Hadoop 的小图片处理技术及其在人脸特征提取上的应用

余 征,龚 勋,李天瑞,张钧波

(西南交通大学 信息科学与技术学院,成都 610031)
E-mail: yu487zheng@126.com

摘 要: 人脸识别算法作为一种较流行的生物识别技术,受到各界人士的关注. 随着人脸识别技术商业化的不断推进,越来越多的应用场景面临实时性的要求. 如何对海量人脸图像快速处理显得格外重要. Hadoop 是为处理大数据而设计,在数据密集型大数据处理上取得了良好成绩. 但它能否应对 I/O 密集型图像数据(海量人脸小图像)带来的挑战,还值得探究. 以人脸识别的重要阶段特征提取为例,根据人脸图片特点,结合 MapReduce 模型和 Hadoop 组合分片方法,设计并实现基于 Hadoop 的人脸特征提取方法,并分析 Hadoop 用于处理海量小图像文件时的性能. 实验证明,Hadoop 对海量小图像文件的处理表现良好.

关 键 词: 小文件; 海量图像; Hadoop; 大数据
中图分类号: TP399 **文献标识码:** A **文 章 编 号:** 1000-1220(2015)08-1891-05

Small Image Processing Techniques in Hadoop and its Application on Facial Feature Extraction

YU Zheng, GONG Xun, LI Tian-rui, ZHANG Jun-bo
(School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: Face recognition algorithm is a popular biometric technology, which has received much attention. With the commercialization of face recognition technology continues to advance, more and more application scenarios are facing the requirement of real-time. How to quickly deal with massive facial images is particularly important. Hadoop is designed for dealing with big data, and it has achieved a good performance in data-intensive processing. But it is still worth exploring whether it can deal with the challenge from I/O intensive image data (massive small face images). An important stage of face recognition feature extraction is taken as an example. According to the characteristics of face images, a facial feature extraction method based on Hadoop is developed by the MapReduce model and Hadoop combine split method. The performance of Hadoop for processing of massive small image files is evaluated. Experimental results show that the Hadoop performs well on the massive small image file processing.

Key words: small files; massive images; Hadoop; big data

1 引 言

在大数据背景下,多种云计算平台逐渐兴起. 其中 Hadoop 是当前较流行的云平台之一. 以其处理大数据的高可靠性、高扩展性、高效性和高容错性等优势受到人们的广泛关注[1]. 在电子商务、移动数据处理、医疗保健等方面都发挥着巨大作用[2-4].

人脸识别是目前应用最广泛的生物识别技术之一. 由于其友好性和非接触性等特点,在搜索罪犯、安全防务、身份识别等方面备受青睐[5]. 特征提取是人脸识别环节的重要阶段. 面对海量人脸数据,能否快速、高效的提取特征尤为重要. 若能利用 Hadoop 平台的高效性,解决人脸识别中的海量小图像文件数据带来的挑战,对提高人脸识别的实时性和拓宽 Hadoop 平台应用面都很有意义.

Hadoop 原本是为处理流式的大文件而设计. 但随着 Hadoop 应用的不断推广,小文件处理成为 Hadoop 平台的一个瓶颈[6],它严重地影响了 Hadoop 的性能. 同时,Hadoop 在图像处理方面没有很好的支持,它并没有为图像实现专门的接口类. 所以当前 Hadoop 不能很好地处理海量小图像文件数据.

对海量的小文件,目前,主要的解决方案是把大量的小文件合并成一个大文件并建立索引,然后应用于 Hadoop 上. 如: Liu 等提出一种在 HDFS 上提高小文件性能方法[7]; Dong 等人提出一种在 HDFS 上有效存取小文件的方法[8]; 张春明等提出一种 Hadoop 小文件存储和读取的方法 HIFM,合并小文件并生成索引[9]. 然而,这些方法改变了原始文件结构,不利于后续工作的使用和查看,且效率一般. 另外,图像文件不能简单地进行分割或组合.

Hadoop 本身也自带了一个处理小文件的方案,即组合分片方法[10]. 它不用考虑文件具体是被怎么组合分片的,且通过对大量小文件的合并大大减少了 Map 任务的启动数量,提升了处理速度. 问题是 Hadoop 没有为组合分片方法提供具

收稿日期: 2014-06-04 收修改稿日期: 2014-06-25 基金项目: 国家自然科学基金项目(F020502)资助. 作者简介: 余 征,女,1988 年生,硕士研究生,研究方向为云计算、计算机视觉、图像处理等; 龚 勋,男,1980 年生,博士,副教授,研究方向为认知图像处理、三维人脸建模、模式识别等; 李天瑞,男,1969 年生,博士后,教授,博士生导师,研究方向为智能信息处理、粗糙集、粒计算和云计算等; 张钧波,男,1986 年生,博士研究生,研究方向为粒计算、粗糙集、数据挖掘和云计算等.

体实现,且当前不支持对图像文件的处理.

本文以人脸特征提取为例,提出基于 MapReduce 的海量小图像文件处理的方案,并在 Hadoop 云平台上实现. 实验选取 FERET 人脸库[1](典型海量小图像文件数据集)作为测试集,进行单机与 Hadoop 并行方案的效率比较,并测试 Hadoop 在处理小图像文件的并行性能. 通过实验表明,Hadoop 在处理大数据上的优势同样也可应用到海量小图像文件的处理上.

2 Hadoop 简介

Hadoop 是一个分布式系统架构,由 Apache 基金会开发[11]. 用户可以不需要了解分布式底层细节情况下就可以开发分布式程序,且能充分利用集群的威力高速运算和存储. 它实现了一个分布式文件系统 Hadoop Distributed File System (HDFS)[12]. HDFS 部署在低廉的硬件上,有着高容错性,且适合超大数据集的应用程序[13]. 同时也实现了 MapRedcue 并行编程模型. MapReduce 可以充分利用集群的计算资源,对存储数据进行高效处理.

2.1 HDFS 架构

HDFS 集群由一个名字节点(NameNode)又称主节点和多个数据节点(DataNode)又称从节点组成[14],并提供应用程序访问接口. 数据节点管理对应节点的数据存储和读写. Hadoop 默认存储块大小为 64M. 像那些小于 64M,甚至十几 KB 的文件统称小文件.

名字节点是整个文件系统的管理节点,负责管理文件命名空间和管理文件系统,同时负责控制客户端的程序文件的操作和存储任务的分配与管理. 在 HDFS 中,文件或目录等均是以对象形式在内存中存储. 每个对象约使用 150 比特内存. 随着小文件数量的增加,耗费的内存也迅速增加. 大量名字节点内存的耗费,严重影响了 Hadoop 的应用性. 另外,访问大量小文件的速度远远慢于访问相同数据量的大文件. 原因是读取大量的小文件,需要不断地在数据节点之间跳跃. 最后,系统会把每个小文件作为一个 Slot,为每个小文件启动一个 Map 任务[15]. 由于启动 Map 任务相当耗时,所以大部分的处理时间是消耗在启动和释放 Map 任务上.

2.2 MapReduce 模型

MapReduce 是一个简单易用的软件框架,用于大规模数据集并行运算的编程模型,可并行处理 TB 级的数据集[16]. MapReduce 过程分为 Map 阶段和 Reduce 阶段. 每个输入分片会调用一个 Map 任务来处理. Map 从输入键值对,产生中间键值对,并将中间结果放在一个环形内存缓冲区内. Reduce 负责把不同 Map 任务传来的中间键值对中相同的键的键值对放到一个 Reduce 中处理,输出结果.

3 特征提取方法

目前人脸识别技术已相对成熟,特征提取方法众多,局部二值模式方法是其中一个较流行的特征提取方法. 本文采用局部二值模式算法的改进算法,即局部定向模式(Local Directional Pattern,简称LDP),作为特征提取方法[17]. LDP相比于局部二值模式更具有鲁棒性. LDP 特征是将一个 8 位的二进制码分配给输入图像的每个像素. 其计算方法是:依次取该图片的每个像素值及其 3×3 邻域的灰度值与 8 个给定 Kirsch 算子[18]相卷积,将卷积结果的高响应位置 1,其余位取 0,得到一个二进制串,二进制串的值即为这个像素点的 LDP 值. 具体计算流程如图 1 所示.

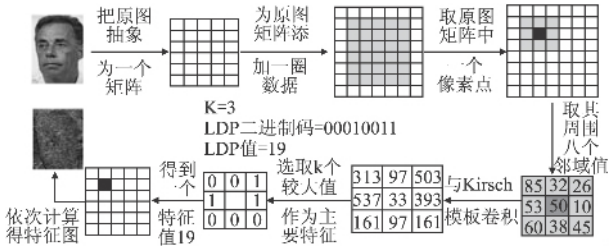

图 1 LDP 计算方法流程图

Fig. 1 Flowchart of the LDP calculation method

4 基于 Hadoop 的小图像文件处理

4.1 组合分片方法简介

组合分片使用组合文件输入格式类作为输入格式,多文件输入分片作为它的输入分片类型,用组合文件记录读取器来生成键值对.

组合文件输入格式类是一个抽象类. Hadoop 没有提供它的具体实现. 组合文件输入格式类可以将来自多个文件的分片组合成一个用户理想大小的分片. 而小图像文件数据集中的每一个文件都相当于一个小的分片,组合文件输入格式类会将多个小文件合并成一个大的输入分片. 这个合并后的输入分片的大小由参数 maxSplitSize 来指定,理想值为数据块大小 64M. 通过对输入小文件的合并,可减少输入文件的数量,从而降低对名字节点内存的消耗,同时也可减少 Map 任务的启动数量.

多文件输入分片(CombineFileSplit)是组合文件输入格式类相对应的输入分片类型. 顾名思义,FileSplit 就是每个文件的输入分片,CombineFileSplit 就是把这些输入分片组合起来的组合分片. Hadoop 会为每一个组合输入分片启动一个 Map 任务.

组合文件记录读取器(CombineFileRecordReader)是组合文件输入格式类对应的记录读取器. 组合文件记录读取器任务是利用 Java 反射机制为来自不同文件的每个分片创建相对应的记录读取器.

4.2 组合图像分片方法

对于图像,由于 Hadoop 没有为其实现 I/O 接口,所以需重写图像 I/O 文件格式类 Image,继承 Writable 通用接口,实现图像的序列化和反序列化,使 Hadoop 可以支持图像处理.

组合图像文件输入分片的具体实现:首先,重写适合于图像文件的组合图像文件输入格式类(CombineImageInputFormat)继承组合文件输入格式类. 在该类中创建组合文件记录读取器,它为每个图像文件都要创建一个图像记录读取器. 由于文件太小且图像文件不易分割,在输入格式文件里设置

[1] FERET 数据集下载自 http://www.nist.gov/itl/iad/ig/colorferet.cfm

输入文件不被分片. 然后,重写图像记录读取器继承记录读取器. 图像记录读取器会为组合分片里的每一个文件分片生成一个键值对,键为图片文件路径,值为图片文件. 组合图像输入格式类伪代码如下:

```
//图像组合文件输入格式类继承组合输入文件格式类
CombineImageInputFormat extends CombineFileInputFormat
<Text,Image>{
  //设置记录读取器为组合文件记录读取器
  RecordReader<Text,Image> createRecordReader(){
return new CombineFileRecordReader <Text,Image>(); }
  //设置不对文件分片
  isSplitable(JobContext context,Path file) {return false;}
}
```

图像记录读取器伪代码:

```
ImageRecordReader extends RecordReader<Text,Image> {
  ImageRecordReader(CombineFileSplit split,TaskAttemptContext con-
text,Integer index){
  //获取一个文件分片
  CombineFileSplit ThisSplit=split;
  //对缓冲区中的图像解码,image 用作键值对的值
  image=new Image(cvDecodeImage(cvMat(1,b.length,CV_8UC1,
new BytePointer(b)),iscolor));
  //获取当前输入文件分片的路径,用作键值对的键
  filePath=ThisSplit.getPath(index);
}
```

通过对分片的组合,Hadoop 为每一个组合后的大分片启动 Map 任务来处理,其将大大减少启动 Map 任务的数量.

4.3　Hadoop 处理小图像文件方法

主函数: 设置组合文件输入格式的参数值,包括组合分片的大小的上界值、单个数据结点输入分片的最小值和单个机架上输入分片的最小值. 设置多输入路径,每张图片都有一个唯一的路径,要把所有图片读进来,需要进行多输入路径的设置. 设置多输出路径是为了把每一个图像特征文件按原结构存储,方便后续的使用.

Map 及 Reduce 函数设计: MapReduce 把输入看作(*Key*, *Value*)键值对. 这里的 *Key* 为图片路径,*Value* 为图片文件. 在 Map 阶段,从记录读取器那里得到键值对,对输入的值提取 LDP 特征,生成新的中间键值对. 产生的中间键值对,键仍为图片路径值,值为特征图. 在 Reduce 阶段,用输入键计算输出存放路径,把特征图以新的键为路径保存,采用多路径输出. 基于 MapReducer 特征提取伪代码如下:

主函数:

```
//设置组合文件输入格式的参数
SPLIT_MINSIZE_PERNODE=64M;
SPLIT_MINSIZE_PERRACK=64M;
"mapreduce.input.fileinputformat.split.maxsize"=64M);
//设置多输入路径
FileInputFormat.addInputPaths(job,pathlist);
//设置多输出路径
MultipleOutputs.addNamedOutput(job,"name",ImageOutputFormat.
class,Text.class,Image.class);
```

Map 函数:

```
//key: 文件路径 value: 图像文件
Map(Text key,Image value)
  //设定特征模板属性
  d=-1,k=-3;
  //为原始图片加一圈数据(使边缘数据可被模板覆盖)
  DstImage=AddData(value);
  //为除加的一圈数据外的所有数据点依次计算 LDP 特征值
  LdpImage=ldp(DstImage);
  context.write(key,LdpImage);
```

Reduce 函数:

```
//key: 路径//value: 特征图
Reduce(Text key,Image value)
//计算新的存放路径,并多路径输出
New Path(key);
context.write(Text key,Image value);
```

Hadoop 处理小图片数据方法的示意图如图 2 所示.

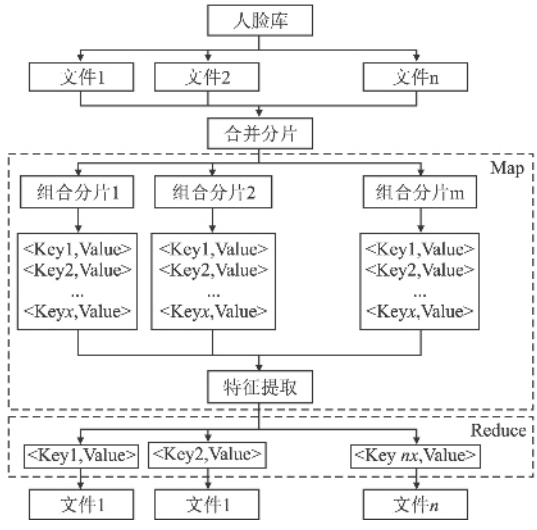

图 2　Hadoop 处理小图片数据方法图

Fig. 2　Methods of small image data processing in Hadoop

5　实验结果分析

5.1　测试集

本文以人脸特征提取算法为例,测试 Hadoop 处理小图像文件的性能. 采用 FERET(Face Recognition Technology 简称 FERET)人脸库. FERET 人脸库是由美国 FERET 项目所创建,共有 14051 张多姿态、光照等条件下的灰度人脸图像,是人脸识别领域应用最为广泛的数据库之一. 其中每张图片均是宽 256 像素、高 384 像素、大小约 100KB、bmp 格式的灰度图片. 从

表 1　数据集

Table 1　Data Sets

测试集/MB	文件数/张	测试集/MB	文件数/张
100	1064	700	7399
200	2128	800	8452
300	3192	900	9513
400	4256	1000	10577
500	5327	1100	11634
600	6342	1200	12712

14051 张(约 1290MB)图像中,分别取 100MB、200MB、300MB、…、1200MB 作为测试集. 详细信息如上页表 1 所示.

5.2 实验环境

本实验是在一台个人超级计算机(ServMax PSC-2n)上,通过安装虚拟机,配置了 5 个 Linux 系统. 搭建 5 个 Hadoop 节点的虚拟机集群,分别是一个 Master,5 个 Slave. Master 节点也同时作为数据节点. 计算机配置如表 2 所示.

表 2 计算机配置

Table 2 Computer configuration

配置	参数
处理器	Intel(R) Xeon(R) CPU E5620 @ 2.4GHz 2.4GHz(2 处理器)
内存	48GB
操作系统	64 位 Win7

每个节点系统配置如表 3 所示. 其中 OpenCv 是常用的视觉库,可以通过 JavaCv 在 Java 平台方便地对其调用. 对每个 Slave 节点设置虚拟机内存 4G、一处理器、一线程,对 Master 节点设置虚拟机内存 8G、二处理器、二线程. 设置每个节点同时可运行的最大 Map 槽数为 4,Reduce 槽数为 1.

表 3 系统配置

Table 3 System configuration

节点配置	参 数
操作系统	ubuntu12.10
平台	Hadoop-1.2.1
软件	Jdk-1.7.0_25 JavaCv-0.2 OpenCv2.4.6.1
实验对象	FERET 人脸数据库

本文的对比实验是在同一台计算机上,目的是测试表 1 中的数据集在单机下的处理效率,并与上述并行方法进行对比. 单机实验环境表 4 所示.

表 4 单机实验环境

Table 4 Single experiment environment

单机环境	参 数
操作系统	64 位 Win7
工具	Visual Studio 2008
所需库	OpenCv2.1
实验对象	FERET 人脸数据库

5.3 实验结果

本实验,采用并行算法的三种评价方法:加速比(Speedup)、可扩展性(Scaleup)、规模增长性(Sizeup)作为评价指标[19].

加速比实验:保持数据集不变,增加计算机数目. 理想的并行算法,受节点间网络数据传输的影响,应该在 Speedup 上呈线性增长. 如图 3 中,横坐标为节点个数,从 1 个节点到 5 个节点. 纵坐标为 Speedup 值. 图中灰色标准线即为理想情况下的增长线,其余四条实验线是分别在 200MB、400MB、600MB、800MB 的数据集下得到的 Speedup 结果.

从图 3 可知,四条实验线的增长趋势接近于线性增长. 数据集越大越接近标准线. 其中存在两个特殊点说明如下:数据集 200M 和 400M 的实验线在第四个节点和第五个节点处,从图上可看出其没有很好地呈线性增长,而是几乎等值的. 这个现象的发生是因为这个数据集在 4 节点情形和 5 节点情形下,各节点任务分配情况类似,运行时间就相差不多. 以 200M 数据集为例说明,前提:所有节点配置的 Map 槽位数均为 4,组合分片大小设置 64M,即 200M 被分为 4 个待执行任务. 在 4 节点时,每个节点均分配到 1 个 Map 任务;在 5 节点时,其中 4 个节点,每个节点一个 Map 任务,另外一个节点没有 Map 任务. 因而,在 200M 时,4 个节点与 5 个节点运行的差别在于 5 节点情形多出来一个空闲节点,其余节点任务情况相同,所以它们的运行时间相差无几. 由此,可以说明所提出的算法在 Speedup 上呈现良好的特性.

可扩展性实验:扩大数据集的同时增加计算机的数目. 理想算法为 Y=1. 以此来测试算法能否应对不同大小数据集. 如图 4 所示,横坐标为节点数,纵坐标为 Scaleup 值. 实验选取(1 节点,100MB)、(2 节点,200MB)、(3 节点,300MB)、(4 节点,400MB)、(5 节点,500MB)五个实验点.

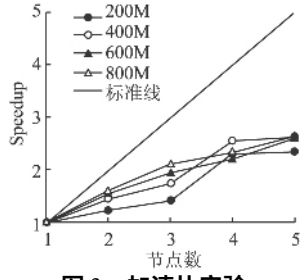

图 3 加速比实验

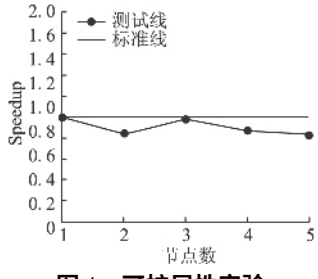

图 4 可扩展性实验

Fig. 3 Speedup experiments Fig. 4 Scalability experiments

从图 4 可知,不同节点的执行有所差异,都在 Y=1 线附近且浮动幅度相对不大,证明其可以适应不同大小的图像数据集.

规模增长性实验:保持计算机数目不变,扩大数据集. 执行的曲线在 Y=X 线下方,则说明算法良好,是用来测试算法本身的时间复杂度.

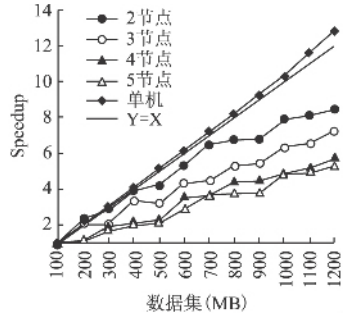

图 5 规模增长性实验

Fig. 5 Sizeup experiments

如图 5 所示,横坐标为数据集大小,纵坐标为 Sizeup 值. 各条线分别代表单机下非并行环境的实验结果和集群下二节点、三节点、四节点、五节点的实验结果. 可以看出单机下的执行时间的 Sizeup 曲线是在 Y=X 上方的,且随着数据集的增加时间也成线性增长趋势. 而并行方法的结果均在 Y=X 线下方,对 Sizeup 指标来说,若并行执行的曲线在 Y=X 线下方则证明并行良好. 由图 5 可以看到,二个节点的情况已经比单

机执行效率高,三个节点比二节点又有明显提高,四五节点有更明显的提升. 说明随着节点数的增加,算法的并行结果趋势发展更好. 因此算法在 Sizeup 上表现良好,通过增加节点个数可以有效地降低算法时间复杂度,与单机相比,效率有大幅提高.

以上实验说明本文所提出的基于 Hadoop 平台的算法可以有效地提高小图像文件的处理效率,且在小图像文件数据集上表现出良好的并行性能.

6　结束语

随着大数据的兴起,Hadoop 成为一种流行趋势,在大数据处理上卓见成效,但对于海量小图像文件的处理却略显乏力. 本文以人脸识别环节中的特征提取阶段为例,设计并实现了基于 Hadoop 的人脸特征提取方法,并在 FERET 人脸库数据集上做测试. 实验结果表明,通过使用组合分片方法,Hadoop 很好地发挥出处理大数据的并行能力. 随着数据集规模的增长,Hadoop 处理时间呈缓慢增长趋势,且在各并行性能测试指标上表现良好,相比于单机的效率有较大的提升. 所以,本文所提出的利用 Hadoop 来处理海量小图像文件方法是可行且有效的. 今后我们的工作主要是大数据环境下基于 Hadoop 平台来实现人脸实时识别,以达到实用的目的.

References:

[1] Dong Xi-cheng. Hadoop internals: in-depth study of MapReduce [M]. Beijing: China Machine Press,2013.

[2] Kuo A M. Opportunities and challenges of cloud computing to improve health care services [J]. Journal of Medical Internet Research,2011,13(3):97-98.

[3] Ma Lin-shan,Zhao Qing-feng,Xiao Xin-guo. Research on mobile cloud information services model based on Hadoop platform [J]. Information Science,2013,31(4):28-32.

[4] Wei Fei-fei. ECLHadoop: efficient big data processing strategy based on Hadoop for electronic commerce logistics [J]. Computer Engineering & Science,2013,35(10):65-71.

[5] Li Wu-jun,Wang Chong-jun,Zhang Wei,et al. A survey of face recognition [J]. Pattern Recognition and Artificial Intelligence,2006,19(1):58-66.

[6] Yuan Yu,Cui Chao-yuan,Wu Yun,et al. Performance analysis of Hadoop for handling small files in single node [J]. Computer Engineering and Applications,2013,49(3):57-60.

[7] Liu Xu-hui,Han Ji-zhong,Zhong Yun-qin,et al. Implementing WebGIS on Hadoop: a case study of improving small file I/O performance on HDFS [C]. IEEE International Conference on Cluster Computing,IEEE,2009:1-8.

[8] Dong Bo,Qiu Jie,Zheng Qing-hua,et al. A novel approach to improving the efficiency of storing and accessing small files on hadoop: a case study by powerpoint files [C]. IEEE International Conference on Services Computing (SCC),IEEE,2010:65-72.

[9] Zhang Chun-ming,Rui Jian-wu,He Ting-ting. An approach for storing and accessing amall files on Hadoop [J]. Computer Applications and Software,2012,29(11):95-100.

[10] Zhang Xin. Depth cloud computing: Hadoop source code analysis [M]. Beijing: China Railway Publishing House,2013.

[11] Shvachko K,Hairong K,Radia S. The Hadoop distributed file system [C]. Incline Village,IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST),NV,2010:1-10.

[12] Shvachko K,Kuang H,Radia S,et al. The hadoop distributed file system [C]. IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST),IEEE,2010:1-10.

[13] Tom White. Hadoop: the definitive guide (2nd Edition) [M]. USA: O´Reilly Media,2011.

[14] Wang Feng,Qiu Jie,Yang Jie,et al. Hadoop high availability through metadata replication [C]. Proceedings of the First International Workshop on Cloud Data Management,ACM,2009:37-44.

[15] Wang Li-jun,Huang Yong-feng,Chen Ji,et al. Institute for network science and cyberspace [C]. Beijing: Tsinghua University,2013 IEEE 15th International Conference on e-Health Networking,Applications & Services (Healthcom),IEEE,2013:1-6.

[16] Ekanayake J,Pallickara S,Fox G. Mapreduce for data intensive scientific analyses [C]. IEEE Fourth International Conference on eScience,IEEE,2008:277-284.

[17] Jabid T,Kabir M H,Chae O. Local directional pattern (LDP)—a robust image descriptor for object recognition [C]. 2010 Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS),IEEE,2010:482-487.

[18] Kim D,Lee S,Sohn M. Face recognition via local directional pattern [J]. International Journal of Security & its Applications. 2013,7(2):191-200.

[19] Zhao Wei-zhong,Ma Hui-fang,He Qing. Parallel k-means clustering based on MapReduce [M]. Cloud Computing,Springer,2009:674-679.

附中文参考文献:

[1] 董西成. Hadoop 技术内幕: 深入解析 MapReduce 架构设计与实现原理 [M]. 北京: 机械工业出版社,2013.

[4] 魏斐翡. ECLHadoop: 基于 Hadoop 的有效电子商务物流大数据处理策略 [J]. 计算机工程与科学,2013,35(10):65-71.

[5] 李武军,王崇骏,张　炜,等. 人脸识别研究综述 [J]. 模式识别与人工智能,2006,19(1):58-66.

[6] 袁　玉,崔超远,乌　云,等. 单机下 Hadoop 小文件处理性能分析 [J]. 计算机工程与应用,2013,49(3):57-60.

[9] 张春明,芮建武,何婷婷. 一种 Hadoop 小文件存储和读取的方法 [J]. 计算机应用与软件,2012,29(11):95-100.

[10] 张　鑫. 深入云计算 Hadoop 源代码分析 [M]. 北京: 中国铁道出版社,2013.

欢迎点击这里的链接进入精彩的Linux公社 网站

Linux公社（www.Linuxidc.com）于2006年9月25日注册并开通网站，Linux现在已经成为一种广受关注和支持的一种操作系统，IDC是互联网数据中心，LinuxIDC就是关于Linux的数据中心。

Linux公社是专业的Linux系统门户网站，实时发布最新Linux资讯，包括Linux、Ubuntu、Fedora、RedHat、红旗Linux、Linux教程、Linux认证、SUSE Linux、Android、Oracle、Hadoop、CentOS、MySQL、Apache、Nginx、Tomcat、Python、Java、C语言、OpenStack、集群等技术。

Linux公社（LinuxIDC.com）设置了有一定影响力的Linux专题栏目。

包括：**Ubuntu 专题** **Fedora 专题** **Android 专题** **Oracle 专题** **Hadoop 专题** **RedHat 专题** **SUSE 专题** **红旗 Linux 专题** **CentOS 专题**