean("option.mapper.setup", false);

      if (is) {
        Record result = context.createOutputRecord();
        result.set(0, "setup");
        result.set(1, 1L);
        context.write(result);
```

```
    }
  }

  @Override
  public void map(long key, Record record, TaskContext context) throws IOException {
    boolean is = context.getJobConf().getBoolean("option.mapper.map", false);

    if (is) {
      Record result = context.createOutputRecord();
      result.set(0, record.get(0));
      result.set(1, 1L);
      context.write(result);
    }
  }

  @Override
  public void cleanup(TaskContext context) throws IOException {
    boolean is = context.getJobConf().getBoolean("option.mapper.cleanup", false);

    if (is) {
      Record result = context.createOutputRecord();
      result.set(0, "cleanup");
      result.set(1, 1L);
      context.write(result);
    }
  }
}

public static void main(String[] args) throws Exception {
  if (args.length != 2 && args.length != 3) {
    System.err.println("Usage: OnlyMapper <in_table> <out_table> [setup|map|cleanup]");
    System.exit(2);
  }

  JobConf job = new JobConf();
  job.setMapperClass(MapperClass.class);
  job.setNumReduceTasks(0);

  InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
  OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);

  if (args.length == 3) {
    String options = new String(args[2]);

    if (options.contains("setup")) {
      job.setBoolean("option.mapper.setup", true);
    }

    if (options.contains("map")) {
      job.setBoolean("option.mapper.map", true);
    }

    if (options.contains("cleanup")) {
      job.setBoolean("option.mapper.cleanup", true);
    }
  }
```

```
    JobClient.runJob(job);
  }
}
```

多路输入输出示例

目前ODPS支持多表的输入及输出。

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

import java.io.IOException;
import java.util.Iterator;
import java.util.LinkedHashMap;

import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;

/**
 * Multi input & output example.
 *
 * To run: jar -resources odps-mapred-example-0.12.0.jar com.aliyun.odps.mapred.open.example.MultipleInOut
 * mr_src,mr_src1,mr_srcpart|pt=1,mr_srcpart|pt=2/ds=2
 * mr_multiinout_out1,mr_multiinout_out2|a=1/b=1|out1,mr_multiinout_out2|a=2/b=2|out2;
 *
 **/
public class MultipleInOut {

  public static class TokenizerMapper extends MapperBase {
    Record word;
    Record one;

    @Override
    public void setup(TaskContext context) throws IOException {
      word = context.createMapOutputKeyRecord();
      one = context.createMapOutputValueRecord();
      one.set(new Object[] { 1L });
    }

    @Override
    public void map(long recordNum, Record record, TaskContext context)
        throws IOException {
      for (int i = 0; i < record.getColumnCount(); i++) {
        word.set(new Object[] { record.get(i).toString() });
        context.write(word, one);
```

```
    }
  }
}

public static class SumReducer extends ReducerBase {
  private Record result;
  private Record result1;
  private Record result2;

  @Override
  public void setup(TaskContext context) throws IOException {
    result = context.createOutputRecord();
    result1 = context.createOutputRecord("out1");
    result2 = context.createOutputRecord("out2");
  }

  @Override
  public void reduce(Record key, Iterator<Record> values, TaskContext context)
      throws IOException {
    long count = 0;
    while (values.hasNext()) {
      Record val = values.next();
      count += (Long) val.get(0);
    }

    long mod = count % 3;
    if (mod == 0) {
      result.set(0, key.get(0));
      result.set(1, count);
      //不指定label，输出的默认(default)输出
      context.write(result);
    } else if (mod == 1) {
      result1.set(0, key.get(0));
      result1.set(1, count);
      context.write(result1, "out1");
    } else {
      result2.set(0, key.get(0));
      result2.set(1, count);
      context.write(result2, "out2");
    }
  }

  @Override
  public void cleanup(TaskContext context) throws IOException {
    Record result = context.createOutputRecord();

    result.set(0, "default");
    result.set(1, 1L);
    context.write(result);

    Record result1 = context.createOutputRecord("out1");
    result1.set(0, "out1");
    result1.set(1, 1L);
    context.write(result1, "out1");

    Record result2 = context.createOutputRecord("out2");
```

```
    result2.set(0, "out1");
    result2.set(1, 1L);
    context.write(result2, "out2");
  }
}

public static LinkedHashMap<String, String> convertPartSpecToMap(
    String partSpec) {
  LinkedHashMap<String, String> map = new LinkedHashMap<String, String>();
  if (partSpec != null && !partSpec.trim().isEmpty()) {
    String[] parts = partSpec.split("/");
    for (String part : parts) {
      String[] ss = part.split("=");
      if (ss.length != 2) {
        throw new RuntimeException("ODPS-0730001: error part spec format: "
            + partSpec);
      }
      map.put(ss[0], ss[1]);
    }
  }
  return map;
}

public static void main(String[] args) throws Exception {

  String[] inputs = null;
  String[] outputs = null;
  if (args.length == 2) {
    inputs = args[0].split(",");
    outputs = args[1].split(",");
  } else {
    System.err.println("MultipleInOut in... out...");
    System.exit(1);
  }

  JobConf job = new JobConf();

  job.setMapperClass(TokenizerMapper.class);
  job.setReducerClass(SumReducer.class);

  job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
  job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));

  //解析用户的输入表字符串
  for (String in : inputs) {
    String[] ss = in.split("\\|");
    if (ss.length == 1) {
      InputUtils.addTable(TableInfo.builder().tableName(ss[0]).build(), job);
    } else if (ss.length == 2) {
      LinkedHashMap<String, String> map = convertPartSpecToMap(ss[1]);
      InputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map).build(), job);
    } else {
      System.err.println("Style of input: " + in + " is not right");
      System.exit(1);
    }
  }
```

```
    //解析用户的输出表字符串
    for (String out : outputs) {
      String[] ss = out.split("\\|");
      if (ss.length == 1) {
        OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).build(), job);
      } else if (ss.length == 2) {
        LinkedHashMap<String, String> map = convertPartSpecToMap(ss[1]);
        OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map).build(), job);
      } else if (ss.length == 3) {
        if (ss[1].isEmpty()) {
          LinkedHashMap<String, String> map = convertPartSpecToMap(ss[2]);
          OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map).build(), job);
        } else {
          LinkedHashMap<String, String> map = convertPartSpecToMap(ss[1]);
          OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map)
              .label(ss[2]).build(), job);
        }
      } else {
        System.err.println("Style of output: " + out + " is not right");
        System.exit(1);
      }
    }

    JobClient.runJob(job);
  }

}
```

多任务示例

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

import java.io.IOException;
import java.util.Iterator;


import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.RunningJob;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;

/**
 * MultiJobs
 *
 * Running multiple job
 *
```

```
 * To run: jar -resources multijobs_res_table,odps-mapred-example-0.12.0.jar
 * com.aliyun.odps.mapred.open.example.MultiJobs mr_multijobs_out;
 *
 **/
public class MultiJobs {

  public static class InitMapper extends MapperBase {

    @Override
    public void setup(TaskContext context) throws IOException {
      Record record = context.createOutputRecord();
      long v = context.getJobConf().getLong("multijobs.value", 2);
      record.set(0, v);
      context.write(record);
    }
  }

  public static class DecreaseMapper extends MapperBase {

    @Override
    public void cleanup(TaskContext context) throws IOException {
      //从JobConf中获取main函数中定义的变量值
      long expect = context.getJobConf().getLong("multijobs.expect.value", -1);
      long v = -1;
      int count = 0;

      Iterator<Record> iter = context.readResourceTable("multijobs_res_table");
      while (iter.hasNext()) {
        Record r = iter.next();
        v = (Long) r.get(0);
        if (expect != v) {
          throw new IOException("expect: " + expect + ", but: " + v);
        }
        count++;
      }

      if (count != 1) {
        throw new IOException("res_table should have 1 record, but: " + count);
      }

      Record record = context.createOutputRecord();
      v--;
      record.set(0, v);
      context.write(record);
      context.getCounter("multijobs", "value").setValue(v);
    }
  }

  public static void main(String[] args) throws Exception {
    if (args.length != 1) {
      System.err.println("Usage: TestMultiJobs <table>");
      System.exit(1);
    }
    String tbl = args[0];
    long iterCount = 2;
```

```
    System.err.println("Start to run init job.");

    JobConf initJob = new JobConf();

    initJob.setLong("multijobs.value", iterCount);
    initJob.setMapperClass(InitMapper.class);

    InputUtils.addTable(TableInfo.builder().tableName("mr_empty").build(), initJob);
    OutputUtils.addTable(TableInfo.builder().tableName(tbl).build(), initJob);

    initJob.setMapOutputKeySchema(SchemaUtils.fromString("key:string"));
    initJob.setMapOutputValueSchema(SchemaUtils.fromString("value:string"));

    initJob.setNumReduceTasks(0);

    JobClient.runJob(initJob);

    while (true) {
      System.err.println("Start to run iter job, count: " + iterCount);

      JobConf decJob = new JobConf();

      decJob.setLong("multijobs.expect.value", iterCount);
      decJob.setMapperClass(DecreaseMapper.class);

      InputUtils.addTable(TableInfo.builder().tableName("mr_empty").build(), decJob);
      OutputUtils.addTable(TableInfo.builder().tableName(tbl).build(), decJob);

      decJob.setNumReduceTasks(0);

      RunningJob rJob = JobClient.runJob(decJob);

      iterCount--;

      if (rJob.getCounters().findCounter("multijobs", "value").getValue() == 0) {
        break;
      }
    }

    if (iterCount != 0) {
      throw new IOException("Job failed.");
    }
  }
}
```

二次排序示例

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

import java.io.IOException;
import java.util.Iterator;

import com.aliyun.odps.data.Record;
```

```
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.SchemaUtils;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.data.TableInfo;

/**
 * This is an example ODPS Map/Reduce application. It reads the input table that
 * must contain two integers per record. The output is sorted by the first and
 * second number and grouped on the first number.
 *
 * To run: jar -resources odps-mapred-example-0.12.0.jar
 * com.aliyun.odps.mapred.open.example.SecondarySort mr_sort_in mr_secondarysort_out;
 *
 **/
public class SecondarySort {

  /**
   * Read two integers from each line and generate a key, value pair as ((left,
   * right), right).
   **/
  public static class MapClass extends MapperBase {
    private Record key;
    private Record value;

    @Override
    public void setup(TaskContext context) throws IOException {
      key = context.createMapOutputKeyRecord();
      value = context.createMapOutputValueRecord();
    }

    @Override
    public void map(long recordNum, Record record, TaskContext context)
        throws IOException {
      long left = 0;
      long right = 0;

      if (record.getColumnCount() > 0) {
        left = (Long) record.get(0);
        if (record.getColumnCount() > 1) {
          right = (Long) record.get(1);
        }

        key.set(new Object[] { (Long) left, (Long) right });
        value.set(new Object[] { (Long) right });

        context.write(key, value);
      }
    }
  }

  /**
```

```
   * A reducer class that just emits the sum of the input values.
   **/
  public static class ReduceClass extends ReducerBase {

    private Record result = null;

    @Override
    public void setup(TaskContext context) throws IOException {
      result = context.createOutputRecord();
    }

    @Override
    public void reduce(Record key, Iterator<Record> values, TaskContext context)
        throws IOException {
      result.set(0, key.get(0));
      while (values.hasNext()) {
        Record value = values.next();
        result.set(1, value.get(0));
        context.write(result);
      }
    }

  }

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: secondarysrot <in> <out>");
      System.exit(2);
    }

    JobConf job = new JobConf();
    job.setMapperClass(MapClass.class);
    job.setReducerClass(ReduceClass.class);

    //将多列设置为Key
    // compare first and second parts of the pair
    job.setOutputKeySortColumns(new String[] { "i1", "i2" });

    // partition based on the first part of the pair
    job.setPartitionColumns(new String[] { "i1" });

    // grouping comparator based on the first part of the pair
    job.setOutputGroupingColumns(new String[] { "i1" });

    // the map output is LongPair, Long
    job.setMapOutputKeySchema(SchemaUtils.fromString("i1:bigint,i2:bigint"));
    job.setMapOutputValueSchema(SchemaUtils.fromString("i2x:bigint"));

    InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
    OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);

    JobClient.runJob(job);
    System.exit(0);
  }

}
```

使用资源示例

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

import java.io.BufferedInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;


import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;

/**
 * Upload
 *
 * Import data from text file into table
 *
 * To run: jar -resources odps-mapred-example-0.12.0.jar,mr_join_src1.txt
 * com.aliyun.odps.mapred.open.example.Upload mr_join_src1.txt mr_join_src1;
 *
 **/
public class Upload {

  public static class UploadMapper extends MapperBase {
    @Override
    public void setup(TaskContext context) throws IOException {
      Record record = context.createOutputRecord();
      StringBuilder importdata = new StringBuilder();
      BufferedInputStream bufferedInput = null;

      try {
        byte[] buffer = new byte[1024];
        int bytesRead = 0;

        String filename = context.getJobConf().get("import.filename");
        bufferedInput = context.readResourceFileAsStream(filename);

        while ((bytesRead = bufferedInput.read(buffer)) != -1) {
          String chunk = new String(buffer, 0, bytesRead);
          importdata.append(chunk);
        }

        String lines[] = importdata.toString().split("\n");
        for (int i = 0; i < lines.length; i++) {
          String[] ss = lines[i].split(",");
          record.set(0, Long.parseLong(ss[0].trim()));
          record.set(1, ss[1].trim());
```

```
          context.write(record);
        }
      } catch (FileNotFoundException ex) {
        throw new IOException(ex);
      } catch (IOException ex) {
        throw new IOException(ex);
      } finally {
      }

    }

    @Override
    public void map(long recordNum, Record record, TaskContext context)
        throws IOException {

    }

  }

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: Upload <import_txt> <out_table>");
      System.exit(2);
    }

    JobConf job = new JobConf();

    job.setMapperClass(UploadMapper.class);

    job.set("import.filename", args[0]);

    job.setNumReduceTasks(0);

    job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint"));
    job.setMapOutputValueSchema(SchemaUtils.fromString("value:string"));

    InputUtils.addTable(TableInfo.builder().tableName("mr_empty").build(), job);
    OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);

    JobClient.runJob(job);
  }

}
```

实际上用户有多种手段设置JobConf：

- 通过SDK中JobConf的接口设置，本示例即是通过此方法。此方法的优先级最高；
- 在jar命令行中通过–conf参数指定新的JobConf文件。此种方式的优先级最低。–conf的使用方式请参考 运行命令 ；

使用counter示例

代码示例中定义了三个Counter：map_outputs，reduce_outputs和global_counts。用户可以在

Map/Reduce的setup， map/reduce及cleanup接口中获取任意自定义Counter，并进行操作。

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

import java.io.IOException;
import java.util.Iterator;

import com.aliyun.odps.counter.Counter;
import com.aliyun.odps.counter.Counters;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.RunningJob;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.SchemaUtils;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.data.TableInfo;

/**
 * User Defined Counters
 *
 * To run: jar -resources odps-mapred-example-0.12.0.jar
 * com.aliyun.odps.mapred.open.example.UserDefinedCounters mr_src mr_testcounters_out;
 *
 **/
public class UserDefinedCounters {

  enum MyCounter {
    TOTAL_TASKS, MAP_TASKS, REDUCE_TASKS
  }

  public static class TokenizerMapper extends MapperBase {
    private Record word;
    private Record one;

    @Override
    public void setup(TaskContext context) throws IOException {
      super.setup(context);
      Counter map_tasks = context.getCounter(MyCounter.MAP_TASKS);
      Counter total_tasks = context.getCounter(MyCounter.TOTAL_TASKS);
      map_tasks.increment(1);
      total_tasks.increment(1);

      word = context.createMapOutputKeyRecord();
      one = context.createMapOutputValueRecord();
      one.set(new Object[] { 1L });
    }

    @Override
    public void map(long recordNum, Record record, TaskContext context)
        throws IOException {
      for (int i = 0; i < record.getColumnCount(); i++) {
```

```
      word.set(new Object[] { record.get(i).toString() });
      context.write(word, one);
    }
  }
}

public static class SumReducer extends ReducerBase {
  private Record result = null;

  @Override
  public void setup(TaskContext context) throws IOException {
    result = context.createOutputRecord();
    Counter reduce_tasks = context.getCounter(MyCounter.REDUCE_TASKS);
    Counter total_tasks = context.getCounter(MyCounter.TOTAL_TASKS);
    reduce_tasks.increment(1);
    total_tasks.increment(1);
  }

  @Override
  public void reduce(Record key, Iterator<Record> values, TaskContext context)
      throws IOException {
    long count = 0;
    while (values.hasNext()) {
      Record val = values.next();
      count += (Long) val.get(0);
    }
    result.set(0, key.get(0));
    result.set(1, count);
    context.write(result);
  }
}

public static void main(String[] args) throws Exception {
  if (args.length != 2) {
    System.err
        .println("Usage: TestUserDefinedCounters <in_table> <out_table>");
    System.exit(2);
  }

  JobConf job = new JobConf();
  job.setMapperClass(TokenizerMapper.class);
  job.setReducerClass(SumReducer.class);

  job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
  job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));

  InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
  OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);

  RunningJob rJob = JobClient.runJob(job);

  Counters counters = rJob.getCounters();
  long m = counters.findCounter(MyCounter.MAP_TASKS).getValue();
  long r = counters.findCounter(MyCounter.REDUCE_TASKS).getValue();
  long total = counters.findCounter(MyCounter.TOTAL_TASKS).getValue();
```

```
    System.exit(0);
  }
}
```

grep示例

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

import java.io.IOException;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.Mapper;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.RunningJob;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;

/**
 * Extracts matching regexs from input files and counts them.
 *
 * To run: jar -resources odps-mapred-example-0.12.0.jar
 * com.aliyun.odps.mapred.open.example.Grep mr_src mr_grep_tmp mr_grep_out val;
 **/
public class Grep {

  /**
   * RegexMapper
   **/
  public class RegexMapper extends MapperBase {
    private Pattern pattern;
    private int group;

    private Record word;
    private Record one;

    @Override
    public void setup(TaskContext context) throws IOException {
      JobConf job = (JobConf) context.getJobConf();
      pattern = Pattern.compile(job.get("mapred.mapper.regex"));
      group = job.getInt("mapred.mapper.regex.group", 0);

      word = context.createMapOutputKeyRecord();
      one = context.createMapOutputValueRecord();
      one.set(new Object[] { 1L });
```

```
  }

  @Override
  public void map(long recordNum, Record record, TaskContext context) throws IOException {
    for (int i = 0; i < record.getColumnCount(); ++i) {
      String text = record.get(i).toString();
      Matcher matcher = pattern.matcher(text);
      while (matcher.find()) {
        word.set(new Object[] { matcher.group(group) });
        context.write(word, one);
      }
    }
  }
}

/**
 * LongSumReducer
 **/
public class LongSumReducer extends ReducerBase {
  private Record result = null;

  @Override
  public void setup(TaskContext context) throws IOException {
    result = context.createOutputRecord();
  }

  @Override
  public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException {
    long count = 0;
    while (values.hasNext()) {
      Record val = values.next();
      count += (Long) val.get(0);
    }
    result.set(0, key.get(0));
    result.set(1, count);
    context.write(result);
  }
}

/**
 * A {@link Mapper} that swaps keys and values.
 **/
public class InverseMapper extends MapperBase {
  private Record word;
  private Record count;

  @Override
  public void setup(TaskContext context) throws IOException {
    word = context.createMapOutputValueRecord();
    count = context.createMapOutputKeyRecord();
  }

  /**
   * The inverse function. Input keys and values are swapped.
   **/
  @Override
```

```
  public void map(long recordNum, Record record, TaskContext context) throws IOException {
    word.set(new Object[] { record.get(0).toString() });
    count.set(new Object[] { (Long) record.get(1) });
    context.write(count, word);
  }
}

/**
 * IdentityReducer
 **/
public class IdentityReducer extends ReducerBase {
  private Record result = null;

  @Override
  public void setup(TaskContext context) throws IOException {
    result = context.createOutputRecord();
  }

  /** Writes all keys and values directly to output. **/

  @Override
  public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException {
    result.set(0, key.get(0));

    while (values.hasNext()) {
      Record val = values.next();
      result.set(1, val.get(0));
      context.write(result);
    }
  }
}

public static void main(String[] args) throws Exception {
  if (args.length < 4) {
    System.err.println("Grep <inDir> <tmpDir> <outDir> <regex> [<group>]");
    System.exit(2);
  }

  JobConf grepJob = new JobConf();

  grepJob.setMapperClass(RegexMapper.class);
  grepJob.setReducerClass(LongSumReducer.class);

  grepJob.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
  grepJob.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));

  InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), grepJob);
  OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), grepJob);

  grepJob.set("mapred.mapper.regex", args[3]);
  if (args.length == 5) {
    grepJob.set("mapred.mapper.regex.group", args[4]);
  }

  @SuppressWarnings("unused")
  RunningJob rjGrep = JobClient.runJob(grepJob);
```

```
    JobConf sortJob = new JobConf();

    sortJob.setMapperClass(InverseMapper.class);
    sortJob.setReducerClass(IdentityReducer.class);

    sortJob.setMapOutputKeySchema(SchemaUtils.fromString("count:bigint"));
    sortJob.setMapOutputValueSchema(SchemaUtils.fromString("word:string"));

    InputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), sortJob);
    OutputUtils.addTable(TableInfo.builder().tableName(args[2]).build(), sortJob);

    sortJob.setNumReduceTasks(1); // write a single file
    sortJob.setOutputKeySortColumns(new String[] { "count" }); // sort by
                                                               // decreasing
                                                               // freq

    @SuppressWarnings("unused")
    RunningJob rjSort = JobClient.runJob(sortJob);
  }

}
```

join示例

ODPS MapReduce框架自身并不支持Join逻辑。但用户可以在自己的Map/Reduce函数中实现数据的Join，当然这需要用户做一些额外的工作。

假设需要Join两张表mr_join_src1(key bigint, value string)及mr_join_src2(key bigint, value string)， 输出表是mr_join_out(key bigint, value1 string, value2 string)，其中value1是mr_join_src1的value值，value2是mr_join_src2的value值。

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
```

```
/**
 * Join, mr_join_src1/mr_join_src2(key bigint, value string), mr_join_out(key
 * bigint, value1 string, value2 string)
 *
 * To run: jar -resources odps-mapred-example-0.12.0.jar
 * com.aliyun.odps.mapred.open.example.Join mr_join_src1 mr_join_src2 mr_join_out;
 */
public class Join {

  public static final Log LOG = LogFactory.getLog(Join.class);

  public static class JoinMapper extends MapperBase {

    private Record mapkey;
    private Record mapvalue;
    private long tag;

    @Override
    public void setup(TaskContext context) throws IOException {
      mapkey = context.createMapOutputKeyRecord();
      mapvalue = context.createMapOutputValueRecord();
      tag = context.getInputTableInfo().getLabel().equals("left") ? 0 : 1;
    }

    @Override
    public void map(long key, Record record, TaskContext context)
        throws IOException {
      mapkey.set(0, record.get(0));
      mapkey.set(1, tag);

      for (int i = 1; i < record.getColumnCount(); i++) {
        mapvalue.set(i - 1, record.get(i));
      }
      context.write(mapkey, mapvalue);
    }

  }

  public static class JoinReducer extends ReducerBase {

    private Record result = null;

    @Override
    public void setup(TaskContext context) throws IOException {
      result = context.createOutputRecord();
    }

    @Override
    public void reduce(Record key, Iterator<Record> values, TaskContext context)
        throws IOException {
      long k = key.getBigint(0);
      List<Object[]> leftValues = new ArrayList<Object[]>();

      while (values.hasNext()) {
        Record value = values.next();
```

```
        long tag = (Long) key.get(1);

        if (tag == 0) {
          leftValues.add(value.toArray().clone());
        } else {
          for (Object[] leftValue : leftValues) {
            int index = 0;
            result.set(index++, k);
            for (int i = 0; i < leftValue.length; i++) {
              result.set(index++, leftValue[i]);
            }
            for (int i = 0; i < value.getColumnCount(); i++) {
              result.set(index++, value.get(i));
            }
            context.write(result);
          }
        }
      }

    }

  }

  public static void main(String[] args) throws Exception {
    if (args.length != 3) {
      System.err.println("Usage: Join <input table1> <input table2> <out>");
      System.exit(2);
    }
    JobConf job = new JobConf();

    job.setMapperClass(JoinMapper.class);
    job.setReducerClass(JoinReducer.class);

    job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,tag:bigint"));
    job.setMapOutputValueSchema(SchemaUtils.fromString("value:string"));

    job.setPartitionColumns(new String[]{"key"});
    job.setOutputKeySortColumns(new String[]{"key", "tag"});
    job.setOutputGroupingColumns(new String[]{"key"});
    job.setNumReduceTasks(1);

    InputUtils.addTable(TableInfo.builder().tableName(args[0]).label("left").build(), job);
    InputUtils.addTable(TableInfo.builder().tableName(args[1]).label("right").build(), job);
    OutputUtils.addTable(TableInfo.builder().tableName(args[2]).build(), job);

    JobClient.runJob(job);
  }

}
```

sleep示例

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;
```

```
import java.io.IOException;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import com.aliyun.odps.OdpsException;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;

/**
 * Dummy class for testing MR framefork. Sleeps for a defined period of time in
 * mapper and reducer. Generates fake input for map / reduce jobs. Note that
 * generated number of input pairs is in the order of
 * <code>numMappers * mapSleepTime / 100</code>, so the job uses some disk
 * space.
 * To run: jar -resources odps-mapred-example-0.12.0.jar com.aliyun.odps.mapred.open.example.SleepJob
 * -m 1 -r 1 -mt 1 -rt 1;
 *
 **/
public class SleepJob {

  private static Log LOG = LogFactory.getLog(SleepJob.class);

  public static class SleepMapper extends MapperBase {

    private LinkedHashMap<Integer, Integer> inputs = new LinkedHashMap<Integer, Integer>();
    private long mapSleepDuration = 100;
    private int mapSleepCount = 1;
    private int count = 0;
    private Record key;

    @Override
    public void setup(TaskContext context) throws IOException {
      LOG.info("map setup called");
      JobConf conf = (JobConf) context.getJobConf();

      mapSleepCount = conf.getInt("sleep.job.map.sleep.count", 1);
      if (mapSleepCount < 0)
        throw new IOException("Invalid map count: " + mapSleepCount);

      mapSleepDuration = conf.getLong("sleep.job.map.sleep.time", 100)
          / mapSleepCount;

      LOG.info("mapSleepCount = " + mapSleepCount + ", mapSleepDuration = "
```

```
        + mapSleepDuration);

    final int redcount = conf.getInt("sleep.job.reduce.sleep.count", 1);
    if (redcount < 0)
      throw new IOException("Invalid reduce count: " + redcount);

    final int emitPerMapTask = (redcount * conf.getNumReduceTasks());

    int records = 0;
    int emitCount = 0;

    while (records++ < mapSleepCount) {
      int key = emitCount;
      int emit = emitPerMapTask / mapSleepCount;
      if ((emitPerMapTask) % mapSleepCount > records) {
        ++emit;
      }
      emitCount += emit;
      int value = emit;
      inputs.put(key, value);
    }

    key = context.createMapOutputKeyRecord();
  }

  @Override
  public void cleanup(TaskContext context) throws IOException {
    // it is expected that every map processes mapSleepCount number of
    // records.
    LOG.info("map run called");

    for (Map.Entry<Integer, Integer> entry : inputs.entrySet()) {
      LOG.info("Sleeping... (" + (mapSleepDuration * (mapSleepCount - count))
          + ") ms left");
      try {
        Thread.sleep(mapSleepDuration);
      } catch (InterruptedException e) {
        throw new IOException(e);
      }

      ++count;
      // output reduceSleepCount * numReduce number of random values, so that
      // each reducer will get reduceSleepCount number of keys.
      int k = entry.getKey();
      int v = entry.getValue();
      for (int i = 0; i < v; ++i) {
        key.set(new Object[] { (Long) ((long) (k + i)) });
        context.write(key, key);
      }
    }
  }
}

public static class SleepReducer extends ReducerBase {

  private long reduceSleepDuration = 100;
```

```
  private int reduceSleepCount = 1;
  private int count = 0;

  @Override
  public void setup(TaskContext context) throws IOException {
    LOG.info("reduce setup called");
    JobConf conf = (JobConf) context.getJobConf();
    reduceSleepCount = conf.getInt("sleep.job.reduce.sleep.count",
        reduceSleepCount);
    reduceSleepDuration = conf.getLong("sleep.job.reduce.sleep.time", 100)
        / reduceSleepCount;
    LOG.info("reduceSleepCount = " + reduceSleepCount
        + ", reduceSleepDuration = " + reduceSleepDuration);
  }

  @Override
  public void reduce(Record key, Iterator<Record> values, TaskContext context)
      throws IOException {
    LOG.info("reduce called");

    LOG.info("Sleeping... ("
        + (reduceSleepDuration * (reduceSleepCount - count)) + ") ms left");
    try {
      Thread.sleep(reduceSleepDuration);
    } catch (InterruptedException e) {
      throw new IOException(e);
    }

    count++;
  }
}

public static int run(int numMapper, int numReducer, long mapSleepTime,
    int mapSleepCount, long reduceSleepTime, int reduceSleepCount)
    throws OdpsException {
  JobConf job = setupJobConf(numMapper, numReducer, mapSleepTime,
      mapSleepCount, reduceSleepTime, reduceSleepCount);
  JobClient.runJob(job);
  return 0;
}

public static JobConf setupJobConf(int numMapper, int numReducer,
    long mapSleepTime, int mapSleepCount, long reduceSleepTime,
    int reduceSleepCount) {
  JobConf job = new JobConf();

  InputUtils.addTable(TableInfo.builder().tableName("mr_empty").build(), job);
  OutputUtils.addTable(TableInfo.builder().tableName("mr_sleep_out").build(), job);

  job.setNumReduceTasks(numReducer);

  job.setMapperClass(SleepMapper.class);
  job.setReducerClass(SleepReducer.class);

  job.setMapOutputKeySchema(SchemaUtils.fromString("int1:bigint"));
```

```
    job.setMapOutputValueSchema(SchemaUtils.fromString("int2:bigint"));
    job.setPartitionColumns(new String[] { "int1" });

    job.setLong("sleep.job.map.sleep.time", mapSleepTime);
    job.setLong("sleep.job.reduce.sleep.time", reduceSleepTime);
    job.setInt("sleep.job.map.sleep.count", mapSleepCount);
    job.setInt("sleep.job.reduce.sleep.count", reduceSleepCount);

    return job;
  }

  private static void printUsage() {
    System.err.println("SleepJob [-m numMapper] [-r numReducer]"
        + " [-mt mapSleepTime (msec)] [-rt reduceSleepTime (msec)]"
        + " [-recordt recordSleepTime (msec)]");
  }

  public static void main(String[] args) throws Exception {

    if (args.length < 1) {
      printUsage();
      return;
    }

    int numMapper = 1, numReducer = 1;
    long mapSleepTime = 100, reduceSleepTime = 100, recSleepTime = 100;
    int mapSleepCount = 1, reduceSleepCount = 1;

    for (int i = 0; i < args.length; i++) {
      if (args[i].equals("-m")) {
        numMapper = Integer.parseInt(args[++i]);
      } else if (args[i].equals("-r")) {
        numReducer = Integer.parseInt(args[++i]);
      } else if (args[i].equals("-mt")) {
        mapSleepTime = Long.parseLong(args[++i]);
      } else if (args[i].equals("-rt")) {
        reduceSleepTime = Long.parseLong(args[++i]);
      } else if (args[i].equals("-recordt")) {
        recSleepTime = Long.parseLong(args[++i]);
      }
    }

    // sleep for *SleepTime duration in Task by recSleepTime per record
    mapSleepCount = (int) Math.ceil(mapSleepTime / ((double) recSleepTime));
    reduceSleepCount = (int) Math.ceil(reduceSleepTime
        / ((double) recSleepTime));

    run(numMapper, numReducer, mapSleepTime, mapSleepCount, reduceSleepTime,
        reduceSleepCount);
  }

}
```

unique示例

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

import java.io.IOException;
import java.util.Iterator;

import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;

/**
 * Unique Remove duplicate words
 *
 * To run: jar -resources odps-mapred-example-0.12.0.jar
 * com.aliyun.odps.open.mapred.example.Unique mr_sort_in mr_unique_out
 * key|value|all;
 *
 **/
public class Unique {

  public static class OutputSchemaMapper extends MapperBase {
    private Record key;
    private Record value;

    @Override
    public void setup(TaskContext context) throws IOException {
      key = context.createMapOutputKeyRecord();
      value = context.createMapOutputValueRecord();
    }

    @Override
    public void map(long recordNum, Record record, TaskContext context)
        throws IOException {
      long left = 0;
      long right = 0;

      if (record.getColumnCount() > 0) {
        left = (Long) record.get(0);
        if (record.getColumnCount() > 1) {
          right = (Long) record.get(1);
        }

        key.set(new Object[] { (Long) left, (Long) right });
        value.set(new Object[] { (Long) left, (Long) right });

        context.write(key, value);
      }
    }
  }
```

```
public static class OutputSchemaReducer extends ReducerBase {
  private Record result = null;

  @Override
  public void setup(TaskContext context) throws IOException {
    result = context.createOutputRecord();
  }

  @Override
  public void reduce(Record key, Iterator<Record> values, TaskContext context)
      throws IOException {
    result.set(0, key.get(0));
    while (values.hasNext()) {
      Record value = values.next();
      result.set(1, value.get(1));
    }
    context.write(result);
  }
}

public static void main(String[] args) throws Exception {
  if (args.length > 3 || args.length < 2) {
    System.err.println("Usage: unique <in> <out> [key|value|all]");
    System.exit(2);
  }

  String ops = "all";
  if (args.length == 3) {
    ops = args[2];
  }

  // Key Unique
  if (ops.equals("key")) {
    JobConf job = new JobConf();

    job.setMapperClass(OutputSchemaMapper.class);
    job.setReducerClass(OutputSchemaReducer.class);

    job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,value:bigint"));
    job.setMapOutputValueSchema(SchemaUtils.fromString("key:bigint,value:bigint"));

    job.setPartitionColumns(new String[] { "key" });
    job.setOutputKeySortColumns(new String[] { "key", "value" });
    job.setOutputGroupingColumns(new String[] { "key" });

    job.set("tablename2", args[1]);

    job.setNumReduceTasks(1);
    job.setInt("table.counter", 0);

    InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
    OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);

    JobClient.runJob(job);
  }
```

```
    // Key&Value Unique
    if (ops.equals("all")) {
      JobConf job = new JobConf();

      job.setMapperClass(OutputSchemaMapper.class);
      job.setReducerClass(OutputSchemaReducer.class);

      job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,value:bigint"));
      job.setMapOutputValueSchema(SchemaUtils.fromString("key:bigint,value:bigint"));

      job.setPartitionColumns(new String[] { "key" });
      job.setOutputKeySortColumns(new String[] { "key", "value" });
      job.setOutputGroupingColumns(new String[] { "key", "value" });

      job.set("tablename2", args[1]);

      job.setNumReduceTasks(1);
      job.setInt("table.counter", 0);

      InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
      OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);

      JobClient.runJob(job);
    }

    // Value Unique
    if (ops.equals("value")) {
      JobConf job = new JobConf();

      job.setMapperClass(OutputSchemaMapper.class);
      job.setReducerClass(OutputSchemaReducer.class);

      job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,value:bigint"));
      job.setMapOutputValueSchema(SchemaUtils.fromString("key:bigint,value:bigint"));

      job.setPartitionColumns(new String[] { "value" });
      job.setOutputKeySortColumns(new String[] { "value" });
      job.setOutputGroupingColumns(new String[] { "value" });

      job.set("tablename2", args[1]);

      job.setNumReduceTasks(1);
      job.setInt("table.counter", 0);

      InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
      OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);

      JobClient.runJob(job);
    }

  }

}
```

sort示例

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

import java.io.IOException;
import java.util.Date;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.example.lib.IdentityReducer;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;

/**
 * This is the trivial map/reduce program that does absolutely nothing other
 * than use the framework to fragment and sort the input values.
 * <p>
 * To run: jar -resources odps-mapred-example-0.12.0.jar com.aliyun.odps.mapred.open.example.Sort
 * mr_sort_in mr_sort_out;
 *
 **/
public class Sort {

  static int printUsage() {
    System.out.println("sort <input> <output>");
    return -1;
  }

  /**
   * Implements the identity function, mapping record's first two columns to
   * outputs.
   **/
  public static class IdentityMapper extends MapperBase {
    private Record key;
    private Record value;

    @Override
    public void setup(TaskContext context) throws IOException {
      key = context.createMapOutputKeyRecord();
      value = context.createMapOutputValueRecord();
    }

    @Override
    public void map(long recordNum, Record record, TaskContext context)
        throws IOException {
      key.set(new Object[] { (Long) record.get(0) });
      value.set(new Object[] { (Long) record.get(1) });
      context.write(key, value);
    }

  }
```

```
  /**
   * The main driver for sort program. Invoke this method to submit the
   * map/reduce job.
   *
   * @throws IOException
   *         When there is communication problems with the job tracker.
   **/
  public static void main(String[] args) throws Exception {

    JobConf jobConf = new JobConf();

    jobConf.setMapperClass(IdentityMapper.class);
    jobConf.setReducerClass(IdentityReducer.class);

    jobConf.setNumReduceTasks(1);

    jobConf.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint"));
    jobConf.setMapOutputValueSchema(SchemaUtils.fromString("value:bigint"));

    InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), jobConf);
    OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), jobConf);

    Date startTime = new Date();
    System.out.println("Job started: " + startTime);

    JobClient.runJob(jobConf);

    Date end_time = new Date();
    System.out.println("Job ended: " + end_time);
    System.out.println("The job took "
        + (end_time.getTime() - startTime.getTime()) / 1000 + " seconds.");
  }
}
```

partition示例

代码示例1，一段简单的将Partition作为输入输出的例子，仅供参考：

```
public static void main(String[] args) throws Exception {

JobConf job = new JobConf();

...

LinkedHashMap<String, String> input = new LinkedHashMap<String, String>();
input.put("pt", "123456");
InputUtils.addTable(TableInfo.builder().tableName("input_table").partSpec(input).build(), job);

LinkedHashMap<String, String> output = new LinkedHashMap<String, String>();
output.put("ds", "654321");
OutputUtils.addTable(TableInfo.builder().tableName("output_table").partSpec(output).build(), job);

JobClient.runJob(job);
}
```

代码示例2，仅供参考：

```
package com.aliyun.odps.mapred.open.example;

...
  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: WordCount <in_table> <out_table>");
      System.exit(2);
    }

    JobConf job = new JobConf();

    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(SumCombiner.class);
    job.setReducerClass(SumReducer.class);

    job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
    job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));


    Account account = new AliyunAccount("my_access_id", "my_access_key");
    Odps odps = new Odps(account);
    odps.setEndpoint("odps_endpoint_url");
    odps.setDefaultProject("my_project");

    Table table = odps.tables().get(tblname);
    TableInfoBuilder builder = TableInfo.builder().tableName(tblname);

    for (Partition p : table.getPartitions()) {
      if (applicable(p)) {
        LinkedHashMap<String, String> partSpec = new LinkedHashMap<String, String>();
        for (String key : p.getPartitionSpec().keys()) {
          partSpec.put(key, p.getPartitionSpec().get(key));
        }
        InputUtils.addTable(builder.partSpec(partSpec).build(), conf);
      }
    }

    OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);

    JobClient.runJob(job);
  }
```

备注:

- 这是一段使用ODPS SDK和MapReduce SDK组合实现MapReduce任务读取范围Partitoin的示例。此段代码不能够编译执行， 仅给出了main函数的示例。示例中applicable函数是用户逻辑，用来决定该Partition是否符合作为该MapReduce作业的输入。

pipeline示例

代码示例，仅供参考：

```
package com.aliyun.odps.mapred.example;

import java.io.IOException;
import java.util.Iterator;

import com.aliyun.odps.Column;
import com.aliyun.odps.OdpsException;
import com.aliyun.odps.OdpsType;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.Job;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.pipeline.Pipeline;

public class Histogram {

 public static class TokenizerMapper extends MapperBase {

  Record word;
  Record one;

  @Override
  public void setup(TaskContext context) throws IOException {
   word = context.createMapOutputKeyRecord();
   one = context.createMapOutputValueRecord();
   one.setBigint(0, 1L);
  }

  @Override
  public void map(long recordNum, Record record, TaskContext context)
      throws IOException {
   for (int i = 0; i < record.getColumnCount(); i++) {
    String[] words = record.get(i).toString().split("\\s+");
    for (String w : words) {
     word.setString(0, w);
     context.write(word, one);
    }
   }
  }
 }

 public static class SumReducer extends ReducerBase {
  private Record num;
  private Record result;

  @Override
  public void setup(TaskContext context) throws IOException {
   num = context.createOutputKeyRecord();
   result = context.createOutputValueRecord();
  }

  @Override
  public void reduce(Record key, Iterator<Record> values, TaskContext context)
      throws IOException {
   long count = 0;
```

```
      while (values.hasNext()) {
        Record val = values.next();
        count += (Long) val.get(0);
      }
      result.set(0, key.get(0));
      num.set(0, count);
      context.write(num, result);
    }
  }

  public static class IdentityReducer extends ReducerBase {

    @Override
    public void reduce(Record key, Iterator<Record> values, TaskContext context)
        throws IOException {
      while (values.hasNext()) {
        context.write(values.next());
      }
    }
  }

  public static void main(String[] args) throws OdpsException {
    if (args.length != 2) {
      System.err.println("Usage: orderedwordcount <in_table> <out_table>");
      System.exit(2);
    }

    Job job = new Job();


    /***
     * 构造Pipeline的过程中，如果不指定Mapper的
OutputKeySortColumns，PartitionColumns，OutputGroupingColumns，
     * 框架会默认使用其OutputKey作为此三者的默认配置
     ***/
    Pipeline pipeline = Pipeline.builder()
        .addMapper(TokenizerMapper.class)
        .setOutputKeySchema(
            new Column[] { new Column("word", OdpsType.STRING) })
        .setOutputValueSchema(
            new Column[] { new Column("count", OdpsType.BIGINT) })
        .setOutputKeySortColumns(new String[] { "word" })
        .setPartitionColumns(new String[] { "word" })
        .setOutputGroupingColumns(new String[] { "word" })

        .addReducer(SumReducer.class)
        .setOutputKeySchema(
            new Column[] { new Column("count", OdpsType.BIGINT) })
        .setOutputValueSchema(
            new Column[] { new Column("word", OdpsType.STRING)})

        .addReducer(IdentityReducer.class).createPipeline();

    job.setPipeline(pipeline);

    job.addInput(TableInfo.builder().tableName(args[0]).build());
```

```
    job.addOutput(TableInfo.builder().tableName(args[1]).build());

    job.submit();
    job.waitForCompletion();
    System.exit(job.isSuccessful() == true ? 0 : 1);
  }

}
```