基于 hadoop 的热传导推荐算法的设计与实现

组员：201820000135 许甜甜
201920000153 徐　园

目录

1. 绪论

1.1. 研究背景

随着互联网的发展，全球的网页服务器数量不断增长，网页的数量也相应地呈爆炸性增长趋势。互联网的发展把我们带入了一个新的世界，我们可以在互联网上读文字、听音乐、看电影和购物等，十分方便，但也让我们在海量的数据面前不知所措。用户很难判断信息的真实价值，可能使真正有价值的信息由于缺乏辅助工具而被埋没，这种现象称为信息超载。为了有效地解决上面的一系列问题，人们研发了搜索引擎，如 Google、百度等，使用网络爬虫收集网页信息，并对其进行筛选、分析和处理，为用户提供服务。但是这仅仅是基于用户的输入而进行的反馈，所以反馈的信息都限制在用户已知的信息范围中，而不能帮助用户找到其他不知道但是很有价值的内容。推荐系统可以从一定程度上弥补传统的搜索引擎的缺陷，分析用户的历史操作，建立用户偏好模型，依靠算法计算用户对未知产品的偏好权重，并根据权重对商品排序，向用户推送一个可能令其感兴趣的作品推荐列表。Hadoop 主要由两大核心模块组成：分布式文件系统 HDFS 和分布式批处理框架 MapReduce 两部分。 HDFS 为我们提供了包括大规模文件存储系统以及对应可靠的备份管理机制的高容错性、高吞吐量的海量数据存储解决方案，而 MapReduce 编程模型为开发人员提供便利的分布式应用开发接口，如通信、同步、 调度计算、 负载均衡、 处理机器异常等分布式计算和分布式系统常常遇到的复杂难题则交由 Hadoop 平台解决，对于上层开发人员来说是透明的。网络推荐算法是最近几年兴起的个性化推荐算法，算法将用户和产品的内容特征不予考虑，而仅仅抽象为节点，发掘隐藏在用户和产品中的选择关系，有效地避免了协同过滤算法中遇到的信息挖掘技术的限制及评分稀疏性和算法可扩展性的问题。

1.2. 主流的推荐算法

推荐引擎的物理理论探索历史可以追溯到上世纪 90 年代，横跨了各个学科的各个领域，优秀的推荐服务依赖于科学先进的推荐算法和大规模的学习数据。一个完善的推荐引擎由历史行为记录模块、分析建模模块和推荐算法模块三个部分组成。

历史行为模块主要是通过网站历史日志的挖掘，来收集用户的偏好信息，推荐系统可以根据偏好信息预测该用户对其他没有选择过的商品的个人观点。分析建模模块主要通过分析用户行为记录，在用户未做出明确行为的情况下，建立模型描述用户的喜好产品和喜好程度等喜好信息，估计用户对某种商品的打分。推荐算法是整个推荐系统核心和关键，模块利用运行在后台的推荐算法，通过一系列预测计算，及时地从大量的候选商品对象中选择出用户可能喜欢推送推荐结果。

目前推荐算法主要包括基于内容的推荐算法、协同过滤算法、混合推荐算法以及最近兴起的网络推荐算法。四种算法的比较如下表所示：

表 1 各种推荐算法的比较

推荐算法	优点	缺点
基于内容	结果容易被用户理解， 没有新商品的冷启动问题， 没有矩阵稀疏性问题， 覆盖率高， 可以推荐新的但不流行的商品， 有成熟的分类学习算法技术支持	无法处理图像、声音多媒体数据， 计算时间复杂度高， 过度特殊化问题， 存在新用户的冷启动问题， 新颖性不足， 可扩展性问题
协同过滤	不需要专业知识和人工干预， 算法随着人数增加性能提高， 发现用户的潜在兴趣点， 可以处理复杂的非结构化数据	新用户、商品加入的冷启动问题， 稀疏性影响推荐质量， 强依赖于数据集本身的属性， 可扩展性问题
混合推荐	比独立算法具有更好的性能	复杂度更高， 混合比例的阈值不易控制
网络推荐算法	不受信息挖掘技术的制约，可以解决矩阵稀疏性问题，准确性高	冷启动问题， 可扩展性问题

1.3. 课题的主要内容

本课题的研究内容主要包括两个，hadoop 分布式平台搭建和网络推荐算法在 hadoop 平台上的 MapReduce 分布式实现。网络推荐算法主要是基于热传导推荐算法。

2. Hadoop 分布式平台

2.1. 简介

Hadoop 是一个得到 Apache v2 认证的支持数据密集型分布式计算的开源软件，支持在大规模商用硬件集群上运行应用，并透明地提供可信性和数据移动。Hadoop 使用简易的编程模型来分布式处理分布在集群机器上的大规模数据集，通过部署廉价的机器集群可以与任何高性能的计算机并驾齐驱。

Hadoop 最核心的是：运行在大型商业机器集群上的分布式文件系统 Hadoop Distributed File System(HDFS)和分布式处理框架和并行运行环境 Hadoop MapReduce。HDFS 能够运行在大规模通用硬件集群上，可靠地存储超大规模的数据文件，而且可以被大量用户高带宽的访问，具有动态无缝扩容的高可扩展性、数据自动检测和备份的高可靠性、高吞吐访问量等特点。集群由数台到数万台低

成本的商用机器组成，每个节点提供本地计算能力和存储单元，利用高效的分布式算法进行整合，从而提供应对甚至 BP 级的数据计算和存储能力。

Hdfs 主要由一个管理者节点 NameNode 和多个工作者 DataNode 组成，分别存储元数据和应用数据，所有节点都通过 TCP 协议族完全连接和互相通信。HDFS 的 DataNode 节点不采用 RAID 数据保护机制，而是通过文件备份来保证数据的可靠性。NameNode 是 HDFS 的关键组件，是中心管理服务器，也是 HDFS 元数据的仲裁者和管理者，主要提供内部元数据服务，负责管理文件系统名称空间、集群配置信息、存储块的复制和外部客户机的访问，NameNode 的失效会导致整个文件系统无法使用。DataNode 是存储块的实际提供者，也是文件存储器的基本单元，受 NameNode 或者客户端调度根据需求对 DataNode 上存储的数据块进行操作，并且保存文件块的元数据，定期向 NameNode 发送自身存储的块列表。HDFS 架构如图 1 所示：

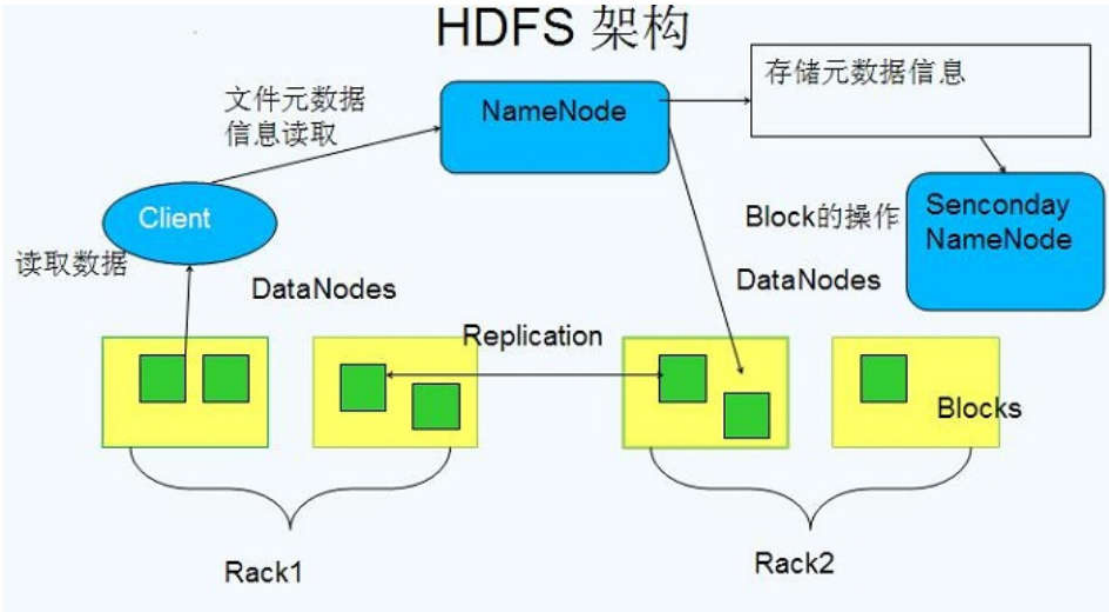


图 1 HDFS 架构图

MapReduce 最早是由 Google 公司研究提出的一种面向大规模数据处理的并行计算模型和方法。MapReduce 分为 Map（映射）和 Reduce（化简）两个严格但有效的编程阶段，是一种典型的分而治之的思想，并且采用重新执行作为提高容错性的主要机制。MapReduce 原理如图 2 所示：

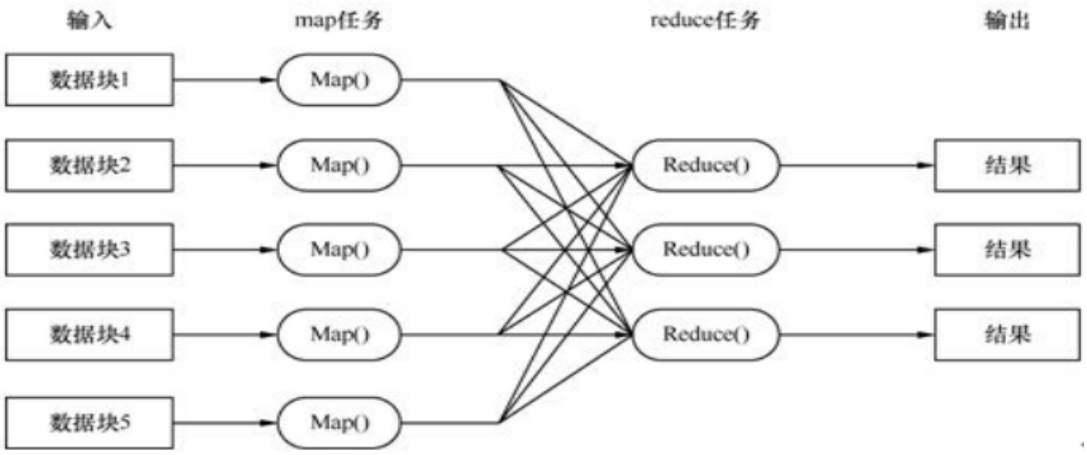


图 2 MapReduce 原理图

输入块在不同的机器上并行，将大的作业任务通过 MapReduce 中的 Splitter 切分为很多小的任务，然后分配到集群的各个工作节点上并行执行，每个任务会被一个 Mapper 处理，节点利用 MapReduce 提供的 RecordReader 从数据块中提取 key/value 形式的键值对，键值对传入用户自定义的 map 函数，map 函数生成的中间 key-value 对序列化缓存在内存中。Map 操作是无状态的，每次只处理一个 key-value 对，这样可以被很容易地映射到多个节点上进行并行计算。Reduce 过程其实是对 map 的输出结构的汇总，遍历排序后的中间 key-value 对，对相同 key 的 value 进行聚合以形成较小的 value 集合，reduce 处理后的结果添加到分区对应的输出文件中。当所有作业都完成时，master 唤醒用户程序，MapReduce 输出存储在 reduce 作业对应的输出文件中，每个输出文件对应一个 Reduce 任务，而且 Reduce 结果可能作为另一个 MapReduce 调用的输入。MapReduce 作业运行原理图如图 3 所示：

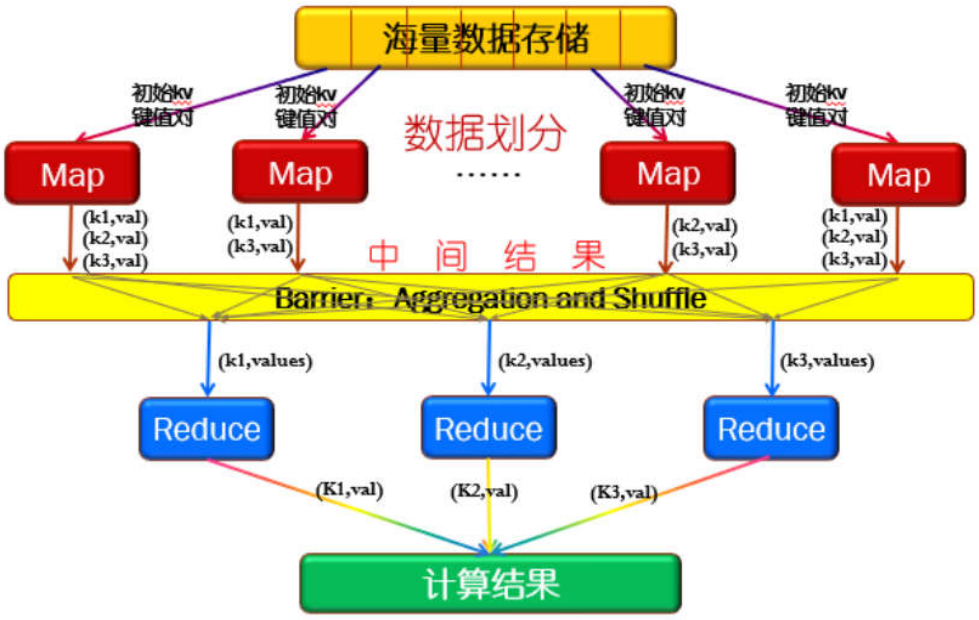


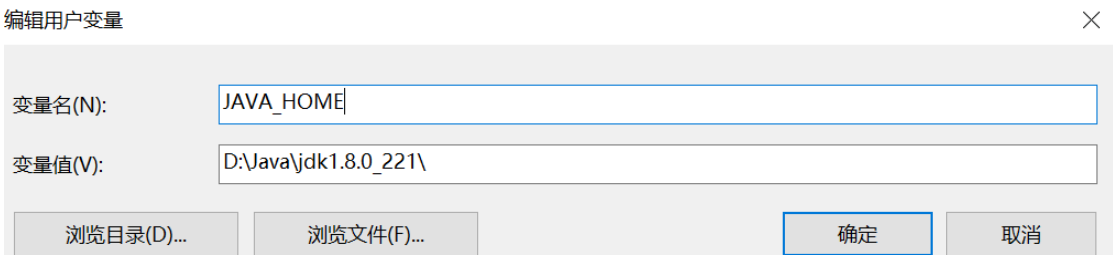
图 3 MapReduce 作业运行原理图

2.2. Hadoop 环境搭建

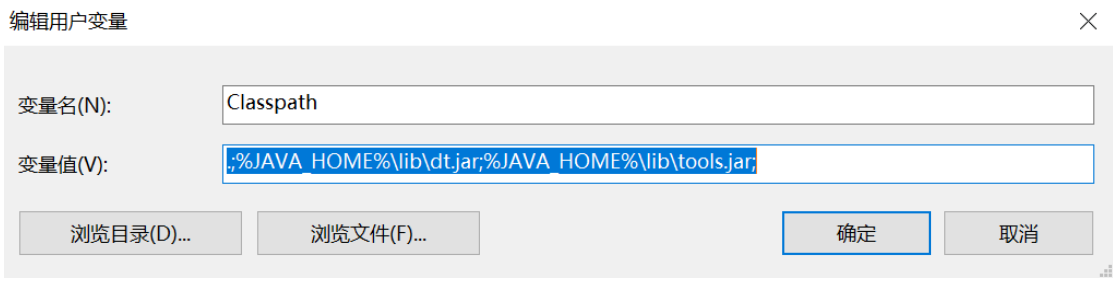
2.2.1. 安装 jdk

双击下载好的 jdk 安装包，进行应用程序的安装。然后配置 jdk 环境变量。

A. 配置 JAVA_HOME，JAVA_HOME 变量值指向 JDK 安装的文件夹。



B. 配置 Classpath。



C. 配置 Path 变量，在 Windows10 系统中直接点击新建按钮，输入%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin 点击确认按钮直到再次回到控制面板。

```
%JAVA_HOME%\bin
%JAVA_HOME%\jre\bin
```

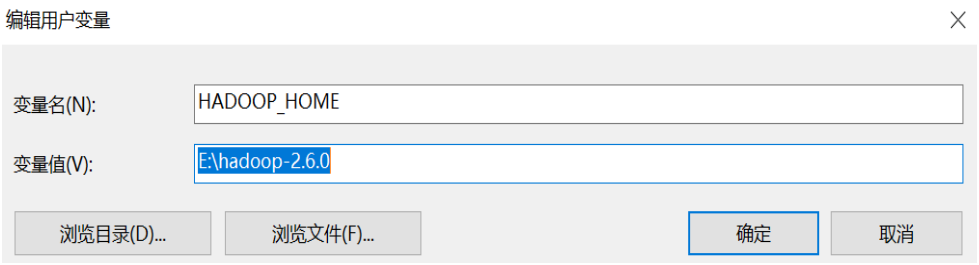
D. 配置完成后，打开 cmd 软件，输入命令 Java -version 如果没有提示错误，则 jdk 安装成功。如下图所示：

```
C:\Users\Lenovo>Java -version
java version "1.8.0_221"
Java(TM) SE Runtime Environment (build 1.8.0_221-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.221-b11, mixed mode)
```

2.2.2. 安装 hadoop

Hadoop 有三种安装模式，单机模式，伪分布式模式和分布式。因为我只有一台电脑，所以采用的是伪分布式模式，伪分布式是在一台电脑上模拟多台电脑进行分布式 MapReduce 操作。具体步骤如下：

1. 将从官网下载好的 hadoop 压缩包解压到磁盘中，我下载的版本是 hadoop2.6.0，解压目录是 E 盘根目录。
2. 配置 hadoop 环境变量。



3. Path 环境下配置【%HADOOP_HOME%\bin;】变量。
4. 配置好环境变量后，打开命令行，输入 hadoop version，显示如下则说明基本环境已经配置完成完成。

```
C:\Users\Lenovo>hadoop version
Hadoop 2.6.0
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r e3496499ecb8d220fba99dc5ed4c99c8f9e33bb1
Compiled by jenkins on 2014-11-13T21:10Z
Compiled with protoc 2.5.0
From source with checksum 18e43357c8f927c0695f1e9522859d6a
This command was run using /E:/hadoop-2.6.0/share/hadoop/common/hadoop-common-2.6.0.jar
```

5. Hadoop 基本文件配置：hadoop 配置文件位于 etc/hadoop 目录下，修改以下四个文件。

(1) 修改 core-site.xml 文件。

```
<!-- Put site-specific property overrides in this file. -->

<configuration>
  <!--指定namenode地址-->
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

(2) 修改 hdfs-site.xml 文件。

```
<configuration>
  <!--指定hdfs保存数据的副本数量-->
  <property>
        <name>dfs.replication</name>
        <value>1</value>
  </property>
  <!--指定hdfs的权限-->
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
  <!--指定hdfs中namenode的存储位置-->
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>/e:/hadoop-2.6.0/data/dfs/namenode</value>

    </property>
  <!--指定hdfs中datanode的存储位置-->
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>/e:/hadoop-2.6.0/data/dfs/datanode</value>
    </property>
</configuration>
```

(3) 修改 mapred-site.xml 文件。

```
<configuration>
  <!--告诉hadoop用户MapReduce运行在YARN上-->
  <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>

</configuration>
```

(4) 修改 yarn-site.xml 文件。

```
<configuration>
<!-- Site specific YARN configuration properties -->
  <!--namenodeManager获取数据的方式是shuffle-->
  <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
</configuration>
```

6. 格式化文件系统，在 hadoop/bin 下执行 hdfs namenode -format 命令。在 sbin 目录下双击 start all 命令脚本。

7. 在命令行中输入 jps，显示如下则说明 hadoop 环境已搭建完成。

```
C:\Users\Lenovo>jps
8132 NameNode
12188 Jps
1388 ResourceManager
2572 NodeManager
9004 DataNode
```

3. 推荐系统

3.1. 简介

在传统的搜索引擎中，用户需要明确自己的具体需求或需求范围的关键字，通过关键字进行内容的搜索。但是当用户对自己的需求不是很明确，或者无法用关键字进行概括和表达时，推荐系统就产生了。推荐系统根据商品信息、用户信息、用户对商品的喜好、商品之间的联系等信息，推测用户对商品的喜好程度，以此为根据向用户进行可能更感兴趣内容的推荐，帮助用户在海量的信息中进行信息过滤。

网络推荐算法主要是基于用户历史日志建立的用户-商品二分网络，不考虑用户和商品的内容特征，而是对用户和商品进行抽象简化为节点，算法关注用户和商品之间已经存在的选择关系，有效地避免了协同过滤算法中遇到的信息挖掘技术的限制及评分稀疏性和算法可扩展性的问题，最适合没有显示打分而只有用

户商品间选择关系的数据，显示评分的数据很容易通过分段处理的方式映射到二分网络中，例如 movielens 数据集采用 1-5 分评分形式，则可以认为大于等于 3 分则有选择关系，小于则没有。二分图是一个具有广泛意义的网络模型，将所有节点分为 X 和 Y 两个集合，只有不同集合之间允许有关系，也就是右边相连，相同集合之间不存在直接联系，而是通过另一个集合而产生间接关联。热传导推荐算法就是基于二分图相互映射关系，最后给用户推荐有价值的作品。

热传导算法通过最近邻居平均分配过程重新分配资源，每个用户节点的温度等于所有该用户选择过的商品节点温度的平均值，然后每个商品节点的温度等于所有选择过该商品的用户的平均温度的平均值，两个步骤的加和被称为商品到商品的热传导的一步，热传导算法过程如图 4 所示：

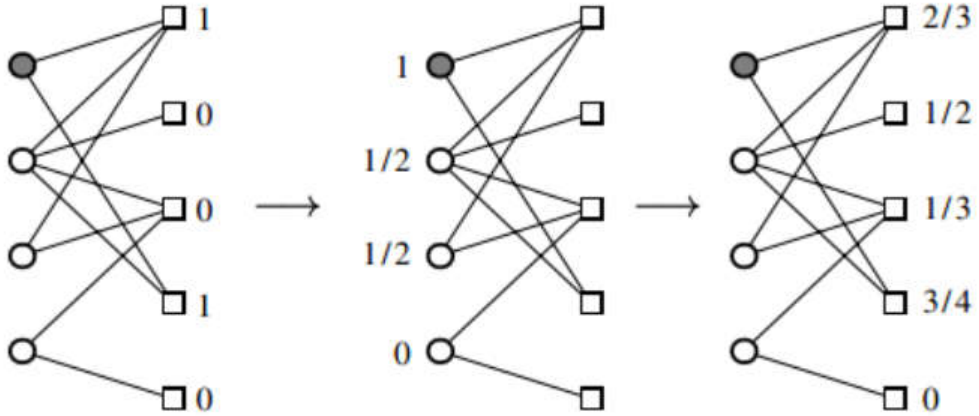
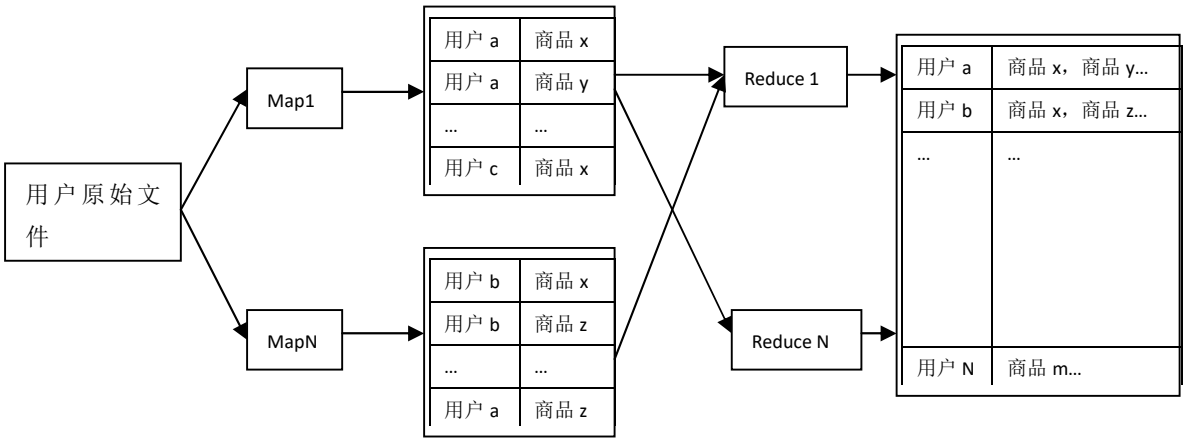


图 4 热传导算法过程

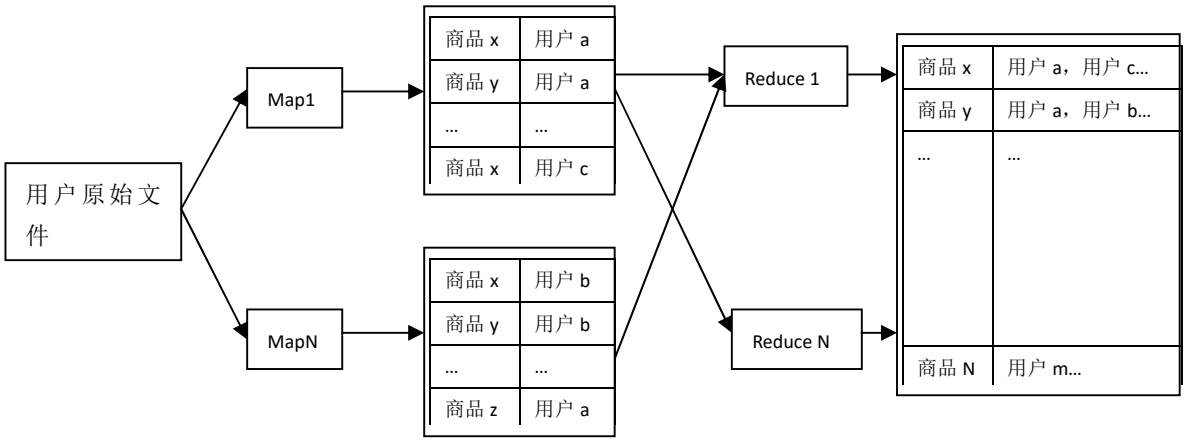
3.2. 算法的 MapReduce 并行化

热传导算法的计算过程不能简单地使用 MapReduce 模型进行分解，在这里我简要地把整个计算任务分解为 6 个 MapReduce 作业，每个作业包括一个自定义的 map 函数和一个自定义的 reduce 函数，输入是用户的历史日志记录，输出是用户的推荐列表，每个作业完成算法的一小部分，多个 MapReduce 会依次按照步骤执行，最终计算得到推荐结果。

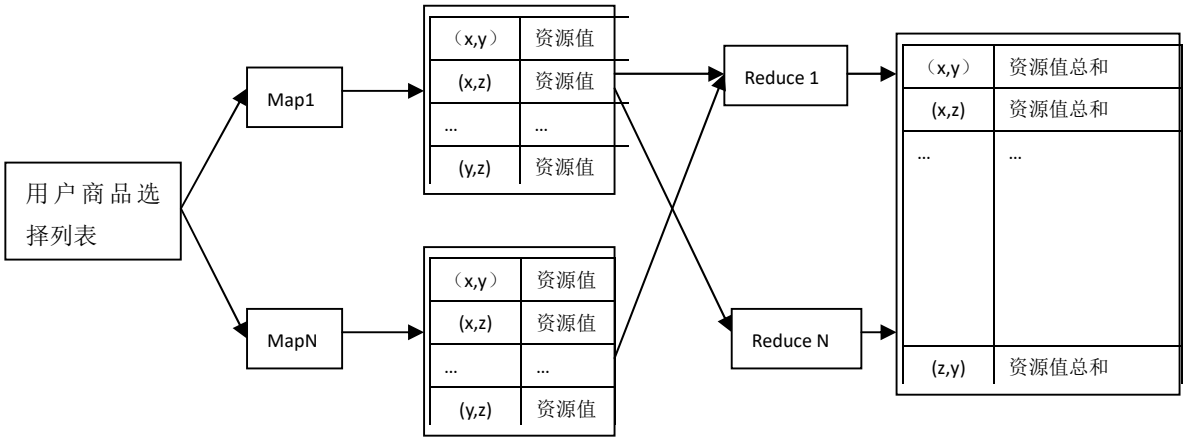
第一步 MapReduce 作业从用户的历史日志文件中选择用户和商品生成每一个用户的商品消费列表。在 Map 阶段，处理过程类似于用户日志的过滤器，输入为用户的历史日志文件，map 函数的输出形式为<用户 ID，商品 ID>形式的键值对，所有的 Mapper 得到的中间文件将会传输到 Reduce 阶段进行加和，Reduce 函数会根据键值对为每个用户生成用户-商品列表{用户 ID {商品 1,商品 2,… ，商品 N}}。过程如下图所示：



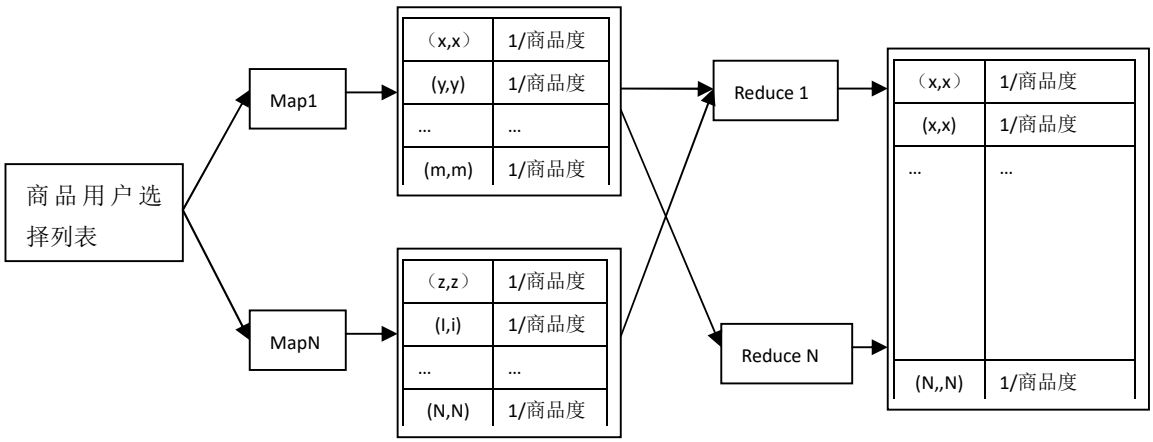
第二步 MapReduce 作业计算每个商品的度列表，商品度在二分图网络结构中可以认为是连接商品节点的边的总数。在 Map 阶段，输入仍然是原始的用户历史日志文件，输出为<商品 ID，用户 ID>键值对。在 Reduce 阶段，对所有的 key 相同的商品 ID 的用户 ID 进行加和，最后输出为每个商品的商品-用户列表{商品 ID，{用户 1，用户 2，… ，用户 N}}.过程如下图所示：



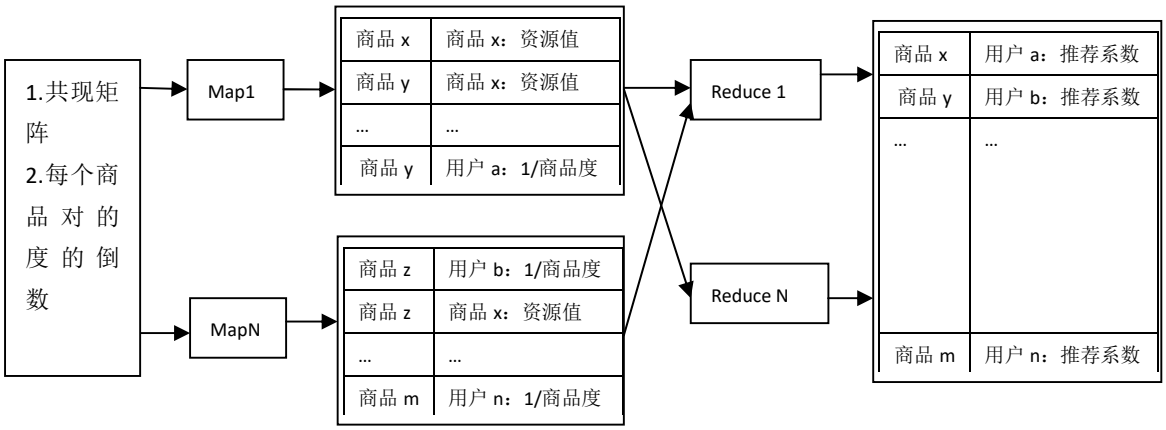
第三步 MapReduce 作业是整个过程的核心和关键，主要目的是计算资源分配矩阵 W。由于二分图中只有两个商品节点有共同用户的选择，才会有商品间的推荐资源流动，所以计算 W 必须先对商品间共现的次数进行计数。例如，假设用户 u 选择过商品 x，y 和 z，u 只对（x，y），（x，z）和（y，z）之间的资源分配过程做出贡献，像（x，y）这样的二元组被称为具有共现关系的商品对。对所有的商品 i 和 j，统计商品对的出现次数，得到 i 和 j 的共现次数。在 Map 阶段，通过第一步的 MapReduce 作业的输出中得到的用户商品选择列表，计算所有商品对的共现次数，并除以用户的度，得到稀释后的部分资源图，由于结果只会返回非零元素，所有有效地解决了矩阵稀疏性。输出的 key 为具有共现关系的商品对，value 为 key 的稀释后的资源分布的非零值。在 Reduce 阶段，将相同 key 的 value 值进行相加，得到商品的资源共现矩阵。过程如下所示：



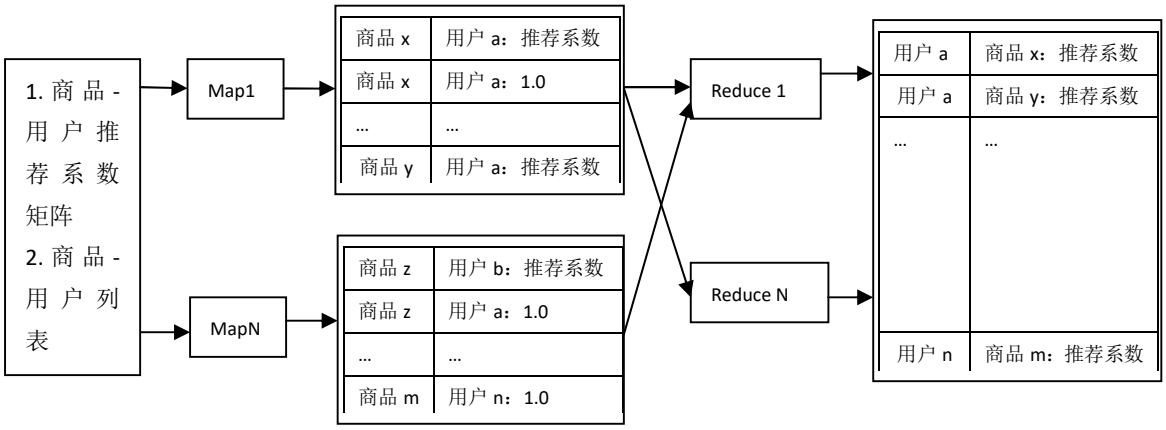
第四步 MapReduce 作业主要计算每个商品度的倒数，结果为每个商品节点度的倒数，以便后面计算矩阵相乘时使用。在 Map 阶段，输入是第二步输出的商品-用户度列表，输出是<（商品 i，商品 i），1/商品度>，在 Reduce 阶段归纳 Map 阶段的输出。专门把这个步骤单独分解成一个 MapReduce 作业步骤，方便下一步运算。过程如下所示：



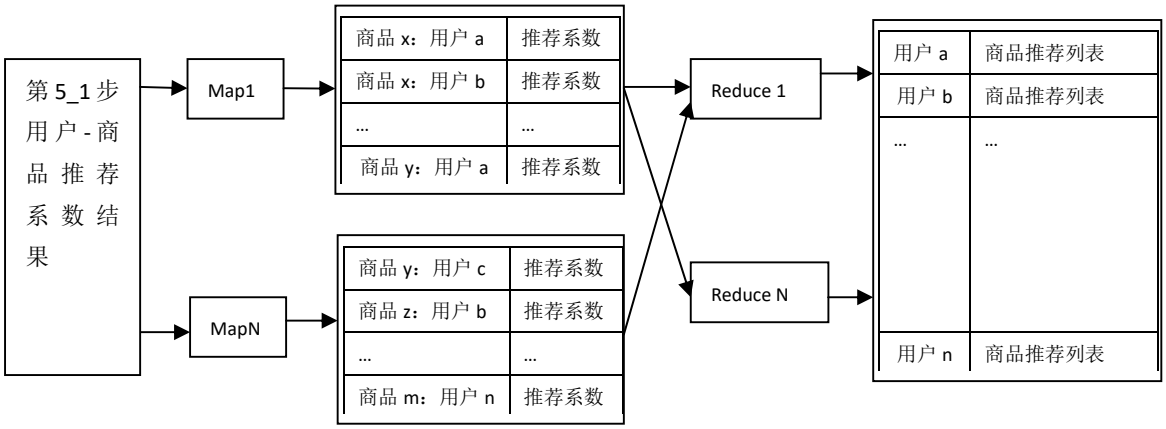
第 5 步 MapReduce 作业的任务是计算每个商品的温度，结果为每个商品对对每个用户的资源分配系数，方便下一步进行矩阵乘法，计算出最终的推荐列表。输入是第三步输出的共现矩阵和第四步的输出的每个商品的度的倒数。在 Map 阶段，分别划分两个矩阵，将它们分别设置为{商品 ID，{商品 ID，资源值}}和{商品 ID，{用户 ID，1/商品度}}，方便 Reduce 阶段进行矩阵乘积。在 Reduce 阶段，对 Map 阶段设置好格式的两个矩阵进行乘法运算，最终得到{商品 ID，{用户，推荐系数}}。过程如下所示：



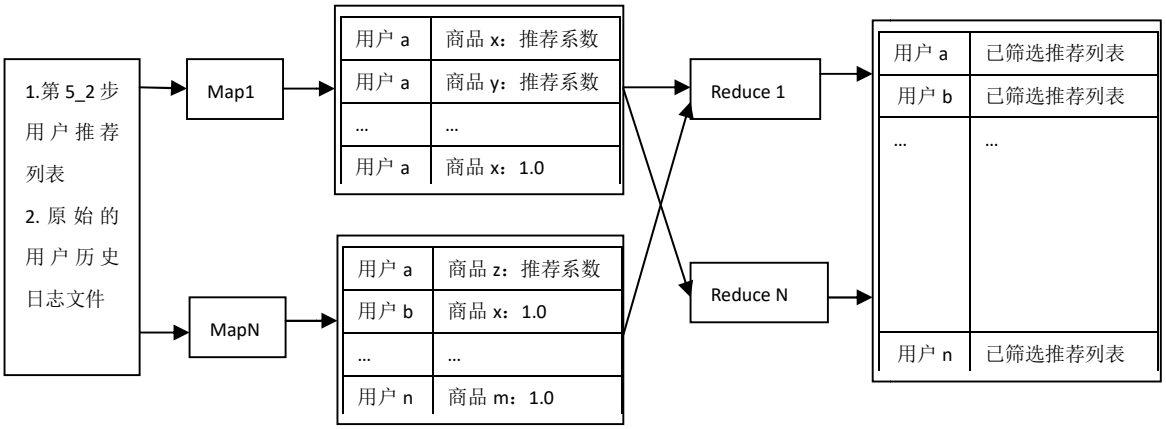
第 5_1 步 MapReduce 作业的主要目标是将第五步的输出与第二步获得的商品-用户列表相乘，以便获得每个商品对每一个特定用户的推荐系数。相乘的原因是每个消费过该商品的用户都会对推荐算法有所贡献。步骤如下所示：



第 5_2 步 MapReduce 作业是把第 5_1 步的结果重新整合，然后相加，最终得到每个用户对每个商品的推荐结果。



第六步 MapReduce 目标是将 5_2 步的用户-商品推荐列表的结果进行过滤和排序，过滤是把用户已经选择过的商品筛选掉，排序是按照推荐系数从大到小依次排序，得到最后的推荐结果。



3.3. 结果

1. 原始用户历史日志文件，第一列为用户 ID，第二列为商品 ID，第三列为用户商品是否选择或评价关系，选择为 1，未选择为 0。

```
1,1,1
1,4,1
2,1,1
2,2,1
2,3,1
2,4,1
3,1,1
3,3,1
4,3,1
4,5,1
```

2. 第 1 步 MapReduce 结果，第一列为用户 ID，第二列为商品和选择关系列表。

```
hdfs://localhost:9000/test/step1/part-r-00000
1       4:1,1:1
2       4:1,3:1,2:1,1:1
3       3:1,1:1
4       5:1,3:1
```

3. 第 2 步 MapReduce 结果，第一列为商品 ID，第二列为用户和选择关系列表。

```
hdfs://localhost:9000/test/step2/part-r-00000
1       3,2,1
2       2
3       4,3,2
4       2,1
5       4
```

4. 第 3 步 MapReduce 结果，第一列为商品共现关系，第二列为商品对的资源值。

```
hdfs://localhost:9000/test/step3/part-r-00000
1:1     1.25
1:2     0.25
1:3     0.75
1:4     0.75
2:1     0.25
2:2     0.25
2:3     0.25
2:4     0.25
3:1     0.75
3:2     0.25
3:3     1.25
3:4     0.25
3:5     0.5
4:1     0.75
4:2     0.25
4:3     0.25
4:4     0.75
5:3     0.5
5:5     0.5
```

5. 第 4 步 MapReduce 结果，第一列为每个商品对，第二列为 1/用户度。

```
hdfs://localhost:9000/test/step4/part-r-00000
1:1     0.3333333333333333
2:2     1.0
3:3     0.3333333333333333
4:4     0.5
5:5     1.0
```

6. 第 5 步 MapReduce 结果，第一列为商品 ID，第二列为用户，推荐系数值。

```
hdfs://localhost:9000/test/step5/part-r-00000
1       1,0.42
1       2,0.08
1       3,0.25
1       4,0.25
2       1,0.25
2       2,0.25
2       3,0.25
2       4,0.25
3       1,0.25
3       2,0.08
3       3,0.42
3       4,0.08
3       5,0.17
4       1,0.38
4       2,0.12
4       3,0.12
4       4,0.38
5       3,0.50
5       5,0.50
```

7. 第 5_1 步 MapReduce 结果，第一列为商品 ID，用户 ID，第二列为推荐系数值。

```
hdfs://localhost:9000/test/step5_1/part-r-00000
1,1     0.42
2,1     0.42
3,1     0.42
1,2     0.25
2,2     0.25
3,2     0.25
1,3     0.25
2,3     0.25
3,3     0.25
1,4     0.38
2,4     0.38
3,4     0.38
2,1     0.08
2,2     0.25
2,3     0.08
2,4     0.12
2,1     0.25
3,1     0.25
4,1     0.25
2,2     0.25
3,2     0.25
4,2     0.25
2,3     0.42
3,3     0.42
4,3     0.42
2,4     0.12
3,4     0.12
4,4     0.12
2,5     0.50
3,5     0.50
```

8. 第 5_2 步 MapReduce 结果，第一步为用户 ID，第二列为商品 ID，推荐系数总和。

```
hdfs://localhost:9000/test/step5_2/part-r-00000
1       1,0.6699999999999999
1       2,0.5
1       3,0.33
1       4,0.76
2       1,1.0
2       2,1.0
2       3,0.8299999999999998
2       4,1.0
2       5,0.5
3       1,0.6699999999999999
3       2,0.5
3       3,0.6699999999999999
3       4,0.5
3       5,0.5
4       1,0.25
4       2,0.25
4       3,0.59
4       4,0.12
4       5,1.0
```

9. 第 6 步 MapReduce 结果，第一列为用户 ID，第二列为在用户历史日志文件中筛选过后的商品 ID 和推荐结果。

```
hdfs://localhost:9000/test/step6/part-r-00000
1	2,0.5
1	3,0.33
2	5,0.5
3	2,0.5
3	4,0.5
3	5,0.5
4	1,0.25
4	2,0.25
4	4,0.12
```

4. 总结

本课题详细地分析和研究了 Hadoop 两大组成元件 HDFS 和 MapReduce 的运行机制和编程原理，介绍了推荐系统的概要流程和常见的推荐算法，接着介绍热扩散推荐算法，设计并实现了算法的 MapReduce 分布式实现，以应对海量日志数据。在本次课题中，我了解了 Hadoop 的具体工作流程，知道如何把一个针对大量数据的算法进行 MapReduce 并行化处理，这对于我来说是很有意义的一次作业。