社交网络好友推荐系统的实现

薛鹏泽[1)] 符静莹[1)] 刘亚鑫[1)] 黄楚瑶[1)]

[1)](澳门城市大学 数据科学学院, 澳门特别行政区 中国 999078)

摘 要 随着社交网络的快速发展，用户生成的数据量呈爆炸式增长，如何从海量数据中为用户推荐可能感兴趣的好友成为了一个重要问题。Hadoop 作为一个开源的分布式计算框架，以其高效性、高可靠性、高可扩展性以及价格低廉等优点，为大规模数据的处理提供了有力支持。本文研究了基于 Hadoop 的 MapReduce 模型社交网络好友推荐系统的实现，数据集来自斯坦福 CS246 课程，旨在通过利用 MapReduce 的分布式计算能力，在 Mac 系统中进行好友推荐。我们的代码及实验结果将在：https://github.com/xuepengze/Experiment/tree/main/PeopleYouMightKnow。

关键词 Hadoop；Map-Reduce；Stanford CS246；社交网络推荐系统；大规模数据处理；分布式计算

Implementation of Social Network Friend Recommendation System

XUE Peng-Ze[1)] FU Jing-Ying[1)] LIU Ya-Xin[1)] HUANG Chu-Yao[1)]

[1)](City University of Macau, Faculty of Data Science, MSAR 999078)

Abstract With the rapid development of social networks, the amount of user-generated data has exploded, making it a crucial issue to recommend potential friends to users from this massive dataset. Hadoop, as an open-source distributed computing framework, offers powerful support for large-scale data processing due to its efficiency, high reliability, scalability, and cost-effectiveness. This paper investigates the implementation of a social network friend recommendation system based on the MapReduce model in Hadoop. The dataset used is from the CS246 course at Stanford , aiming to leverage the distributed computing capabilities of MapReduce to perform friend recommendations within a Mac system. Our code experimental results will be available at：https://github.com/xuepengze/Experiment/tree/main/PeopleYouMightKnow.

Key words Hadoop;Map-Reduce;Stanford CS246;social network recommendation system; large-scale data processing; distributed computing

1 引言

1.1 研究背景及意义

1.1.1 研究背景

随着社交网络的迅猛发展，平台用户数量和交互行为呈现指数级增长，产生了海量的社交数据。这些数据蕴含着丰富的信息，为用户兴趣挖掘、关系推荐等领域提供了广泛的研究机会。好友推荐系统作为社交网络的核心功能之一，能够通过分析用户的关系网络，为其推荐潜在的好友，提升用户体验和社交网络的粘性。然而，随着数据规模的增加，传统的单机计算方式已难以满足大规模社交数据处理的需求。

为应对这一挑战，分布式计算技术得到了广泛应用。Hadoop 作为一种开源的分布式计算框架，其核心组件 MapReduce 能够将任务分解为多个并行作业，以高效处理大规模数据。基于此框架，可以

薛鹏泽(第一作者)，男，2002 年生，本科,主要研究领域为遥感图像变化检测、+853 66691102.E-mail: 2543213144@qq.com. 符静莹（第一作者），女，2002 年生，本科，主要研究领域为机器学习、+853 63768136、E-mail:1348855789@qq.com. 刘亚鑫（第一作者），女，2002 年生，本科,主要研究领域为多媒体信息安全、+853 63764257.E-mail: 1322908657@qq.com. 黄楚瑶（第一作者），女，2001 年生，本科,主要研究领域为机器学习、+853 62756792.E-mail: 1047503947@qq.com.

在合理的硬件资源下完成对海量社交网络数据的分析与挖掘。

1.1.2 研究意义

本实验基于 Hadoop MapReduce 框架，构建了一个社交网络好友推荐系统。通过分析用户的好友关系，系统生成每个用户的潜在好友推荐列表，并根据共同好友数量进行排序，体现了推荐的合理性与准确性。本实验的意义主要体现在以下几个方面：

1. 分布式计算的实践价值：通过实验，验证了分布式计算在处理大规模社交数据时的高效性与可扩展性，为类似的大数据处理任务提供了技术支持。

2. 社交网络推荐的现实意义：好友推荐系统能够帮助社交网络用户拓展社交关系，提升用户体验，同时为平台运营提供有效的用户粘性提升工具。

3. 大规模数据处理的算法研究：通过 Map-Reduce 的设计与实现，深入探讨了社交网络数据的处理方法，为进一步优化好友推荐算法奠定了基础。

4. 实验数据的学术参考：本实验所使用的数据集来自 Stanford CS246 课程该数据集具有真实的大规模社交网络结构，为算法验证和学术研究提供了重要的参考价值。

1.2 技术选择及优势

在本实验中，选择使用 Hadoop MapReduce 作为分布式计算的实现框架，以处理大规模的社交网络数据。Hadoop 是一种开源的分布式计算框架，具有高度的扩展性和容错性，其核心组件 Map-Reduce 提供了一种简单而高效的并行计算模型。

1.2.1 技术选择

Hadoop MapReduce 作为实验的技术核心，是因为其能够满足以下需求：

大规模数据处理，社交网络中的用户关系数据通常规模庞大，单机计算在存储和处理能力上难以胜任，而分布式计算框架可以充分利用多节点的计算资源。简单的计算模型，MapReduce 采用“分而治之”的思想，将大规模任务分解为多个并行的 Map 和 Reduce 作业，简化了编程复杂度，适合解决如好友推荐这种键值对关联性较强的问题。开源易用，Hadoop 是当前主流的开源分布式计算平台，拥有广泛的社区支持和完善的文档，可以快速构建和部署实验环境。

1.2.2 Hadoop MapReduce 的优势

本实验选择 Hadoop MapReduce，不仅因为其成熟性和易用性，还因为其在大规模数据处理中的以下独特优势：

1. 并行计算能力

Map-Reduce 将任务分解为多个 Map 和 Reduce 作业，可以在多台计算机上并行执行。对于像 soc-LiveJournal1Adj 这样的大规模社交网络数据，MapReduce 的并行处理能力可以显著提高计算效率，缩短任务运行时间。

2. 良好的容错性

Hadoop 具有内置的容错机制，当某个计算节点出现故障时，系统会自动重启任务并重新分配资源，确保任务的高可靠性。这一点对于长时间运行的大规模任务尤为重要。

3. 灵活的数据分区与分布式存储

Hadoop 的分布式文件系统（HDFS）能够自动将大文件分块存储到多个节点中，每个块都有多个副本，保证了数据的高可用性。Map-Reduce 作业会根据数据分布智能调度任务，使计算更高效。

1.3 应用场景

在日常生活中，Hadoop Map-Reduce 技术广泛应用于各类社交网络平台的推荐系统和数据分析。比如，在好友推荐功能中，社交媒体（如 Facebook、微博）通过分析用户的好友列表、共同好友关系，为用户推荐可能认识的人，帮助用户拓展社交圈；电商平台（如 京东、Amazon）基于用户的购买记录和浏览行为，推荐相关产品，提升购物体验。此外，在短视频平台（如抖音、YouTube），通过用户的观看历史和点赞评论关系，Map-Reduce 能快速处理海量用户数据，生成个性化视频推荐列表。这些应用场景背后，Map-Reduce 提供了高效的数据处理和并行计算能力，为日常生活中的数字服务提供了坚实的技术支持。

2 相关技术

2.1 Hadoop

Hadoop 是由 Apache 开源软件基金会开发的，运行在大规模服务器上用于大数据存储、计算、分析的分布式并行编程框架。Hadoop 由三部分组成，分别是分布式存储系统 HDFS、资源管理和任务调

度框架 YARN 和分布式计算框架 MapReduce。

Hadoop 允许用户使用简单的编程模型实现跨机器集群对海量数据进行分布式计算处理。其设计目标是能够在廉价集群上高效稳定运行，支持 PB 级数据存储与计算。它并发处理数据，节点间高速运算，冗余存储确保高可靠性，自动维护副本和重算任务，应对数据丢失或节点故障。Hadoop 搭建的分布式系统能挖掘社交网络大数据价值，展现其高可靠、高效、可扩展、容错且成本低的特点。

该研究基于 Hadoop 框架实现社交网络好友推荐系统，可以充分利用其分布式存储和并行计算的能力，提高数据处理和分析的效率，优化好友推荐算法，为用户提供更加准确和个性化的推荐服务。

2.2 HDFS

HDFS（Hadoop Distributed File System）是基于流数据访问模式的分布式文件存储系统，专为处理大量数据而设计，是 Hadoop 的核心组件。我们使用 HDFS 处理和存储 soc-LiveJournal1Adj 社交网络好友数据集，为处理和分析社交网络中的大规模数据提供了高效、可靠的存储解决方案。

HDFS 采用了主从（Master/Slave）结构模型，HDFS 的组成架构主要包括 NameNode（名称节点）和 DataNode（数据节点）等组件：NameNode 负责管理文件系统的命名空间，记录每个文件的元数据信息，如文件块的位置等；而 DataNode 则负责存储文件的实际数据块，把数据块以 Linux 文件的形式保存在磁盘上，并根据 Block 标识和字节范围来读写块数据。

HDFS 能存 PB 级数据，满足社交网络海量数据的存储需求。它分块存储数据并自动平衡，优化读写性能，保证节点负载均衡。通过副本存储，HDFS 提升数据可靠性和容错性，即使节点故障也能保持数据完整。在推荐系统中，HDFS 存预处理数据供算法使用，支持批处理及实时分析。它还提供加密和访问控制，保障用户隐私和数据安全。随着社交网络用户数量的增加，HDFS 可以轻松地扩展存储节点和数据量，以满足系统的增长需求。

2.3 MapReduce

MapReduce 是一种面向大规模数据并行处理的分布式文件计算框架，用于处理和生成大规模数据集。它具有良好的易编程性，高容错性，并且适合 PB 级以上海量数据的离线处理，可以轻松地扩展计算节点和数据量，以满足社交网络用户规模的不断增长，并且可以与 HDFS 系统高效的并行处理。将 MapReduce 技术与社交网络好友推荐系统的融合，为处理和分析社交网络中的大规模用户数据提供了强大的计算能力。

MapReduce 的核心思想是分而治之，它将复杂的计算任务分解为一系列的 Map 和 Reduce 操作。具体来说：

Map 阶段：主要负责以对社交网络中的用户数据进行拆分、处理和转换，在此阶段可以计算每个用户的共同好友数，生成一系列的键值对，其中键为用户 ID，值为该用户的共同好友列表和共同好友数量。将处理后的数据将作为中间结果传递给 Reduce 阶段进行进一步的处理和分析。

Reduce 阶段：负责对这些中间键值对进行汇总、排序和归约，生成每个用户的潜在好友列表，并按照共同好友数量进行排序，最终生成所需的输出结果。

通过优化 Map 和 Reduce 阶段的算法和数据处理流程，实现了对用户数据的快速处理和推荐结果的准确生成，为用户提供了更加个性化的好友推荐服务。

3 推荐算法介绍

社交网络交友数据将以邻接表的形式输入，有多个以<用户 ID>,<用户 ID 列表>格式的行，其中<用户 ID>是唯一用户的唯一标识符，<用户 ID 列表>是由逗号分隔的用户列表，这些用户是<用户 ID>的朋友。以下是一个示例：假设我们有 7 位用户，分别以 0，1，2，3，4，5，6 进行编号。

表 1 用户好友列表

用户 ID	用户的朋友
0	1，2，3
1	0，2，3，4，5
2	0，1，4
3	0，1，4
4	1，2，3
5	1，6
6	5

在图表中，用户 0 不是用户 4 和 5 的朋友，但用户 0 和用户 4 有共同的朋友 1、2 和 3；用户 0 和用户 5 有共同的朋友 1。因此，我们希望推荐用户 4 和 5 作为用户 0 的朋友。

推荐给用户的朋友们将以以下格式给出：<推

荐给用户的朋友（共同好友数量：[共同好友的ID，...]),...>。输出结果根据共同好友的数量进行排序，并且可以从下表进行验证。

表 2 好友推荐列表

用户 ID	推荐给用户的朋友
0	4 (3: [3, 1, 2]),5 (1: [1])
1	6 (1: [5])
2	3 (3: [1, 4, 0]),5 (1: [1])
3	2 (3: [4, 0, 1]),5 (1: [1])
4	0 (3: [2, 3, 1]),5 (1: [1])
5	0 (1: [1]),2 (1: [1]),3 (1: [1]),4 (1: [1])
6	1（1：[5]）

用户 0 有朋友 1、2 和 3；因此，<1,2>、<2,1>、<2,3>、<3,2>、<1,3>和<3,1>这些对是用户 0 的共同朋友。因此，我们可以发出<key, value>=<1, r=2;m=0>、<2, r=1;m=0>、<2, r=3;m=0>...，其中 r 表示推荐的朋友，m 表示共同朋友。我们可以在 reduce 阶段聚合结果，并计算用户和推荐用户之间有多少共同朋友。为了克服朋友重复问题，我们在发出的值中添加另一个属性 isFriend，并且如果在 reduce 阶段我们知道他们已经是朋友，我们就不推荐这个朋友了。在接下来的实现中，当他们已经是朋友时，使用 m=-1 而不是使用额外的字段。

我们定义 fromUser 是输入数据中的一个，而 toUser 是其中的一个：

1. Map 阶段：首先，发出<fromUser, r=toUser;m=-1>给所有 toUser。假设存在 n 个 toUser；那么我们将发出 n 条记录来描述 fromUser 和 toUser 已经是朋友了。注意，他们已经在发出的键和 r 之间成为了朋友，所以我们把 m 设置为-1。其次，对于所有可能的 toUser1 和 toUser2 组合，从 toUser 中发出<toUser1, r=toUser2;m=fromUser>，并且他们之间有共同的朋友 fromUser。这将发出 n(n-1) 条记录。总共，在 map 阶段有 n^2 条发出的记录，其中 n 是朋友的数量。

2. Reduce 阶段：计算他们之间有多少共同好友，键和值中的 r。如果他们中的任何一个有共同好友是-1，我们不会推荐，因为他们已经是朋友了。最后根据共同朋友的数量对结果进行排序。

4 研究过程

4.1 开发环境

本实验运行于 macOS 系统，搭建了分布式计算环境，具体系统环境配置如下：

Hadoop 版本：3.3.2，Java 版本：1.8.421.09（arm64），OpenJDK 版本：18.0.1，硬件环境：MacBook，搭载 M1 芯片，支持 ARM 架构，该环境能够高效支持 Hadoop 的分布式计算，并满足实验对大规模数据处理的需求。

4.2 MapRduce 代码编写

4.2.1 FriendCountWritable 类

好友推荐对的信息包含用户和共同好友两个关键元素。在社交网络好友推荐的场景中，为了确定向某个用户推荐哪些好友，需要知道目标用户以及他们之间的共同好友关系，FriendCountWritable 类中的 user 字段用于存储目标用户的标识，mutualFriend 字段用于存储共同好友的标识。

在 MapReduce 计算模型中，数据需要在 Map 阶段和 Reduce 阶段之间进行传递和处理。为了保证数据的一致性和可操作性，需要定义一种合适的数据结构来存储中间结果和最终结果。FriendCountWritable 类实现了 Writable 接口，这使得它能够在 Hadoop 的分布式环境中进行序列化和反序列化操作。序列化是将对象转换为字节流以便在网络上传输或存储在文件中的过程，反序列化则是相反的操作。通过实现 Writable 接口，FriendCountWritable 类的对象可以在 MapReduce 任务的不同阶段之间进行高效的传输和处理，确保数据的完整性和准确性。

```java
public class FriendCountWritable implements Writable {
    public Long user;
    public Long mutualFriend;

    public FriendCountWritable(Long user, Long mutualFriend) {
        this.user = user;
        this.mutualFriend = mutualFriend;
    }

    public FriendCountWritable() {
        this(-1L, -1L);
    }

    @Override
    public void write(DataOutput out) throws IOException {
        out.writeLong(user);
        out.writeLong(mutualFriend);
    }

    @Override
    public void readFields(DataInput in) throws IOException {
        user = in.readLong();
        mutualFriend = in.readLong();
    }
```

图 1 FriendCountWritable 类代码（部分）

4.2.2 Map 阶段

Map 类在整个 MapReduce 框架中进行输入数据初步处理和转换，为后续的 Reduce 阶段提供合

适的数据格式和中间结果。在社交网络好友推荐系统中，具体体现为从原始的社交数据中挖掘出每个用户的好友关系，并基于这些关系生成可能的好友推荐记录。通过对输入数据的逐行解析和处理，Map 类将用户及其好友信息转化为特定的键值对形式，以便在后续计算中能够准确识别和统计好友之间的关系。

```
public class Map extends Mapper<LongWritable, Text, LongWritable, FriendCountWritable>
implements Tool {
    private Configuration conf;

    @Override
    public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
        String line[] = value.toString().split("\t");

        Long fromUser = Long.parseLong(line[0]);
        List<Long> toUsers = new ArrayList<>();

        if (line.length == 2) {
            StringTokenizer tokenizer = new StringTokenizer(line[1], ",");
            while (tokenizer.hasMoreTokens()) {
                Long toUser = Long.parseLong(tokenizer.nextToken());
                toUsers.add(toUser);
                context.write(new LongWritable(fromUser), new FriendCountWritable(toUser,
-1L));
            }

            // Generate combinations of friends and their mutual friends
            for (int i = 0; i < toUsers.size(); i++) {
                for (int j = i + 1; j < toUsers.size(); j++) {
                    context.write(new LongWritable(toUsers.get(i)), new
FriendCountWritable(toUsers.get(j), fromUser));
                    context.write(new LongWritable(toUsers.get(j)), new
FriendCountWritable(toUsers.get(i), fromUser));
                }
            }
        }
    }
```

图 2 Map 阶段代码（部分）

4.2.3 Reduce 阶段

通过对 Map 阶段输出数据的有效聚合、统计和处理，Reduce 类将原始的社交关系数据转化为具有实际应用价值的好友推荐信息，为提升社交网络用户体验和促进用户互动提供了重要的功能支持。

```
public class Reduce extends Reducer<LongWritable, FriendCountWritable, LongWritable, Text>
{
    @Override
    public void reduce(LongWritable key, Iterable<FriendCountWritable> values, Context
context)
            throws IOException, InterruptedException {
        Map<Long, List<Long>> mutualFriends = new HashMap<>();

        // Process each value
        for (FriendCountWritable val : values) {
            Long toUser = val.user;  // 直接访问 public 字段
            Long mutualFriend = val.mutualFriend; // 直接访问 public 字段
            if (mutualFriends.containsKey(toUser)) {
                if (mutualFriend != -1) {
                    mutualFriends.get(toUser).add(mutualFriend);
                }
            } else {
                if (mutualFriend != -1) {
                    mutualFriends.put(toUser, new ArrayList<>
(Arrays.asList(mutualFriend)));
                }
            }
        }

        // Sort by number of mutual friends (descending order)
        TreeMap<Long, List<Long>> sortedMutualFriends = new TreeMap<>((key1, key2) -> {
            int size1 = mutualFriends.containsKey(key1) ? mutualFriends.get(key1).size() :
0;
            int size2 = mutualFriends.containsKey(key2) ? mutualFriends.get(key2).size() :
0;
            return Integer.compare(size2, size1); // descending order
        });
```

图 3 Reduce 阶段代码（部分）

4.2.4 编译、打包

```
pengzexue at pengzedeMacBook-Pro in ~/Desktop/PeopleYouMightKnow (master●)
$ javac -classpath $(hadoop classpath) -d bin src/FriendCountWritable.java src/Map.j
ava src/Reduce.java
```

图 4 编译

```
pengzexue at pengzedeMacBook-Pro in ~/Desktop/PeopleYouMightKnow (master●)
$ jar -cvf FriendRecommendation.jar -C bin/ .
已添加清单
正在添加：.DS_Store(输入 = 6148) (输出 = 178)(压缩了 97%)
正在添加：FriendCountWritable.class(输入 = 1498) (输出 = 772)(压缩了 48%)
正在添加：Map.class(输入 = 4210) (输出 = 1909)(压缩了 54%)
正在添加：Reduce.class(输入 = 3860) (输出 = 1732)(压缩了 55%)
(base)
```

图 5 打包

验证完整性：

```
pengzexue at pengzedeMacBook-Pro in ~/Desktop/PeopleYouMightKnow (master●)
$ jar -tf FriendRecommendation.jar
META-INF/
META-INF/MANIFEST.MF
.DS_Store
FriendCountWritable.class
Map.class
Reduce.class
(base)
```

图 6 验证完整性

编写后与数据放在桌面文件夹中路径类似：

```
PeopleYouMightKnow/
    ├── bin                 # 编译输出目录
    ├── src                 # 源代码目录
    │   ├── FriendCountWritable.java
    │   ├── Map.java
    │   └── Reduce.java
    └── input.txt           # 输入数据文件
```

4.3 启动 Hadoop 集群

在伪分布式环境下，启动 Hadoop 服务，执行以下命令启动 Hadoop 集群，包括 HDFS 和 yarn：

```
pengzexue at pengzedeMacBook-Pro in ~/Desktop/PeopleYouMightKnow (master●)
$ start-dfs.sh
Starting namenodes on [localhost]
localhost: ERROR: Cannot set priority of namenode process 44416
Starting datanodes
localhost: ERROR: Cannot set priority of datanode process 44516
Starting secondary namenodes [pengzedeMacBook-Pro.local]
2024-12-18 21:22:51,416 WARN util.NativeCodeLoader: Unable to load native-hadoop lib
rary for your platform... using builtin-java classes where applicable
(base)
```

图 7 启动 Hadoop 集群

检查 HDFS 状态：打开浏览器，访问 http://localhost:9870，查看 HDFS Web UI。

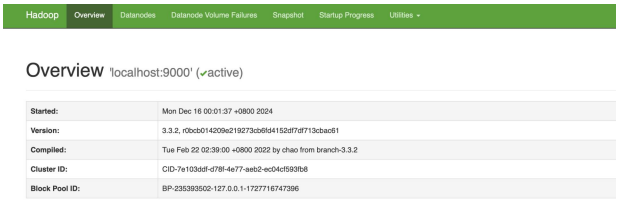

图 8 检查 HDFS 状态

之后启动 yarn 出现错误首先配置 yarn 文件将日志文件储存在日志目录 /opt/homebrew/opt/hadoop/log 中在查看运行后的日志文件：

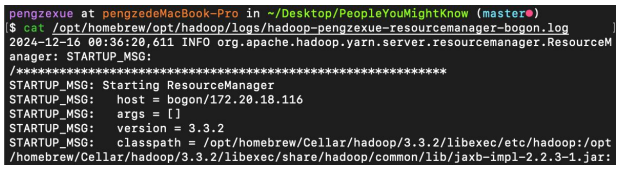

图 9 查看运行后的日志文件

从日志信息来看，ResourceManager 启动失败的主要原因是 Java 模块限制，导致 java.lang.reflect.InaccessibleObjectException 异常。这种问题常见于 Hadoop 与较新的 Java 版本（如 Java 17 或更高版本）之间的不兼容。所以要么降级 java 版本或者继续修改 yarn 配置文件：

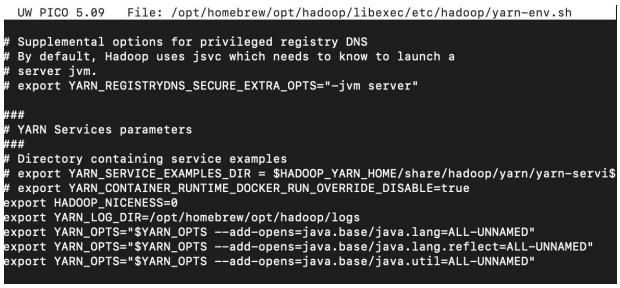

图 10 修改 yarn 配置

解决问题之后启动 yarn：

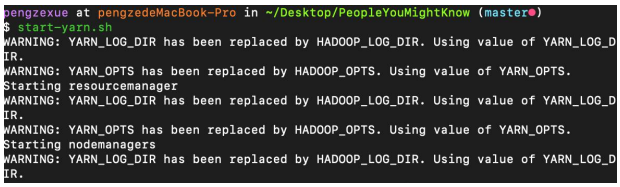

图 11 启动 yarn

打开浏览器，访问 http://localhost:8088，查看 yarn 资源管理界面：

图 12 查看 yarn 状态

查看运行情况：

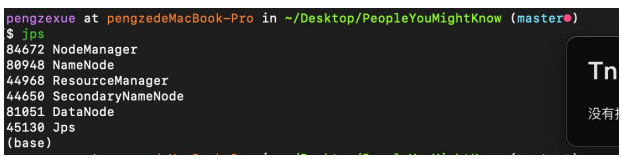

图 13 查看运行情况

4.4 创建 HDFS 目录，上传数据到 HDFS

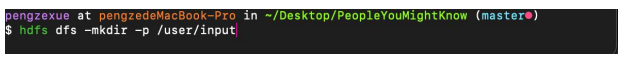

图 14 创建 HDFS 目录

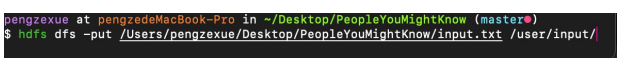

图 15 上传数据文件到 HDFS

验证数据是否成功上传：

图 16 验证数据是否成功上传

运行 MapReduce 作业：

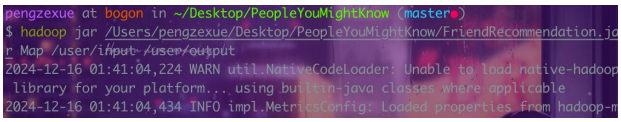

图 17 运行 MapReduce 作业

导出结果并看输出（图片展示部分输出）：

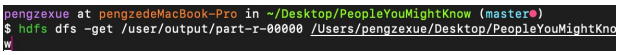

图 18 导出结果

图 19 查看输出（部分）

5 总结

本文详细阐述了基于 Hadoop 的社交网络好友推荐系统的实现过程。该系统通过 MapReduce 编程模型，高效地处理和分析大规模的社交网络数据，从而为用户推荐可能认识的新朋友。

系统架构与组件：

实现了 FriendCountWritable 类，用于存储每个好友推荐对的信息，包括用户 ID 和共同好友 ID。

设计了 Map 类，负责生成每个用户的好友推荐记录。通过解析输入数据，生成用户与潜在好友之间的推荐关系，并输出为键值对形式。

实现了 Reduce 类，用于聚合并输出推荐结果。根据共同好友的数量，对用户进行排序，并生成最终的推荐列表。

技术实现与优化：

利用 Hadoop 的分布式文件系统（HDFS）存储大规模数据，提高了数据处理的效率和可靠性。

通过 MapReduce 框架，实现了数据的并行处理和分布式计算，有效降低了数据处理的时间复杂度。

在 Reduce 阶段，采用 TreeMap 对共同好友数量进行排序，保证了推荐结果的准确性和有序性。

系统验证与测试：

编译并打包源代码，生成可执行的 JAR 文件。

在 Hadoop 集群上部署并运行 MapReduce 作业，验证了系统的正确性和性能。

通过上传数据到 HDFS、运行 MapReduce 作业以及查看输出结果等步骤，对系统进行了完整的测试。

致 谢 本次小组作业的顺利完成，离不开我们每一位成员的辛勤付出与团结合作。在此，我代表小组向所有成员表达最深的感激之情。也特别感谢在过程中给予我们宝贵建议和悉心指导的老师，为我们的作业增添了更多的深度与广度，以及在我们遇到难题时伸出援手的同学。从最初的任务分配，到资料搜集、讨论交流，再到最终的整合与提交，每一步都凝聚了大家的智慧与汗水。正是凭借着我们各自的专业知识、积极态度和不懈努力，我们才能够克服重重困难，高质量地完成了这次作业。同时，也要感谢学校提供的良好学习环境和丰富资源，为我们的作业完成提供了有力的支持。最后，再次感谢我们小组内的每一位成员。在合作中，我们相互学习，共同进步，展现了团队的力量。这份作业不仅是我们学习成果的体现，更是我们团队协作精神的见证。感谢大家，让我们共同见证了这一份成功的喜悦！

参 考 文 献

[1] 杨婷.基于MapReduce的好友推荐系统的研究与实现[D].北京邮电大学,2013.

[2] 黄学峰.基于Hadoop的电影推荐系统研究与实现[D].南京师范大学,2016.

[3] 李文海,许舒人.基于Hadoop的电子商务推荐系统的设计与实现[J].计算机工程与设计，2014 ,35(1):8.DOI:10.3969/j.issn.1000-7024.2014.01.025.

[4] 杨瑞仙,刘莉莉,楚晨等.社交网络中的好友推荐研究——概念、方法与展望[J].现代情报,2023,43(04):28-38.

[5] 贺超波,汤庸,陈国华等.面向大规模社交网络的潜在好友推荐方法[J].合肥工业大学学报(自然科学版),2013,36(04):420-424.

附录X.

数据来源：

https://www.dbtsai.com/assets/blog/2013/01/soc-LiveJournal1Adj.txt

第一作者照片
(高清照片)

Author1,XUE Peng-Ze Born in 2002, with a bachelor's degree, the main research area is Remote Sensing Change Detection.
Author1,FU Jing-Ying Born in 2002, with a bachelor's degree, the main research area is machine learning.
Author1,LIU Ya-Xin Born in 2002, with a bachelor's degree, the main research area is multimedia information security.
Author1,HUANG Chu-Yao Born in 2001, with a bachelor's degree, the main research area is machine learning.

Background

In today's digital era, social networks are thriving, with renowned platforms like Facebook and Twitter accumulating astonishing amounts of user data, encompassing intricate and vast information on user social connections. Take Facebook as an example; it boasts billions of users, with a complex web of friend connections among them. In this scenario, traditional single-machine data processing methods struggle to effectively analyze friend recommendations when confronted with such massive data scales.

As social networks continue to expand, both the number of users and the volume of inter-user relationships soar, imposing stricter requirements on the real-time performance and accuracy of friend recommendations. Given the inherent limitations of computational resources on a single machine, it becomes infeasible to handle frequent friend recommendation calculations against large-scale data. Consequently, there is an urgent need for a distributed computing model to surpass current performance bottlenecks and achieve efficient processing of massive data.

Enter MapReduce, a distributed computing framework that emerged at just the right time. It decomposes processing tasks for large datasets into multiple subtasks, executes them in parallel across cluster nodes, and ultimately aggregates the results of these subtasks. This approach offers a practical solution for efficiently processing massive social network data, significantly driving the development and advancement of social network data processing technologies.

The research pertains to the domain of social network analysis and distributed computing. The problem at hand is to develop an efficient friend recommendation system in the context of large-scale social networks. Internationally, while there have been advancements in traditional recommendation algorithms, handling the vast amounts of data and the computational complexity in social networks remains a challenge. Existing methods often struggle with scalability and performance when dealing with big data.

This paper presents a novel solution by integrating the MapReduce paradigm into the friend recommendation process. By distributing the data processing tasks across multiple computing nodes, our approach significantly improves the system's scalability and processing speed. We are able to handle large datasets more effectively and generate accurate friend recommendations in a timely manner, which overcomes the limitations of previous methods.

The project holds great significance for the social network industry. It enables social platforms to provide more reliable and efficient friend recommendations, thereby enhancing user experience and satisfaction. This, in turn, can lead to increased user activity and retention, as well as open up new avenues for targeted advertising and content delivery. From a technical perspective, it also contributes to the development of distributed computing applications in the social network domain.