|  |  |
| --- | --- |
| **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，Bron-Kerbosch）对比，验证了算法的可行性及高效性。在此基础之上，根据生物基因数据的特征提出了一种BK辅助BGP的Hybrid算法，在各种数据上都能够有更优的表现。

在并行环境中，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。

**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 I](#_Toc405367898)

[Abstract II](#_Toc405367899)

[目录 1](#_Toc405367900)

[第一章 绪论 1](#_Toc405367901)

[1.1 研究背景 1](#_Toc405367902)

[1.2 选题意义 1](#_Toc405367903)

[1.3 研究现状 1](#_Toc405367904)

[1.3.1 完全图单机算法 2](#_Toc405367905)

[1.3.2 完全图并行算法 2](#_Toc405367906)

[1.3.3 近似完全图单机算法 2](#_Toc405367907)

[1.3.4 近似完全图并行算法 2](#_Toc405367908)

[1.3.5 Hadoop系统负载均衡 2](#_Toc405367909)

[1.4 论文的内容及组织 2](#_Toc405367910)

[1.4.1 主要研究内容及成果 2](#_Toc405367911)

[1.4.2 论文组织结构 2](#_Toc405367912)

[第二章 相关理论及技术 5](#_Toc405367913)

[2.1 完全图和近似完全图 5](#_Toc405367914)

[2.1.1 图论相关概念 5](#_Toc405367915)

[2.1.2 完全图 5](#_Toc405367916)

[2.1.3 近似完全图 6](#_Toc405367917)

[2.1.4 负载均衡的基本概念 6](#_Toc405367918)

[2.3 并行计算模型MapReduce 6](#_Toc405367919)

[2.4 开源分布式计算平台Hadoop 6](#_Toc405367920)

[2.5 图数据表示 6](#_Toc405367921)

[2.6 本章总结 7](#_Toc405367922)

[第三章 完全图和近似完全图检测算法 8](#_Toc405367923)

[3.1 完全图检测算法 8](#_Toc405367924)

[3.2 近似完全图检测算法 8](#_Toc405367925)

[3.3 Hybrid算法 9](#_Toc405367926)

[3.6 本章总结 9](#_Toc405367927)

[第四章 并行环境下的负载均衡 10](#_Toc405367928)

[4.1 完全图和近似完全图检测的负载均衡 10](#_Toc405367929)

[4.1.1中间结果集的膨胀问题 10](#_Toc405367930)

[4.2 并行平台Hadoop中通用的负载均衡 10](#_Toc405367931)

[4.3 本章总结 10](#_Toc405367932)

[第五章 实验部分 11](#_Toc405367933)

[5.1 实验条件及环境搭建 11](#_Toc405367934)

[5.1.1 实验条件 11](#_Toc405367935)

[5.1.2 搭建Hadoop集群 11](#_Toc405367936)

[5.2实验内容以及结果分析 14](#_Toc405367937)

[5.2.1 多表连接策略优化前后系统的性能对比 14](#_Toc405367938)

[5.2.3 Hadoop负载均衡系统参数调优及与原Hadoop的性能对比 16](#_Toc405367939)

[5.4 本章总结 16](#_Toc405367940)

[第六章 总结与展望 17](#_Toc405367941)

[6.1 本文研究总结 17](#_Toc405367942)

[6.2 课题研究展望 17](#_Toc405367943)

[参考文献 19](#_Toc405367944)

[在校期间发表论文 20](#_Toc405367945)

[致谢 22](#_Toc405367946)

[知识产权说明书 24](#_Toc405367947)

第一章 绪论

1.1 研究背景

随着互联网尤其是移动互联网的快速发展，社会化交际网络的快速普及，物联网、商业销售、Web语义分析、生物网络信息等相关应用的丰富，云计算相关技术的快速发展，更多的设备被添加到网络中，数据资源呈指数增长。正如人们所说：大数据时代到来了。在大数据时代，对于企业来讲最重要的是获得和使用数据，能否从现有的海量数据中快速、高效、准确地分析提取有用信息已经成为企业能否在新环境下取得成功的关键因素。早在2010CNNIC统计时，仅中国的网页规模就已经达到600亿，且还在以每年78.6%的比率增长；Web搜索领域Google索引了全球超过500亿的网页，每天有20PB数据要处理；同时社交网络的后来居上，Facebook已经超过7亿用户[39]；在环境和气象领域，需要从数以万计的传感器中接受大量数据以进行监控和预测。正是有如此庞大和迫切的需求，快速高效的海量数据处理日益受到学术界和科技界的关注。

在各种各样的海量数据应用中普遍存在网络连接结构，如Web中网页的链接关系，社交网络中人与人之间的好友关系，基因表达数据中各个基因之间的协同表达关系等。而图正是数学上描述各个实体之间的网络关系的一种经典结构，如此总总应的用都可以抽象成关于图的计算。

图论作为数学的一个重要分支，以图为研究对象，研究顶点和边组成的图形的数学理论和方法。图论中所研究的图是对现实网络的抽象，由顶点和边构成，图中的顶点代表实现事物，用连接两点的边来表示顶点所代表的事物之间具有某种关系。早在1736年数学家欧拉关于柯尼斯堡七桥问题的论著中就有关于图论的研究记录，这个问题后来被推广为著名的欧拉回路问题。经过二百多年的发展，图论已经成为一个独立的数学研究分支。图论中的问题主要可以概括为子图问题、染色问题、路径问题、网络流与匹配问题以及覆盖问题。这些问题里各自出现了许多经典的代表，如哈密顿回路、最大团、四色问题、斯坦纳树、最短路径、中国邮递员问题、最小覆盖集等，也出现了许多如戴克斯特拉算法、克鲁斯卡尔算法、拓扑排序等等经典算法。然而经过研究证明，很多图相关的问题都是NP或者NP Complete问题，算法有很高的时间和空间复杂度。在过去的现实需求中数据量较小，传统算法可以具有较好的实用性。但在大数据时代，数据的采集、获取变得非常方便，数据中所蕴含的科学与商业价值推动着对于海量数据的图处理需求迅速提高。传统的单机算法受制于内存和磁盘的限制，尽管不断优化，还是不能够在有意义的时间内完成处理。同时现有的多机并行算法通常利用冗余来保证计算的完整和正确，很难达到较高的并行度，或者在计算过程中带来严重的数据膨胀问题。

1.2 选题意义

随着图在社交网络、生物信息、实体识别等领域广泛应用，业界对高效的图处理算法有着迫切的现实需求。完全图和近似完全图常常用于进行分类和聚类分析，例如在社交网络中，通过检测完全图和近似完全图，从而找到关系十分紧密的群体，群体之间往往具有相同的兴趣爱好或者消费习惯，因此在个性化服务和投放广告方面能做的更加地准确有效，从而增加用户的粘性，提升网站的流量和商业价值都；在生物信息领域，不同基因之间的协同表达是一个重要的研究问题，常常作为新型药物研制的突破口，是生物工程中常用的技术，完全图和近似完全图的检测则是分析基因协同表达中的重要技术，这些技术的应用能够有效缩短新药的研制时间和成本。

在大规模数据的背景下，TB级甚至PB级的大规模数据是比较常见的情况，这些问题难以在单机上进行计算，并行算法成为必然选择。并行算法的过程中经常会出现数据不均衡或者计算不均衡的情况，严重制约平台的吞吐量和计算效率，负载均衡已经成为并行和分布式计算平台研究中最重要的方面之一。

尽管对于完全图与近似完全图的枚举、并行环境的负载均衡等问题现在已经有许多优秀的算法和方案[10][12][14][16][18][20][33]，然而都存在性能瓶颈和难以适用情况，又或者是针对某些具有特定特征的数据定制，没有较好的通用性。文献[1]是Bron和Kerbosch完全图枚举的经典算法（BK），在BK的基础上发展、衍生了一系列算法，文献[3][5]通过实际应用显示BK算法的效率要普遍好于现有的其他算法。然而在本文的研究过程中发现BK算法在许多数据上依然存在搜索空间大，速度慢等等问题。文献[2],[4]等并行完全图算法虽然一定程度上缓解了单机计算能力的不足，但存在中间数据过度膨胀，冗余状态量大，需要通过大量的机器数目来存储和计算，资源利用效率低。另外现有的分布算法并行程度低，且当出现负载不均衡的情况，如社交网络数据中少数名人的好友关系非常大，而大部分普通用户关系简单，这种差异会导致系统少数机器有性能瓶颈从而影响整体执行效率，降低系统吞吐量。现有的近似完全图算法一般是借鉴完全图的解决方案稍作调整，但是目前还没有很好的并行算法。已有的近似完全图并行算法[6][7]是在特定的限制环境下的特殊解决方案。同时在近似完全图中同样也存在和完全图枚举过程中的负载不均衡问题。并行完全图和近似完全图枚举问题中所共有的负载不均衡的问题通常有两种解决思路：根据特定问题（本文中即完全图和近似完全图枚举）设计方案，在各个机器节点间互相分担高负载节点的任务量；对计算平台做相应修改，使计算平台本身可以达到自动均衡负载的功能。第一种方案具有简单、易实现的优势，但是需要设计者对输入数据特征和特定算法有较好的了解，而且这样的算法并没有通用性，需要为每个算法重新设计均衡策略；第二种方案，从平台上解决了负载均衡的问题，可以有较好的通用性，只要任务是可以切分的，系统可以自动去均衡各个机器节点的负载，而无需用户介入，缺点是实现比较复杂。

完全图枚举和近似完全图枚举都是经典的NP Complete问题一个问题的有效解决可以对其他NP Complete问题有借鉴意义，同时本文所使用的并行环境下的负载均衡方案既有针对完全图和近似完全图的特定应用方案，也有通用的Hadoop系统中负载均衡问题解决方案，可以用来解决分布式计算平台Hadoop中大部分负载不均衡问题。综上所述，本文的选题具有较大的研究意义。

1.3 研究现状

本文主要涉及图论中的两个典型问题极大完全图枚举（Maximal Clique Enumeration，MCE）和最大近似完全图枚举（Maximal Qusi-Clique Enumeration）。这两个问题自提出以来就已被广泛研究，各有单机和并行算法两类算法。分布式计算平台Hadoop上的负载均衡问题近几年也有不少研究成果，下面将从这五方面介绍现有的研究现状做介绍。

1.3.1 完全图单机算法研究现状

完全图中的所有节点都与除自身外的其他节点相邻。在实际应用过程中尤其是计算科学相关领域中最有价值的是从杂乱无章的大量数据图中找出所有的极大完全图。这个问题也因Luce和Perry在社交网络分析中使用Clique表示团体中每一个人都认识其他所有人也成为极大完全图枚举（MCE）。

抽象的图论算法中有多种极大完全图枚举算法，以及一些与之相关的数学性质。这些算法主要分为三类：

第一类是以Bron-Kerbosch为典型代表的基于回溯搜索并应用剪枝算法降低搜索空间。这类算法[1][15]利用完全图必然是邻接点子集的特征，算法每次从候选点集中选取一个节点与现有的节点构成完全图，直到候选节点中不再有点可以与现有结果组成更大的完全图，最后在递归回溯搜索。

第二类类算法[9][11][13]使用反向搜索策略，这类算法的主要特征是可以较容易地得到算法关于结果集中完全图个数的复杂度上界。

第三类算法是基于“组装”的算法[21][22]，将简单的子图自动组装拼接成更大的单元，最终形成极大完全图。该算法首先将图分解，分解过程中如若产生完全图则直接输出，其他的子图按照特定的方式组装若能组装成完全图则输出。

另外还有如Thang Nguyen Bui等提出的基于遗传编程算法的完全图检测[17]。算法将网络连接关系图用树的形式表示，完全图即使从根到叶子节点的路径上所涉及到的点集，找出一个点的邻接节点非常方便。根据完全图的性质结合遗传编程的特点可以较快的剪去许多不符合条件的节点，减少候选节点数量提高搜索速度。同时由于图以树的形式表示可以借鉴许多在树上研究得较为成熟的技术。James Cheng等根据大规模图数据中完全图检测的前提下提出一种优化的存储结构H\*--graph[19]，该结构定义图的核心部分及核心部分所邻接的节点。将其他节点数据存在外存磁盘中，可以有效的限制大规模数据的情况下单机完全图检测过程中对内存的消耗。同时H\*--graph只保存了数据图中的核心部分，数据的更新代价大幅降低，避免了传统算法中数据更新时需要重新计算的缺点。

在图论领域有不少关于完全图的性质可以应用到实际的完全图检测算法中。Turon指出，如果一个图有足够多的边，那么它一定包含一个最大的完全图；若一个图中包含条边，则必然好汉一个三个节点的完全图。Moon和Moser指出，在一个包含3n个节点的图中，至多包含个最大完全图。拉姆齐理论指出每个图或者其补图包含一个至少有对数个节点的完全图。

1.3.2 完全图并行算法研究现状

随着近十几年互联网、生物信息学、社交网络、移动互联网等众多新兴技术的迅速发展，数据的规模变得越来越大。为了适应变化的需求，原本在单机环境中的完全图检测算法也在近些年被人们扩展到并行环境中。这些并行完全图检测算法[23][24][25][26][27]的基本思想都是将图或者搜索空间切分为独立的、较小的子图或子空间，然后对于这些切分出来的子图或子空间将它们分散到多个机器节点上各自进行计算，以提高整体的计算效率。

S Szabo提出一种基于分解邻接矩阵的并行完全图检测算法[29]，该算法充分利用完全图的性质将邻接矩阵不断切分的数据单元，并将其分散到不同的节点上计算。

Matthew C.Schmidt等人提出一种基于多核并行的完全图检测算法[28]。该算法使用共享内存来存储全局信息，每个核负责处理部分图数据，各个核将处理出来的结果汇总合并成为最终结果。此算法可以处理较大数据量的图且可以通过核数量的扩展有接近线性的并行性，但是对系统的硬件性能有很高的要求。

之前的算法大多运行在多核机器或者MPI平台上，自从MapReduce计算模型的开源平台Hadoop兴起后，其良好的扩展性、稳定性，适应于异构集群、普通计算节点的特性带来了一波新的浪潮。Shengqi Yang等首先提出了使用MapReduce模型在Hadoop平台上进行完全图检测的并行算法[4]。该算法首先提取数据图中每个节点的两跳邻接点（相邻节点以及相邻节点的相邻节点），并将其发往不同的计算节点进行计算，各个计算节点上使用传统单机算法进行检测。

1.3.3 近似完全图单机算法研究现状

近似完全图相对完全图而言其对图的特征要求相对较低，因此在现实生产环境中有着更加广泛的应用。对于近似完全图的研究一般是完全图的一种变形，一遍的完全图算法都可以稍加修改，减弱限制条件使之成为近似完全图检测的算法[34]。近年来，由于社交网络如Facebook、Twitter等推动，近似完全图在这种场景下对于公司的广告投放、社区发现有着重要的指导依据，文献[30][31]对此有详细阐述。近似完全图在实际应用中一般都有个条件限制（结果集大小超过一定限值或者最多与k个顶点不相邻即K-plex[34]），由于这些条件有确确实实的意义，如低于4个人以下的近似完全图基本没有实用价值，因此本文也采用这样的限制条件。现在已经有不少研究关注近似完全图的挖掘算法，除去对完全图的BK算法做相应改进的方法外，有下面这些典型的算法和优化方案：

Guimei Liu等提出了使用度数过滤数据图中不可能满足近似完全图条件的节点，同时提出了近似完全图挖掘中的多种优化方案，包括关于图直径、设定最小节点数目阀值、子图中近似完全图能够添加的节点上下界的优化等等[32]。

James Abello等提出了一种先期在外存中使用广度优先搜索，限制输入子图的大小，使得原本由于数据量巨大而不能完全放到内存中图数据可以在有限的内存条件下完成近似完全图检测的工作。算法在对内存中的节点的边进行搜索，同时应用一些剪枝条件和优化策略，使得处理大规模数据成为可能。算法还利用随机适应性贪心搜索的方法通过抽样每次选取可以放入内存的最小子图进行计算[33]。

与完全图检测相比而言，近似完全图由于其限制条件较低，其搜索复杂度、搜索空间以及结果数量都较完全图更大。在大数据环境下，单机算法面临着更加严重的性能问题。

1.3.4 近似完全图并行算法研究现状

近似完全图的并行算法目前还不是很多，主要有两种。

第一种是Bin Wu等在多核并行环境下并行算法优化。整体上该算法同时使用经典算法的主体结构，每个搜索树节点包含三个集合：Res，Cand, Not，分别表示已经形成K-plex的节点集，可扩展的候选节点集，之前已经扩展过不再需要扩展的节点集。文章中提出两种优化方案：若是一个k-plex，如果中节点数目大于()，则图的直径小于等于；若极大k-plex节点数目大于，如果包含节点，则必然在()跳之内[35]。

第二种是Arash Khosraviani等提出的在MapReduce并行框架中做近似完全图检测，这也是首次提出在MapReduce平台中进行相关工作。算法将数据图分解为各自独立的子图，使用MapReduce框架将数据发散到各个机器节点上各自做检测，这样可使得算法可以获得较好的扩展性。算法在分解数据图的过程中使用了数学特性限定的，这样对于所有的搜索工作只需要在两跳数据集里面进行搜索即可。这样的限制使得算法不能够有效检测到较小或者节点数目较少的近似完全图，但是由于这些不能检测到的图基本没有使用价值，因此该优化假设是可以利用的[36]。

现有的并行近似完全图枚举算法首先没有从单机算法上将计算时间减下来，同时还没有能够较全面的考虑并行环境中的负载不均问题。本文从这两方面都做了优化工作。

1.3.5 分布式计算平台Hadoop系统负载均衡研究现状

本文中所涉及到的并行图算法处理工作都是在MapRedue模型的Hadoop平台上完成的，因此本文探讨的平台负载均衡也是针对Hadoop系统而言。

对于在分布式计算平台Hadoop上进行完全图和近似完全图枚举的负载不均衡问题本文首先采用了关于这两种算法特定场景下设计的均衡策略。同时注意到这种方案并不能够通用地解决在Hadoop平台上使用MapReduce模型进行大规模数据条件下图数据并行处理问题中的负载不均问题。因此本文进一步探讨如何在Hadoop平台上实现一种通用的负载均衡机制。

MapReduce模型将计算分为Map和Reduce两个阶段，一般情况下Map主要对数据进行分区操作，Reduce完成对分区的计算。Hadoop系统默认采用的是Hash分区法，同时支持Range和用户自定义分区法，但采用的都是一次分区机制，即对元组仅进行一次划分，并且采用分区与Reduce一一对应的随机指派策略。对于均匀分布的数据集，该分区机制能很好地实现各Reduce接收数据的均衡性，但对于个别值密集的倾斜数据，默认分区方法很难实现对数据的一次性均匀划分。一旦发生数据的划分倾斜，势必会造成Reduce运行的不均衡，从而影响整个作业的运行时间。因此如何保证分区后数据的均衡性，成为近年来研究的热点。

为解决原MapReduce自身一次分区生成机制带来的弊端，研究者们提出了两阶段分区机制，即首先按照原机制生成数据分区，然后对数据量发生倾斜的分区进行一次调整。分区调整策略如文献[37]中的方法：在Map运行到一定时机时，根据采样所得到的分布信息对发生数据量倾斜的分区进行一次拆分，然后在保证数据一致性的基础上将拆分后的分区与较少数据量的分区进行合并，从而产生均衡分区。该方法的关键是何时对分区进行拆分，较早对分区拆分势必会增大采样误差，而较晚调整又会延迟数据从Map端到Reduce端的传输时机，对于不同的数据分布，该方法很难给出一个最优的调整时机。文献[20]提出了一种预处理分区调整策略，即在运行用户的作业前，启动一个采样任务负责调整默认分区策略，以此保证使用均衡的划分策略来运行用户的作业。由于一个作业的启动和停止需要较大的开销，对短作业和数据集经常发生变化的作业，该方法往往不能取得最优的运行效率另外，这种方法需要多一次对于输入数据集的访问，这也就增大了系统的文件访问和数据传输开销，在繁忙的集群中会带来集群资源的浪费。由此可见，一次分区生成和一次分区调整机制，在处理分布类型复杂的各类数据集时往往存在一定的局限性。

Hadoop上的负载均衡工作还有一类是基于动态调整策略的。华盛顿大学YongChul Kwon等人提出了一种基于负载切分转移的动态均衡方案[38]--SkewTune。任务的划分仍按照Hadoop默认的方式，负载均衡工作时机发生在当检测到系统中有Slot任务完成时。系统使用代价估计模型首先从本地现有的任务中取出一个最繁重的任务，当本地没有需切分的任务时会通过Master节点分析从其他节点选取最繁重的任务，这个选择出来的任务称为Straggler。系统之后通过对Straggler的任务划分，将其中一部分任务转移到出现空闲的Slot中运行，以此达到负载均衡的目的。

1.4 论文的内容及组织

1.4.1 主要研究内容及成果

本文主要关注大规模数据环境下并行图处理算法的研究以及并行计算平台中的负载均衡问题。主要包括完全图枚举、近似完全图枚举和Hadoop系统的负载均衡问题，期望能够为并行环境中高效解决图处理算法和良好的系统均衡提供思路和经验。

本文分析了现有的经典完全图和近似完全图枚举算法的优劣，提出了一种新的完全图枚举和近似完全图枚举切分算法（BGP，Binary Graph Partitioning）。通过实验证明新算法在大多数据情况下其搜索空间以及搜索效率都优于现有算法。

在分析BGP和经典算法BK各自优势的基础上结合两者的特征将两种算法结合，提出一种Hybrid算法，该算法兼具BGP和BK的优势。经验证Hybrid算法在所有实验数据集上都有最优的表现。

契合当前大规模数据的处理需求，结合之前提出的单机算法，本文提出了基于并行计算平台Hadoop的BGP的并行算法。通过实验证明了并行BGP算法的并行性、高效性。

最后由于在并行计算平台Hadoop上实现BK和BGP的并行图算法都会涉及到负载不均衡问题，本文从图算法本身和计算平台两个方面提出了负载均衡方案。

1.4.2 论文组织结构

本文的组织结构如下：

第一章是绪论部分，从整体上介绍本文的研究背景，主要介绍了完全图枚举、近似完全图枚举和负载均衡三个问题的研究现状，最后简要介绍了本文的研究内容和研究成果。

第二章是相关理论及技术介绍，包含完全图、近似完全图、负载均衡相关概念，MapReduce并行计算框架和其开源实现Hadoop。

第三章对完全图和近似完全图枚举算法进行了深入分析，选取最具代表性的经典算法做为对比实验。分析了现有算法的优缺点。提出了一种新的完全图和近似完全图算法。并结合两种算法各自的优劣提出一种可以充分发挥两者优势的Hybrid算法。同时论述了这几种算法的并行实现。

第四章在第三章的基础之上分析了算法并行化实现过程中发现的负载均衡问题，提出了针对完全图和近似完全图算法的负载均衡方案。最后进一步深入探讨了并行平台中的负载均衡问题，提出静态和动态两种通用的负载均衡方案。

第五章是论文的实验部分。本章选取了一系列真实数据和生成数据通过多种维度的实验验证了本文提出的算法的高效性。同时通过实验验证了负载均衡方案的有效性。

第六章是对本文工作的总结以及未来工作的展望，总结全文的主要内容以及今后可以改进的地方及提升的空间。

第二章 相关理论及技术

本文选取的研究对象时图论中经典的两个问题—完全图和近似完全图。在实际研究中，研究者们主要关注的方面是如何从原始数据图中找出所有的且最大的完全图或近似完全图，也就是通常所说的极大完全图枚举（Maximal Clique Enumeration， MCE）和最大近似完全图枚举（Maximal Qusi-Clique Enumeration，也成为K-plex）。本章将本文涉及到的如图论相关概念、完全图、近似完全图、并行计算模型MapReduce、开源平台Hadoop以及负载均衡等相关技术作简要介绍。

2.1 图论相关概念

作为数学的一个重要研究领域，图论自1736年欧拉在著作中首次研究柯尼斯堡七桥问题开始经过上百年发展现在已经形成了超图理论、拟阵理论、拓扑图论和代数图论等分支，并广泛应用于计算机科学、心理学、商业挖掘、运筹学等等方面。

图论的主要研究对象时图。图是由顶点的有穷非空集合和顶点之间的边的集合组成，通常表示这样一个二元组：，其中，G表示一个图，V是图G中顶点的非空有限集合，E是图G中边的可空有限集合。图中的边是关于节点的一个二元组()，边的两个元素a和b之间可以是有序的也可以是无序的。当边的元组之间无序时将边称之为无向边，相似地边的元组间有序时称边为有向边。由有无向边组成的图为无向图，由有向边组成的图称之为有向图。本文根据实际的应用需求以及普适实用的研究意义将研究对象限定为无向图，如无特别说明在正文中所提到的相关图都是无向图。下图2-1中(a)表示一个有向图，(6)表示一个无向图。

图论中两点之间距离并不是指从一个点到达另一个点的路径中边的权值和最小值，而是指的图的直径是指图中任意两个顶点之间可以到达的路径中所经过的最少节点个数。图的直径值图中任意两点之间的距离的最大值，如图2-1 (b)中图的直径是1。

在有向图中，以节点为始点的边个数称为的出度，以节点为终点的边个数称为的入度，他们的和称为的度。无向图中，与节点相邻的边个数称为的度，记为。给定两个图，，若且，则称图是图的子图。

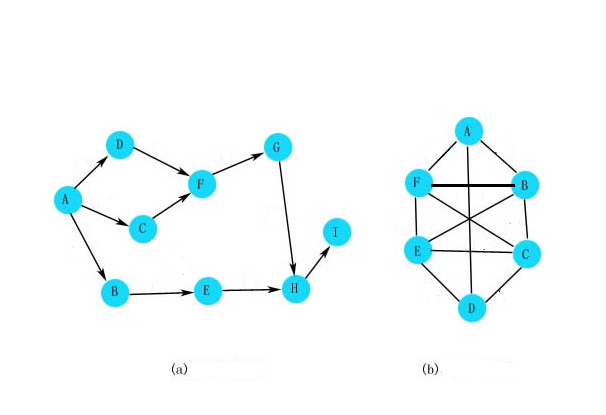


图2-1 有向图和无向图

2.2 完全图和近似完全图

本文的主要研究对象完全图和近似完全图都是图的一种稠密子图，下面将介绍完全图和近似完全图的相关定义和性质。

2.2.2 完全图相关概念

完全图指图中任意两个节点之间都有边相连，在有个顶点的完全图中应有条边，也称这个图是完全的。完全图是它本身的团（Clique）。在上图2-1中B、C、E、F四个点组成的子图就是一个完全图。图2-2直观上展示了-的完全图。

关于完全图有两个相似的容易混淆的概念，极大完全图（Maximal Clique）和最大完全图（Maximum Clique）。一个完全图称为极大完全图指不能够通过从这个图的邻接点中扩展出一个新的更大的完全图，也就是说这个完全图不是其他任意一个完全图的子图。最大完全图指一个图中大小最大的一个完全图，也就是说完全图的顶点数目大小在这个图的所有完全图中是最大的。指定一个顶点，包含顶点的极大完全图可以有很多个，但是包含顶点的最大完全图只要找到最大一个即可。在图2-1中对于点C来说，C、D、E是三个点的一个极大完全图，但是最大完全图是C、B、E、F。

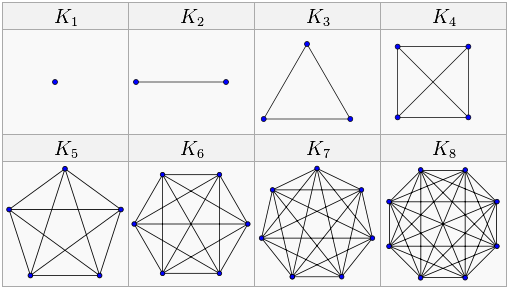


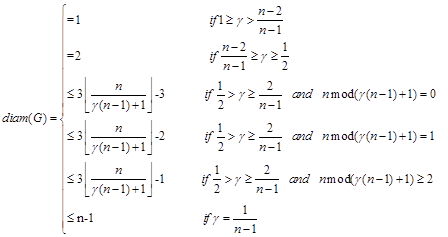
图2-2 完全图

2.2.3 近似完全图相关概念

近似完全图也是一种稠密图，但是可以容许图中可以有部分点之间不相邻。由于完全图对图的连接结构要求非常严格，现实场景中数据很难达到完全图的标准。近似完全图由于具有相似的特性但是要求不那么严格，在现实环境中的应用会更加普遍。一个近似完全图是图的一个子集，包含至少条边，其中。我们称quasi-clique为，其中参数用于表示近似完全图的疏密程度，越小，近似完全图中平均每个节点的邻接点个数越少，也就是说，近似完全图更加稀疏；反之，平均每个节点邻接的点越多，近似完全图更加稠密。特别地，当时，即上一节中提到的完全图。

由于近似完全图要求较松，粗糙的定义给实际挖掘工作带来了较大的麻烦。在实际生产应用中我们通常对和的值进行限制。若节点数目过少，挖掘出来的近似完全图应用价值不高，通常我们会限制。鉴于多数现实应用意义本文中的算法同样使用了这个限制条件。

图论中还有另外一个度量图的密集程度的方式，即图的直径。文献[40]中指出与之间的关系，具体如下面公式2-1所示，其中n为图中节点的个数。由公式2-1可知，为了满足的条件，只需保证即可。因此，在近似完全图的搜索过程中，可以只使用两跳数据集，而无需全部图数据，这从一定程度上减少了搜索过程的时空代价。



公式 2-1

通常对于近似完全图，还有一种较为简洁的表示方法，K-Plex。一个近似完全图K-Plex定义为包含N个节点的图中，每个节点最多与（N-K）个节点不相邻（节点认为与自身不相邻）。当K=1时，所有节点只与自身不相邻，因此就是完全图。实际上，完全图是近似完全图的一种特殊情况。

2.3 并行计算模型MapReduce

随着计算规模的不断扩大涌现了许多优秀的并行计算模型，MapReduce、BSP(Bulk Synchronous Parallel Computing)、LogP等。其中MapReduce模型以其简洁性和通用性获得了长足的发展。MapReduce[41]是Google的J Dean等提出的一种函数式编程模型，可以使得程序员方便地使用简单的函数实现完成复杂的分布式并行计算。其设计主要包括两个模块，分布式存储\*DFS和MapReduce计算框架。

MapReduce框架将计算过程分为Map和Reduce两部分。Map过程称为映射，映射过程就是对逻辑上一致的列表中每个独立元素进行指定的操作，每个元素的操作是独立的，输出使用新的列表保存操作结果。由于操作并没有改变输入列表，Map是可以高度并行的，这对并行计算领域中高性能和高并行度的需求非常有用。Reduce过程称为化简，化简过程是对输入列表中的元素进行适当地合并，归约出最终结果[42]。

在MapReduce模型中输入文件一般是存储在分布式文件系统中的，文件在分布式文件系统中被切分为同样大小的一块块的分片Split，每个Split都有多个备份。

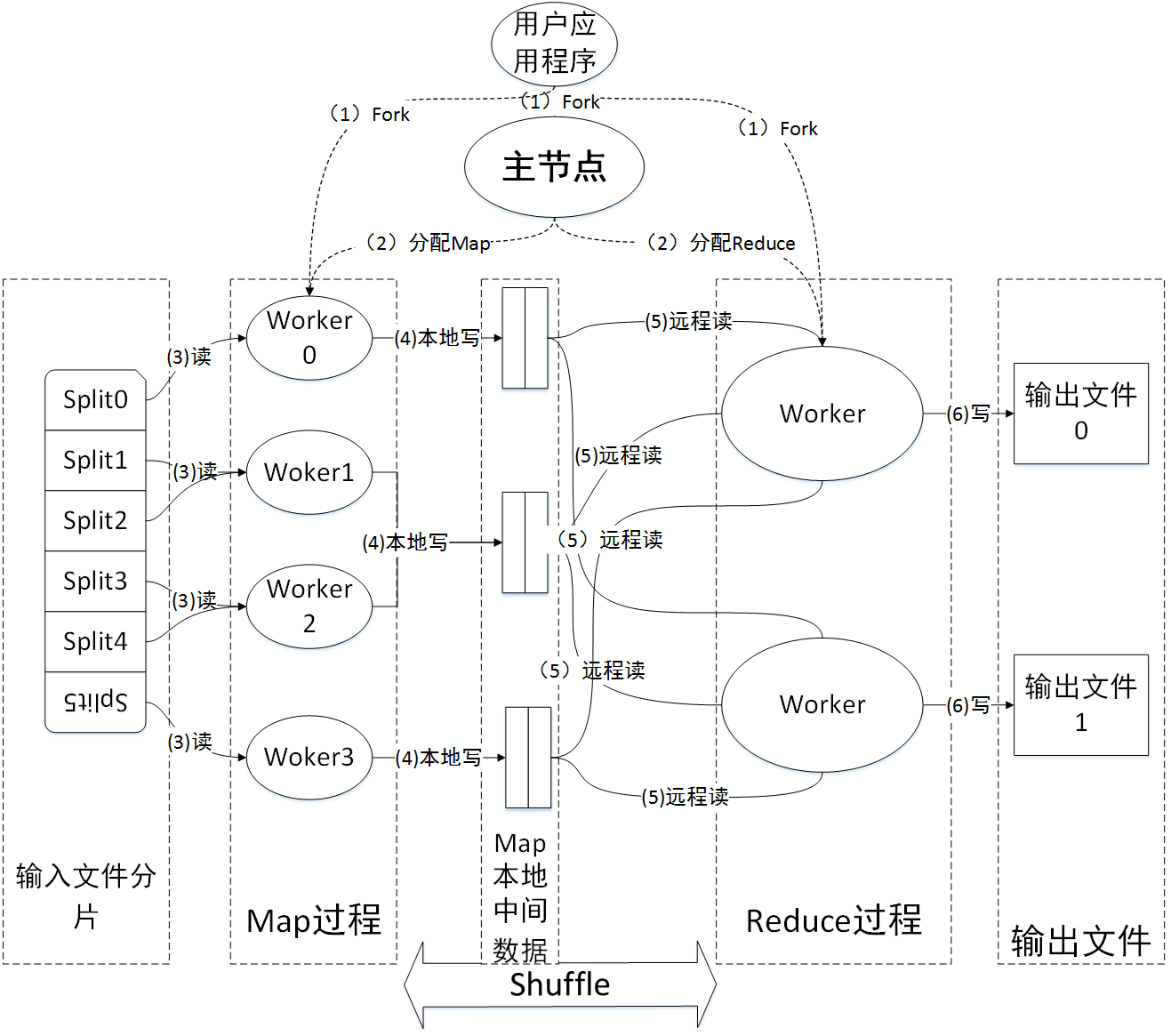


图2-3 MapReduce架构图

MapReduce计算框架设计为Master/Slave架构，图2-3显示了MapReduce的架构设计。MapReduce中涉及到的有这样几个元素：用户Client，Master，Worker。MapReduce计算模型的框架如下：

1. 用户提交作业任务给主节点。
2. 主节点检查作业配置，分配Map和Reduce任务。
3. Map任务从输入文件中读入指定的分片数据，一般每个Map处理一个分片数据。
4. Map任务执行用户指定的map()操作，并将数据写入到本地磁盘中。
5. Reduce通过Shuffle过程从完成任务的Map端主动获取属于自己要处理的Partition。
6. Reduce取到所有Map的对应Partition输出后执行用户指定的reduce()工作并将结果写到分布式文件系统中。

Map和Reduce过程中输入数据逻辑上都是<Key, Value>格式。Map在处理完一条输入数据后会将输出数据先写入到本地内存环形缓冲区中，当缓冲区中的数据达到一定程度（默认为80%）时，Map会由Spill线程将内存中输出数据写到本地磁盘中。在Map运行过程中可能会出现多个输出文件，这些中间输出文件最终将Merge成一个输出文件。Merge过程总是按Partition聚集，按Value值排序。Partition是针对Key的聚集标准，通常情况下，这样可以保证Key值相等的记录会发送到同一个Reduce上处理，这是保证MapReduce计算模型正确性的基本要求。用户可以定制划分方法，但是必须保证达到这个要求。

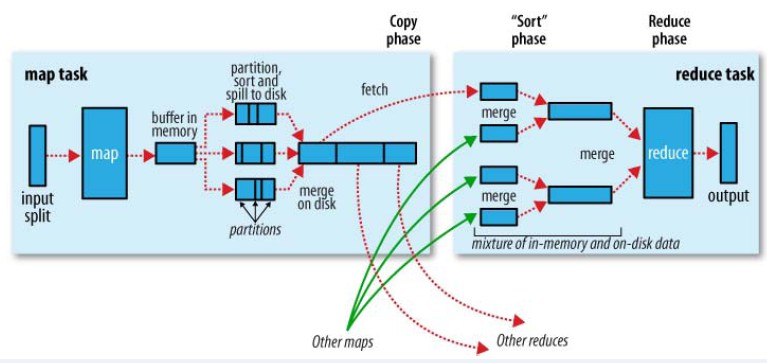


图2-4 Shuffle过程

每一个Reduce都有一个指定的需要处理的Partition数据。Reduce要能正确完成计算任务就需要从所有的Map输出文件中取得它所处理的Partition相关输出。Reduce通过Http协议从Map端拷贝文件到本地，拷贝过程中同时做Merge工作，Merge同样保证按Value有序。可见在MapReduce的Shuffle过程中一直保证了Value的有序性。

2.4 开源分布式计算平台Hadoop

Hadoop[43]是MapReduce的一个开源实现，最初是Apache开源搜索引擎Nutch的一个子项目。在大数据时代，Hadoop以其良好的扩展性、稳定性、容错性以及简洁的编程接口获得了巨大的成功。现在Hadoop已经成为大数据时代的标准。利用Hadoop可以方便有效地对大型数据集进行复杂的操作。

Hadoop系统主要有两个组成部分：分布式文件系统HDFS(Hadoop Distributed File System)和并行计算框架MapReduce。HDFS的设计目标是在通用的异构的廉价硬件上，具有高度容错性、可扩展性，针对一次存多次读，提供高吞吐量并发读的分布式文件系统。适合存储少量的大文件。HDFS放宽了（relax）POSIX的要求以此实现流的形式访问文件系统中的数据

HDFS设计为使用普通的廉价硬件，且大规模的数据必然会使用到大量的磁盘，因此系统中磁盘发生Failure的情况被认为是通常情况，因此HDFS在设计之初就针对经常的磁盘Failure做容错。HDFS有着高容错性的特点是通过对数据块的多备份机制实现的。HDFS中的数据都默认地被切分64M大小的分片，每个分片默认有3个备份，分别存储在一台机器上，与此机器同机架的一台机器上以及其他机架的一台机器上。当发现任意一台机器上发生数据丢失后，系统会自动从其他备份中去读取、复制，保证系统的三个备份。如此每个数据会有多个分片及其多个备份，这就使得在分配Map任务的时候可以保证较大的并发性。

Hadoop现在依然不仅仅是一个计算平台，随着Hadoop生态圈的不断成长，有越来越多的优秀项目加入到其中。如Hive、HBase、Pig等，它们或是对Hadoop的改进或者是对Hadoop的高层次包装抽象，使得其满足一些简单的应用接口定义。Pig提供了更加丰富的数据结构，一般为多值和嵌套的结构，还提供了更加强大的数据变换操作，包括MapReduce中忽略的连接(join)操作，其优点在于可以使用简单的控制台命令轻松处理TB级数据，对于研究人员和工程师而言操作数据更加方便简单；Hive则是构建于Hadoop之上的数据仓库框架，其设计目的是让SQL使用者能更加容易的操作HDFS上的海量数据，由于建立在Hadoop之上，很多组织将它作为一个通用且可伸缩的数据处理平台；HBase是对Google提出的BigTable的一种开源实现，是一个在HDFS上开发的面向列的分布式数据库，它不支持SQL，也不是关系型数据库，它自底向上地进行构建，在廉价的硬件集群上管理超大规模的稀疏表，却能够简单地通过增加节点达到线性扩展，是在HDFS上实时地随机读/写超大规模数据集的有效方案。

由于Hadoop的广泛认同性，且大量基于Hadoop之上的开源实用系统，对Hadoop本身的优化也可以使得这些系统获得进一步提升。因此本文选择Hadoop作为并行计算平台。

2.5 负载均衡

2.6 图数据表示

图数据具有极强地表示能力，

2.7 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 完全图和近似完全图枚举算法

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 完全图枚举算法

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 近似完全图枚举算法

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。

|  |
| --- |
| **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）  *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 Hybrid算法

信息整合模块主要负责

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. 并行环境下的负载均衡

4.1 完全图和近似完全图枚举的负载均衡

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

4.2 并行平台Hadoop中通用的负载均衡

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。

[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. C.Bron and J.Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. Communications of the ACM, 16(9):575–577, 1973.
2. Li Lu and Yunhong Gu. D MaximalCliques: A distributed algorithm for enumerating all maximal cliques and maximal clique distribution. IEEE International Conference on Data Mining Workshops. 2010.
3. F.Cazals and C.Karande. A note on the problem of reporting maximal cliques. Theoretical Computer Science, 407(1):564–568, 2008.
4. Shengqi Yang, Bai Wang and Haizhou Zhao. Efficient dense structure mining using MapReduce. IEEE International Conference on Data Mining Workshops. 2009.
5. D.Eppstein and D.Strash. Listing all maximal cliques in large sparse real-world graphs. In 10th International Symposium on Experimental Algorithms, 2011.
6. Bin Wu and Xin Pei. A parallel algorithm for enumerating all the maximal k-plexes. PAKDD 2007 workshops, LNAI 4819, 476-4831.
7. Arash Khosraviani and Mohsen Sharifi. A distributed algorithm for γ-Quasi-Clique extractions in massive graphs. INCT, 422-431, 2011.1
8. E.Tomita, A.Tanaka, and H.Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. Theoretical Computer Science, 363(1):28–42, 2006.
9. S.Tsukiyama, M.Ide, and I.Shirakawa. A new algorithm for generating all the maximal independent sets. SIAM Journal on Computing, 6(3):505–517, 1977.
10. F.Cazals and C.Karande. A note on the problem of reporting maximal cliques. Theoretical Computer Science, 407(1-3):564–568,2008.
11. N.Chiba and T.Nishizeki. Arboricity and subgraph listing algorithms. SIAM Journal on Computing, 14(1):210–223, 1985.
12. E.A.Akkoyunlu. The enumeration of maximal cliques of large graphs. SIAM Journal on Computing, 2(1), 1973.
13. K.Makino and T.Uno. New algorithms for enumerating all maximal cliques. In SWAT, Lecture Notes in Computer Science 3111, pages 260–272, 2004.
14. V.Stix. Finding all maximal cliques in dynamic graphs. Computational Optimization and Applications, (27):173–186, 2004.
15. Robert E. Osteen and Julius T. Tou. A clique detection algorithm based n neighborhoods in graphs. International Journal of Computer and Information Sciences, Vol.2, No.4, 1973.
16. D. Eppstein, M. L ofﬂer, and et al. Listing all maximal cliques in sparse graphs in near-optimal time. In ISAAC(1), pages 403–414,2010.
17. Thang Nguyen Bui and Paul H. Eppley. A hybrid genetic algorithm for the maximum clique problem. In Larry Eshelman, editor, Proceedings of the Sixth International Conference on Genetic Algorithms, 478-484, 1995.
18. S.Tsukiyama, M.Ide, and I.Shirakawa. A new algorithm for generating all the maximal independent sets. SIAM Journal on Computing,6(3):505–517, 1977.
19. James Cheng, Yiping Ke, Ada Wai-Chee Fu and Jeffrey Xu Yu. Finding maximal cliques in massive nerworks by h\*-graph. SIGMOD, June 6-11, 2010.
20. S. C. Racha. Load balancing map-reduce communications for efficient executions of applications in a cloud. 2012.
21. R.Rama, Suresh Badarla and Kamala Krithivasan. Clique detection algorithm using clique-self-assembly. 6th International Conference on Bio-Inspired Computing: Theories and Applications, 2011.
22. Jose L. Walteros and Panos M. Pardalos. A decomposotion approach for solving critical clique detection problems. SEA 2012, LNCS 7276, 393-404, 2012.
23. B. Wu, S. Yang, and et al. A distributed algorithm to enumerate all maximal cliques in mapreduce. In International Conference on Frontier of Computer Science and Technology, pages 45–51, 2009
24. L. Lu, Y. Gu, and et al. dmaximalcliques: A distributed algorithm for enumerating all maximal cliques and maximal clique distribution. In IEEE International Conference on Data Mining Workshops, pages 1320–1327, 2010.
25. S. Yang, B. Wang, and et al. Efﬁcient dense structure mining using mapreduce. In IEEE International Conference on Data Mining Workshops, pages 332–337, 2009.
26. M.C.Schmidt, N.F.Samatova, K.Thomas, and B.H.Park. A scalable, parallel algorithm for maximal clique enumeration. Journal of Parallel and Distributed Computing, (69):417–428, 2009.
27. J. Cheng, L. Zhu, and et al. Fast algorithms for maximal clique enumeration with limited memory. In KDD, 2012
28. Mattew C.Schmidt. A scalable, parallel algorithm for maximal clique enumeration. Journal of Parallel and Distributed Computing, Vol 69, pp 417-428, 2009.
29. S Szabo. Parallel algorithms for finding cliques in a graph. 5th International Workshop on Multi-Rate Processes and Hysteresis. Journal of Physics: Conference Series, 2011.
30. B.Balasundaram, S.Butenko, I.V.Hicks, and S.Sachdeva. Clique relaxations in social network analysis: the maximum k-plex problem. Operations Research, 59(1):133-142, 2011.
31. M.Haraguchi and Y.Okubo. A method for pinpoint clustering of web pages with pseudo-clique search. Federation over the web, Vol.3847 of Lecture Notes in Computer Science, pp 59-78, 2006.
32. Guimei Liu and Limsoon Wong. Effective prunning techniques for mining quasi-cliques. Machine Learning and Knowledge Discovery in Databases Lecture Notes in Computer Science, volume 5212, pp 33-49, 2008.
33. James Abello, Mauricio G.C. Resende, and Sandra Sudarsky. Massive Quasi-clique detection. LATIN’02 LNCS 2286, 586-612, 2002.
34. B.McClosky and I.V.Hicks. Combinatorial algorithms for the maximum k-plex problem. Journal of Combinatorial Optimization, (23):29–49, 2012.
35. Bin Wu and Xin Pei. A parallel algorithm for enumerating all the maximal k-plexes. PAKDD 2007 workshops, LNAI 4819, 476-483.
36. Arash Khosraviani and Mohsen Sharifi. A distributed algorithm for γ-Quasi-Clique extractions in massive graphs. INCT, 422-431, 2011.
37. G. Benjamin, A. Nikolaus, R. Angelika, and A. Kemper. Handing data skew in mapreduce. In Proceedings of the 1st International Conference on Cloud Computing and Services Science, volume 146, pages 574–583, 2011.
38. Kwon Y C, Balazinska M, Howe B, et al. Skewtune: mitigating skew in mapreduce applications[C]//Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM, 2012: 25-36.
39. 于戈, 谷峪, 鲍玉斌, 等. 云计算环境下的大规模图数据处理技术[J]. 计算机学报, 2011, 34(10): 1753-1767.
40. Jian Pei, Daxin Jiang and Aidong Zhang. On mining cross graph quasi-cliques. KDD’05 August 21-24, 2005.
41. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.
42. MapReduce in <http://zh.wikipedia.org/wiki/MapReduce>
43. Hadoop in <http://hadoop.apache.org/>

# 在校期间发表论文

# 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 ，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