doi: 103969/j. issn. 0490-6756. 2017. 01. 011

基于 Spark 的并行医学图像处理研究

兰云旭[1], 王俊峰[1], 唐　鹏[2]
(1. 四川大学计算机学院, 成都 610065; 2. 西南交通大学电气工程学院, 成都 610031)

摘　要: 随着医学图像规模的不断增长,为了快速且有效的处理医学图像并使各类图像处理算法得到应用. 文章将传统的医学图像处理方法与 Spark 整合起来,提出了基于 Spark 的并行医学图像处理方法. 首先,采用基于二进制的图像预处理转换方法,存储图像到分布式文件系统 HDFS 中;其次,应用传递函数的方法,避免了图像处理算法进行 MapReduce 转化,实现了快速的通用图像并行处理;最后,以肺叶 DR 图像分割算法为实例证明了基于 Spark 医学图像并行处理有较好的适应性和较高的效率,并适应大规模图像的并行处理.

关键词: 医学图像处理; Spark; 并行处理; 大数据
中图分类号: TP391.4　　**文献标识码**: A　　**文章编号**: 0490-6756(2017)01-0065-06

Parallel processing researches of medical image based on Spark

LAN Yun-Xu[1], *WANG Jun-Feng*[1], *TANG Peng*[2]
(1. College of Computer Science, Sichuan University, Chengdu 610065, China;
2. School of Electrical Engineering, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: With the growing number of medical images, in order to effectively and quickly process medical images and make all kinds of medical images processing method get quickly transplant and apply. A medical image parallel processing method based on Spark is proposed, which integrate traditional medical image processing method with the Spark. Firstly, preprocess images base on binary conversion method, store images in the HDFS distributed file system. Secondly, implements a transfer function way, which realized image parallel processing quickly, avoid convert the image processing algorithms to the MapReduce algorithms. Thirdly, according to the experiment, medical image parallel processing method based on Spark has preferable adaptability and efficiency, which can adapt to the large-scale image processing.
Keywords: Medical images processing; Spark; Parallel processing; Big data

1　引　言

随着医学图像规模的不断增长,计算机辅助诊断技术[1]的深入研究,通过对医学图像进行高效处理,实现计算机辅助诊断的方法越来越多. 现今已提出肺部分割、病灶检测等多种研究方法以及原型系统[2]. 针对庞大图像规模的现状,提出了诸如基于 GPU、MPI[3] 等主流图像并行处理方案,由于 GPU 计算需要较高的构造成本,MPI 的编程过于复杂,如何在有效利用已提出的各类医学图像处理的前提下,结合云计算、大数据处理技术的发展成果,实现快速且有效的医学图像处理与应用毫无疑问成为现阶段亟待解决的问题之一.

大数据时代的来临,使得大数据处理技术已应

收稿日期: 2016-02-28
基金项目: 国家重大科技专项(2012ZX10004-901); 四川省科技支撑计划项目(2013SZ0002, 2014SZ0109)
作者简介: 兰云旭(1992－), 男, 江西赣州人, 硕士生, 研究方向为计算机应用.
通讯作者: 王俊峰. E-mail: wangjf@scu. edu. cn

用到诸多领域,包括网络日志分析、用户行为分析、各类流量分析等,而在大数据处理技术中则以 Hadoop[4] 及 Spark[5] 最为活跃. 大数据技术现已逐渐应用到医学领域,包括医疗数据仓库的建立、大规模医学图像的处理等. 在医学图像并行处理方面,提出了不同于 GPU、MPI 等方案的基于 Hadoop 架构的图像并行处理方法,文献[6]提出了通过设计基于 MapReduce[7] 的图片处理格式运用 Hadoop 架构实现并行处理,文献[8]则提出了应用 MapReduce 计算模型,实现对大规模的医学图像处理. 基于 Hadoop 架构的图像处理方法均是通过 MapReduce 来实现图像并行处理,其高吞吐也带来了高延时,对于任务及资源的分配缺乏公平性,没有考虑异构环境以及区分长任务短任务,影响图像处理的效率,同时,运用 Hadoop 架构进行图像处理涉及到各类医学图像处理算法的转化,涉及大量的编程工作. 相对于 Hadoop 架构,Spark 提供了更好的数据重用机制、较强的数据集抽象、丰富的编程工具,解决了 Hadoop 架构的高延时缺陷,引入弹性数据集概念代替 MapReduce,提供了 Scala、Java、Python 三种 API 实现与其他程序的整合,并通过全局资源来进行任务及资源的有效分配,提高图像处理效率.

本文提出基于 Spark 的并行医学图像处理方案,其由图像处理接口、Spark 集群、分布式文件系统三部分构成,通过图像处理接口对图像预处理转换后读入 Spark 集群进行并行处理,得到结果存储到分布式文件系统. 图像处理接口通过传递函数的方式实现与 Spark 整合,可添加任意图像处理算法实现图像的并行处理. 通过实验证明,方法可实现图像的并行处理,且适应大规模图像的处理.

2 Spark 平台

Spark 是由 UC Berkeley AMP lab 所开发,是现今大数据领域内较为流行、高效的并行计算框架. 其在 MapReduce 的基础上,提出了一种新的弹性分布式数据集 (Resilient Distributed Datasets RDD)作为内存抽象来实现类似 MapReduce 模型所做的工作,并定义了各种操作类型来实现 RDD 的转换或返回结果. Spark 操作类型可分为转换操作(Transformations)和动作操作(Actions),转换操作是惰性求值的,即是通过构造一个 RDD 间相互依赖的有向无环图(DAG),并在最后通过动作操作来触发这个任务返回结果. Spark 中的任务运行流程如图 1 所示.

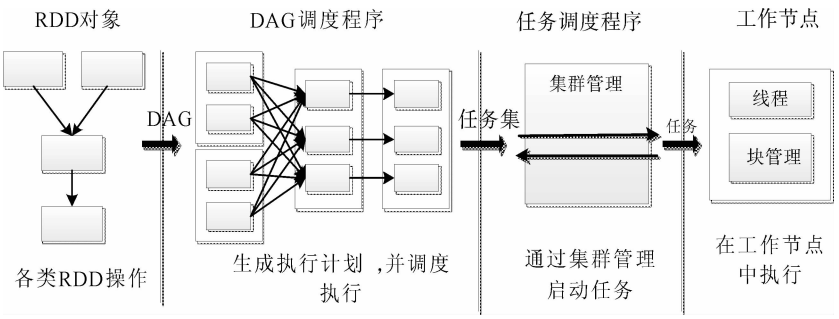

图 1 Spark 任务运行流程
Fig. 1 Spark task flow

Spark 平台可简单认为由主驱动程序、集群管理程序、节点执行程序构成. 主驱动程序来进行集群上程序的初始化,定义 RDD,并链接集群管理程序;集群管理程序负责任务的统一调度,根据用户定义的各实例配置及各工作节点的负载情况,把任务分成相应的分片合理分配到各工作节点上执行;节点执行程序主要负责对任务的处理,并返回处理结果.

3 基于 Spark 的并行处理架构与流程

基于 Spark 的医学并行处理架构可分为分布式文件系统(HDFS[9])、Spark 集群、图像处理接口三个部分,整体构架如图 2 所示.

1) 分布式文件系统 HDFS:负责预处理后的图像及各类输出结果储存,并支持根据图像的规模不断增加存储节点,保证读取速度及存储规模.

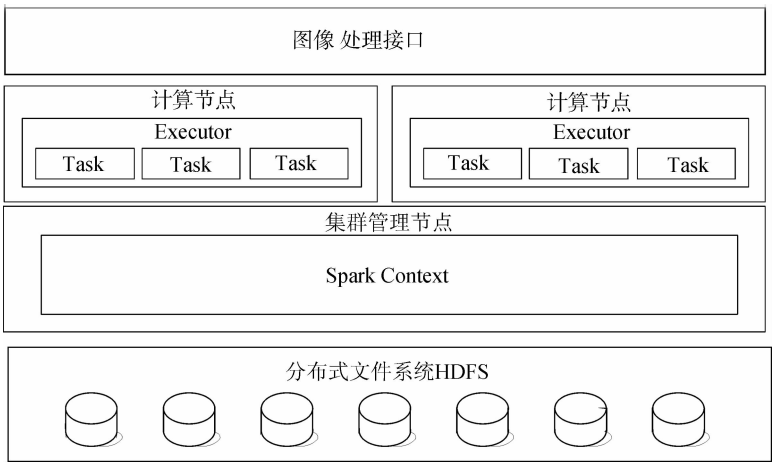

图 2　整体构架
Fig. 2　Overall architecture

2）Spark 集群：负责图像的读入及并行处理，还包括集群上的作业调度及资源分配.

3）图像处理接口：用于实现图像的预处理、转换操作以及图像处理算法与 Spark 主驱动程序的整合.

图像处理接口将预处理后的图像存入 HDFS，Spark 读取 HDFS 中的图像文件并将每张图像创建对应的 RDD，之后对 RDD 进行转换并接收图像处理接口传递的图像处理方法，进而触发动作操作返回结果，实现图像的并行处理，并输出结果到 HDFS. 处理流程如图 3 所示.

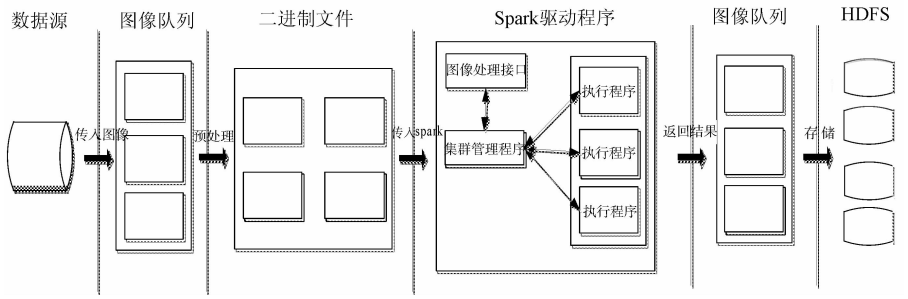

图 3　并行处理流程
Fig. 3　Parallel processing flow

3.1　图像的预处理与转换

在现有的海量医学图像中，多数图像现阶段存储在医院的各类服务器文件系统中，导致存在大量异构图像数据，同时，Spark 并不支持直接读入 DRI、JPG 等格式的图像数据源并创建分布式数据集. 为了让异构图像数据能够被各工作节点访问并且被 Spark 读入，本文通过定义的图像处理接口对图像数据源进行预处理转换操作，将每个图像依据像素转换为二进制值文本文件并写入 HDFS 分布式文件系统.

对于远端图像数据源，可以使用 Spark Streaming[10] 通过读入网络流的方式来进行远端图像的预处理与转换，Spark Streaming 进行预处理需依赖 Spark 集群实现，而若在图像接收不连续的情况下，使用此方法无疑加重了 Spark 集群的负担，图像数据均需要先由 Spark 集群接收并进行作业调度，接收及调度的成本远大于预处理的成本. 而把不经过预处理的图像数据通过流的方式直接传入 Spark 集群，异构数据不能被有效处理整合，对网络带宽的要求也有所增加，无形加大了图像处理的难度.

通过把图像转换为二进制值文本文件并存储，即保证了图像的读取效率及质量，又保证了图像便于被各节点访问，解决了图像源数据的异构性问题. 二进制值文本文件在确定编码方式的情况下可通过读入二进制流的方式恢复为图像，在许多图像数据库中均采用这种方式来进行图像的存储.

3.2 图像并行处理的实现

Spark 对于文本文件提供了统一的外部数据源文本接口,将整个文本文件读成一个行集合,即定义了一个基本的 RDD,并在之后进行一系列的 RDD 操作.

将图像对应的二进制值文本文件批量读入 Spark 平台.在文件读入后,每个文件对应一个基本的 RDD,即每个图像对应一个 RDD.图像的处理操作即可视为简单的 RDD 转换操作.

Spark 中 RDD 的转换操作,实质上可以看作是在既定的方法下进行的集合或类型的转换操作,本文提出传递所需的图像处理算法函数的方式来构造 RDD 的转换操作,进而实现 Spark 下的并行图像处理.在图像处理接口上充分利用这一点,实现了各类医学图像算法与 Spark 平台的整合.具体的实现流程如图 4 所示.

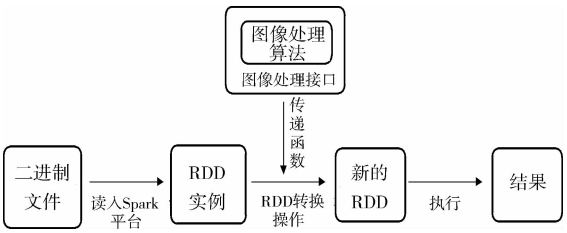

图 4 函数传递流程

Fig. 4 Function transfer flow

在 Spark 下的驱动程序可用如下方式实现:

```
sc=SparkContext(appName="images1");
init=sc.textFile("hdfs://image1.txt");
img = init. map (lambda x: ImageAlgorithm (x));
img.collect();
def ImageAlgorithm(x):
…(此处添加所需图像处理算法)
```

通过函数传递来实现整合,保证了 Spark 平台和图像处理接口都是独立的模块.图像处理接口起到万能接口的作用,接口会在 RDD 转换过程中将图像处理算法传入 Spark 平台.根据不同的需要,在图像处理接口上可编写不同的图像处理算法进行不同的图像处理操作.

在 RDD 转换完成后,触发动作操作执行实例,Spark 批量处理传入图像,并根据图像数量决定任务数量,再根据集群的规模,把任务负载均衡地分配到各个工作节点.各工作节点处理完毕后将结果写入分布式文件系统(HDFS)或分布式数据库(HBase[11]).通过这种方式,保证了图像与任务的一一对应关系,同时,避免了在进行图像并行处理时,需要对图像进行序列化,快速有效地读入各图像.

3.3 通用的并行图像处理方法

通过传递函数的方法把 Spark 和图像处理接口相整合,保证了基于 Spark 的并行图像处理方法对各类图像处理算法具有通用性,改变了图像处理算法转换为并行算法需要根据 MapReduce 模型进行转化的现状,这是很多并行处理方法所不具备的,大大缩小了代码编写学习成本,拥有"即写即用"的特性.以 Python 下的图像处理算法为例,Python 的面向对象、可移植、丰富的图像处理工具、弱数据类型等特性,使得它被各类图像处理方法所青睐,用 Python 编写的图像处理算法已应用到各类医学图像处理上.这些算法只需在图像处理接口上添加,图像处理接口即通过传递函数方式将图像处理算法传递到 Spark 主驱动程序上,实现了图像的并行处理.

通过此方法,对于任意图像处理算法,都可快速实现图像并行处理,对原有图像处理算法的效率提升有积极意义.毫无疑问,基于 Spark 的并行医学图像处理成为了一种通用的并行处理方法,为搭建通用的并行处理图像系统提供了可能.

4 在医学图像处理上的应用

本文提出的基于 Spark 的并行图像处理是一种通用的并行处理方法,可在图像处理接口上添加任意图像处理算法,实现图像的并行处理.在医学领域,计算机辅助诊断技术可以通过 DR 胸片来检测胸腔疾病,而进行肺轮廓的分割是进行胸腔疾病检测的基础.本文在图片处理接口上添加了一种基于样例组合的肺轮廓分割的图像处理方法,其主要思想即是首先通过人工建立肺叶轮廓的样例集,并获取每个轮廓的点坐标序列,分别取肺叶轮廓的上半部分和下半部分作为肺轮廓匹配的样例模板.之后读取 DR 图像,用多尺度滑动窗方式从 DR 图像中提取出若干区域子图像,采用距离搜索从大量匹配结果中收集最佳的若干个上下肺轮廓的候选匹配,用 3 阶 Bezier 曲线拟合上下肺轮廓,并根据 Beizer 曲线的控制点判断其可能性,将上下肺轮廓线的匹配值和 Beizer 轮廓的可能性相组合,得到综合评价,选取最佳评价的候选轮廓作为输出,分割流程如图 5 所示.

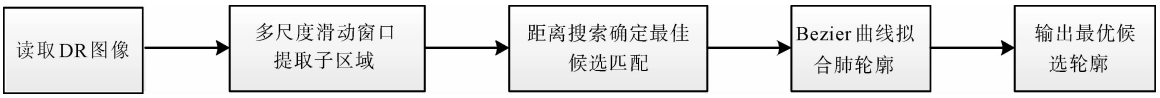

图 5　肺轮廓分割处理流程

Fig. 5　Lung segmentation processing flow

4.1　实验环境及数据来源

基于 Spark 的医学图像并行处理方法在实验集群中实现. 实验集群由 1 个主节点，8 个从节点构成. 其中主节点在 Spark 中担任 Master，从节点担任 Worker. 操作系统均为 Centos7，硬件配置均为 Intel Core(TM) i3-3240 CPU@3.40 GHz，2 GB DDR3 内存. 所有节点均已成功配置 Spark 1.5.0并行处理框架、Python 2.7 以及各类 Python 下的图像处理包，如 Scipy、Matplotlib 等. 实验中使用 Spark 的 Python 接口，通过 Python 语言撰写 Spark 主驱动程序及医学图像实现方法. 任务部署模式采用 Spark Standalone. 网络环境为局域网内网环境并通过主机名形成内网 IP 映射.

表 1　集群配置

Tab. 1　Cluster configuration

节点名@IP 地址	角色	CPU	内存
Node1@192.168.10.216	Master	2 核	2GB
Node2@192.168.10.217	Worker	2 核	2GB
Node3@192.168.10.218	Worker	2 核	2GB
Node4@192.168.10.219	Worker	2 核	2GB
Node5@192.168.10.220	Worker	2 核	2GB
Node6@192.168.10.221	Worker	2 核	2GB
Node7@192.168.10.222	Worker	2 核	2GB
Node8@192.168.10.223	Worker	2 核	2GB
Node9@192.168.10.224	Worker	2 核	2GB

本文中所有 DR 图像数据来自国家“十二五”重大传染病专项四川示范区信息平台，平台共有图像数据 30 万余张，每天新增体检 DR 图像数据 1000 余张，每张图像以 JPEG 及 DRI 图像格式存于文件系统中，图像大小在 10 M 以内，本文选取其中 10 天新增的共计 10000 张大小相似的需要处理图像进行实验.

4.2　实验设计与内容

实验通过在方法上添加基于样例组合的肺轮廓分割方法的 Python 实现算法来进行，并根据实验的目的选取若干医学 DR 图像作为样本进行实验.

4.2.1　*与传统方法的性能对比*　把线性传统处理方式和文中提出的并行处理方式进行对比，在 8 个工作节点的情况下，图像数 N(单位：张)与时间 T(单位：min)关系如图 6 所示.

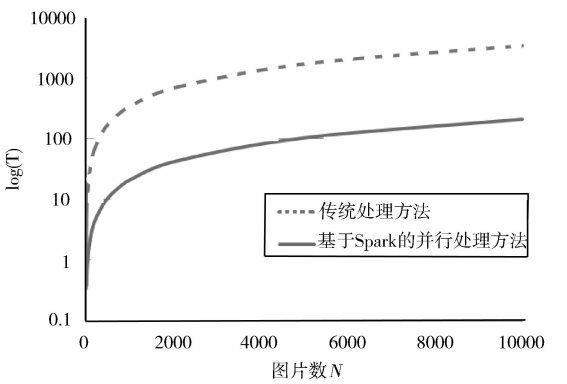

图 6　传统处理方法与并行处理方法测试结果

Fig. 6　Test results between tradition processing and parallel processing

从图 6 中可以看出，基于 Spark 的并行处理方法相对于传统方式，在多张图片处理上有着明显的优势. 随着图像的增加，基于 Spark 的并行处理方法的优势也越发明显，可实现图片的快速处理.

4.2.2　*多节点下的处理效率*　本文在重大传染病专项数据库中选取 1000 张、2000 张、5000 张医学图像的情况作为数据源进行实验，通过改变集群工作节点数目，基于 Spark 的并行处理方法实验结果如图 7 所示.

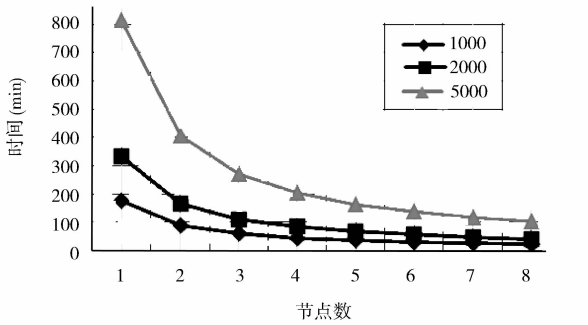

图 7　不同节点数目的处理时间

Fig. 7　Multi-node processing time

通过实验表明，在单工作节点情况下处理 5000 张图像需要 13.5 h，而在 8 个工作节点下处理同样的图片仅需要 1.7 h，大大减小了处理时间. 随着节点数的增加，图像的处理速度逐渐增加. 不同节点与不同图像数的处理时间如表 2 所示.

表 2 图像处理时间

Tab. 2 Image processing time

处理时间								
	节点数							
图像数	1	2	3	4	5	6	7	8
1000 张	176.7min	88.3min	59.0min	44.0min	35.3min	29.0min	26.0min	22.0min
2000 张	332.7min	166.3min	110.8min	83.5min	67.0min	55.8min	48.3min	41.3min
5000 张	813.3min	406.7min	271.3min	205.0min	163.5min	138.0min	117.0min	102.1min

4.2.3 *加速比评估* 在分布式系统中,加速比是衡量分布式系统性能优劣的重要指标之一,以公式 $S_p=T_1/T_p$ 表示加速比,S_p 代表加速比,T_1 代表单机下的处理时间,T_p 代表在 P 个机器的并行系统中的处理时间. 通常,当 S_p 和 P 相等时为理想加速比. 选取表 2 中 5000 张图像不同节点下的处理时间进行加速比评估,加速比如图 8 所示.

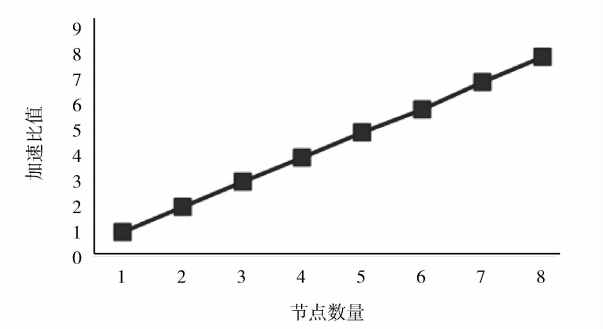

图 8 加速比

Fig. 8 Speed-up ratio

如图 8 所示,由于节点的增多,数据的传输及磁盘读写的时间也相应的增多了,但其耗时远远小于图像处理的时间,加速比仍是随着节点个数的增加呈现近似线性的增长. 这也证明了基于 Spark 的医学图像并行处理方法拥有良好的集群扩展性,可通过增加节点的方式,适应大规模的医学图像处理.

5 结 论

基于 Spark 的并行医学图像处理架构有效解决了各类医学图像处理算法对海量医学图像的快速并行处理,并可通过增加节点的方式,不断提升处理速度及处理规模. 文中提出了基于 spark 的并行图像处理架构,通过基于二进制值的分布式图像存储、转换方法,保证了图片数据可简单读入 Spark 平台,并应用传递函数的方法,避免了 MapReduce 转化,成为了一种通用的图像并行处理方法,实现各类图像处理算法的快速应用. 通过实验表明,方法可添加任意处理算法实现并行处理,且适应大规模图像的处理.

参考文献:

[1] Iwase Y, Imamura H. Medical image processing apparatus, medical image processing method, and program: US 8560341 B2[P]. USA:[s. n.][P]. 2013.

[2] Maduskar P, Hogeweg L, Philipsen R, *et al*. Automated localization of costophrenic recesses and costophrenic angle measurement on frontal chest radiographs[J]. Proc Spie Int Soc Opt Eng, 2013, 8670(4): 867038.

[3] Forum M P. MPI: A message-passing interface standard[M]. USA: University of Tennessee, 1994.

[4] White T. Hadoop: The definitive guide[M]. USA: O'Reilly Media, Inc, 2012.

[5] Zaharia M, Chowdhury M, Franklin M J, *et al*. Spark: cluster computing with working sets[J]. HotCloud, 2010, 10: 10.

[6] 张良将, 宦飞, 王杨德. Hadoop 云平台下的并行化图像处理实现[J]. 信息安全与通信保密, 2012(10): 59.

[7] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters.[J]. Commun ACM, 2008, 51(1):107.

[8] Markonis D, Schaer R, Eggel I, *et al*. Using MapReduce for large-scale medical image analysis[C]// Proceedings of 2012 IEEE Second International Conference on Healthcare Informatics, Imaging and Systems Biology. USA: IEEE, 2012.

[9] Shvachko K, Kuang H, Radia S, *et al*. The Hadoop distributed file system[C]// Proceedings of 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). USA: IEEE Computer Society, 2010.

[10] Zaharia M, Das T, Li H, *et al*. Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters[C]// Usenix Conference on Hot Topics in Cloud Ccomputing. USA: USENIX Association, 2012.

[11] Vora M N. Hadoop-HBase for large-scale data[C]// Proceedings of 2011 International Conference on Computer Science and Network Technology (ICCSNT). China: IEEE, 2011.