Computer Science and Application 计算机科学与应用, 2017, 7(7), 645-653
Published Online July 2017 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2017.77075

Hans汉斯

Implementation of Natural Language Processing Platform Based on Hadoop Cluster

Ning Xie, Wei Guo, Huifeng Tang

PLA University of Foreign Language, Luoyang Henan
Email: tanghuifengom@163.com

Received: Jun. 14th, 2017; accepted: Jun. 28th, 2017; published: Jul. 4th, 2017

Abstract

This paper proposes a Hadoop-based program of natural language processing platform, collecting user needs and data via web interface, automatically calling the cluster into processing, and returning processing results. The platform supports Mapreduce function packages programmed by third-party developers, and is easy to expand. Experiments confirm the platforms ability of timely responding to user needs, accurately calling relevant programs, and returning processing results.

Keywords

Hadoop, MapReduce, Natural Language Processing Platform, Plug-In

一种基于Hadoop集群的自然语言处理平台实现方案

谢　宁，郭　威，唐慧丰

解放军外国语学院，河南 洛阳
Email: tanghuifengom@163.com

收稿日期：2017年6月14日；录用日期：2017年6月28日；发布日期：2017年7月4日

摘　要

本文提出一种基于Hadoop集群的自然语言处理平台实现方案，通过网页收集用户需求和数据，自动调用集群进行处理，返回处理结果。且支持运行第三方开发者编写的Mapreduce插件，平台易扩展。通过

文章引用: 谢宁, 郭威, 唐慧丰. 一种基于 Hadoop 集群的自然语言处理平台实现方案[J]. 计算机科学与应用, 2017, 7(7): 645-653. https://doi.org/10.12677/csa.2017.77075

实验验证，方案能实现及时响应用户需求，准确调用相关程序，并返回处理结果。

关键词

Hadoop，MapReduce，自然语言处理平台，插件

1. 引言

随着互联网技术的迅猛发展，数据的智能化处理获取越来越重要。在自然语言处理领域，大规模语料库技术和其他基于概率统计的研究方法蓬勃发展，为自然语言的研究提供了新的思路和工具。各种新模型、新技术、新应用层出不穷，对计算机的计算和存储能力提出了更高的要求。当前流行的一些数据密集型的计算方法，处理过程比较复杂，处理耗时较长。如对大规模语料进行 word2vec 模型训练，往往一次就需要数天时间。传统的单机计算模式，其计算速度和存储能力不能满足大规模数据处理的需求。

目前，现有的算法成果主要是各种程序包，一般对应某种编程语言，包含一个或多个功能，针对特定领域，具有较高的处理效率和精确度。使用前需要先下载相应的工具包，再针对所处理的语言、计算机硬件环境和具体处理需求等进行参数配置。这种方法需要算法使用者维护数据文件和执行文件，还要掌握相应的编程语言。由于开发习惯不尽相同，不同程序包内文件组成方式千差万别，参数配置过程复杂且不通用，不便于使用者学习掌握，提高了使用的门槛。

综合考虑上述情况，提出建设自然语言处理平台的方案。借助 Hadoop 集群，解决大规模数据存储的问题，同时提升计算能力；使用插件式的应用开发模式，整合改进自然语言处理领域的各种成熟算法，缩短了产品开发周期；实现友好的交互界面，用户通过便捷的操作，即可上传数据，调用算法，得到处理结果。

2. 相关工作

2.1. 自然语言处理平台

2002 年，刘群[1]等提出，我国的自然语言处理研究缺乏一些公共的基础设施，很多研究工作都要花费大量的精力从底层模块做起，这使研究在很大程度上处于一种低水平重复的状态，造成研究工作效率低下且难以深入的问题。他还提出，可以借鉴开放式的开发模式，并给出了一个可以共享代码、语料、语言知识库等资源，并支持协作开发的自然语言处理开放平台的设计。

2006 年，国内语言信息处理平台的雏形出现。哈尔滨工业大学的郎君、刘挺等[2]对一套中文语言处理平台进行了描述，平台基于 xml 面向 Web 实现，命名为“语言技术平台 LTP (Language Technology Platform)”。平台集成了词法、词义、句法、语义、篇章分析等 10 项中文处理核心技术，旨在向初学者提供一套系统化工具，初学者进行一些初步的处理，也可进而研究一些更高级的应用。该平台对外免费共享。到 2010 年，该平台的设计者已尝试提供网络服务。

近年来，随着自然语言处理研究升温，不少科技公司结合云计算、深度学习或人工智能技术，推出了可远程使用的语言处理平台。例如百度云的自然语言处理平台天智[3]，腾讯云的文智自然语言处理平

台[4]等，提供外部接口，供用户付费或免费调用。出于研究目的，不少高校、科研院所也推出了自己的处理平台，部署有少量功能进行在线展示。其中比较著名的有清华大学自然语言处理与社会人文计算实验室网页的在线展示部分[5]，哈尔滨工业大学联合科大讯飞公司推出的“哈工大－讯飞语言云”[6]，北京理工大学的自然语言处理与信息检索共享平台[7]等。

2.2. Hadoop

Hadoop 是一个成熟的分布式系统的基础架构，搭载了分布式存储系统(HDFS)和分布式计算框架MapReduce，分别为海量数据的存储和计算提供支持[8]。Hadoop 可以高吞吐量的访问数据，所以最初被应用于网页搜索、日志分析、广告计算等大规模数据处理工作。百度和淘宝在国内最先使用 Hadoop 进行电子商务数据和离线的日志的统计分析和处理。国内早期对基于 Hadoop 海量数据处理系统的研究，主要是实现海量数据的存储和分布式计算，处理的对象主要是系统的日志文件[9] [10]。

随着 Hadoop 技术的进一步发展，功能逐渐完善，应用范围日趋扩大，以 Hadoop 为基础的数据分析系统数量猛增。针对日志文件处理的研究势头不减，以更为非格式化的自然语言，如网络文本，为处理对象的系统开始出现。这些研究的方向主要包括文本过滤、文本分类和聚类、网络文本分析、情感分析、句法分析、自动文摘、文档相似度计算等[11] [12] [13]。

2.3. 插件化开发

插件是一种根据统一规范编写而成的应用程序，被主程序通过接口调用，拓展主程序的功能[14]。按照插件化模式开发的软件包含主程序和插件两部分，在主程序基本不变的情况下，通过调整插件数量或修改插件执行顺序调整软件的功能。任何人都可以按照接口标准编写插件，实现“即插即用”的功能开发[15]。插件可单独使用，也可通过插件输出输入接口的拼接，实现多个插件组合使用，完成较为复杂的应用。

相对于传统开发模式，每一个插件可以独立开发、测试、部署和升级，系统开发不再需要反复进行代码合并和整体发布，开发效率大幅提高。自然语言处理领域的很多常见应用，可以拆分为多个处理环节。例如中文文本分类应用，可分为中文分词、词频统计、关键词提取、文档相似度计算、文本分类等几个环节，每个环节由一个插件实现。

2.4. 本方案与以往研究的区别

首先是用户的使用方式更便捷。之前的研究中，向集群提交请求的方式一般是通过程序调用或用户编写 xml 文件。这保证了传递需求的准确性和多样性，但对用户的要求还显得较高，需要用户掌握相应的编程语言和函数调用规则，至少是各个配置参数的取值和意义。本方案尝试使用 Web 网页作为用户提交请求、调用系统功能的界面。用户在系统设定好的下拉框选择功能，在文本框输入参数，使用难度大大降低。

其次是能自动提交任务执行。平台启动后，以收到用户请求为信号，自动触发任务提交过程。支持向平台连续提交多个任务，全过程不再需要管理员人为参与，降低维护难度，提高执行效率。

最后是平台功能易扩展。使用插件化的功能开发模式，功能之间的设计实现互不干扰，降低开发难度。可以借鉴自然语言处理领域成熟的算法，进行分布式改写后整合成即插即用的功能插件，将已有成果快速转化为可供用户远程调用的功能。

3. 平台设计

平台的设计分为三个模块，其关系如图 1 所示。

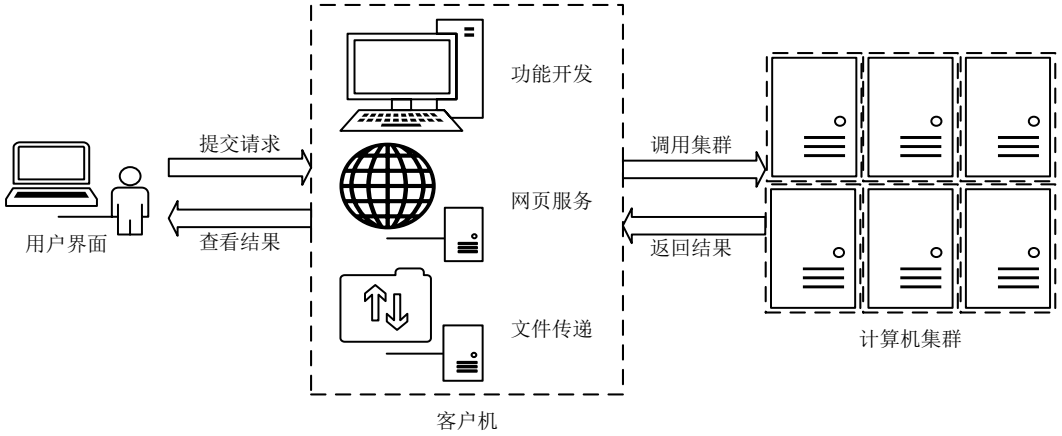

Figure 1. System framework
图 1. 系统框架图

运行在网络服务器上的用户界面，对用户展示平台，屏蔽功能实现细节。界面收集用户处理需求和数据，在处理完成后向用户提供处理结果的下载链接；平台的计算和存储能力由一个包含若干台计算机的 Hadoop 集群提供。集群中参与运算的计算机数量可根据硬件条件和任务需求增减；界面和集群之间的连接由一台客户机承担。客户机作为中转，实现界面、第三方开发者和集群之间文件和信息的传递。

前端交互界面的作用已比较明确，Hadoop 集群的搭建也不是本文讨论的重点，故详细介绍客户机的作用：1) 客户机负责管理由第三方开发者完成的 Jar 包；2) 客户机实现接受请求、向集群上传下载数据、调用集群进行运算等功能；3) 在客户机搭建网页服务器，用户上传的待处理数据直接可下载到客户机本地，减少数据传递的时间开销；4) 客户机使用 Ubuntu 桌面版操作系统，平台管理员通过网页查看集群的运行状态。客户机安装 JDK 和 Eclipse 后，可直接用于功能的开发和调试。

3.1. 数据准备

在客户机的本地文件系统指定路径，分别存放输入和输出文件。

客户机的输入文件有 3 类，分别指定 3 个目录：1 个目录存放已经开发完成的 Jar 文件和介绍文档，文件可以是第三方开发者编写完成后通过网络传输得到，也可以是开发者在客户机本机开发生成；另外 2 个目录分别存放从交互界面得到的用户数据和包含任务请求的配置文件。

客户机的输出文件只有一类，即处理结果。考虑到平台实际运行中，可能有多个任务被执行，在输出目录中，按任务名称建立任务目录作为区分。某一任务可能被执行多次，生成处理结果文件时，在命名中加上任务开始执行的时间，以区分多次执行的结果。

3.2. 作业流程

平台主要的作业流程有两种，一是调用插件。对于已经完成开发的插件，可根据界面传递来的用户的需求直接调用，完成相应的功能。二是插件更新。将新开发的插件和有升级的插件及时更新到客户机。

3.2.1. 用户需求收集

用户登录网页向平台提出需求，首先选择所需要的功能。根据所选功能的不同，网页展示若干参数的下拉框或文本框，供用户点击选择或手动输入。最后提供数据上传窗口，用户提交符合相应格式的文件。若用户不上传文件，则平台将使用客户机默认位置存放的示例文件继续执行。

用户需求确认后，平台在客户机本地按功能名命名建立文件夹，存放用户数据文件。数据存放完毕

后，将用户需求生成一个文本文件下载到客户机本地硬盘中指定位置。文件中每个参数占一行，参数名和参数值以“:”隔开，文件名加“.mission”后缀表示是任务配置文件。至此，完成用户需求收集。

3.2.2. 集群处理

现有向 Hadoop 集群远程提交作业主要有三种方法：1) 手动将程序打包成 Jar 文件，使用系统命令行提交；2) 使用 Eclipse 的 Hadoop 插件提交；3) 使用代码自动生成 Jar 包和任务提交。

第一种方法直接对 Jar 包进行操作，是 Hadoop 的标准的做法。但整个过程由人工操作，不能满足平台自动提交任务的要求；后两种方法操作简单，但都借助了 Eclipse 开发工具，提交任务前需针对功能的使用场景编写驱动类。驱动类的设置涉及程序内部处理细节，实现过程较为繁琐，且不同功能的设置不通用。

本平台对几种任务提交方式进行整合，选择一种使用 Java 程序通过 SSH 调用 Shell 命令的方式，远程执行 Jar 包的方法。任务提交流程如图 2 所示。

1) Java 实现的主程序持续运行，对客户机存储任务请求的文件夹进行监控，当发现有新文件传入时自动触发任务提交流程；2) 获取新传入文件的文件名，根据其后缀名验证文件类型；3) 若文件后缀是“.mission”，则判定是任务请求文件，按行读取文件，保存参数。否则不执行，结束提交；4) 建立客户机与集群 HDFS 文件系统的连接，新建 HDFS 输入目录，上传本地的数据文件；5) 通过 Java 程序执行 Shell 命令，命令中包含从任务请求文件中读取到的各项参数；6) 处理完成后，将处理结果下载到客户机本地。

3.2.3. 功能更新

有新的功能插件完成了开发，将相应的 Jar 文件放入客户机的指定位置，即可被平台调用。若平台现有功能有了更新升级，用该功能新的 Jar 文件替换同名的原文件。

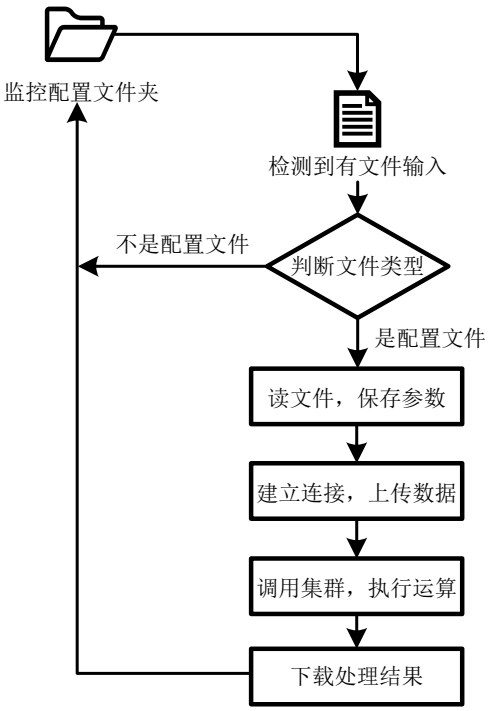

Figure 2. The process of client submitting job to cluster
图 2. 客户机向集群提交任务流程图

4. 实验结果及性能分析

4.1. 实验目的

本文实验目的主要有两个：

1) 通过实验验证按照本文提出的方案实现的系统是否运行正常；

2) 修改集群节点数目，运行同一个功能，比较不同数据量的处理时间，对集群平台的性能做出总结。

为验证本文提出的平台构建方案的可行性，搭建了 7 台节点的 Hadoop 集群。实验时 1 台主机作为 master，参与运算的 slave 数量可根据实验需求灵活调整。所有主机的硬件配置完全相同，排除无关因素对实验的影响，保证实验严谨性。试验环境如下：

主机配置：CPU 为 Intel Core 2 Duo CPU E7300 @ 2.66GHz×2，内存 1.9 GB，155,3 GB 硬盘。

操作系统：采用 Ubuntu 14.04.3 LTS(64Bit)，客户机为桌面版，集群中的主机为服务器版。

软件及相应版本：Java 环境为 jdk1.8+，Hadoop 2.7.1。

因为实验主要验证方案可行性，实验采用了相对简单的算法和数据。

采用算法为 WordCount，被称为单词计数或词频统计，是 Hadoop 编程中最基本也最常见的算法。算法包含 Map 和 Reduce 两个阶段。Map 被称为“映射”，按行读取数据，根据空格将数据拆分为处理单元(单词)，再保存成数据对“处理单元，1”的形式，供后续处理使用；Reduce 被称为“规约”，处理单元相同的数据对被分配给同一个 Reduce，对其附带的数值进行累加，就得到了处理单元在整个数据集中出现的频数。

选择英文 Bible 语料库作为测试数据。该语料库文件大小仅为 4.4 MB，不能发挥集群运算的速度优势，故对其进行了重复复制。为验证集群数量对集群调度和通信时间的影响，特设置一个数据量极小的特殊数据，仅包含一个文本，文本中只有一个单词。可以预见，处理该数据时，文件传输和处理时间极短，程序的总运行时间大体可以作为集群调度时间的基准。最终确定了 5 组测试数据，除极小数据外，其余 4 组的大小分别为 512 MB、1024 MB、2048 MB、4096 MB。集群规模有 4 种，参与运算的 slave 节点数量分别为：1、2、4、6。

为解决单一实验的偶然因素对实验结果造成影响，使用控制变量和均值思想，本试验采用 6 种不同节点数量的集群，分别测试 5 组数据执行同一算法。每次处理时，分别统计系统准备时间、数据上传时间、MapReduce 处理时间和结果下载时间。

4.2. 实时性分析

为验证集群响应任务请求的实时性，统计了从提交任务到开始向集群上传数据的时间，作为集群的响应时间。在 4 个规模不同的集群下处理 5 组数据，每组实验重复 3 次取平均值。在不同集群规模和不同数据量情况下，集群响应时间如表 1 所示。

由表 1 得，8 组实验的平均响应时间均在 58 ms 到 144 ms 之间。集群的响应时间随数据量增大和集群规模减小有增加的趋势，超过 100 ms 的几个结果集中出现在集群规模较小且数据量较大的情况下。

4.3. 数据传输时间分析

为验证客户机与不同规模的集群间传递数据速度的差异，选取 5 组数据，对 4 个集群分别进行数据上传和下载操作，记录传输时间。每组实验同样进行 3 次，结果取平均值。在不同集群规模和不同数据量情况下，数据传输时间如表 2 所示。

由表 2 得，极小数据量相应的上传和下载时间都远小于其他几组数据，因为上传和下载的文件都极小。

Table 1. Cluster response time (ms)
表 1. 集群响应时间(ms)

数据量(MB)	1	2	4	6	均值
极小	58	69	65	67	65
512	84	76	61	87	77
1024	108	86	82	97	93
2048	144	107	64	69	96
4096	127	100	74	114	104

Table 2. Data transfer time (ms)
表 2. 数据传输时间(ms)

数据量(MB)	传输类型	1	2	4	6	平均值
极小	上传	69	79	83	88	80
	下载	20	15	14	13	16
512	上传	63,333	57,276	60,050	57,141	59,450
	下载	105	96	100	101	101
1024	上传	133,939	160,328	179,701	199,982	168,488
	下载	179	116	106	81	121
2048	上传	226,321	275,834	319,555	333,674	288,846
	下载	178	191	157	99	156
4096	上传	441,329	517,547	571,201	573,108	525,796
	下载	97	81	128	181	122

其余 4 组数据相应的上传时间都较长，基本与数据量大小呈正相关。下载时间相差不大，因为处理结果的文件大小基本相同。实验中运行的是 WordCount 程序，且被处理的各组数据是由同一文件复制得来，可以推断输出文件行数相同，区别仅在于处理结果中每个单词的数量不同，所以文件大小相差不大。所以得出结论，文件传输时间基本与要传输的文件大小呈正相关关系。

针对同一组数据，文件传输时间随集群规模扩大有增加的趋势。因为数据按块的形式在客户机和集群各个节点间进行传输，传输时间长短取决于最后完成传输的节点。在不能排除各节点主机性能有差异的情况下，集群规模扩大导致总的传输时间延长。

4.4. 性能分析

比较不同大小的数据在不同集群规模情况下的处理时间。实验选取了 5 组数据，在 4 个集群中分别进行 3 次处理，处理时间取平均值。不同集群规模下的数据处理时间如表 3 所示。

由表 3 得，随处理的数据量增大，处理时间相对增长。随集群规模增加，处理时间相对缩短。以 slave 数为 1 的处理时间作标准，计算集群加速比，如图 3 所示。

由图 3 所示，当处理的数据量较小时，集群不能对运算产生加速效果。因为集群节点间通讯开销等负面因素，集群处理用时甚至比单机处理用时更长。当处理数据的大小为 512 MB 及以上时，集群运算加速比随集群规模增加有明显提升。在本实验中，当集群 slave 数为 6，数据量为 2048 MB 时，计算加速比达到最大值。

6 节点的理论最高加速比为 6，实验中最大加速比与之相比仍有较大差距。分析原因：一是实验所用机器比较老旧，单机性能有限；二是运算中 reduce 数太少。实验中对 map 数和 reduce 数采用默认设定。通过查看运行日志，发现执行时系统自动为每个数据块分配了 1 个 map，但每次任务都只有 1 个 reduce，形成了处理瓶颈。为验证 reduce 数对处理速度的影响，设置如下实验：使用 6 节点集群处理 1024 MB 数据，分别设置 reduce 数为分别为 slave 数的 1、2、3 倍，另设 1 组对照。不同 reduce 数集群处理时间如表 4 所示。

实验结果与 reduce 数和 slave 数都为 1 的集群，对 1024 MB 数据的处理时间对比，得到的加速比如图 4 所示。

由图 4 得，随集群 reduce 数增加，集群运算加速比又有提高。当 reduce 数为 slave 数的 2 倍时，加速比达到最大值。可见 reduce 数的设定确实影响集群处理效率。在本实验中，reduce 数设为 slave 数 2 倍时，集群加速效果最好。

Table 3. Data processing time (ms)
表 3. 数据处理时间(ms)

数据量(MB)	1	2	4	6
极小	25,388	26,690	26,775	25,486
512	2,135,665	1,497,060	1,175,610	732,215
1024	4,288,210	3,247,376	2,140,819	1,556,942
2048	9,410,533	6,530,532	3,865,268	2,963,231
4096	18,787,454	11,406,213	8,040,599	5,943,141

Table 4. Data processing time (ms)
表 4. 数据处理时间(ms)

reduce 数	1	6	12	18
处理时间	1,556,942	1,356,639	1,166,978	1,422,078
加速比	2.754	3.161	3.675	3.015

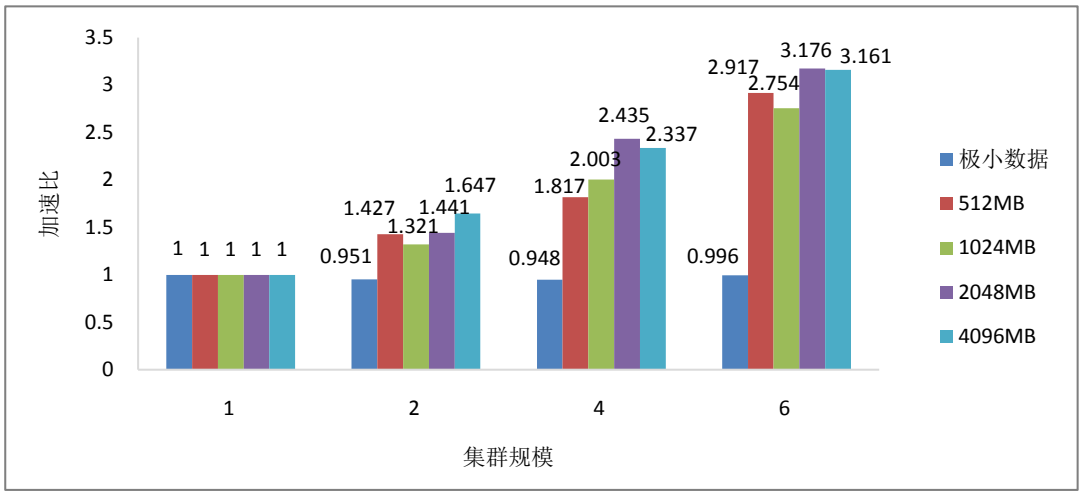

Figure 3. The speed-up ratio of cluster of different size
图 3. 不同规模集群加速比

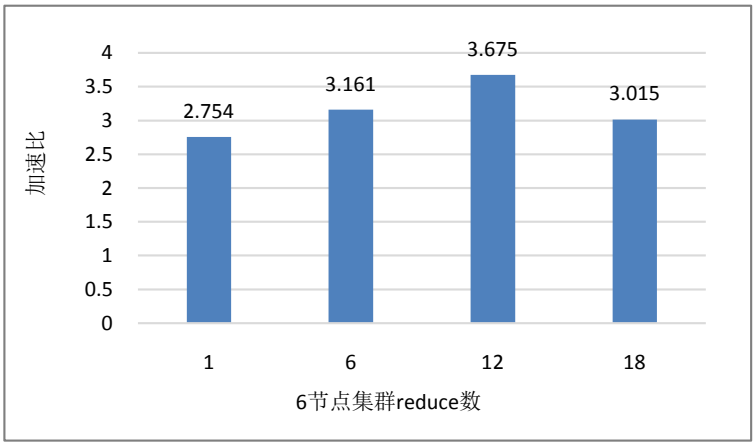

Figure 4. The speed-up ratio of cluster with different reduce number
图 4. 不同 reduce 数集群加速比

5. 结束语

本文研究对比了多种远程向 Hadoop 集群提交任务的方法，提出了一种自然语言处理通用平台的实现方案，降低了用户使用门槛，插件式的开发模式降低了功能开发难度，任务自动提交集群执行提高了运行效率。实验验证了本方案具有可行性。

参考文献 (References)

[1] 刘群, 张浩, 白硕. 中文信息处理开放平台的设计[C]//中国中文信息学会. 第一届学生计算语言学研讨会论文集. 北京: 中国中文信息学会, 2002: 7.

[2] 郎君, 刘挺, 李生, 张会鹏. 基于 XML 的开放式语言技术平台: LTP [C]//中国中文信息学会. 中文信息处理前沿进展——中国中文信息学会二十五周年学术会议论文集. 北京: 中国中文信息学会, 2006: 12.

[3] 百度云天智——基于百度大脑打造的人工智能平台. 自然语言处理[EB/OL]. https://cloud.baidu.com/product/bls.html?from=featuresBoard, 2017-05-30.

[4] 文智自然语言处理 NLP——腾讯云[EB/OL]. https://www.qcloud.com/product/nlp, 2017-05-30.

[5] 清华大学自然语言处理与社会人文计算实验室. [EB/OL].http://www.thunlp.org/, 2017-05-30.

[6] 语言云(语言技术平台云) [EB/OL]. http://www.ltp-cloud.com/, 2017-05-30.

[7] 北京理工大学的自然语言处理与信息检索共享平台[EB/OL]. http://ictclas.nlpir.org/nlpir/, 2017-05-30.

[8] Ghemawat, S., Gobioff, H. and Leung, S. (2003) The Google File System. *Proceedings of the* 19*th ACM Symposium on Operating Systems Principles*, 3-7. https://doi.org/10.1145/945445.945450

[9] 万至臻. 基于 MapReduce 模型的并行计算平台的设计与实现[D]:. 杭州: 浙江大学, 2008.

[10] 李云桃. 基于 Hadoop 的海量数据处理系统的设计与实现[D]: [硕士学位论文]. 哈尔滨: 哈尔滨工业大学, 2009.

[11] 姚卫国, 张东波. 基于 Hadoop 分布式平台的 Web 文本关键词提取方案[J]. 湘潭大学自然科学学报, 2016(2): 79-83.

[12] 崔富明. 基于 Hadoop 的文本聚类并行化研究[D]: [硕士学位论文]. 广州: 华南理工大学, 2016.

[13] 陈炎龙, 段红玉. 基于改进 Hadoop 云平台的海量文本数据挖掘[J]. 湖南师范大学自然科学学报, 2016(3): 84-88.

[14] 洪新军. 插件技术、分层技术应用于计算机软件技术中的价值探讨[J]. 电脑编程技巧与维护, 2016(2): 10-12.

[15] 王天舒. 浅谈插件技术在计算机软件技术中的应用[J]. 电脑知识与技术, 2017(2): 86-88.