Hadoop相关介绍：

**Mapreduce**：大规模数据集的并行运算。是一种编程模型，用于大规模数据集（大于1TB）的并行运算。概念"Map（映射）"和"Reduce（归约）"，是它们的主要思想，都是从函数式编程语言里借来的，还有从矢量编程语言里借来的特性。它极大地方便了编程人员在不会分布式并行编程的情况下，将自己的程序运行在分布式系统上。 当前的软件实现是指定一个Map（映射）函数，用来把一组键值对映射成一组新的键值对，指定并发的Reduce（归约）函数，用来保证所有映射的键值对中的每一个共享相同的键组。

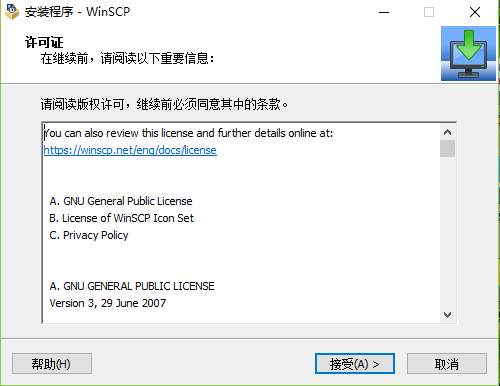
在hadoop中用于执行mapreduce任务的机器角色有两个，一个是jobtracker；另一个是tasktracker。jobtracker是用来调度工作的，tasktracker是用来执行工作的。一个hadoop集群中只有一台jobtracker。

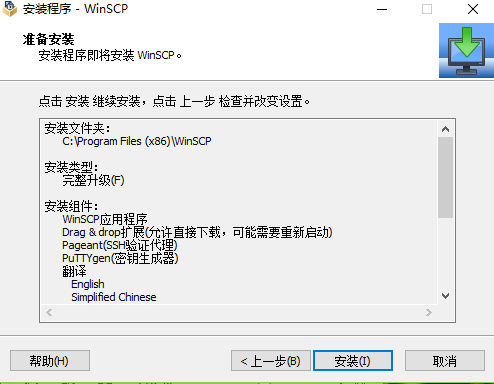
**HDFS**：分布式文件系统（Hadoop Distributed File System），HDFS是一个高度容错性的系统，适合部署在廉价的机器上。HDFS能提供高吞吐量的数据访问，非常适合大规模数据集上的应用。一个HDFS集群是由一个namenode和若干个datanode组成。其中namenode作为主服务器，管理文件系统的命名空间和客户端对文件的访问操作；集群中的DataNode管理存储的数据。HDFS允许用户以文件的形式存储数据。从内部来看，文件被分成若干个数据块，而且这若干个数据块存放在一组DataNode上。Namenode执行文件系统的命名空间操作，比如打开、关闭、重命名文件或目录等，它也负责数据块到具体DataNode的映射。DataNode负责处理文件系统客户端的文件读写请求，并在namenode的统一调度下进行数据块的创建、删除和复制工作。

**Yarn**：另一种资源协调者（Yet Another Resource Negotiator），是一种新的 Hadoop 资源管理器，它是一个通用资源管理系统，可为上层应用提供统一的资源管理和调度，它的引入为集群在利用率、资源统一管理和数据共享等方面带来了巨大好处。YARN的基本思想是将JobTracker的两个主要功能（资源管理和作业调度/监控）分离，主要方法是创建一个全局的ResourceManager（RM）和若干个针对应用程序的ApplicationMaster（AM）。

WinScp安装：

双击.exe文件进行安装

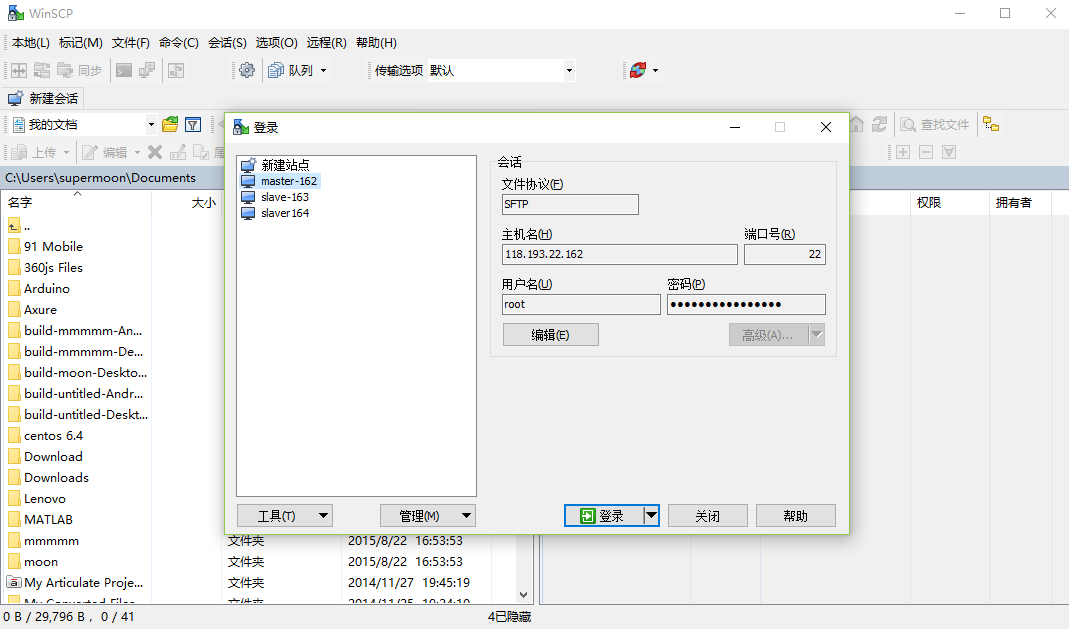






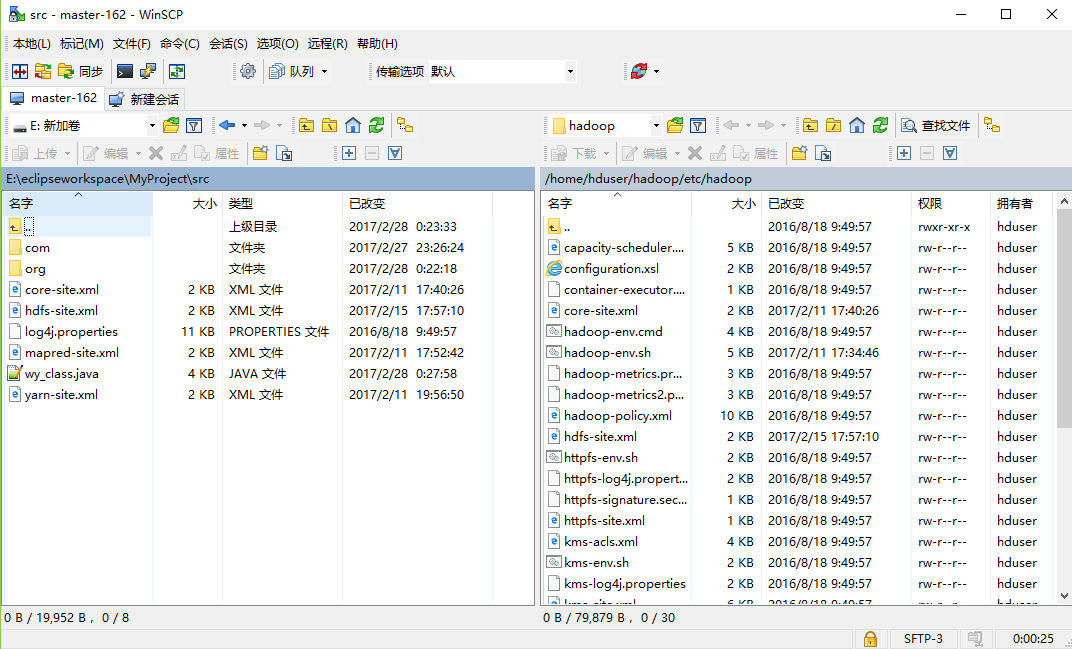
安装完成

进行相关配置，保存对话

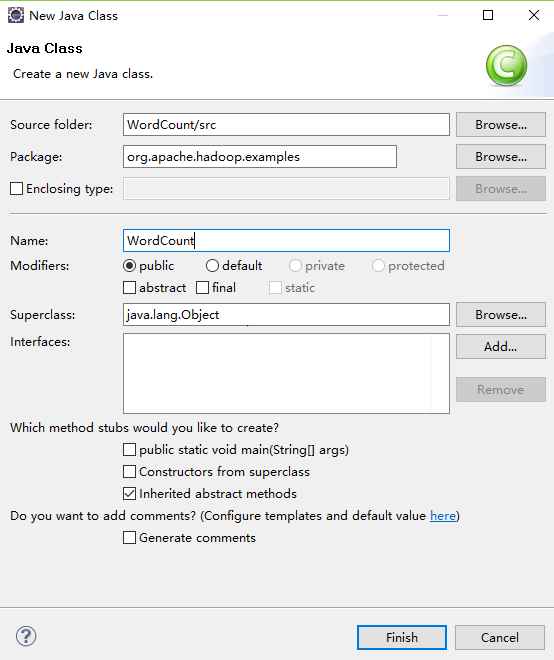


创建MapReduce项目并运行：

在运行 MapReduce 程序前，还需要执行一项重要操作（也就是上面提到的通过复制配置文件解决参数设置问题）：将 /usr/local/hadoop/etc/hadoop 中将有修改过的配置文件，以及 log4j.properties 复制到 WordCount 项目下的 src 文件夹



需要填写两个地方：在 Package 处填写 org.apache.hadoop.examples；在 Name 处填写 WordCount。



创建 Class 完成后，在 Project 的 src 中就能看到 WordCount.java 这个文件。将如下 WordCount 的代码复制到该文件中。

package org.apache.hadoop.examples;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

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

Configuration conf = new Configuration();

String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();

if (otherArgs.length != 2) {

System.err.println("Usage: wordcount <in> <out>");

System.exit(2);

}

Job job = new Job(conf, "word count");

job.setJarByClass(WordCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(otherArgs[0]));

FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

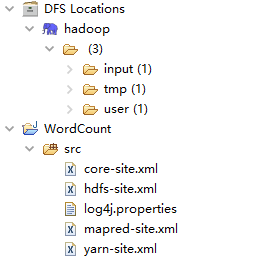
}

}

出现的问题：

Package出现错误，无法运行

复制完成后，务必右键点击 WordCount 选择 refresh 进行刷新（不会自动刷新，需要手动刷新），可以看到文件结构如下所示：

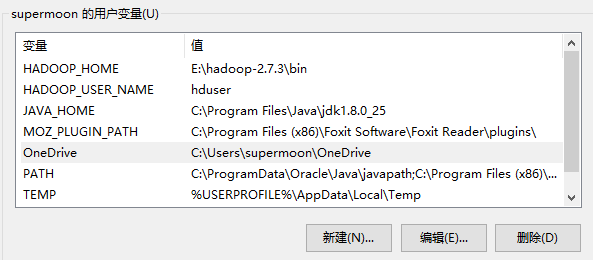


1.右击New->Map/Reduce Project

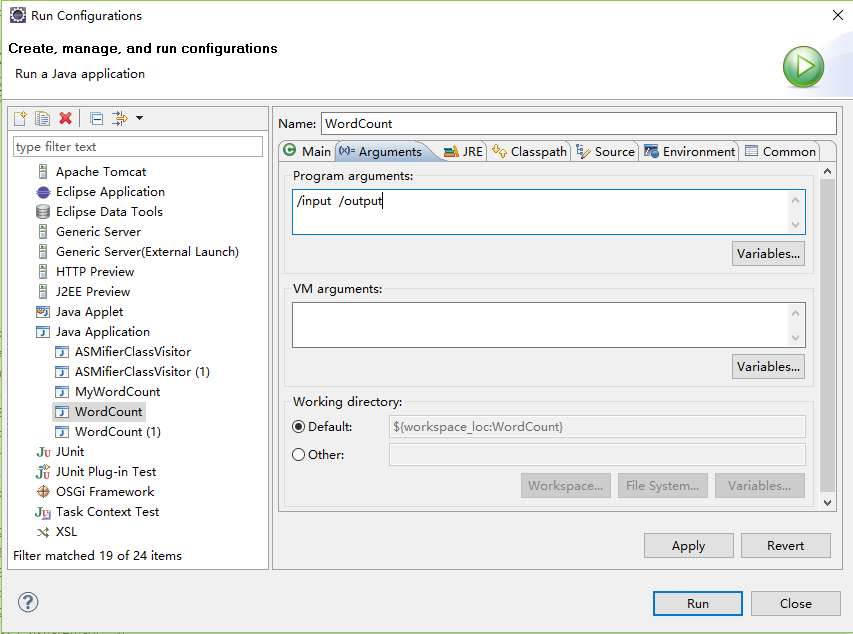
2.新建WordCount.java

3.在hdfs创建一个input目录（输出目录output可以不用创建，运行MR时会自动创建）

配置windows用户及系统环境变量：



设置运行配置



推荐算法：

随着网络的普及，用户上网的成本越来越低，上网时间越来越多，使得整个信息世界的信息越来越多，人们面对这海量的信息的时候就会难以找到自己所需要的东西，这就产生了信息过载问题。推荐系统的出现代表解决信息过载问题未来发展方向，根据海量的信息通过算法和规则来为用户推荐有用的信息，并给用户提供个性化的服务。在学术界推荐系统已经逐渐成了一门独立学科。

当下推荐算法的多种多样，虽然有的推荐引擎已经在商业上大获成功，有的还只是在实验室研究阶段。对于推荐引擎来说没有绝对的优秀的推荐引擎，每个推荐算法都有其优点以及缺点。本论文主要研究各种主流的推荐算法的优缺点，并在工程上设计推荐引擎组合。使用推荐引擎组合将所有最优组合结果推荐给用户。推荐引擎的设计需要区别离线计算以及在线设计部分，并需要单独的数据预处理部分。使用 Hadoop 平台运行推荐引擎可以解决大数据量的问题，并提供快速的计算方式。

在大型的电子商务网站亚马逊，阿里巴巴，社交网站 Facebook,新浪微薄，以及个性化音乐网站豆瓣，电影租凭网站 Netflix都大量使用了推荐系统。 推荐系统一般包括数据数据收集，推荐算法的计算和为用户推荐三个部分。数据收集一般是通过与用户的交互获得数据，比如用户购买了该物品，或者用户对物品进行了浏览以及对物品进行了评分，点赞，收藏等动作。通常这些数据也可以从网站的日志中获得，不过更为快捷的方式就是直接收集这些数据，存储到指定的地方，这样方便推荐系统使用这些数据。接着推荐系统对收集到的数据结合商品数据对数据进行整理，计算出用户的行为模式，或者计算出用户以及物品的相似度。通过上面求得到的相似度对用户进行物品推荐。

所谓推荐算法就是利用用户的一些行为，通过一些数学算法，推测出用户可能喜欢的东西。推荐系统一般分为一下几种:

1. 协同过滤推荐：

协同过滤的基本思想就是用户获取的喜好会影响到他未来的喜好。而这种相似性可以用来作为推荐的依据。

1. 基于内容的推荐

推荐系统被看做是用于解决从目标大集合中选择最有意义的数据，因此推荐系统也植根于信息检索和信息过滤领域。这些领域就是强调区分相关和不相关的文档。基于内容的推荐的核心就是物品的描述以及特征，并根据这些特征与用户的喜好，特点进行匹配。从而向用户进行推荐。

该算法根据用户过去喜欢的物品，为用户推荐和他过去喜欢的产品相似的产品。例如，一个推荐书籍的系统可以依据某个用户之前喜欢很多的哲学的书而为他推荐亚里士多德的书。CB最早主要是应用在信息检索系统当中，所以很多信息检索及信息过滤里的方法都能用于CB中。

CB 的过程一般包括以下三步：

1. Item Representation：为每个item抽取出一些特征来表示此 item；

2. Profile Learning：利用一个用户过去喜欢的 item 的特征数据，来学习出此用户的喜好特征；

3. Recommendation Generation：通过比较上一步得到的用户 profile 与候选item的特征，为此用户推荐一组相关性最大的item。

基于内容过滤的推荐算法可以保证用户之间相对的独立性，也可以生成很好的推荐理由。但是物品的特征比较难抽取，因为依靠的是文档，同时这个算法也无法挖掘用户的潜在兴趣。



图 基于内容过滤概述图

3、基于知识的推荐

在消费类电子产品中，涉及大量的单次购买者，这意味着无法以来基于用户过去的记录来为用户做推荐。根据使用户知识以及产品信息，通过算法推导出什么产品能满足用户的需求，从而作出推荐，这种推荐系统不依赖用户的评分也不依赖用户的历史数据。所以也不存在冷启动的问题。

4、混合推荐方法

就是以上几种方法的混合使用，以上的推荐系统各有利弊，混合的推荐系统就是综合各种情况进行整合，对用户进行推荐，特别是大型的电子商务网站，就需要各种推荐算法为商品推荐进行服务。

大数据处理：

我们现在处于一个大数据时代，推荐系统面对的是海量的用户数据，海量的商品数据，以及每天不断产生的新的网站日志，用户行为记录，新的内容等等。如何存储以及处理这些庞大的数据也是一个棘手的问题。使用传统的 RDBMS 对于大数据量的读写都一个瓶颈，而且传统的数据库并不适合进行分布式存储，PB 级的数据量需要均匀的存储到不同的节点，同时还需要一个并行计算的框架来处理这些数据。Hadoop 是这方面的理想选择，从 2004 年发展至今，Hadoop 在大数据处理方面的应用及其广泛。Hadoop提供一个分布式文件系统 HDFS，以及并行编程模型 MapReduce，在这个文件系统上可以轻松的完成分布式计算。