|  |  |
| --- | --- |
| **学校代码** | **10699** |
| **分 类 号** | **TP311** |
| **密 级** | **公开** |
| **学 号** | **2012201676** |

题目 大规模数据并行图处理算法与计算平台研究

|  |  |
| --- | --- |
| **作者** | **尤立** |

|  |  |
| --- | --- |
| **学科、专业** | **计算机软件与理论** |
| **指 导 教 师** | **陈群** |
| **申请学位日期** | **2015年2月** |

# 

西北工业大学

硕士学位论文

(学位研究生)

题目：大规模数据并行图处理算法

与计算平台研究

作 者： 尤立

学科专业：计算机软件与理论

指导教师： 陈群

二零一五年二月

Northwestern Polytechnical University

A Dissertation Submitted for the

Master Degree

Title: Research of parallel graph algorithms over big data and processing platform

Author YouLi

Specialty Computer Software and Theory

Supervisor CHENQun

February 2015

# 摘要

近年来，大数据已经成为各大公司追逐的热点领域。随着大数据时代的到来，大量、多样、变化的数据给业界带来诸多机遇和挑战。很多热门研究领域，如社交网络分析、Web文档聚类、实体识别、基因表达分析等应用中，都必须对海量数据进行分析和挖掘。这些应用中通常都包含较复杂的网络结构，实际生产过程中，人们通常使用图来抽象系统中的各个实体之间的关系结构，并通过对抽象的图挖掘从数据中提取有价值的知识。在这些应用中关于图的完全图和近似完全图挖掘已经成为不可或缺的重要组成部分。

完全图和近似完全图是经典的NP Complete问题，算法时空复杂度都很高。人们一直以来就致力于找出它们的高效解决算法。过去的研究已经提出了一些在较小数据集上表现良好的算法。但是在大数据环境下，单机算法受制于内存和磁盘的限制往往不能够满足现实生产需求，并行化是必然趋势，因此研究集群环境中大规模数据的并行图处理算法尤为重要。本文在充分了解现有的算法基础上，提出了一种基于图分割的完全图和近似完全图枚举算法（BGP，Binary Graph Partitioning），并分别在单机及并行环境中在多种实际和生成数据上，通过实验与经典算法（BK，Broc-Kron）对比，验证了算法的可行性及高效性。提出Hybrid

本文选择利用当今流行的大数据计算框架MapReduce的开源实现Hadoop作为并行算法的运行平台。针对并行环境中完全图和近似完全图计算过程中出现的负载不均问题，本文提出了一种负载均衡的算法，实验证明此负载均衡算法得到了较好的均衡性。最后，本文进一步探讨了Hadoop系统中的负载均衡问题，提出静态和动态两种系统级负载均衡方案。

通过在多种实际数据和生成数据上的实验表明，本文提出的单机算法。

**关键词：**完全图，近似完全图，并行，负载均衡，MapReduce, Hadoop

# Abstract

The era of big data has come. A large,diverse data is generated so quickly that it presents a tremendous challenge to IT industry. MapReduce model which is proposed by Google becomes the initial model for the processing of large-scale data. Apache's Hadoop is an open source implementation of MapReduce model.Because of its high reliability, high scalability, high efficiency and high fault tolerance ,Hadoop becomes a hot platform rapidly. Hadoop cluster load balance has also become a hot topic in the field of distributed computing. This paper studies methods of Hadoop cluster load balance management in data skew environment.

**Keywords:**data skew,load balance,Hadoop,multi-table join,data localization.

# 目录

[摘要](#_Toc13735)

[Abstract](#_Toc11250)

[目录](#_Toc6823)

[第一章 绪论](#_Toc27447)

[1.1 研究背景与意义](#_Toc12075)

[1.2 国内外研究现状](#_Toc24409)

[1.3 论文的内容及组织](#_Toc17380)

[1.3.1 研究内容](#_Toc15214)

[1.3.2 组织结构](#_Toc6138)

[第二章 Hadoop及负载均衡相关技术综述](#_Toc24779)

[2.1 Hadoop技术综述](#_Toc13098)

[2.1.1 分布式文件系统（HDFS）](#_Toc7429)

[2.1.2 MapReduce模型](#_Toc17984)

[2.2 负载均衡理论概述](#_Toc14618)

[2.2.1 负载均衡的基本概念](#_Toc27671)

[2.2.2负载均衡算法](#_Toc27373)

[2.3 本章总结](#_Toc14262)

[第三章 Hadoop负载均衡系统的设计和实现](#_Toc875)

[3.1 系统总体设计](#_Toc8268)

[3.2 系统详细设计之数据采样模块](#_Toc18201)

[3.2.1 采样的必要性](#_Toc8727)

[3.2.2 采样器的设计](#_Toc3037)

[3.3 系统详细设计之信息整合模块](#_Toc25467)

[3.3.1 信息分类](#_Toc20029)

[3.3.2 信息去重](#_Toc26580)

[3.3.3 信息归并](#_Toc16748)

[3.3.4 整合模块的设计](#_Toc26049)

[3.4 系统详细设计之决策模块](#_Toc14550)

[3.4.1 数据倾斜](#_Toc17672)

[3.4.2 代价模型](#_Toc32532)

[3.5 系统详细设计之消息传输模块](#_Toc10295)

[3.5.1 心跳信息的调制](#_Toc3434)

[3.5.2 加载配置文件](#_Toc18606)

[3.6 本章总结](#_Toc28729)

[第四章 Hadoop负载均衡系统优化](#_Toc24516)

[4.1 一种基于Hadoop的多表连接优化策略](#_Toc18449)

[4.1.1中间结果集的膨胀问题](#_Toc28178)

[4.1.2 优化方案的整体设计](#_Toc5190)

[4.1.3 NIHDFS索引设计](#_Toc31285)

[4.1.4 基于NIHDFS的“替换-查询”方法](#_Toc6964)

[4.1.5 “替换-查询”方法的系统实现](#_Toc27269)

[4.1.6 若干细节优化](#_Toc9621)

[4.2 一种基于Hadoop的数据本地优化策略](#_Toc31119)

[4.2.1 数据本地化的必要性](#_Toc11669)

[4.2.2 数据本地化的设计和实现](#_Toc853)

[4.3 本章总结](#_Toc26511)

[第五章 实验部分](#_Toc6563)

[5.1 实验条件及环境搭建](#_Toc32764)

[5.1.1 实验条件](#_Toc8212)

[5.1.2 搭建Hadoop集群](#_Toc29668)

[5.2实验内容以及结果分析](#_Toc12704)

[5.2.1 多表连接策略优化前后系统的性能对比.................................................](#_Toc3224)

[5.2.2 数据本地优化前后系统的性能对比........................................................](#_Toc16413)

[5.2.3 Hadoop负载均衡系统的参数调优及与原Hadoop的性能对比](#_Toc5494)

[5.3实验结论](#_Toc20004)

[5.4 本章总结](#_Toc8347)

[第六章 总结与展望](#_Toc32671)

[6.1 本文研究总结](#_Toc15239)

[6.2 课题研究展望](#_Toc10339)

[参考文献](#_Toc15812)

[致谢](#_Toc28851)

[知识产权说明书 7](#_Toc28851)1

第一章 绪论

随着移动无线带宽的快速提升，社会交际网络的快速普及，物联网相关应用的更加丰富，云计算相关技术的快速发展，更多的设备被添加到网络中，数据的产生速度飞速增长。快速化、大量化和多样化是大数据的显著特征，快速、大量、多样的数据给IT业提出了巨大挑战，对大规模数据的处理和分析也成为这个领域的研究热点。

Enomalism是开源的计算平台，该平台提供了对集群弹性的计算能力的管理能力，该系统基于Linux，提供类似于亚马逊的弹性计算能力。

1.1 研究背景与意义

大数据催生了众多的计算机技术，其中最有影响力是Google提出的MapReduce框架模型，该模型为大数据的处理提供了最初的模型。它由两部分构成：编程模型和运行时环境

任务分配策略的不合理也是造成集群负载失衡的重要原因。Hadoop中Reduce任务分配采用的是随机策略，即某一时刻，哪个任务被分配是随机的，该任务被分配到哪个节点上也是随机的。如果某一个时刻，某个重负载任务被分配到现有的重负载节点上的时候，那么该节点就成为了集群的瓶颈节点，随机策略对该节点分配的过重的负载导致这个节点要花费相当长的时间来完成任务，因此任务的分配策略也是集群负载的重要影响因素之一。

1.2 国内外研究现状

对于Hadoop的负载均衡问题，国内的众多学者对其进行了研究。集群的负载均衡对HDFS同样有着非常显著的影响，HDFS中本身自带的负载均衡策略不够灵活，常常导致HDFS的负载均衡效率非常低下，针对这个问题，文献[1]描述了一种动态的负载均衡策略,该策略

同节点也将导致不同的数据传输延迟，据此集群的最小代价节点被挑选出来，当分配一个新任务时，系统优先将该任务分配给这个节点，达到提升系统性能的目的。

1.3 论文的内容及组织

1.3.1 研究内容

(1)对MapReduce模型进行进一步研究，将理论和实验相结合，确定Hadoop存在的各种类型的负载不均衡以及产生的机理。

(2)对Hadoop环境下集群的负载均衡问题进行具体研究，分析影响负载均衡的各个因素，并建立负载均衡的代价模型。

(3)在该模型基础上，针对模型中的各种问题，设计解决方法并实现相应的算法。如负载均衡的衡量算法，任务调度算法，采样器的采样算法，消息传递算法等，并在此基础上搭建Hadoop负载均衡系统。

(4)引入数据本地化的思想，对系统进行优化，从而减少机器节点之间过多数据传输造成的网络代价。在系统处理多表连接问题时，中间结果的快速增长导致系统性能下降明显，针对这个问题，笔者设计了“替换-查询”的Hadoop多表连接策略，通过对连接表建立索引对系统进行优化，以减少中间结果爆炸对系统性能的影响。

(5)设计实验，与一些相关工作进行对比，调整系统的参数，说明各个参数的设定对算法的影响，总结了本文所设计的系统的适用场合、优势与不足及后续需要完善补充的地方。

1.3.2 组织结构

第二章，主要介绍了Hadoop系统以及负载均衡的相关技术理论。

由于本文选用的系统平台是开源的Hadoop系统，本章节分别对Hadoop的分布式文件系统（HDFS）和MapReduce编程模型进行了介绍。本章节还对集群负载均衡的基础概念，基本原理和相关算法进行了介绍。

第三章对Hadoop负载均衡系统中的数据采样，决策，消息传递以及信息整合四部分的关键技术，设计细节进行详细的描述，主要介绍设计方法，功能，以及相关的算法等。

第四章对Hadoop负载均衡系统进行优化。优化主要解决两个问题，(1)系统在处理多表连接问题时，由于中间结果急剧膨胀，导致系统I/O代价过高的问题。(2)由于数据迁移而导致的节点间大量的数据传输，致使网络代价过高，降低系统性能的问题。

第五章主要介绍了实验条件，Hadoop系统搭建，并且设计实验，验证了系Hadoop负载均衡系统的有效性。

第六章是对本文工作的总结以及未来工作的展望，说明本文方法的局限性与不足之处，今后可以改进的地方及提升的空间。

第二章 Hadoop及负载均衡相关技术综述

2.1 Hadoop技术综述

Hadoop是一个开源项目，该项目由Apache倡导并开发的。HDFS是Hadoop平台的文件系

拷贝一个，来保证正确副本的数量。

(5)低成本。

2.1.1 分布式文件系统（HDFS）

HDFS，全称Hadoop Distributed File system，是Hadoop生态圈的分布式文件系统。

**(1)HDFS的架构**

如图2-1所示，HDFS本身采用的Master/Slave架构，Master是集群的主控节点，一个集群中存在一个主控节点，主控节点又叫NameNode节点。一个HDFS集群中可以存在多个子

图2-1 HDFS架构图

**(2)HDFS相关概念**

1)数据块

数据块是HDFS最基本的单元，默认情况下，每一个数据块的大小是64M。数据块比磁盘块大得多，降

NameNode，即名字节点，该节点时HDFS的主控节点，负责对文件和数据节点的管理。

2.1.2 MapReduce模型

**(1)MapReduce架构**

同HDFS一样，Hadoop MapReduce也采用了Master/Slave（M/S）架构，具体如图2-2所示。它主

同节点的数据进行排序，归并。Reduce函数运行完毕之后，所有的结果会被存储在系统的HD

**(2)MapReduce相关概念**

1)Client

Hadoop中所

duce Task两种，均由TaskTracker 启动并完成。

**(3)MapReduce执行机制**

MapReduce由两个阶段组成：Map和Reduce。Hadoop将HDFS中数据块切分成Split之后，被

化简操作（Reduce）主要负责将隶属于一个列表的元素进行合并，它的并行度没有map那么高

图

2.2 负载均衡理论概述

2.2.1 负载均衡的基本概念

**（1）负载均衡的含义**

负

单台机器的处理时间太长，将任务切分并派发到多台机器上执行，通过任务的并行执行来降低相应时间，提升用户体验。

**（2）负载均衡分类**

1）

衡集群负载，因此这种方式非常高效，但是成本也高。

2.2.2负载均衡算法

对于负

平衡器（长期）记录各节点的请求处理数量并对其进行更新，当一个新的请求到来是，将新请求分配给历史上处理请求次数最少的节点。

2.3 Hadoop技术与负载均衡的关系

由于Hadoop的易扩

用到Hadoop集群环境中，通过对Hadoop集群中负载均衡机制的研究、设计和实现，达到平衡集群负载的目的，从而提升系统效率。

2.4 本章总结

本章主要介绍了相关的背景技术，主要有两方面的内容，一方面讲述了Hadoop的总体结构，对Hadoop中的分布式文件系统，即HDFS，以及MapReduce相关理论、架构进行了描述。另一方面介绍了负载均衡的基本概念，理论以及相关的算法。

第三章 Hadoop负载均衡系统的设计和实现

本章重点介绍Hadoop负载均衡系统的设计和实现，该系统包含了采样，信息整合，决策和消息传递描四个模块。本章从系统的总体设计和详细设计两方面入手，对系统的设计思路、实现方法以及相关算法做了详细的描述。

3.1 系统总体设计

Hadoop负载均衡系统，即Hadoop\_Load\_Balance，主要包含采样，信息整合，决策控制和消息传递

要完成的任务，并即刻调集数据开始执行。整个过程的详细流程图如图3-2所示。

3.2 系统详细设计之数据采样模块

采样模块主要是对系统的输入数据以及Map任务输出的数据进行采样。该部分在Map阶段进行。采样结果作为决策模块的输入。

具体要对那些数据进行采样，用户可以自定义，本文设计的Hadoop负载均衡系统主要是采集元组记录个数和所占用的字节数。

3.2.1 采样的必要性

造成Hadoop负载不均衡的主要原因在于数据倾斜，在用户提交任务之后，系统对任务

使的系统在分配Reduce任务之前，对于输入数据和Map Task的输出结果有个比较清晰的了解，更好的掌握数据的倾斜状况。

3.2.2 采样器的设计

**(1)采样器**

设计采样器的目标是采集数据，并利用采集的数据准确的估算出系统所处理的数据的整体情况。采

1. **定时采样算法**

启动一个定时器线程，并设定一个阈值，当定时器的值不小于该阈值的时候就将该条记录元组采集到样本中，并给定时器值为0，知道整个程序完成为止。算法伪码如下。

|  |
| --- |
| **Sampler** |
| **Input:**  *M*:Map Task’s Output  *N*:Job’s split data  *Threshold*:timer’s Threshold  **Output:**  *Scale[M/N]*:*M/N*’s scale（The number of tuples and the size of the space occupied）  Start Timer’s thread  **for** *i* from *1* to *N* **do**  Map();  **for** *j* from *1* to *M* **do**  **if** Timer >= *Threshold* **do**  *Threshold=0；*  Pick *M[j]* update to *Scale[M/N]*;  Pick *N[i]*;  **If** *Scale[M/N] don’t contain N[i]***do**  将*N[i]* update to *Scale[M/N]；*  **end if**  **end if**  **end for**  **end for**  **return** *Scale[M/N]*; |

3.3 系统详细设计之信息整合模块

信息整合模块主要负责采样信息的分类，去重和归并操作。

3.3.1 信息分类

对于样本

据用户关心的信息对数据进行分类。

3.3.2 信息去重

采样信息中常存在着重复信息，比如采集某个学生的数学成绩，采集得到的信息通常会包含很多其他的信息比如性别信息，当需要再次采集该同学的英语成绩时，那么他的性别信息就再次被采集。当用户对该学生的性别信息比较关心的时候，那么该样本中存在着多个相同的信息。

信息去重，就是删除样本中重复出现的信息，这样可以避免信息冗余，节省数据的存储资源，同时也便于对采样信息进行管理和维护。

3.3.3 信息归并

由于样本数据众多，种

息整合，样本数据将被准确分类和大幅的压缩，减少存储开销和读写磁盘的I/O代价。

3.3.4 整合模块的设计

**(1)结构设计**

整重之后的这些信息归并到一起，自此信息整合模块的任务才算是真正的完成。信息整合模块的整体框架如图3-5所示。

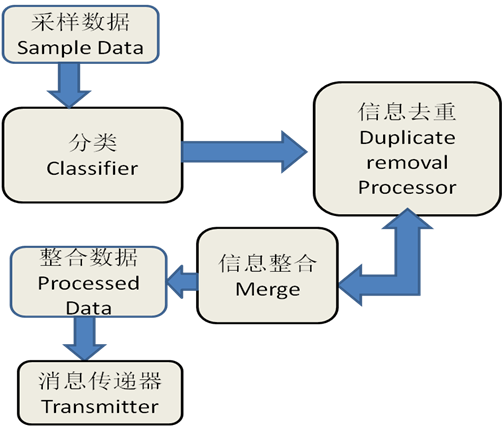


图3-5 信息这个和模块设计图

3.4 系统详细设计之决策模块

子节点的采样信息，经过信息整合模块处理之后，经由消息传递模块发送到决策模块。决策模块的主组，使得重组后的各逻辑分区的负载量比较均匀，从而达到平衡Hadoop集群负载的目的。

3.4.1 数据倾斜

**(1)瓶颈任务**

1）瓶颈任务的概念

并行计算

a. Partition分

本单位进行Reduce任务分配的。

4）任务推送器，负责将最终的任务分配方案，推送到Reduce的任务执行节点。

3.5 系统详细设计之消息传输模块

消息传输模块是联络数据采样，信息整合和决策模块的桥梁，它主要负责各个模块之间的联络和消息传递，它是原型系统不可或缺的组成部分。

该部分的是通过调制心跳信息和加载配置文件实现的。下面的小节中会对两部分做详细的介绍。

3.5.1 心跳信息的调制

3.5.2 加载配置文件

**(2)序列化和反序列化**

查找到配置文件信息以后，系统需要将该信息传回到主进程（NameNode，JobTracker，DataNode，TaskTracker）。Hadoop系统为了提升消息传递效率，用户可以选择对数据进行序列化和反序列化处理。序列化的目标是对信息进行排序和压缩，反序列化是将目标数据解析成原始数据，是序列化的逆过程。

3.6 本章总结

本章主要介绍了Hadoop负载均衡系统，即Hadoop\_Load\_Balance系统的总体设计和详细设计，对系统中的数据采样模块，信息整合模块，决策模块和消息传输模块的功能，原理，架构和涉及到的相关算法都做了详细的介绍。

1. Hadoop负载均衡系统优化

4.1 一种基于Hadoop的多表连接优化策略

Hadoop\_Load\_Balance系统在处理多表连接问题时，大规模的中间结果加剧了集群负载不均衡的状况，从而严重降低了系统效率。为解决该问题，笔者提出一种“替换-查询”方法，该方法通过对连接表建立索引，将预输出的元组集替换为索引信息输出到中间结果，以索引的形式参与多表连接，以此减少中间结果规模，；并运用缓冲池、二次排序和多线程技术对索引信息进行优化管理，加快索引的查询速度，提升系统的整体性能。

4.1.1中间结果集的膨胀问题

多表连接的性能。

4.1.2 优化方案的整体设计

输入的信息依照“替换-查询”方法进行处理。该处理包含三个部分，分别是对连接表建立NI

行查询，恢复出原始记录元组并进行解析，然后再进行Map端的连接操作。Map端和Reduce端连接操作的操作内容由具体的多表连接算法来决定。

4.1.3 NIHDFS索引设计

HDFS（Hadoop Distributed File System）是被设计成适合运行在通用硬件上的分布式文件

4.1.4 基于NIHDFS的“替换-查询”方法

“替换-查询”的多表连接处理是通过索引化，替换和查询三个模块来实现的，如图4-4所示。

**(1) 索引化**

在处理多表连接问题时，Map-Reduce任务的输入有两种类型，一种是前面某个Map-Red

连接属性信息。此时要对NIHDFS索引执行查询操作，到达获取连接属性信息的目的，系统将启动专门线程来完成该查询操作。

4.1.5 “替换-查询”方法的系统实现

查询过程如图4-4所示，TaskTracker是多表连接操作的任务执行节点，该节点中运行着多个M

4.3 本章总结

本章对Hadoop\_Load\_Balance系统做了两方面的优化。首先，针对系统处理多表连接问题时，中间结果极速膨胀导致降低系统性能下的问题，本章详细介绍了“替换-查询”的优化方法。其次，数据不在本地所引发了集群内大规模的数据迁移，大量的网络带宽被占用导致系统能明显下降，针对该问题，本章介绍了数据本地化的优化方法，并对Reduce阶段本地化的优化方法，算法，设计和实现都做了详细的介绍。

第五章 实验部分

5.1 实验条件及环境搭建

5.1.1 实验条件

软件环境： Ubuntu10.10，JDK1.6，Hadoop-0.20.205.0以及Hadoop负载均衡系统（Hadoop\_Load\_Balance）。

开发环境：Eclipse3.6，JDK1.6，Shell脚本语言，TPC-H，Hadoop集群环境。

硬件环境：5台1.6GHz的八核CPU，16G内存，150G硬盘的曙光系列服务器。

5.1.2 搭建Hadoop集群

利用试验环境中介绍的各种软件和硬件资源，搭建一个拥有五个节点的Hadoop集群环境，该集群中包含一个Master节点，四个Slave节点。Hadoop版本为Hadoop-0.20.205.0。这5个节点的IP地址是172.19.0.118~122，对应的节点的主机名是test118~122，test118作为Master节点，其他的四台机器作为Slave节点,搭建Hadoop集群环境，具体的搭建方法如下：

(1)新建用户dic并将该用户添加到sudoers组中。

1. $:su - //进入root用户
2. $:useradd -m dic //-m的意思是强制创建用户的目录
3. $:passwd dic
4. $:vi /etc/sudoers //添加“dic ALL=(ALL) ALL”到文件中。

(2)安装SSH

ssh免密码登陆原理：ssh证书登录方式:假设想要在ubuntuA访问另外一台ubuntuB, 于是我们需要在A上生成公钥密钥对. 然后把公钥拷贝到B上, 在B上把公钥加入当前用户的.ssh/authorized\_keys里。然后在A上就可以免密码登录B了。如果A需要免密码登陆C,D,E,F...等机器, 只要把在A上生成的公钥文件id\_rsa.pub拷贝到C,D,E,F...上即可。

1. $ ssh-keygen -t rsa -P '' -f ~/.ssh/id\_rsa
2. 将产生的id\_rsa.pub拷贝到其他机器上的/home/dic/.ssh目录下（注意修改文件名称，防止本地的id\_rsa.pub被覆盖,如test118.pub）
3. 在/home/dic/.ssh/创建authorized\_keys文件，然后将复制过来的test118.pub内容追加到authorized\_keys中（cat test118.pub >> authorized\_keys）。
4. 将“172.19.0.118 test118”写入到/etc/hosts中。
5. 注意务必将~/.ssh文件夹的权限设置成700，将authorized\_keys的权限设置成600，不然认证信息将无法加载。

(3)安装Java

1. JDK的版本：JDK1.6。
2. 安装路径：/home/jvm/jdk1.6.0\_26。
3. 在/etc/profile中配置JAVA\_HOME等环境变量如下所示。

|  |
| --- |
| *export JAVA\_HOME=/home/jvm/jdk1.6.0\_26*  *export JRE\_HOME=/home/jvm/jdk1.6.0\_26/jre*  *export CLASSPATH=.:$JAVA\_HOME/lib:$JRE\_HOME/lib:$CLASSPATH export PATH=$JAVA\_HOME/bin:$JRE\_HOME/bin:$PATH:$ANT\_HOME/bin* |

(4)配置文件

1. 配置core-site.xml如下所示。

|  |
| --- |
| *<?xml version="1.0"?>*  *<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>*  *<!-- Put site-specific property overrides in this file. -->*  *<configuration>*  *<property>*  *<name>hadoop.tmp.dir</name>*  *<value>/home/dic/Hadoop-0.20.205.0/hadooptmp</value>*  *<description>*  *A base for other temporary directories.</description>*  *</property>*  *<property>*  *<name>fs.default.name</name>*  *<value>hdfs://172.19.0.118:9000</value>*  *</property>*  *</configuration>* |

1. 配置hdfs-site.xml如下所示。

|  |
| --- |
| *<?xml version="1.0"?>*  *<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>*  *<!-- Put site-specific property overrides in this file. -->*  *<configuration>*  *<property>*  *<name>dfs.replication</name>*  *<value>3</value>*  *</property>*  *<property>*  *<name>dfs.http.address</name>*  *<value>172.19.0.118:50070</value>*  *</property>*  *<property>*  *<name>dfs.web.ugi</name>*  *<value>dic,supergroup</value>*  *</property>*  *</configuration>* |

1. 配置mapred-site.xml如下所示。

|  |
| --- |
| *<?xml version="1.0"?>*  *<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>*  *<!-- Put site-specific property overrides in this file. -->*  *<configuration>*  *<property>*  *<name>mapred.job.tracker</name>*  *<value>172.19.0.118:9001</value>*  *</property>*  *<property>*  *<name>mapred.tasktracker.map.tasks.maximum</name>*  *<value>4</value>*  *</property>*  *<property>*  *<name>mapred.tasktracker.reduce.tasks.maximum</name>*  *<value>2</value>*  *</property>*  *</configuration>* |

1. 配置填写master如下所示。

|  |
| --- |
| *172.19.0.118 test118* |

1. 配置填写slave如下所示。

|  |
| --- |
| *172.19.0.119 test119*  *172.19.0.120 test120*  *172.19.0.121 test121*  *172.19.0.122 test122* |

(5) 启动Hadoop

1. 格式化一个新的分布式文件系统：

$ bin/hadoop namenode -format

1. 在分配的NameNode上，运行下面的命令启动Hadoop集群：

$ bin/start-all.sh

(6)停止Hadoop

1. 关闭Hadoop系统：

$ bin/stop-all.sh

5.2实验内容以及结果分析

[5.2.1 多表连接策略优化前后系统的性能对比](#_Toc380793448)

（1）实验目的：验证Hadoop负载均衡系统“替换-查询”的多表连接策略的有效性。

（2）实验数据：TPC-H的基准数据集，选用nation, region，customer，orders四张表作为输入数据，数据规模为397.6M。

（3）实验用例：多表连接。

（4）Hadoop系统：基于Hadoop的负载均衡系统。

（5）实验内容及及结果分析。

优化前后的两套系统分别执行以上四张表的连接操作，连接的具体过程如图5-1所示。连接算法是经典的Reduce Join算法，两个系统分别执行了五遍该算法，将各个阶段的统计信息整合求取平均值，并将其填入表5-1中，由于优化后系统的最终结果是一系列NIHDFS索引信息，需要将其恢复为记录元组，该阶段消耗的时间是t4，产生的中间结果是size4，原系统在该阶段不需要做任何操作。

表5-1 实验结果汇总表一

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | t1 | t2 | t3 | t4 | 时间合计 | size1 | size2 | size3 | size4 | 空间合计 |
| 优化前 | 11s | 336s | 520s | 0s | 876s | 4K | 469.1M | 630.2M | 0M | 1099M |
| 优化后 | 12s | 237s | 271s | 244s | 762s | 265b | 35.5M | 38.5M | 630.2M | 704M |

优化后的系统产生的中间结果size1，size2，size3远比优化之前要小，连接操作所消耗的时间也比优化前的系统要小很多，空间和时间消耗对比如图5-3和图5-4所示。与优化之前相比，空间性能提升35.5%,时间性能提升12.9%，时间和空间整体对比如图5-2所示。本实验表明，“替换-查询”的表连接处理方法能够有效的减少中间结果的数据规模和增长速度，提升系统效率。

|  |  |
| --- | --- |
| 图5-1 连接过程示意图 | 图5-2 空间和时间对比图 |

|  |  |
| --- | --- |
| 图5-3 各阶段空间消耗对比图 | 图5-4 各阶段时间消耗对比图 |

[5.2.3 Hadoop负载均衡系统参数调优及与原Hadoop的性能对比](#_Toc380793450)

1. **Hadoop负载均衡系统参数调优实验**

实验目的：Hadoop负载均衡系统决策时机的参数调优，即map运行到什么进度时执行

越明显。良好的系统性能，对代价模型的有效性也进行了验证。

5.4 本章总结

本章主要介绍了实验条件，以及Hadoop集群环境的搭建步骤，并设计实验验证了Hadoop\_Load\_Balance系统的有效性。经验证，该Hadoop\_Load\_Balance能够有效的平衡Hadoop集群任务负载，系统的性能得到了明显的提升。

第六章 总结与展望

6.1 本文研究总结

Hadoop系统处理用户提交的工作时，常出现集群负载不均衡的问题，瓶颈任务大大拖长了整个任务的执行周期。针对这个问题本文提出了一套基于Hadoop的负载均衡系统，该系统通过数据采样来探知系统中的数据分布信息，样本数据经过信息分类，去重，归并之后被传输到的系统的决策模块中，该模块对其进行分析，得知系统中是否存在数据倾斜以及数据倾斜的严重程度，并据此对分区进行重构，通过降低重构后的逻辑分区负载的方差值，来达到均衡各个Reduce任务负载的目的。

针对Hadoop\_Load\_Balance系统处理连接问题时，因中间结果爆炸，导致的负载不均衡问题，通过对中间结果建立索引，用小巧的索引替代冗长的记录元组，从而大大降低了系统I/O代价。在系统上实现了Reduce端的数据本地化技术，减少了系统Shuffle阶段远程拷贝的数据量，提升了系统性能。

6.2 课题研究展望

本文提出的负载代价模型，忽略了集群中各个机器节点的异构性，也没有关注机器CPU，Memory等资源的占用率与系统性能的变化关系。在以后的研究中可以将异构性考虑在内，在为系统分配任务之前，根据机器节点的状态确定是否为其分配新任务，从而对集群负载做到更加精细的控制。另外，本文中只考虑了Map和Reduce任务一轮就执行完毕的情况，针对Map和Reduce任务需要多轮才能执行完毕的情况，将作为下一步要研究的点。

# 参考文献

[1]林伟伟 , 刘波 . 基于动态带宽分配的 Hadoop 数据负载均衡方法[J] . 华南理工大学学报: 自然科学版, 2012, 40(9): 42-47.

[2]吕良干. 云计算环境下资源负载均衡调度算法研究 [D][D]. 鸟鲁木齐: 新疆大学, 硕士学位论文,2010.

[3]程斌. 负载均衡调度系统的设计与实现[D]. 华中科技大学,硕士学位论文, 2011.

[4]Liu Y, Li M, Alham N K, et al. Load balancing in MapReduce environments for data intensive applications[C]//Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on. IEEE, 2011, 4: 2675-2678.

# 在校期间发表论文

论文题目：《一种基于Hadoop的多表连接策略》

作者：徐剑，陈群，王卓，李战怀，潘巍

发表时间：2014.1.15

发表状态：已录用(2014年8期)

期刊名：现代电子技术

ISSN：1004-373X

CN：61-1224/TN

# 致谢

在本文完成后，我即将结束为期三年的研究生生涯，我衷心地感谢在我硕士期间，所有对我的学习、工作及生活给予帮助的人。

首先，感谢我的硕士生导师陈群教授，感谢他在我研究生期间对我的关心和帮助。生活中，陈老师像家长一样，常常与我进行沟通与交流，并给予我鼓励与安慰，使我倍感温暖；工作中，陈老师一丝不苟，从严治学，他严谨的工作风格给我留下了深刻的印象，是我以后科研及工作的榜样。我毕业论文的顺利完成，与陈老师的悉心指导是分不开的。再一次向陈老师致以真诚的敬意。

我还要感谢我的另一位导师，潘巍老师。在进行项目研究的过程中，潘老师丰富的经验以及扎实的理论功底，使我受益匪浅，收获颇丰。无论是研究点的设立、算法的设计，还是科研文章的撰写、做科研的态度，潘老师都会一一耐心地教导我，使我逐步走进科研的殿堂，同时，潘老师活跃的思想，谦逊的性格以及追求真理的态度是我以后科研道路上的榜样。

同时，我要感谢教研室的博士师兄王卓、索勃、樊峰峰、硕士师弟陈肇强、尤立、白松、师妹勾志营、以及与我共同科研、共同成长的硕士同学徐东海、杜晶、胡嘉琪、徐曜和王浩，是他们在我遇到技术难关时，及时给予帮助，在我困惑于一个科研问题时，牺牲时间与我讨论，再一次感谢他们。

最后，我要特别感谢我的父母，我的家人，他们是我学习、科研的坚强后盾，一直在为我默默地付出，无私地奉献。无论我遇到什么困难，什么挫折，他们总会站在我身边，给我最强有力的支持，他们的支持是我顺利完成学业的保证。谨以此文献给他们。

# 知识产权说明书