FusionInsight

# MapReduce应用开发

www.huawei.com



HUAWEI

# 目标

- 学完本课程后，您将能够:
  - 掌握**MapReduce**的业务过程；
  - 掌握开发环境搭建；
  - 进行**MapReduce**应用开发；

# 目录

# MapReduce 基本定义

**MapReduce**是面向大数据并行处理的计算模型、框架和平台，其资源调度由**Yarn**完成，任务资源隐含了以下三层含义：

**1）MapReduce**是一个基于集群的高性能并行计算平台（**Cluster Infrastructure**）。

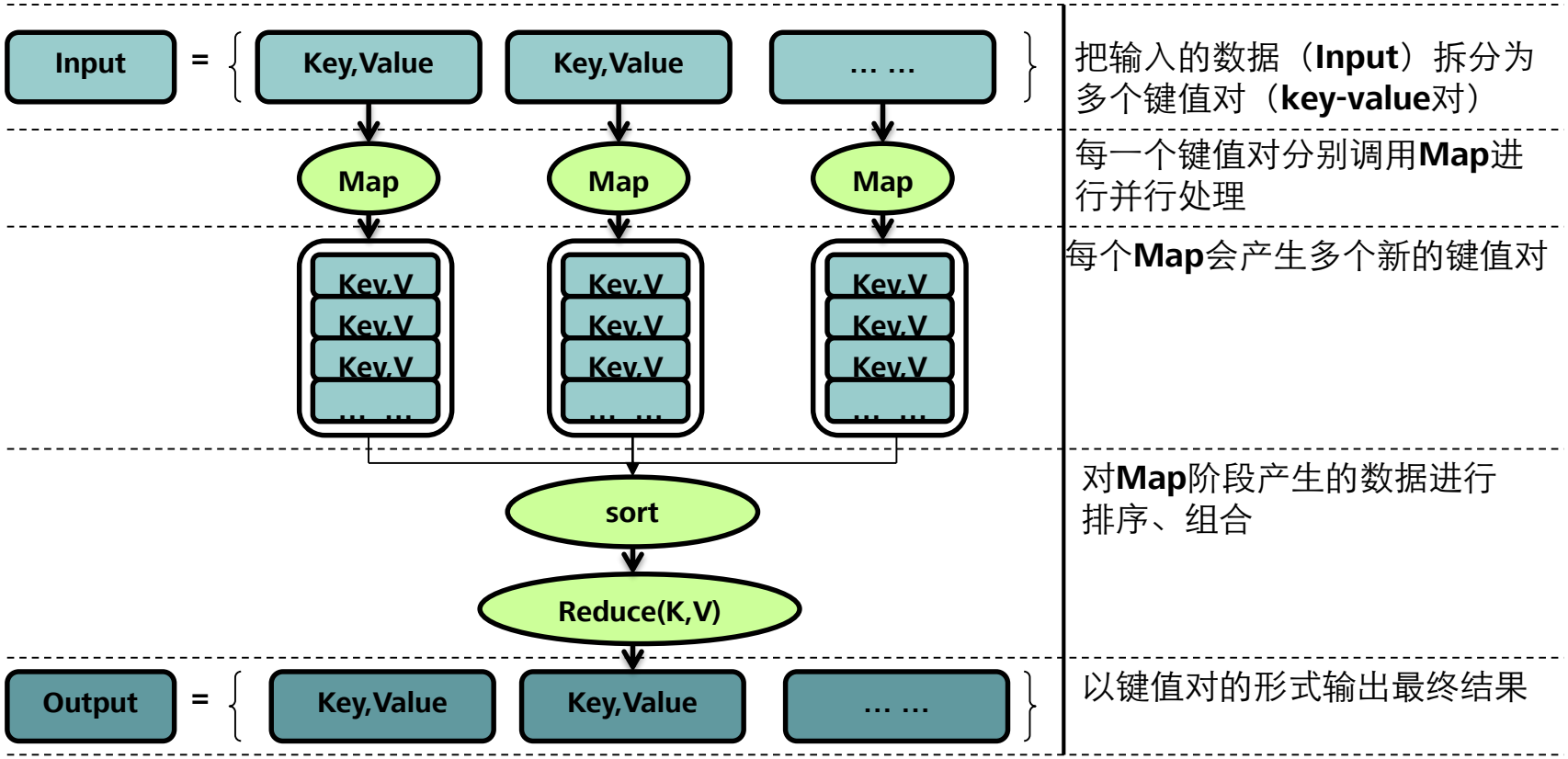
**2）MapReduce**是一个并行计算与运行软件框架（**Software Framework**）。

**3）MapReduce**是一个并行程序设计模型与方法（**Programming Model & Methodology**）。

# MapReduce 应用场景

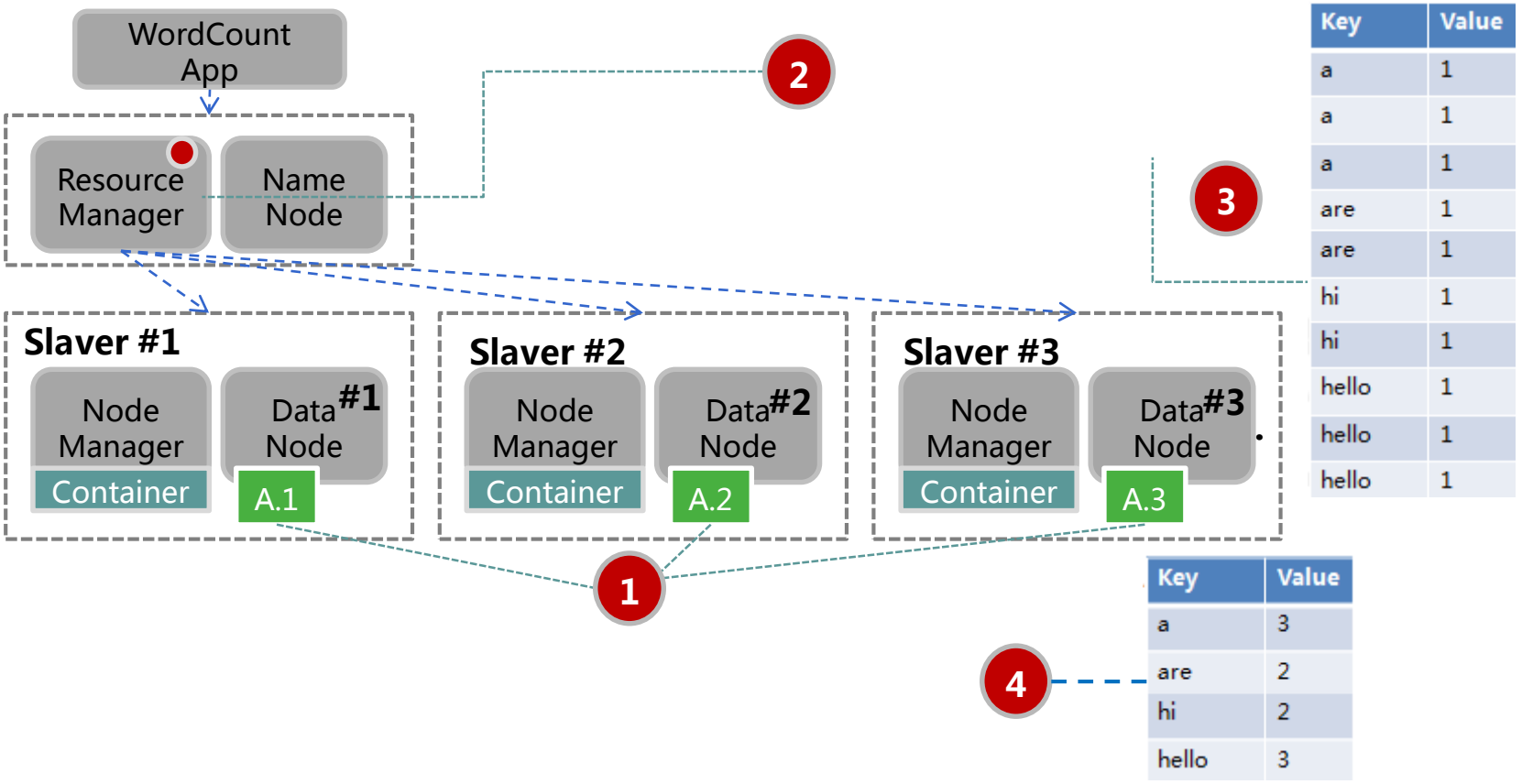
- **MapReduce**基于**Google**发布的分布式计算框架**MapReduce**论文设计开发，用于大规模数据集（大于**1TB**）的并行运算，特点如下：
  - 易于编程：程序员仅需描述做什么，具体怎么做就交由系统的执行框架处理。
  - 良好的扩展性：可以添加机器扩展集群能力。
  - 高容错性：通过计算迁移或数据迁移等策略提高集群的可用性与容错性。

# MapReduce的过程-图解MR

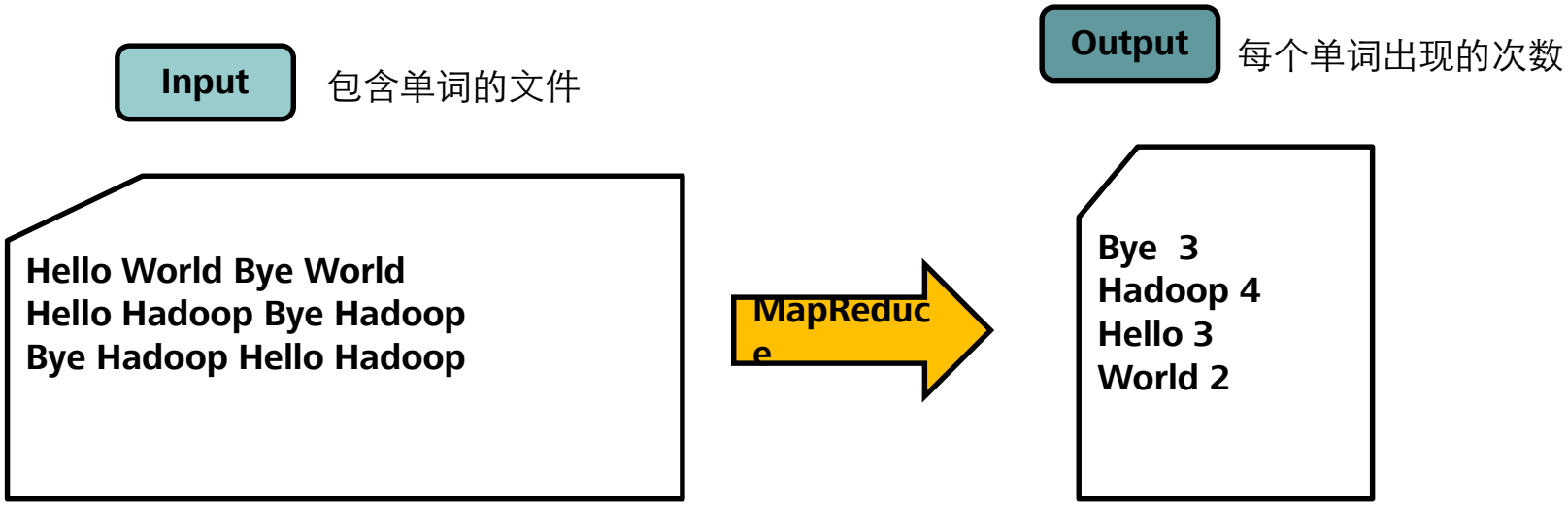


# 典型程序WordCount举例

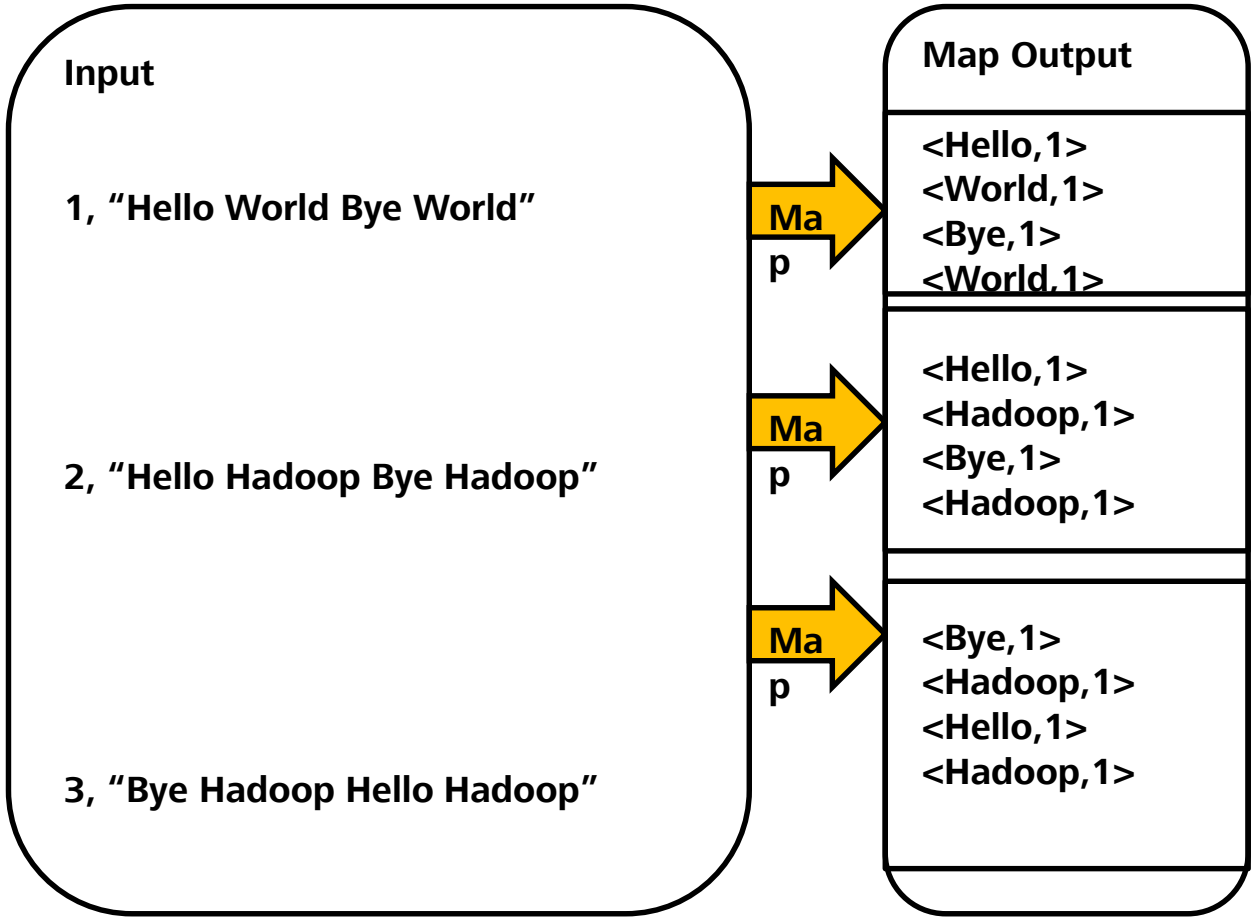
假设要分析一个大文件**A**里每个英文单词出现的个数，利用**MapReduce**框架能快速实现这一统计分析。



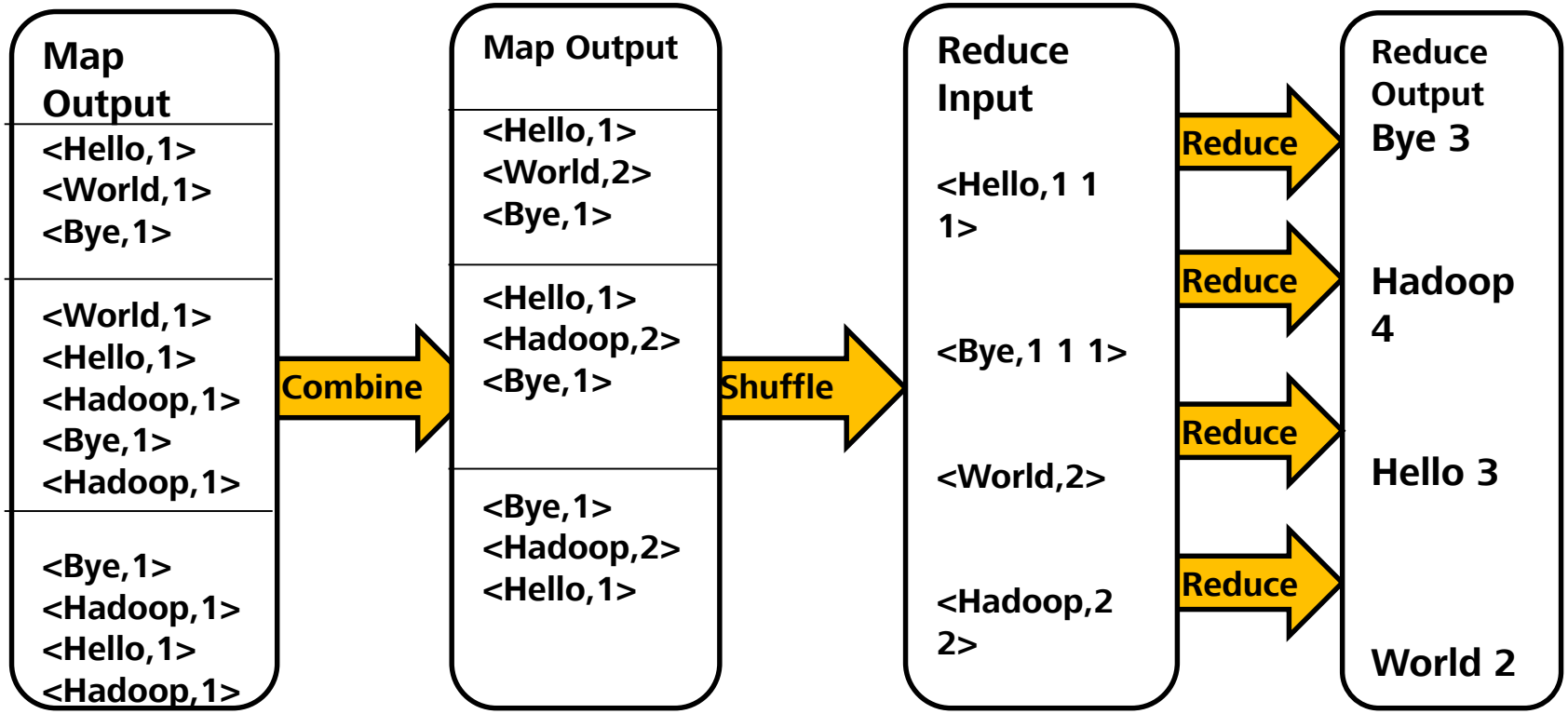
# WordCount程序功能



# WordCount 的Map过程



# WordCount的Reduce过程



# 目录

# 准备开发环境

开发环境安装**Eclipse**软件，安装要求如下：
**Eclipse**使用**3.6**及以上版本。

开发环境安装**JDK**程序，安装要求如下：
**JDK**使用**1.7**或**1.8**版本。

# 搭建开发环境

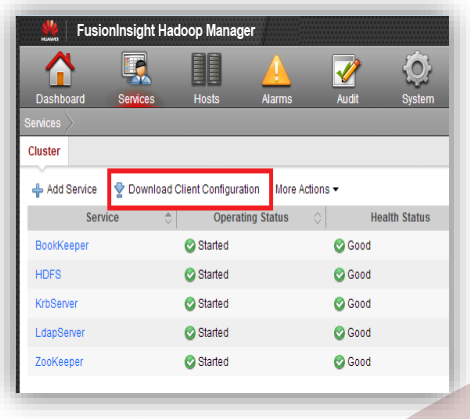
**1.**确认**Yarn**组件和**MapReduce**组件已经安装，并正常运行。

**2.**客户端机器安装**Eclipse**和**JDK**程序，安装要求如下：
**Eclipse**使用**3.0**及以上版本，并安装**JUnit**插件。
**JDK**使用**1.7**版本。

**3.**客户端机器的时间与**FusionInsight**集群的时间要保持一致，时间差要小于**5**分钟。
**FusionInsight**集群的时间可通过登录主管理节点（集群管理**IP**地址所在节点）运行**date**命令查询。

**4.**需要在**Yarn**服务页面下载**MapReduce**客户端程序到客户端机器中。

# 搭建开发环境



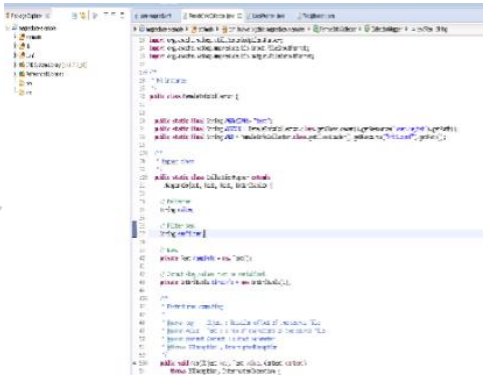

下载客户端

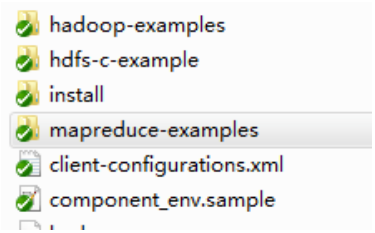
获取样例工程

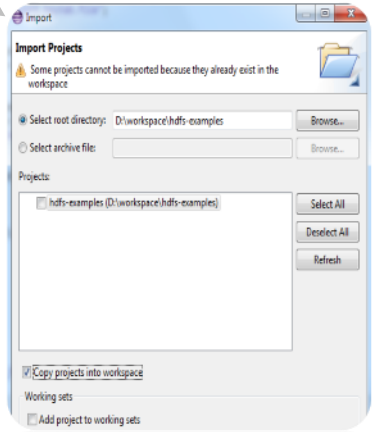
生成样例工程

导入eclipse

编码








HUAWEI

# 目录

# 代码示例-场景说明

假定用户有某个周末网民网购停留时间的日志文本，基于某些业务要求，要求开发**MapReduce**应用程序实现如下功能：

- 统计日志文件中本周末网购停留总时间超过**2**个小时的女性网民信息。
- 周末两天的日志文件第一列为姓名，第二列为性别，第三列为本次停留时间，单位为分钟，分隔符为**”,”**。

周六网民停留日志：

**LiuYang,female,20**
**YuanJing,male,10**
**GuoYijun,male,5**
**CaiXuyu,female,50**
**Liyuan,male,20**
**FangBo,female,50**
**LiuYang,female,20**
**YuanJing,male,10**
**GuoYijun,male,50**
**CaiXuyu,female,50**
**FangBo,female,60**

周日网民停留日志：

**LiuYang,female,20 YuanJing,male,10**
**CaiXuyu,female,50 FangBo,female,50**
**GuoYijun,male,5 CaiXuyu,female,50**
**Liyuan,male,20 CaiXuyu,female,50**
**FangBo,female,50 LiuYang,female,20**
**YuanJing,male,10 FangBo,female,50**
**GuoYijun,male,50 CaiXuyu,female,50**
**FangBo,female,60**

# 代码示例- 业务分析

统计日志文件中本周末网购停留总时间超过2个小时的女性网民信息。
主要分为三个部分:

1.从原文件中筛选女性网民上网时间数据信息，通过类CollectionMapper继承Mapper抽象类实现。

2.汇总每个女性上网时间，并输出时间大于两个小时的女性网民信息，通过类CollectionReducer继承Reducer抽象类实现。

3.main方法提供建立一个MapReduce job，并提交MapReduce作业到hadoop集群 。

# 代码示例-map方法实现

类**CollectionMapper**定义**Mapper**抽象类的**map()**方法，**Map**过程需要继承**org.apache.hadoop.mapreduce**包中的**Mapper**类，并重写**map**方法，实现读取字符串数据，获取上网停留时间

```
/**
* 分布式计算
*
* @param key Object : 原文件位置偏移量。
* @param value Text : 原文件的一行字符数据。
* @param context Context : 出参。
* @throws IOException , InterruptedException
*/
   public void map(Object key, Text value, Context context)
           throws IOException, InterruptedException
   {

       String line = value.toString();

       if (line.contains(sexFilter))
       {

               // 读取的一行字符串数据。
                 String name = line.substring(0, line.indexOf(delim));
                 nameInfo.set(name);
               // 获取上网停留时间。
                 String time = line.substring(line.lastIndexOf(delim) + 1,
                               line.length());
                  timeInfo.set(Integer.parseInt(time));

           // map输出key，value键值对。
         context.write(nameInfo, timeInfo);
       }
   }
```

最后输出为**key,value**键值对

# 代码示例-reduce方法实现

类**CollectionReducer**定义**Reducer**抽象类的**reduce()**方法。**Reduce**过程需要继承**org.apache.hadoop.mapreduce**包中的**Reducer**类，并重写**reduce**方法。

判断输出条件。

**reduce**输出为**key**：网民的信息，**value**：该网民上网总时间。

```
    /**
     * @param key Text : Mapper后的key项。
     * @param values Iterable : 相同key项的所有统计结果。
     * @param context Context
     * @throws IOException , InterruptedException
     */
      public void reduce(Text key, Iterable<IntWritable> values,
              Context context) throws IOException, InterruptedException
{
          int sum = 0;
          for (IntWritable val : values) {
              sum += val.get();
          }

          // 如果时间小于门槛时间，不输出结果。
          if (sum < timeThreshold)
      {
              return;
          }
          result.set(sum);

          // reduce输出为key: 网民的信息，value: 该网民上网总时间。
          context.write(key, result);
       }
```

# 代码示例-初始化及提交任务

在**MapReduce**中，由**Job**对象负责管理和运行一个计算任务，并通过**Job**的一些方法对任务的参数进行相关的设置。

此处设置了使**CollectionMapper**完成**Map**过程和使用**CollectionReducer**完成**Combine**和**Reduce**过程。还设置了**Map**过程和**Reduce**过程的输出类型：**key**的类型为**Text**，**value**的类型为**IntWritable**。

任务的输入和输出路径则由命令行参数指定，并由**FileInputFormat**和**FileOutputFormat**分别设定。完成相应任务的参数设定后，即可调用**job.waitForCompletion()**方法执行任务。

```
// 初始化Job任务对象。
@SuppressWarnings("deprecation")
Job job = new Job(conf, "Collect Female Info");
job.setJarByClass(FemaleInfoCollector.class);

// 设置运行时执行map，reduce的类，也可以通过配置文件指定。
job.setMapperClass(CollectionMapper.class);
job.setReducerClass(CollectionReducer.class);

// 设置combiner类，默认不使用，使用时通常使用和reduce一样的类。
// Combiner类需要谨慎使用，也可以通过配置文件指定。
job.setCombinerClass(CollectionReducer.class);

// 设置作业的输出类型。
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

// 提交任务交到远程环境上执行。
System.exit(job.waitForCompletion(true) ? 0 : 1);
```

# 代码示例-Mapreduce跨组件调用（1）

在编写的**Mapreduce**二次开发程序中。如果需要访问**Hbase**，**Hive**等组件，这些组件需要进行访问**zookeeper**。下面以访问**Hbase**为例，要分别在**map**及**reduce**方法中提供**jaas**和**krb5**文件。在**main**方法中进行了鉴权后，**map**方法中只要传入**jaas**以及**krb5**文件，并调用**system.setProperty**方法即可。在链接**zookeeper**时，会自动认证。此方法会使**map**起的**JVM**带有这些**r**认证信息。

```
// 对于需要访问ZooKeeper的组件，需要提供jaas和krb5配置
// 在Map中不需要重复login，会使用main方法中配置的鉴权信息
String krb5 = "krb5.conf";
String jaas = "jaas_mr.conf";
// 这些文件上传自main方法
File jaasFile = new File(jaas);
File krb5File = new File(krb5);
System.setProperty("java.security.auth.login.config", jaasFile.getCanonicalPath());
System.setProperty("java.security.krb5.conf", krb5File.getCanonicalPath());
System.setProperty("zookeeper.sasl.client", "true");
```

# 代码示例-Mapreduce跨组件调用（2）

**Reduce**方法访问**zookeeper**与**map**类似，在**main**方法中进行了鉴权后，**reduce**方法中只要传入**jaas**以及**krb5**文件，并调用**system.setProperty**方法即可。在链接**zookeeper**时，会自动认证。此方法会使**reduce**起的**JVM**带有这些认证信息。

```
// 对于需要访问ZooKeeper的组件，需要提供jaas和krb5配置
// 在Map中不需要重复login，会使用main方法中配置的鉴权信息
String krb5 = "krb5.conf";
String jaas = "jaas_mr.conf";
// 这些文件上传自main方法
File jaasFile = new File(jaas);
File krb5File = new File(krb5);
System.setProperty("java.security.auth.login.config", jaasFile.getCanonicalPath());
System.setProperty("java.security.krb5.conf", krb5File.getCanonicalPath());
System.setProperty("zookeeper.sasl.client", "true");
```

说明：详细的开发示例请参看配套**CPI**文档。

# 开发相关类总结

下面介绍**MapReduce**的二次开发部分参数及其默认设置：

**（1）InputFormat**类

该类的作用是将输入的数据分割成一个个的**split**，并将**split**进一步拆分成**<key, value>**对作为**map**函数的输入。在执行一个任务的时候， **MapReduce**会将输入数据划分成**N**个**Split**，然后启动相应的**N**个**Map**程序来分别处理它们。数据如何划分？**Split**如何调度？划分后的数据又如何读取**?**此类将负责这些过程。

**（2）Mapper**类

实现**map**函数，根据输入的**<key, value>**对生产中间结果。

**（3）Combiner**

实现**combine**函数，合并中间结果中具有相同**key**值的键值对。

**（4）Partitioner**类

实现**getPartition**函数，在**Shuffle**过程按照**key**值将中间数据分成**R**份，每一份由一个**Reduce**负责。

**（5）Reducer**类

实现**reduce**函数，将中间结果合并，得到最终的结果。

**（6）OutputFormat**类

该类负责输出最终的结果， **MapReduce**使用**OutputFormat**类将数据输出存入文件中，每个**Reducer**将它的输出直接写到自己的文件中。

# 运行程序-Windows环境

- 编译并运行程序
  - 在开发环境**Eclipse**中，右击**LocalRunner.java**，单击“**Run as > Java Application**”运行对应的应用程序工程。
  - 若是自己开发的程序，就运行调用任务的类即可。即使用**job.waitForCompletion** 方法启动任务的类。

# 运行程序-Linux环境

1.导出**MapReduce**应用**Jar**包“**mapreduce-example**”。

2.拷贝生成的应用包到**Linux**客户端节点上。例如保存在客户端安装目录。

3.导入运行应用所需要使用的组件客户端环境变量。

导入环境变量：**source /opt/yarn_client/bigdata_env**。

4.在**HDFS**中新建目录，例如创建“**/user/input**”。上传样例工程中“**log1.txt**”和“**log2.txt**”到该目录。

5.在**Linux**客户端环境下提交**MapReduce**任务。

执行**yarn jar mapreduce-example.jar <input> <output>**命令，运行样例工程。

# 查看运行结果

- 查看结果
  - 查看**Yarn API**返回结果是否符合预期。
  - 若有运行**Yarn shell**或者访问**Yarn webUI**权限，可以查看程序运行结果。（程序运行最终状态输出在**HDFS**路径下的结果）。

# 调试代码

**MapReduce**开发调试采用原理是**Java**的远程调试机制，在**Map/Reduce**任务（运行在一个**container**中的进程）启动时，添加**jar**远程调试命令。

**1.**在工程中**conf**目录下的**mapred-site.xml**配置文件中找到 这两个**"mapreduce.map.java.opts"**、**"mapreduce.reduce.java.opts"**参数。 在这两个参数的值中分别加入调试命令“**-Xdebug -Xnoagent -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=y"**。

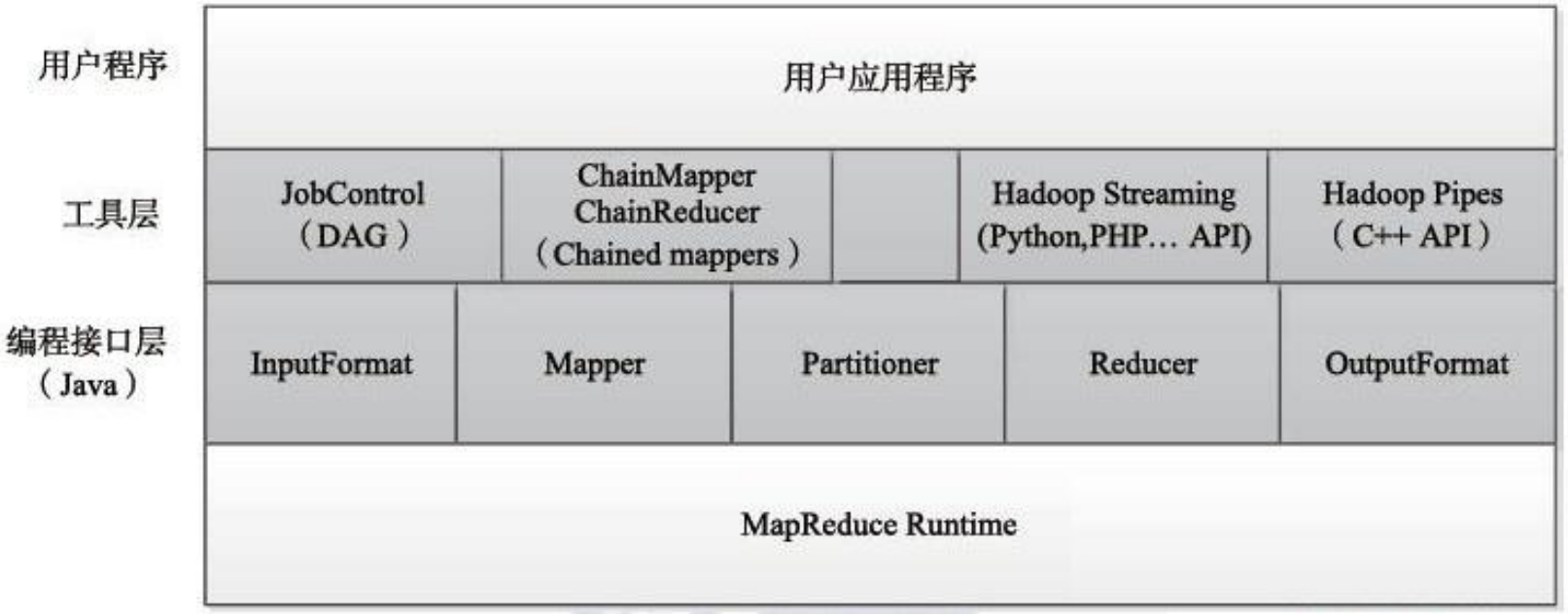
**2.**当通过**IDE**提交**MapReduce**任务后，**Yarn**启动各个**container**实例，**container**实例启动时读到上述远程调试命令，则监听本机**8000**端口。

**3.**在**IDE**上，选择**MapReduce**任务的实现类，通过配置远程调试信息，执行**debug**。

- 增加了调试参数后，按步执行后提交**job**，**job**会在**map 0% reduce 0%**的时刻停止以等待远程调试的**debugger**。此时在**Eclipse**中设置相应断点和远程调试，对**localhost**的**8000**端口进行远程调试。
- 进入**Yarn**的**Web**界面，查看**Map/Reduce**任务的**container**信息，查看**Map**任务所在的节点**IP**并记录。
- 在**IDE**上选择**Map/Reduce**类，然后选择“**Debug As > Debug Configurations...**”。
- 在弹出的页面，单击“**Remote Java Application**”并选择“**FemaleInfoCollector**”类，**host**为**Map**所在节点，端口号为**8000**，然后单击“**debug**”。

# 目录

# MR接口介绍




第一层是工具层，位于基本**Java API**之上，主要是为了方便用户编写复杂的**MapReduce**程序和利用其他编程语言增加**MapReduce**计算平台的兼容性而提出来的。

第二层是最基本的**Java API**，主要有**5**个可编程组件，分别是**InputFormat**、**Mapper**、**Partitioner**、**Reducer**和**OutputFormat**。**Hadoop**自带了很多直接可用的**InputFormat**、**Partitioner**和**OutputFormat**，大部分情况下，用户只需编写**Mapper**和**Reducer**即可。

# MR接口介绍-工具层

**JobControl**：方便用户编写有依赖关系的作业，这些作业往往构成一个有向图，所以通常称为**DAG**（**Directed Acyclic Graph**）作业。

**ChainMapper/ChainReducer**：方便用户编写链式作业，即在**Map**或者**Reduce**阶段存在多个**Mapper**，形式如下：**[MAPPER+ REDUCER MAPPER*]**。

**Hadoop Streaming**：方便用户采用非**Java**语言编写作业，允许用户指定可执行文件或者脚本作为**Mapper/Reducer**。

**Hadoop Pipes**：专门为**C/C++**程序员编写**MapReduce**程序提供的工具包。

# MR接口介绍-Java API（1）

| 序号 | 方法 | 说明 |
| --- | --- | --- |
| 1 | Job(Configuration conf, String jobName), Job(Configuration conf) | 新建一个MapReduce客户端，用于配置作业属性，提交作业。 |
| 2 | setMapperClass(Class < extends Mapper> cls) | 核心接口，指定MapReduce作业的Mapper类，默认为空。也可以在“mapred-site.xml”中配置“mapreduce.job.map.class”项。 |
| 3 | setReducerClass(Class< extends Reducer> cls) | 核心接口，指定MapReduce作业的Reducer类，默认为空。也可以在“mapred-site.xml”中配置“mapreduce.job.reduce.class”项。 |
| 4 | setCombinerClass(Class< extends Reducer> cls) | 指定MapReduce作业的Combiner类，默认为空。也可以在“mapred-site.xml”中配置“mapreduce.job.combine.class”项。需要保证reduce的输入输出key，value类型相同才可以使用，谨慎使用。 |
| 5 | setInputFormatClass(Class< extends InputFormat> cls) | 核心接口，指定MapReduce作业的InputFormat类，默认为TextInputFormat。也可以在“mapred-site.xml”中配置“mapreduce.job.inputformat.class”项。该设置用来指定处理不同格式的数据时需要的InputFormat类，用来读取数据，切分数据块。 |
| 6 | setJarByClass(Class<> cls) | 核心接口，指定执行类所在的jar包本地位置。java通过class文件找到执行jar包，该jar包被上传到HDFS。 |

# MR接口介绍-Java API（2）

| 序号 | 方法 | 说明 |
| --- | --- | --- |
| 7 | setJar(String jar) | 指定执行类所在的jar包本地位置。直接设置执行jar包所在位置，该jar包被上传到HDFS。与setJarByClass(Class<?> cls)选择使用一个。也可以在“mapred-site.xml”中配置“mapreduce.job.jar”项。 |
| 8 | setOutputFormat(Class<? extends OutputFormat> theClass) | 核心接口，指定MapReduce作业的OutputFormat类，默认为TextOutputFormat。也可以在“mapred-site.xml”中配置“mapred.output.format.class”项，指定输出结果的数据格式。例如默认的TextOutputFormat把每条key，value记录写为文本行。通常场景不配置特定的OutputFormat。 |
| 9 | setOutputKeyClass(Class<?> theClass) | 核心接口，指定MapReduce作业的输出key的类型，也可以在“mapred-site.xml”中配置“mapreduce.job.output.key.class”项。 |
| 10 | setOutputValueClass(Class<?> theClass) | 核心接口，指定MapReduce作业的输出value的类型，也可以在“mapred-site.xml”中配置“mapreduce.job.output.value.class”项。 |
| 11 | setPartitionerClass(Class<? extends Partitioner> theClass) | 指定MapReduce作业的Partitioner类。也可以在“mapred-site.xml”中配置“mapred.partitioner.class”项。该方法用来分配map的输出结果到哪个reduce类，默认使用HashPartitioner，均匀分配map的每条键值对记录。例如在hbase应用中，不同的键值对应的region不同，这就需要设定特殊的partitioner类分配map的输出结果。 |

# MR接口介绍-Java API（3）

| 序号 | 方法 | 说明 |
| --- | --- | --- |
| 12 | setMapOutputCompressorClass(Class<? extends CompressionCodec> codecClass) | 指定MapReduce作业的map任务的输出结果压缩类，默认不使用压缩。也可以在“mapred-site.xml”中配置“mapreduce.map.output.compress”和“mapreduce.map.output.compress.codec”项。当map的输出数据大，减少网络压力，使用压缩传输中间数据。 |
| 13 | setJobPriority(JobPriority prio) | 指定MapReduce作业的优先级，共有5个优先级别，VERY_HIGH,HIGH,NORMAL,LOW,VERY_LOW，默认级别为NORMAL。也可以在“mapred-site.xml”中配置“mapreduce.job.priority”项。 |
| 14 | setQueueName(String queueName) | 指定MapReduce作业的提交队列。默认使用default队列。也可以在“mapred-site.xml”中配置“mapreduce.job.queuename”项。 |
| 15 | setNumMapTasks(int n) | 核心接口，指定MapReduce作业的map个数。也可以在“mapred-site.xml”中配置“mapreduce.job.maps”项。<br>注意：指定的InputFormat类用来控制map任务个数，注意该类是否支持客户端设定map个数。 |
| 16 | setNumReduceTasks(int n) | 核心接口，指定MapReduce作业的reduce个数。默认只启动1个。也可以在“mapred-site.xml”中配置“mapreduce.job.reduces”项。reduce个数由用户控制，通常场景reduce个数是map个数的1/4。 |

**说明：**

开发接口类：**org.apache.hadoop.mapreduce.Job**上面仅介绍了一些常用的接口。

# 总结

本次主要讲解了**mapreduce**的基本定义以及如何编写、运行一个**mapreduce**开发程序。最后介绍了开发过程中用到的一些常见接口。

# 练习

1.以下哪个是MapReduce的特点?

A.易于编程　　B.良好扩展性

C.实时计算　　D.高容错性

2.MapReduce程序的资源由哪个组件调度?

A.HDFS　　B.Yarn

C.Hbase　　C.Hive

# 练习

**3.**编写**Mapreduce**二次开发程序时，经常要实现**Inputformat**类，那么此类一般包含哪些功能?

**4.**一个**Mapreduce**程序从**input**到**output**包含哪些过程?

# Thank you

www.huawei.com