分类号 密级

UDC

学 位 论 文

面向异步大规模图计算模型的图划分方法研究

|  |  |  |  |
| --- | --- | --- | --- |
| 作者姓名： | 李玉国 | | |
| 指导教师： | 张岩峰 教授 | | |
|  | 东北大学计算机科学与工程学院 | | |
| 申请学位级别： | 硕士 | 学科类别： | 工学 |
| 学科专业名称： | 计算机应用技术 | | |
| 论文提交日期： | 2016年12月 | 论文答辩日期： | 2016年12月 |
| 学位授予日期： | 2017年1月 | 答辩委员会主席： | 高某某 |
| 评阅人： | 蒋某某、刘某某、张某某、黄某某 | | |

东 北 大 学

2017年1月

##### A Thesis in Computer Software and Theory

## **The Research of Graph Partitioning Methods in Large Scale Asynchronous Graph Processing Model**

By Wang Mou

Supervisor: Professor Wang Moumou

**Northeastern University**

**January 2016**

# 独创性声明

本人声明，所呈交的学位论文是在导师的指导下完成的。论文中取得的研究成果除加以标注和致谢的地方外，不包含其他人己经发表或撰写过的研究成果，也不包括本人为获得其他学位而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均己在论文中作了明确的说明并表示谢意。

学位论文作者签名：

日 期：

学位论文版权使用授权书

本学位论文作者和指导教师完全了解东北大学有关保留、使用学位论文的规定：即学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人同意东北大学可以将学位论文的全部或部分内容编入有关数据库进行检索、交流。

作者和导师同意网上交流的时间为作者获得学位后：

半年 □ 一年□ 一年半□ 两年□

学位论文作者签名： 导师签名：

签字日期： 签字日期：

# 摘 要

随着web网络、社交网络、生物信息网络等规模的不断增加，隐藏在数据中的价值也逐渐得到了关注,如何从海量的数据中发掘出数据的价值以成为当前一个重大挑战，如广告投放、社区发现、搜索排序等需求的实现都需要对海量的数据进行处理。为应对这一挑战，Google提出了一个针对于图数据处理的分布式系统Pregel，之后针对图数据的处理系统得到了学术界和工业界的广泛研究和关注。Maiter是DAIC计算模型的实现，是一个完全异步的大规模分布式图处理框架。因其低通信量和计算的高效性，Maiter框架的性能要优于当前主流的图处理框架，也得到了学术界的广泛认可。为了进一步提高Maiter框架的性能和可用性，本文的工作将从以下两个方面展开。

为进一步提升Maiter框架的性能，本文采用多副本顶点的计算引擎，实现了更小粒度的计算调度和提高了消息的流通性。1）在现有多副本顶点计算模型的基础上结合Maiter消息传递的非阻塞性，设计和实现了MR-DAIC计算模型，使消息在计算过程中得到更快的传递。2）从理论上证明了MR-DAIC中副本顶点之间的消息通信机制，保证了计算模型的正确性。3）通过区分副本顶点（通信顶点和计算顶点），设计实现了副本顶点之间高效的通信机制，降低了计算过程中通信量。

现实世界中图数据的顶点度往往服从扭曲分布，这种分布会导致分布式计算框架在计算、通信、存储等方面负载的严重不均衡，为提高Maiter面对这类数据的可用性，本文采用Vertex-Cut的图划分方法来解决这一问题。1）在研究和分析Balanced p-way，Greedy、HDRF等图分区算法后，结合DAIC计算模型中消息传递通信，首先提出了消息流通性的概念，并基于此实现了一个面向异步图处理框架的图分区算法PAGraph（Partiting For Asynchronous Graph Processing Model），实验证明该分区算法使得算法的收敛速度得到了有效提升。2）Vertex-Cut的图划分方法会导致大量的副本顶点被创建，从而给框架增加了大量的计算负载，在处理低度顶点大量存在的图数据时性能非常的低效。通过实验分析和借鉴现有的一些图分区算法（.e.g.Powerlyra）,提出了一种基于动态决策的点/边切割相结合的图分区算法VE-PAGraph。通过对Maiter图划分方法的改进，使其在处理扭曲分布或者正态分布的图数据都具有很好的可行用性。

综上，经过本文的一系列工作，Maiter框架的性能和可用性进一步提升，为Maiter框架的实际应用创造了有利的条件。

关键词：云计算；Partitioning Algorithm；DAIC；MR-DAIC；Edge-Cut；Vertex-Cut；PAGraph；VE-PAGraph

# Abstract

The rapid development of Internet and communication technology has brought tremendous changes to the way people work and live. In a larger, faster, more secure, more conventient NGI(Next Generation Internet) that integrates wired and wireless access technologies, enjoying high-quality network service anywhere and at any time has become people’s necessary requirement, which indicates that NGI must provide QoS(Quality of Service) guarantee and support ABC(Always Best Connected). Therefore, people can choose the “best” way to access to network and enjoy the service.

**Key words:** next generation internet; always best connected; handoff; quality of service; utility; genetic algorithm

目 录

[独创性声明 I](#_Toc466277918)

[摘 要 II](#_Toc466277919)

[Abstract IV](#_Toc466277920)

[第1章 绪 论 - 1 -](#_Toc466277921)

[1.1研究背景 - 1 -](#_Toc466277922)

[1.2 目的及意义 - 5 -](#_Toc466277923)

[1.3 国内外研究现状 - 6 -](#_Toc466277924)

[1.4 主要研究内容 - 9 -](#_Toc466277925)

[1.5 论文章节安排 - 10 -](#_Toc466277926)

[第2章 DAIC模型及Maiter框架 - 12 -](#_Toc466277927)

[2.1 同步迭代计算 - 12 -](#_Toc466277928)

[2.2 传统异步迭代计算 - 14 -](#_Toc466277929)

[2.3 DAIC计算模型 - 16 -](#_Toc466277930)

[2.3.1 DAIC计算模型的前提条件 - 16 -](#_Toc466277931)

[2.3.2 DAIC计算模型的推导 - 17 -](#_Toc466277932)

[2.3.3 DAIC计算模型的优先级迭代计算 - 18 -](#_Toc466277933)

[2.4 图数据处理系统Maiter - 19 -](#_Toc466277934)

[2.5 本章小结 - 24 -](#_Toc466277935)

[第3章 MR-DAIC计算模型及实现 - 25 -](#_Toc466277936)

[3.1 MR-DAIC模型 - 25 -](#_Toc466277937)

[3.1.1 MR-DAIC模型 - 25 -](#_Toc466277938)

[3.1.2 MR-DAIC模型与多副本顶点模型的比较 - 25 -](#_Toc466277939)

[3.2 MR-DAIC模型正确性证明 - 25 -](#_Toc466277940)

[3.3 计算引擎的实现 - 27 -](#_Toc466277941)

[3.3.1 存储结构设计 - 27 -](#_Toc466277942)

[3.3.2`计算副本顶点和计算副本顶点 - 27 -](#_Toc466277943)

[3.3.3 计算顶点顶点和通信副本顶点 - 27 -](#_Toc466277944)

[3.4 本章小结 - 27 -](#_Toc466277945)

[第4章 面向异步图计算模型的图划分方法PAGraph - 28 -](#_Toc466277946)

[4.1 现有图分区方法介绍 - 28 -](#_Toc466277947)

[4.1.1 Edge-Cut（random、kcore） - 28 -](#_Toc466277948)

[4.1.2 Vertex-Cut（random、Greedy、hdrf） - 28 -](#_Toc466277949)

[4.2异步计算模型中消息的流通性 - 28 -](#_Toc466277950)

[4.2.1不同计算模型中数据流分析 - 28 -](#_Toc466277951)

[4.2.2数据流通性和通信量之间的关系 - 29 -](#_Toc466277952)

[4.2.3 消息流通量 - 29 -](#_Toc466277953)

[4.3 一种面向异步图计算模型的图分区方法PAGraph - 29 -](#_Toc466277954)

[4.3.1 PAGraph分区算法 - 29 -](#_Toc466277955)

[4.3.2 算法有效性证明 - 30 -](#_Toc466277956)

[4.3.3 算法实现 - 30 -](#_Toc466277957)

[4.4 改进的图划分方法VE-PAGraph - 31 -](#_Toc466277958)

[4.4.1 VE-PAGraph分区算法 - 31 -](#_Toc466277959)

[4.4.2 参数设置 - 31 -](#_Toc466277960)

[4.4.3 算法实现 - 31 -](#_Toc466277961)

[4.5 本章小结 - 32 -](#_Toc466277962)

[第5章 实验及评价 - 33 -](#_Toc466277963)

[5.1 实验的软硬件环境 - 33 -](#_Toc466277964)

[5.1.1 实验环境 - 33 -](#_Toc466277965)

[5.1.2 实验数据集 - 33 -](#_Toc466277966)

[5.2 MR-DAIC性能分析 - 33 -](#_Toc466277967)

[5.3 图分区算法性能验证 - 33 -](#_Toc466277968)

[5.3.1 PAGraph性能验证 - 33 -](#_Toc466277969)

[5.3.2 EV-PAGraph性能验证 - 33 -](#_Toc466277970)

[5.4 本章小结 - 33 -](#_Toc466277971)

[第6章 总结及展望 - 34 -](#_Toc466277972)

[6.1总结 - 34 -](#_Toc466277973)

[6.2 展望 - 35 -](#_Toc466277974)

[参考文献 - 36 -](#_Toc466277975)

[致 谢 - 1 -](#_Toc466277976)

[攻读硕士学位期间的论文和项目情况 - 1 -](#_Toc466277977)

[- 1 -](#_Toc466227366)

# 第1章 绪 论

## 1.1研究背景

网络社交、网络通信、网上购物等应用已经成为人们生活当中必不可少的一部分，数以亿计用户规模互联网产品正在不断涌现，这些应用和用户所产生的海量数据已经成为各个互联网公司最宝贵的财富（如商品推荐、广告投放等业务的实现）。随着社会信息化程度和信息收集技术的不断提升，未来人们将面对更加海量的数据，那么如何对如此海量的数据进行高效存储、快速访问、计算处理，已成为当前工业界和学术界着力解决的难题。从人类社会发展史来看，人们探索世界的方式已经从计算科学转换到了数据科学，人类社会进入了DT时代，大数据的相关处理技术也就应运而生。

面对海量的数据，通过升级单机的硬件配置以无法应对这一挑战，目前人们通常采用分布式技术来横向的扩展计算能力。随着高性能互连技术(High Performance Interconnects, HPI)的发展（如万兆以太网和 Infiniband），以及EC2等云平台的发展，这使得传统的高性能计算如地震分析、生物医学等，都可以采用大规模廉价服务器集群来代替超级计算机完成任务[16]。目前，采用大规模集群来解决海量数据的挑战已成为主流的解决途径，同样对于海量图数据处理也是如此。

在巨大的应用需要和学术研究价值下，近些年来大数据处理的分布式系统得到了迅速的发展，如“家喻户晓”的Hadoop、Spark，在Google公司提出图数据处理系统Pregel之后，图数据处理领域得到了学术界和工业界极大关注，出现了一大批高效的图数据处理系统，如典型的图计算系统有Giraph、GraphLab、GraphX等。但是，面对这图数据规模的不断增大，数以亿计顶点的大图已随处可见，这就需要对现有的图数据处理系统进行不断地优化和改进。其中，如何对海量图数据进行合理的划分，如何提高计算并行性、计算效率是这个非常困难问题，现已成大数据领域研究的热点。本次硕士论文的主要工作将围绕这两个方面展开，下面将首先对分布式计算、图计算、多粒度的计算模型、图划分方法进行简要介绍。

（1）分布式计算

分布式计算技术起源于美国的Intel公司[13]，目前已发展成为一门重要的计算机科学。分布式计算采用分治策略思想，研究如何把一个需要非常巨大的计算能力才能解决的问题分成许多小的部分，再将这些部分分配给多个计算机进行处理，最后把这些计算结果综合起来得到最终的结果[11-12]。如图1.1。目前典型的分布式计算技术有中间件技术、移动Agent技术、P2P、Web Services、网格计算、普适计算、云计算等。

根据对大数据的处理形式可以将分布式计算技术以下三个领域：静态数据的批量处理、在线数据（包括流数据和交互式数据）的实时处理、对图数据的处理[14,15]。其中批量数据处理系统，处理的数据往往具有数据静态储存、数据价值



图1.1 分布式计算原理

Fig. 1.1 The principles of distributed computing

密度低、数据量大等特征。它的应用领域非常广泛，例如对搜索引擎系统产生的海量的日志数据进行处理分析用户查询与点击行为，来改善与提高搜索引擎的用户体验和性能，目前具有代表性的批量数据处理系统有Google的MapReduce [28,29]、Apache软件基金会的Hadoop生态系统[19]和微软的Dryad [20]等。流数据处理系统处理的数据往往具有动态性、价值具有时效性、易失性，其应用领域也很多，例如对日志信息进行实时的采集、处理，这样能够及时的挖掘出有价值的信息、实时的展示信息变化、达到动态提醒与预警等功能，目前具有代表性的流数据处理系统有Twitter的Storm [21]、Facebook 的 Puma [22]等。交互式数据处理系统具有数据需要能及时地进行处理，同时结果可能需要立刻被返回等特征，它的应用主要分为两类:人机间交互和人际间交互，如即时通讯、百度知道等。目前具有代表性的交互式处理系统有Facebook的Hive [23,24]、Google 的 Dremel [25]等。图处理系统是对具有图这种数据结构的数据进行处理，操作的基本单位是顶点和边。对图数据处理的分析和挖掘算法往往采用迭代计算求解的方式解决问题，所以图数据处理系统对迭代计算具有很好的支持，目前具有代表性的图数据处理系统有Google 的Pregel[26]、卡内基梅隆大学的GraphLab(单机版本[27]和分布式版本[28])。本次硕士论文主要对图数据处理系统进行研究，在下一小节将会对图数据处理系统作进一步的介绍。

（2）图数据处理系统

图这种数据结构具有很强的抽象和对事物的表达能力，图中的顶点抽象为一个基本的事物，边抽象为两个事物之间的联系。例如在社交网络中的用户用顶点表示，边来刻画用户之间的关系。当前社会中的很多应用服务及其产生的数据都可以很自然的表达为图这种数据结构，例如社交网络、web网络、传感器网络等，这些服务已经深入到了人们的生活当中，因此对这类数据进行分析和信息的挖掘将会极大的提升用户体验，这使得图数据处理系统的研究具有十分重要的理论研究和应用价值。对图数据处理系统的研究和优化主要包含以下两个方面：图数据存储和计算模型。

图计算具有多次迭代计算和频繁数据访问的特性，这要求图数据处理系统必须设计良好的数据存储结构来减少数据访问带来的开销。从存储角度看，图数据处理系统可以分为面向分布式存储和面向单机多核共享存储两类架构。对于分布式存储架构，一个最核心的问题就是如何对大规模的图数据进行划分，尽量的减少网络通信的开销，并同时保证各个数据分区的负载均衡。对于单机环境的存储架构，一般会通过设计高效的数据结构来较少IO开销，例如单机版本的GraphLab [29]采用全内存的存储方式来加快数据的访问速度，GraphChi [30]则利用滑动窗口的设计实现对数据的顺序访问，减少了数据的随机访问提高了IO性能。其中分布式存储架构是目前研究的主流方式，但调查显示目前很大一部分数据通过单机仍然是可以处理，因此对于这两种框架各有利弊，对它们进行研究应用到不同的领域都是具有重要价值意义的。

从计算模型的通信角度看，图数据处理系统分为同步框架和异步框架两种。对于同步图处理框架，在迭代计算中必须保证上一轮迭代计算全部完成后，才能进行下一轮的迭代计算。同步框架的BSP模型与图计算中的迭代计算模型具有天然的匹配性，因此很容易保证计算的正确性，这使得其应用的范围很广，具有代表性的框架有Pregel与Giraph [31];。而在异步框架中没有迭代次数的概念，图计算中可以使用最新的结果作为输入，这样相对于同步图处理框架可以加快算法的收敛速度，因此研究异步图数据处理框架能有效对改进图数据处理框架的性能具有具有重要的价值意义。其缺点是在算法的正确性上很难保证，其具有代表性的框架有GraphLab、Maiter[32,33]。

（3）多粒度的计算模型

从计算模型的计算粒度看，可以分为以边为中心、以点为中心、以路径为中心、以块为中心和以子图为中心等计算模型，它们通过采用灵活的调度策略、提高Cache命中率等方面对图数据处理系统进行优化。

对于优先级调度计算，对于边的权重有区分的算法最基本的计算单位是边，而边权重无区分的算法最基本的计算单位是点，大部分的图挖掘算法属于后者。对于优先级计算调度的粒度越小，那么理论上优先级获取的性能提升也就越大，而实际应用中为实现优先级计算需要获取一个优先级队列，那么粒度越小获取优先级的开销也就越大，因此在实际应用中应该统筹处理好这两者之间的关系。

对于并行性，现有的图数据处理框架最基本的计算单位都是顶点，而现实世界中数据的扭曲分布使得每个顶点计算的开销差别很大，这使得顶点之间的并行性很差。如果采用优先级的调度，那么就不能对优先级高的高负载顶点快速完成计算，从而影响优先级计算的性能。

对于数据访问，以路径为中心、和以子图为中心的计算模型，通过尽量避免对顶点或者边的切割，减少计算过程中的通信量；以数据块为中心的计算模型，通过采用对一个数据块的多次计算的调度方式，提高对cache的命中率从而提高系统的性能。

（4）图划分方法

在分布式环境下，对海量的图数据进行处理，首先面临的问题就是如何将数据比较均匀地分配到各个机器节上。对于非图数据来说，数据中的数据记录相互独立，这个问题的解决就非常的直观，只需将数据均衡的分配到各个机器上就可以了，数据切分算法不用考虑其他的约束条件。对于图数据来说，数据记录之间往往有着很强的耦合性。因此，对于图数据分片的问题，不仅仅要考虑机器之间任务量分配是否均衡的问题，还要考虑机器节点之间通信量的问题。图划分已经被证实是一个NP-Hard问题，目前是图数据处理领域的一个热点研究问题。

根据在图划分中切割单位的不同分为：Edge-Cut和Vertex-Cut。Edge-Cut是指通过切割顶点之间的边，把一个大图划分成若干个小的分区，在这中方法中通信量等价于切割边的数量，采用这种方式的图数据处理系统有Pregd和分布式版本的GraphLab。Vertex-Cut是指通过切割顶点，把一个大图划分成若干个小的分区，在这种方法中通信量等价于顶点的副本数量，在面对扭曲分区的图数据时Vertex-Cut相对于Edge-Cut产生更少的通信量，采用这个方式的图数据处理系统有PowerGraph[34]，GraphX [35,36]。随着人们对图划分方法的研究，提出了Edge-Cut和Vertex-Cut相结合的图划分方法，来减少通信量的同时也避免产生过多的副本顶点带来计算负载的增加，采用这种方式的图数据处理系统有Powerlyra[37]。

## 1.2 目的及意义

课题研究目的：本次硕士论文的主要工作旨在通过研究异步图处理框架特性、多粒度的计算模型、图数据自身特性及图分区算法，基于异步图处理框架Maiter实现一个支持多粒度计算的图处理框架，实现一种面向异步图计算模型的图分区算法，最终达到进一步提升Maiter框架性能和可用性的课题研究目标。

理论意义和应用价值：同步类型的图处理框架可以支持所有迭代计算的图算法，具有通用性的优点，但在迭代计算中超级步之间的同步往往会使框架的整体性能取决于集群中性能最差的那个机器，而异步图处理框架很好的避开“流浪者”问题，这使得异步类型的分布式图处理框架在性能方面要往往优于同步类型的分布式图处理框架，因此基于异步类型的图处理框架进行深入的研究具有十分重要的意义。

在分布式的迭代计算中，如何能够使用最新的中间结果进行下一个超级步的计算，将会极大的提升算法的收敛速度。本次硕士论文中，采用多副本顶点计算技术，为顶点在多个计算节点上创建副本，副本顶点在收集了本地消息后不用通过网络的传输就能够立即对消息应用，实现了计算的本地化。同时在大规模的计算中，对计算进行优先级调度会很大程度上提升计算的效率，而在优先级调度中的一个关键因素就是调度的粒度。在处理好调度带来开销的问题后，通常调度的粒度越小越能获得更好的性能提升。本次硕士论文中，将副本顶点作为优先级计算调度的基本粒度，在处理幂率分布的图数据时，这种细粒度的优先级调度更能取得更好的效果。

在对大规模的图数据进行分布式处理之前，需要首先将数据进行划分，图的划分方式的不同将会直接影响分布式计算过程中通信量和负载均衡问题。本次硕士论文将结合异步框架的特性和多种图分区算法，提出了更为高效的图分区算法，这对进一步提升异步图处理框架在处理大规模图数据时的性能和可用性具有十分重要的价值意义。

## 1.3 国内外研究现状

大数据处理系统研究现状：Hadoop仍然是目前研究最为深入，应用最为广泛的大数据处理框架。在对Hadoop的优化中，研究深入到负载均衡、任务调度、容错处理、参数调优、HDFS[3,7]存储结构优化等方面，提升了Hadoop的性能和健壮性。在对Hadoop应用的扩展中，围着这HDFS、MapReduce这两大核心模块，根据具体的应用领域的需要实现了很多的上层框架如数据仓库Hive、实时数据库系统HBase、脚本Pig等形成了一个Hadoop生态系统。为了获取大数据处理的更好性能和提供框架的易用性，Spark作为新一代的大数据处理领域的一颗新星正在崛起，它采用使用内存存储的SSD实现数据的存储，并提高了灵活的API，它也像Hadoop一样逐渐形成了自己的生态圈，包括Mlib、GraphX、Streaming等。

没有任何一个框架可以使用于所有的应用并保持高效，每个特定的应用领域都有着本领域的特性，这也是本文前面介绍的Hadoop、Spark等生态系统建立的原因。所以针对于不同的领域进行具体的研究是非常有意义的，对于图数据的处理系统来说也是如此，目前针对于图数据的处理系统已成为大数据处理领域的热点研究问题。

图数据处理系统研究现状：对于图数据处理系统的研究最早由Google提出，它的图处理系统Pregel以顶点为中心计算模型，利用超级步来保证算法的正确性，这种编程模型易于将算法分布式实现，但迭代计算来说，每两个超级步之间的都需要一个同步操作，这极大的影响了算法的收敛速度，所以更为高效的异步处理框架应运而生。异步处理框架的典型代表是Graphlab框架，在计算过程中以子图为中心，创新的使用不同的封锁机来保证算法正确性的同时实现异步的计算，避免了同步系统的同步开销，但是这种锁机制也带来了很多的开销，使得框架在整体性能上得到很好的提升。本论文中研究的Maiter也是一个异步模图数据处理系统，它是对DAIC (Delta Based Incremental Computation)计算模型的实现，DAIC 从理论上保证了各个计算节点之间完全自治，无需任何同步操作，实现了完全的异步计算。这使得Maiter相对于Graphlab的异步化更加的彻底，并且Maiter中不需要任何的加锁操作避免了Graphlab中加锁而带来的大量开销。同时Maiter利用DAIC的特性很好的实现了优先级计算，加快了算法的收敛速度。同样Maiter也存在着它的缺点，它只能实现符合DAIC计算模型约束条件的算法，约束条件的最小集还有待提取和应用范围还有待扩展。

本次硕士论文主要针对图数据处理系统中的计算粒度和图划分方法的研究，来提升大规模图处理的效率。下面的内容将会对图计算中的计算粒度和图划分方法的研究现状进行介绍。

对于图计算中计算粒度的研究，按照计算单位的不同可以将现有的图数据处理系统分为以下类型。

（1）以点为中心的图处理模型：以顶点为中心的图处理模型，是目前最主流采用的处理模型，该模型最典型的代表框架是Pregel[2]。它在对图的处理过程将顶点作为处理的基本单元，按照顶点对数据进行分片和对数据分布式的计算。在数据分片阶段，采用Edge-Cut的分片方法，往往采用哈希顶点的id，将图数据分割成若干份并将其分发给相应的机器节点。其典型特征是，顶点只出现在一个机器上，而边会可能出现在多个机器上。在图分布式计算阶段，一般采用GAS的计算模型（GAS是对以顶点为中心计算模型的细化）。在整个图处理过程中，图处理模型一直将顶点作为中心进行操作。

（2）以边为中心的图处理模型：随着人们对大规模图数据的认识，我们在实际应用中产生的大部分图数据都是自然图。这些图数据（如，web图、社交网络图）都有着幂率分布的特性，这使得以顶点为中心的图处理模型显得不是很高效。因此，以边为中心的处理模型被逐渐应用到分布式图处理框架中，其中该模型最典型的分布式框架是PowerGraph[]。在该模型中，进行图数据的分片时是以边为中心，采用Vertex-Cut的分片方法，将图数据中的边均衡的分配到相应的机器上。该模型典型特征是，边只会出现在一个机器上，而顶点可能会出现在多个机器上。相对于以点为中心的图处理模型来讲，该模型是对实际应用考虑后，是对图处理模型的一次改进。

（3）以点边结合的图处理模型：综合以点为中心的图处理模型和以边为中心的图处理模型，而提出了点边相结合的图处理模型，实现该模型的分布式框架有GraphX[]。在数据分片中，分别以点和边为中心对图数据进行切分。其典型特征是，顶点和边都只出现在一个机器上，没有副本的存在。相对于以边为中心图处理模型来讲，这种点边结合的图处理模型既具有了其优点，同时由于顶点没有副本而减少了对数据一致性维护的开销。

（4）以路径为中心图处理模型：基于图数据中，数据位置信息不足的问题，提出了一种以路径为中心的图计算模型，分布式框架PathGraph[]就是对这一模型的一个实现。

（5）以子图为中心图处理模型：由于在图数据中，数据记录之间存在着很强的耦合性，在图分布式计算过程中机器之间会进行信息的交互。在以顶点为中心的计算模型中（包括GAS计算模型），会产生大量的消息而影响图计算性能。为了减少消息的通信量，在GoFFish[]中提出了以子图为中心的计算模型。这种计算模型可以有效的减少通信量，减少迭代计算的次数。

由于图数据内部元素的强关联性，划分方法的不同将直接影响计算过程中通信量的大小，这使得图划分方法一直以来都是专家们重点关注和研究的热点领域。但遗憾的是到目前为止，图数据划分问题尚未得到很好的解决。从切分图数据的不同角度来看，图数据分片算法主要分为两类：边切法（Edge-Cut）和点切法（Vertex-Cut）[16]。

边切法：切边法是图数据分片中最常见的一种思路，它通过切割图中的边将一个大的图分割成若干个子图，每个顶点仅出现在一个子图中。在分割过程中，要尽量将每个子图中分配到数量大致相等的顶点，也要尽量减少切割的边数，即在分割过程中要考虑负载和通信量两个因素。对于切边法的优化目标定义如下式（1）所示：

参数说明：表示边w(e)表示被切割边的数量，A(v)=m表示机器m上分配顶点的数量，表示不平衡因子。

目前，采用边切法实现图划分的图数据处理系统有Pregel、Giraph、Graphlab等。

1. 点切法：通过切割图中的顶点将一个大的图分割成若干个子图，每个边仅出现在一个子图中，一个顶点会出现在多个子图中。在分割过程中，要尽量保证每个子图中分配到数量大致相等的边，也要尽量减少切割的顶点数（顶点的副本数少），即在分割过程中要考虑负载和通信量两个因素。目前，采用边切法实现图划分的图数据处理系统有Pregel、Giraph、Graphlab等。对于切点法的优化目标定义如下：

参数说明：表示顶点A副本的个数，表示机器A上分配边的数量，表示不平衡因子（）。

目前，采用点切法实现图划分的图数据处理系统有GraphX、PowerGraph、HDRF等。

## 1.4 主要研究内容

异步大规模图数据处理系统Maiter实现了完全异步性的图计算，通过差值累积的计算方式、优先级计算极大的提升了算法的收敛速度。但是Maiter在处理度扭曲分布的图数据时，会出现负载不均衡和通信量大等问题，同时这也会降低优先级计算的性能，因此本次硕士论文将围绕解决这一系列的问题展开，主要工作包括以下两个部分：Maiter多粒度计算模型及通信模型的研究和面向异步图数据处理系统的图分区算法的研究。

（1）MR-DAIC计算模型的设计与实现

在分布式迭代计算中，如何能尽快的使用当前最新的中间结果，将会极大的提高分布式算法的收敛速度，如何保证优先级计算中基本调度粒度之间计算的并行性，才能保证优先级计算发挥其最大性能。图被划分成若干个分区之后，一个顶点与它相连接的顶点很可能分布在不同的机器上，那么如果为这些顶点在不同的机器上创建副本顶点，副本顶点就能立即的利用本地的消息，这将会提高算法的收敛速度；同时高度顶点被切分成若干个副本顶点，各个副本之间的度就不会差别很大，这样就保证了基本的计算粒度的并行性，从而保证了优先级计算的性能。在DAIC计算模型中，迭代计算是通过差值累积、异步计算的形式进行，这就为上述设计的实现提供了很好的“着力点”。

在本次论文中，我们提出了多副本顶点的差值累积计算模型（MR-DAIC），通过多副本顶点的计算来提高消息被利用的及时性和计算粒度之间的并行性。在论文的后续章节，将会研究如何从理论上证明MR-DAIC的正确性和如何设计顶点之间高效的通信模型。

（2）高效的图划分方法研究

在对图划分方法的研究中，主要考虑分区的负载和切割图而产生的通信量这两方面的因素，这通常是对于同步系统来讲的。在同步系统的迭代计算过程中，必须等上一轮的计算全部完成后，才能进行下一轮的计算（即使消息已经被传输到了目的顶点也必须等到同步之后才能被应用），而上一轮计算的完成是取决于各个机器的计算量和通信量，所以对于同步图处理系统用负载均衡和通信量来考量图分区算法是无可争议的。 但是在异步图处理系统中，消息的传递和利用没有任何的阻塞性，理论上一个消息一旦产生就能够被利用，那么在本次论文中，提出了一种面向异步图数据处理系统的图分区算法，使用各个分区的负载均衡和消息的流通性来度量算法的优劣。本论文的后续工作将会对流通性如何定义，如何证明其有效性等问题进行研究。

目前图划分方法主要有Vertex-Cut和Edge-Cut两种方法，在早期的图数据处理系统中往往采用Edge-Cut的图划分方法，但是随着现实世界中图数据扭曲分布的特性后，这种划分方法会导致负载不均衡、通信量大等问题。为应对这一数特性带来的挑战，目前的图数据处理系统通常采用Vertex-Cut的划分方法，使得上述问题得到了有效的解决，但该方法同样存在着缺点，它会创建大量的副本顶点从而给系统带来额外的负载。本论文采用Vertex-Cut和Edge-Cut相结合的图划分方法，实现在图划分中保证各个分区负载均衡、通信量低的前提下，尽量减少副本顶点的创建。本论文的后续工作将会对如何决策对于一个顶点是采用Edge-Cut还是采用Vertex-Cut这一复杂问题进行研究。

## 1.5 论文章节安排

本文一共分为六章，各章的内容组织如下：

第一章为绪论。首先介绍了大数据处理的需求和本课题的研究背景，对分布式计算、图计算、不同粒度的计算模型、图分区算法等概念进行了简要介绍；然后指出了本次论文要完成的目标和分析本课题研究的价值意义；然后对当前的研究现状进行介绍，之后对本文的研究内容及思路进行介绍；最后给出了本文的组织结构。

第2章对Maiter框架进行介绍。由于本文的主要工作都是围绕着Maiter框架展开的，为了让读者更好的理解本文的工作，本章将对Maiter框架的计算模型DAIC和Maiter框架的实现进行详细的介绍。

第3章对MR-DAIC模型及其实现进行介绍。为提高DAIC模型的并行性和加快消息被应用速率，本文提出了向副本顶点计算的迭代计算模型MR-DAIC。首先，本章将对该模型的原理、高效性、正确性进行介绍，然后在对计算模型的系统实现进行介绍。

第4章对面向异步图计算模型的图划分方法的介绍。本章中，首先介绍了本文提出中提出的消息流通性及流通量的概念，然后给出基于此而提出的图划分PAGraph的介绍，最后给出了对PAGraph的改进算法EV-PAGraph。

第5章为实验分析。本章采用大量不同类型的真实和合成数据作为实验数据集，通过大量的实验数据向读者展示了MR-DAIC模型和面向异步图计算模型的图划分方法PAGraph、EV-PAGraph的有效性，并对实验结果给出了细致的分析。

第6章，总结和展望。首先对本次论文完成的工作给出了总结性的描述，然后给出了本文需要继续研究和改进的问题。

# 第2章 DAIC模型及Maiter框架

在传统的同步或者异步迭代计算模型中，都是基于中间结果进行迭代计算。不同于传统的迭代计算模型，DAIC(delta-based accumulative iterative computation) 模型是一种基于“变化量”而设计的一种计算模型，在迭代计算中不断的累积这些变化量最终求得最终解。图数据处理系统Maiter是对DAIC模型的一个实现，基于DAIC模型的特性和系统的优化设计，使得Maiter成为图数据处理领域的佼佼者。

基于DAIC模型的累积特性，Maiter是一个完全异步性执行的图计算框架，在理论上消息的传递和利用不存在任何的阻塞性，避免了因同步操作而带来的巨大开销；基于DAIC模型的变化量（差值）特性，Maiter在计算过程中只需要对变化量进行传递和对具有变化量的顶点进行操作，避免了系统中可能出现的无用计算。同时基于DAIC差值计算的特性，Maiter也“自然”的实现了优先级迭代计算，提升了算法的收敛速度。

Maiter在系统设计方面，基于图计算中迭代计算和随机访问的特性，Maiter存储结构采用全内存实现，避免了计算过程中出现大量耗时的IO操作，同时也提高了数据定位、寻址的速率。在消息传递方面，Maiter采用消息通信机制有效的减小了分布式环境下各节点之间的的通信量。在语言实现方面，Maiter采用高效的程序设计语言C++实现，使系统在很多底层操作方面具有很好的性能。

基于Maiter独特的创新性和优秀的性能，目前越来越多的收到国内外学者的认可和引用。本章的余下部分将首先介绍传统的同步迭代计算模型和传统的异步迭代计算模型，然后再对差值累积的异步迭代计算模型（DAIC）的原理和Maiter框架的实现做详细的介绍。

## 2.1 同步迭代计算

迭代计算是计算机解决问题的一种常用方法，所谓的迭代计算就是为解决一个问题循环执行若干次相同的操作，上一轮的计算结果作为下一轮计算的输入，直到计算结果达到设定的终止条件。同步迭代计算是指在启动下一轮计算之前，需要通过全局同步来确保上一轮中所有活跃顶点的计算已经完成，从而保证计算的正确性。在算法2.1中，V表示本轮计算的顶点集合，V'表示本轮激活的顶点集合，VA是顶点V激活的顶点集合，M是本轮计算所产生的消息集合，v.update()表示对顶点v执行用户实现的更新操作。一次迭代计算过程如算法所示，首先对本轮中所有活跃顶点执行update操作（第5-7行），之后执行send（M）and receive()将消息传递到目的顶点（第9行），最后执行全局的同步操作barrier（第10行）确保所有活跃顶点的计算和消息更新完成。

在同步迭代计算模型中，每一轮的迭代计算只需要遍历执行当前所有的活跃顶点即可，所以调度开销小。同时同步模型中消息只会对下一轮的计算才可见，所以不需要对顶点执行任何的加锁操作。另一方面，同步迭代计算模型同样存在着其缺点：在数据流的控制方面，上一轮的消息只能在同步操作之后才能可见并被利用，不能使用当前最新的状态这就增加了迭代计算的次数；在控制流方面，只有上一轮中所有活跃顶点计算完成后，才能进行下一轮的计算，这就使得很多时候集群中很多的机器要等待最晚完成计算的那个机器，由于计算负载、通信负载的不均衡，硬件性能差异、机器当前负载不同等因素会导致最快完成任务的机器和最晚完成任务的机器存在一个很大的时间差，这就造成硬件资源的空闲，同时同步模型的同步操作也会造成一定的同步开销，这就降低了计算速度，延长了算法的收敛时间。

|  |
| --- |
| **算法2.1** 同步计算模型  输入：图*G*  输出：求解的属性集合 |
| 1. **while iterationNum <= max\_iterationNum do** |
| 1. ***if V= then break*** |
| 1. ***V*** |
| 1. foreach v V do |
| 1. {v,m,}  v.compute() |
| 1. ***M***  *M m* |
| 1. V' V' |
| 1. done |
| 1. send（M）and receive() |
| 1. barrier() |
| 1. V V' |
| 1. **iterationNum** |
| 1. done |

## 2.2 传统异步迭代计算

从上一小节的分析中可以发现，对于迭代计算来说同步模型并不是非常高效。同步图数据处理系统在迭代计算中的低效主要由迭代计算过程中的同步过程引起，为了解决这一问题，异步性的迭代计算被提出来。按照消息的获取方式的不同，异步迭代计算模型的具体实现主要有push和pull两种。

在图的同步迭代计算中，采用全局的同步路障来保证分布式算法的正确性，这是一种简单易实现的同步方式，但这种大规模的同步路障操作并不是保证算法正确性的必要条件。其实，图计算中通常将顶点作为计算的基本粒度，只要与该顶点相关联顶点的计算完成并收到这些消息，此时该顶点进行计算就已经保证了正确了。实质上，这种方式相对于同步模型是一种细粒度的同步模型，下面结合算法2.2对该模型的计算过程做一个形象的说明。从调度队列V中取出一个顶点v进行计算(第2行)，当缓存的消息数量达到设定的阈值后进行消息的发送和接收，同时判断消息的目的顶点是否接收到了所有必须的消息，如果接收完成则将该顶点加入到调度队列中（第5-8行）。

|  |
| --- |
| **算法2.2** 传统异步计算模型-Push方式  输入：图*G*  输出：求解的属性集合 |
| 1. while V != do |
| 1. v dequeue(V) |
| 1. {v,m } v.compute() |
| 1. ***M***  *M m* |
| 1. if size(M) > buffer\_size then |
| 1. push(M) and v'.receive() |
| 1. if v' .isSchedulable() then |
| 1. enqueue(V, v') |
| 1. done |

该异步方式相对于同步方式来说，提高了计算的并行性，减少了迭代计算过程中资源的闲置时间，但是它需要对每个调度单位进行加锁来保证算法的正确性，这相对于同步方式中只有一个全局同步锁需要付出更多的开销。实际应用中，为避免大量同步锁带来的开销采用了子图的思想，这其实是一种相对于同步方式和以顶点为粒度异步方式的一种平衡。在以子图为中心的计算过程中，当某个子图需要的消息都接收后，该子图就可以被调度进行计算。目前使用这一方式的图数据处理系统的典型代表是Graphlab。在数据的划分和分配方面，Graphlab将图划分成许多个子图，一个机器负责若干个子图的计算；在迭代计算过程中，以子图为调度单位，初始化所有子图为可调度状态，周期性的检测每个子图的状态，当某个子图收到所有其必须的消息后就将该子图标记为就绪状态，并加入到调度队列中等待系统引擎的执行，执行完成后该子图被标记为等待状态。从上述过程中可以看出，Graphlab采用以子图为粒度的异步调度方式，提高了计算的并行性，同时又避免了大量加锁而带来巨大开销的问题，这使得Graphlab相对于同步模型具有更出色的性能表现。

在上述的异步模型中，即使采用以子图为调度的粒度，加锁还是会带来很大的开销，为应对这一挑战，另一种通过主动拉去消息的异步方式（pull）被提出。Pull方式相对于上一种被动等待消息的异步方式（push），不再需要加锁的操作来保证算法的正确性。算法2.3给出该pull方式的异步模型的执行过程。Thread1负责计算任务，Thread2负责就绪顶点的检测。

在pull异步方式，顶点要不断地去探测自己计算所必须的值是否已经产生，需要付出更大的通信量，因此，pull方式的异步迭代计算只适用于通信负载可忽略不计的情况下。

相对于同步迭代计算模型，异步模型主要有三个方面的优势。1）顶点计算能够使用最近的计算结果，减少了迭代计算次数；2）不需要全局的同步，消除了最晚完成任务机器造成的延迟，提高了资源的利用率。3）异步迭代模型可以很好的支持优先级计算，进一步加快算法的收敛速度。另一方面，异步迭代计算模型需要在分布式环境下维护调度队列和保证数据访问的一致性、计算结果的正确性，这使得异步模型也付出一定的开销。实践证明，在通常情况下异步迭代计算模型性能要优于同步迭代计算模型。

|  |
| --- |
| **算法2.3** 传统异步计算模型-Pull方式  输入：图*G*  输出：求解的属性集合 |
| compute: |
| 1. while V != do |
| 1. v dequeue(V) |
| 1. {v,m } v.compute() |
| 1. ***M***  *M m* |
| 1. done |
| pull: |
| 1. **while (interval > threshhold ) do** |
| 1. *foreach (v* V0*) do* |
| 1. *v.pull()* |
| 1. if v' .isSchedulable() then |
| 1. enqueue(V, v') |
| 1. done |
| 1. done |

## 2.3 DAIC计算模型

在上一小节介绍的同步迭代计算模型和传统异步迭代计算模型中，都是利用上一轮或者最新的计算结果来更新顶点的当前属性值。在Maiter一文中通过分析大量的算法发现，有一类迭代算法的下一轮计算结果可以通过上一轮计算结果累加上本轮计算结果的变化量得到，DAIC模型就是对这类计算的抽象。DAIC模型是一个支持完全异步性计算、动态计算、并行计算的高效的图数据计算模型。本小节将从DAIC模型的原理及特性、前提条件及推导、优先级计算三个方面对DAIC模型进行全面的介绍。

### 2.3.1 DAIC模型原理及特性

（1）DAIC模型原理及定义

Delta-based Accumulative Iterative Computations（DAIC）是对一类迭代算法计算的抽象，迭代计算过程中不断累积“变化量”使顶点属性值最终达到收敛值。从计算模型的原理来看，DAIC 模型的核心思想就是“变化量”，基本原理就是对计算过程中变化量的不断累积，而不同于传统计算中依赖中间结果的更新方式。从编程模型来看，DAIC模型同其他的图数据处理系统一样，采用以顶点为中心的计算模型，即面向顶点的编程模型，用户实现对顶点的处理程序vertex-program，在迭代计算中vertex-program运行在每个顶点，最终完成对整个图的计算。

为实现DAIC模型抽象的计算，定义了两个更新函数如式2.1所示，2.1中第一个更新函数是结果对变化量的累积，更新收敛值；2.1中第二个更新函数是变化量对变化量的累积，更新变化量。k=1，2，3表示迭代次数，表示顶点j在第k轮迭代计算的状态，表示顶点j在第k+1变化量，是一个抽象运算符。公式2.1中第一个更新函数定义了顶点j在第k轮的结果通过第k-1轮的结果累积上第k轮的变化量计算得到，式2.1中第二个更新函数定义了顶点j在k+1轮的变化量是通过累积邻接顶点在第k的变化量计算得到。

（2.1）

上述的更新函数是以同步迭代计算形式进行的描述，这只是为了能够容易的描述DAIC模型，其实真正在DAIC模型迭代计算中是没有任何的同步的概念。下面将将以伪代码形式对DAIC模型的实际计算过程进行描述，DAIC模型也是一个以顶点为中心的图数据计算模型，Vertex-Program在每一个顶点运行，它包括receive操作和update操作，迭代计算过程如算法2.4所示。

|  |
| --- |
| **算法2.4** DAIC异步计算模型  输入：图*G*  输出：求解的属性集合 |
| Receive: |
| 1. while receiveing do |
|  |
| 1. done |
| Update: |
| 1. while !isConvergence() do |
| 1. foreach v V do |
|  |
| 1. for any i in neis() do |
| 1. Send value ) |
| 1. done |
|  |
| 1. done |
| 1. done |

Receive操作负责消息的接收，每当顶点接收到消息后，将消息值累积到当前的变化量中（第2行）；Update操作进行顶点属性值的更新，依次遍历每一个需要继续计算的顶点，首先更新该顶点的属性值（第3行），然后本次计算的变化量发送给相关的邻居顶点（第4~6行），最后将顶点累积的变化量清空（第7行）。

（2）DAIC模型计算的完全异步性

并行计算的完全异步性是DAIC模型在迭代计算中体现出的一个最重要的特性，相对于传统的异步图迭代计算模型（如Graphlab）具有更彻底的异步性。接下来本文将目前主流的三种图数据计算模型（Pregel的BSP模型、Graphlab的多粒度锁实现的异步计算模型和Maiter的DAIC模型）的数据流进行分析，最终使读者对分布式计算的同步操作及DAIC模型的完全异步性有一个清晰的理解。

调度次序：

{A,B}:t1

{A,C}:t1

{A,D}:t2

{B,A}:t2

{B,D}:t3

{C,A}:t4

{D,B}:t5

{D,C}:t6

消息产生

消息等待

消息应用

D

C

B

A

（a）

* t0----t1----t2----t3----t4----t5----t6----



A

BN

C

D

A

BN

CN

DN

* t0----t1----t2----t3----t4----t5----t6----



A

BN

C

D

A

BN

CN

DN

* t0----t1----t2----t3----t4----t5----t6----



A

BN

C

D

A

BN

CN

DN

（b）

（d）

（c）

图2.1 迭代计算数据流

Fig. 2.1 dataflow of iterate computing

从上图2.1（a）是一个简单图，包括四个顶点8条边，并在右边给出了计算的调度次序。图2.1（b、c、d）描述了在迭代计算过程中的一轮计算的数据流（为了便于描述异步模型也采用同步中轮数的概念进行描述），（b）描述的是同步计算模型中数据流，可以看出本轮的消息要想被下一轮计算应用，存在一个全局的同步保证所有的消息都被目的顶点接，其消息的总延迟时间t=4+2+5+1+5+4+3=24；（c）描述的是传统异步计算模型中的数据流，可以看出本轮中的消息是能否被目的顶点应用，只需要以该顶点为单位进行一次同步保证该顶点收到所有的消息即可, 其消息的总延迟时间t=2+4+5+1=12；（d）描述的是DAIC异步计算模型，可以看出消息一旦产生就可以可以被目的顶点立即应用，不存在任何的同步。综合对这三种计算模型数据流的分析，可以得出在迭代计算中同步模型的同步粒度是整个图、传统异步模型的同步粒度是图顶点、DAIC异步模型的同步粒度是边（无任何同步），其消息的总延迟时间t=0。

### 2.3.2 DAIC计算模型的前提条件及推导

DAIC模型是对一类算法计算的抽象，并不是所有的迭代计算都能转化DAIC模型的计算形式。下面将给出DAIC模型的一个充分条件，并结合大家熟知的PageRank算法进行具体化的说明。

（1）前提条件

条件1：累积性表示

迭代计算中的更新函数一般形式：

DAIC模型要求的第一个条件就是上述更新函数可以表示成累积的形式，如公式2.3所示：

是顶点j在第k轮迭代计算的状态，f()为算法的更新函数，表示累积运算符。在PageRank算法中，该累积条件的表示形式如下。

PageRank算法的更新函数为：

上式中，d表示阻尼系数，|N(i)|表示网页i链出的网页数，(i,j)是网页i指向网页j的一个连接，从上式可以看出更新函数是通过“+”运算符累加各个指向它的邻接顶点的值来更新顶点当前值，很明显满足DAIC模型累积性表示的前提条件。

条件2：传递函数g()的满足分配率：

那么对应PageRank算法的表现形式为:

上式明显成立，所以PageRank也满足DAIC模型的条件2传递函数的分配率。

条件3：运算符具有交换律和结合律：

（2.4）

对应Pagerank算法中，“”运算符是“+”运算符，很明显“+”运算符是满足交换律和结合律的，因此Pagerank算法也满足DAIC模型的第三个前提条件。

条件4：满足等同律，即存在抽象值满足：

对应Pagerank算法中“+”运算符，存在自然数0满足上述条件。

上述这四个条件，是判断图算法是否可以转换为DAIC模型的一个判定依据，但在推导过程中可以很容易的看出，这些条件均为充分条件而非充要条件。

（2）DAIC模型推导

DAIC模型的核心思想就是对“变化量”不断的累积，公式2.6给出了变化量的以形式化定义，表示顶点j在第k轮的结果等于其在第k-1轮中的结果累积上他们之间的变化量。

上文中提的条件1（更新函数的累积性）保证了算法更新函数可以表示成：

将变化量的定义公式2.6带入上式得：

根据前提条件2传递函数的分配率，上式转化为：

根据前提条件3运算符的结合律和交换律，上式可转化为：

再根据前提条件1，上式可以被进一步转化为：

(2.7)

根据上式2.7和公式2.6可以得到：

(2.8)

对公式2.7、公式2.8进行进一步的推导，再根据四个前提条件，就能得到完全异步性执行的计算形式。我们可以观察到计算中的变化量并不需要必须是两次迭代的差值，而是可以是任意的粒度，不存在迭代轮数的概念，这就使得DAIC模型成为一个完全异步化的迭代计算模型。

在DAIC计算模型中，实现了最细粒度的计算，只要顶点接收到一个消息，就可以对消息进行利用和进一步的传递，这相对于传统的异步迭代计算模型（以顶点为最小粒度）具有更强的异步性和并行性。同时，DAIC计算模型的实现不需要任何的加锁操作来实现对算法正确性的控制，也不需要浪费大量的通信来获取所必须的消息，这使得DAIC具有同步迭代计算模型调度低开销和传统异步迭代计算并行性高的优点。

### 2.3.3 DAIC计算模型的优先级迭代计算

优先级的概念在计算机世界和现实世界中都具有重要的应用价值，例如操作系统中多任务的优先级调度，它总是选择当前优先级最高的任务来优先执行，最大程度的利用计算机资源，尽可能的满足交给计算机的任务。再例如人们在日益忙碌的生活中，也需要为多个要做的工作按照重要性设定一个日程表，同样在迭代计算中，优先级计算也具有这十分重要的意义。所谓的优先级迭代计算，就是为每一个基本的计算单元（如图顶点）都设定一个调度计算优先级，在图算法迭代计算的过程中，优先的选择优先级高的计算单元进行调度，以达到加快算法收敛的目的。实际实验中发现，有些顶点迭代计算五六次后，其值就基本达到收敛值；而有些顶点则需要迭代十几次甚至几十次，才能达到收敛值。因此优先级计算研究对于优化迭代计算性能具有重要的意义。

对于同步类型的分布式迭代算法来说，它要求所有的图顶点必须同时进行调度，不存在先后计算的概念，所以在同步框架中无法采用优先级迭代。而对于传统异步模型，只有具有邻接关系的图顶点之间才存在计算的先后逻辑关系，而在DAIC异步模型中，所有的顶点不存在任何先后执行的逻辑关系，这就为优先的计算一些顶点创造了条件。

在优先级迭代计算的实现中存在一个关键性的问题，就是到底优先的计算哪些图顶点才会加快分布式算法的收敛呢？在传统的图迭代计算模型中，对于优先级很难给出一个明确的定义，通常的采用将出/入度高的图顶点定义为高优先级，实践证明这种策略是有效的，但是这种定义跟实际的优先级并非是完全一致的。但是实际上DAIC计算模型在计算的过程中有一个非常重要的特性，就是多有的值在计算的过程中是单调递增的，也就是说，在计算的过程中，所有的值都会从初始值不断的递增，直到收敛值。既然DAIC计算模型采用的累积差值的方式进行计算的，那么在异步计算过程中，优先的去计算那些当前差值大的顶点，就可以加快分布式算法的速度。DAIC计算模型对于优先级的定义的有效性给出了理论上的证明，详见文献[11]，DAIC模型中优先级计算的加入有效的提升了算法了的收敛速度。同时，DAIC计算模型的提出者也给出了优先级迭代的有效性证明。

## 2.4 图数据处理系统Maiter

本文的上一小节对Maiter的理论依据（DAIC模型）进行了详细的介绍，本小节的内容将会对Maiter框架的具体实现继续进行介绍。Maiter是在分布式框架picoolo的基础上实现的，使用高效的程序设计语言C++实现。下面对图数据处理系统Maiter的介绍将从系统实现、优先级调度计算的实现和系统API三个方面展开。

（1）Maiter框架实现

Maiter的系统架构同Hadoop一样，分布式集群采用主从的管理模式，包括一个Master节点和若干个Worker节点，Master节点用于控制和监控Worker节点的工作状态，并进行任务分配。若干个Worker节点是并行工作的。但是和Hadoop文件传输不同的是，Maiter框架使用MPI消息通信进行计算节点间的通信，因此Maiter采用的是消息通信形式，减小了通信量，增加了通信的效率。

Maiter框架的主要任务就是如何使每个Worker节点并行的完成异步累计迭代运算，其实也就是我们上一节讨论的两步运算。所以我们下面重点讨论一下Worker节点的设计。

分布式运算的第一步，就是将待处理的原始图数据平均的分配到每个Worker节点上，Maiter也同样有这样的过程。每个Worker节点会将原始数据的一个子集载入到内存中，等待处理，Worker任务是处理自己所负责的图顶点，在这些顶点上执行异步累积迭代计算，每个图顶点由一个全局唯一的*key*来标识。*key*为*j*的数据元素会被分配给第*h*(*j*)个Worker节点，*h*(*j*)是一个哈希函数，这样采用Hash函数划分数据，会使得数据定位更加容易。为了分配均匀，我们可以让*h*(*j*)为取余运算。除此之外，对数据分片进行预处理也可以使得数据分配的更均匀，避免出现负载不均衡问题，比如我们可以使用轻量聚集算法对图进行预处理，把一些关联的比较紧密的图节点分配到一个Worker上，可以大大的减少通信。

每个Worker节点分配到的图数据会放在内存中的状态表中，状态表中的每一行保存一个图顶点的相关数据。为了获得更快的计算效率，状态表采用哈希表实现，并全部存放在内存中。因为在计算的过程中，会不断的在状态表中对特定的*key*进行定位，这种情况下，哈希表的复杂度是O(1)。状态表的结构如图2.1所示。

每个状态表都包含五个域，*k*，*v*，*v*，*pri*，*data*。其中，*k*域用来保持上文所提到的全局唯一的*key*值，用来唯一标识这个状态表。第二个域用来存储，第三个域用来存放，*pri*域用来存放状态表的优先级，系统会根据该值的不同，按不同的顺序进行更新状态表中的一些域。*data*域则是用来存放分配给它的信息。

接下来的问题就是，如何在每张状态表上完成异步累积迭代计算呢？Maiter框架为每张状态表设计了两个线程，分别是Receive线程和Update线程。它们也相应的完成Receive操作和Update操作。每个Worker节点通过Receive进程不断的接收从其他计算节点上传给它的消息，然后把这些信息累积起来，Update进程则使用该累积值来更新状态表的*v*域，并产生新的消息传送出去。其整个过程如图2.2所示。



图2.2 Maiter执行过程

Fig. 2.2 Executing Processes of Maiter

首先Receive操作将其他Worker节点发来的消息用运算⊕将其累积到*v*域上，当累积的一定程度的时候， Receive进程停止对*v*域进行修改，并将以后发来的值先缓存起来。这时Update进程开始工作，Update操作将*v*中的累积值仍用⊕运算累积到*v*域上，然后在用*v*中的值完成函数*(v)*，然后将该值发给*i*节点之后，将*v*置**0。**

至此，Maiter的理论、设计、实现我们就讨论完了。从上文的Receive操作和Update操作可以明显的发现，Maiter的计算只依赖于当前所接收到的消息，就可以开始运算。并不像同步迭代计算那样，必须等到所有的值到达，下一步的运算才能开始进行。这样就实现了完全的异步迭代计算。

（2）Maiter框架的优先级迭代的实现

在2.3节中提到优先级迭代计算可以加快算法的收敛速度，Maiter系统中也实现了优先级计算的调度策略，同遍历所有顶点的调度策略相比其性能得到了很大的提升。

在Maiter优先级计算的实现中第一个关键问题就是如何定义和表示图顶点的优先级。在对DAIC模型的介绍中已经给出了将“变化量”作为优先级的有效性，在上一节Maiter框架实现的介绍中，Maiter框架为每一个图顶点创建一个对象来保存顶点的相关信息，其中有一个数据成员*pri，*就是用来保存图顶点的优先级。该值是从上一次该顶点被调度执行后到目前为止，累积的变化量，累积的值越大该顶点就具有越高的优先级。在DAIC模型迭代计算过程中，顶点的value值是通过累积接收到的消息不断递增的，那么对累积了变化量值大的顶点进行调度计算，就会使顶点值递增的越来，也就能更快的达到收敛值，因此这种优先级的设计和实现是合理的。

在Maiter优先级计算的实现中另一个关键性问题，就是如何使用尽量小的开销来获取优先级调度队列。Maiter优先级的实现并不是采用为图中所有的顶点维护一个优先级队列的方式，而是采用了另一种的方式，在调度器需要的时候再去获取当前的优先级高的顶点。如果在调度计算中以顶点为粒度进行调度，那么就需要频繁的遍历全局获取一个优先级高的顶点进行计算，这么大调度开销是无法承受的。为了避免了这种频繁调度的开销，Maiter的优先级调度采用批处理的形式，一次对一批优先级高的顶点进行计算。

在对优先级队列的获取的实现中，如果采用精确的方式找到优先级最高的那部分顶点，计算复杂度仍然很高。Maiter采用了一种很巧妙的设计，利用采样的方法来近似获取的当前的优先级调度队列。该方法的具体实现过程是：首先在计算节点的状态表中随机采取一定量的样本顶点，对这些样本顶点进行排序，然后根据用户配置的参数，选取多少优先级高的进行调，例如10%，那么就将采样点中的排在10%位置的图顶点的*pri*值作为临界值，Maiter框架在调度计算时，将所有*pri*域大于该临界值的图顶点调度进行计算，这样Maiter就实现了对优先级计算的支持。

优先级计算策略的使用是有助于提升系统的性能，加快算法的收敛速度。但是如果在优先级调度实现上付出过多的开销，很多时候可能会得不偿失，Maiter在实现中利用采样方式来近似的获取优先级队列，极大的降低了调度的开销，同时也保证了调度的精度。Maiter对优先级调度计算低开销的实现，为本论文后面多副本顶点计算模型实现优先级计算奠定了条件。

（3）Maiter框架API介绍

目前，最被大家所熟知的和应用最广的大数据处理系统Hadoop、Spark都是对MapReduce编程模型的实现，算法在Hadoop上实现，用户只需实现map和reduce接口。而图数据处理系统普通采用以点为中心的编程模型，例如算法在Pregel、Giraph上实现，用户只需实现为顶点定义的compute接口。Maiter作为一个图处理系统也自然采用了这一编程模型，但不同的是Maiter框架并没有顶点的操纵函数接口，而是由一些其他的API来完成对顶点的操作，并且这些API在Maiter所实现的DAIC计算模型中都有其固定的意义。并且部分API只是需要设定某些值就可以完成，不需要复杂的操作。因此Maiter编程模型编程更加便利。下面介绍一下Maiter框架提供的API。

Maiter框架的API分为两部分，一部分API用于完成Receive和Update操作，另一部分用于控制计算何时结束。第一部分如下所示：

* + *void read\_data(string& line, K\* k, D\* data) ;*
  + *void init\_c(const K& k, V\* delta)*
  + *const V& default\_v() const*
  + *void accumulate(V\* a, const V& b)*
  + *void priority(V\* pri, const V& value, const V& delta) ;*
  + *void g\_func(const V& delta, const D& data, vector<pair<K, V> >\* output)*

其中*read\_data*()用于每个Worker节点读取图数据分片的信息，并用这些信息来初始化自己状态表中的*data*域和*key*域；inti\_c()用于初始化状态表中的*v*域；*default\_v*()用于初始化状态表的*v*域，同时用来设置Maiter运算模型中的满足等同率*o*。*accumulate*()用于设置Maiter运算模型中的⊕运算；*priority*()用来设置Worker状态表中每一项的计算优先级，在Maiter框架运行时，会选择优先级比较高的状态表中的项优先进行计算，这有助于快速收敛；*g\_func*()则是用*v*域中的信息来运算的到，并将算出的其他节点的信息发送给相应的节点。

控制程序终止的API如下所示：

* *double estimate\_prog(LoaclTableTterator<K,V> \*table\_itr)*
* *bool terminate(list<double> local\_report)*

由于Maiter特殊的异步迭代形式，实际上Maiter框架是没有迭代步的概念的，在一次运算中用到的数据可能是*k*次迭代的结果，也可能是*k*+1此迭代的结果，因此它不能像Hadoop框架一样用迭代的次数来控制程序的终止。

Maiter框架采用的方式是：在固定的时间内所有节点的状态表中*v*域的总和的变化情况来决定程序什么时候结束。其中*estimate\_prog*()是用来设置时间间隔和变化的大小的，*terminate*()用来决定是否停下来。这两个API并不是必须的，在未指定的情况下，Maiter框架会调用默认的API。

## 2.5 本章小结

本章，首先详细的介绍了DAIC计算模型的原理、应用条件及其推导，DAIC模型是本次论文中提出的多副本计算模型和图分区方法的理论基础，只有理解DAIC模型才能更好的理解本文的研究工作。本章还对优先级调度的DAIC模型的有效性和实现进行了介绍，这将有助于读者理解本文研究内容中多副本计算模型是如何提升系统的性能。API部分的介绍，将会给读者一个对DAIC模型和Maiter框架更加清晰的认识。本章是后续章节研究内容的基础，所以请读者仔细阅读和理解本章内容。

# 第3章 MR-DAIC计算模型及实现

在分布式迭代计算中，新的计算中间结果能够越早迭代在下一轮的迭代计算中被应用和传递，越有利于算法尽快的收敛。分布式计算中计算单元之间的并行性也将直接影响资源的利用率，同时它还跟优先级计算息息相关。本文中提出了多副本版本的MR-DACI模型来改进DAIC模型在以上方面的不足，本章内容将从MR-DAIC模型、MR-DAIC模型的正确性证明和MR-DAIC模型的系统实现，这三个方面进行介绍。

## MR-DAIC模型

### 3.1.1 多副本顶点计算模型

在传统的图计算模型中，数据被抽象成一个有向图，是图中所有顶点集合，是图中所有边集合，每个顶点v都运行一个vertex-program，以此来完成对整个图的计算。而在多副本顶点的图计算模型中，每一个顶点都以多副本顶点的形式存在，数据可以被抽象的表示为G={V,E}，其中V={={},={},…={}}，，副本顶点并行的完成原顶点所要完成的任务，最终完成对整个图的计算。

在多副本顶点的图结构中，每个数据分区都构成了一个完整的子图，网络通信发生在顶点内的副本之间。在这种图结构中，所有的顶点都被称为副本，为了实现副本之间高效、正确的信息交换，会选择其中一个副本作为master顶点，其余副本都被称为mirror顶点。对于如何从若干个副本顶点中选择master顶点的问题，在PowerGraph、PowerLyra等系统中都是采用随机选择的策略，而在MR-DAIC模型中为了提高消息被应用的速率采用了一种启发式的策略，具体实现将在系统实现一节中进行介绍。

为了更好的理解单顶点的图计算模型和多副本顶点的图计算模型，下面通过一个例子来形象认识这两种模型的异同。例如，例如，图3.1中左边是一个简单的图，将该图分配到两个机器上进行计算。其中右上边的图3.1(a)DAIC模型对应的是单顶点的图计算模型，在该模型中顶点唯一存在，在迭代计算过程中通信发生在顶点A、D之间和顶点A、E之间，通信量为4。右下边的图3.1(b)MR-DAIC对应的是多副本顶点的图计算模型，在该模型中图顶点会以多个副本顶点的形式存在，masterA和mirrorA共同完成图顶点A的计算任务，在迭代计算中通信发生在

图3.1 DAIC模型和MR-DAIC模型

M=2

M=4

A

sample graph

B

C

A

D

E

（a）DAIC模型

B

D

C

E

（b）MR-DAIC模型

B

D

A

A

C

E

master

mirror

Fig. 3.1 Models of DAIC and MR-DAIC

副本顶点masterA和mirrorA之间，通信量为2。同时，在表3.1中总结了这两中模型之间的几处重要不同点。

表3.1 单顶点的图计算模型和多副本顶点的图计算模型特点对照表

|  |  |  |
| --- | --- | --- |
|  | 单顶点计算模型 | 多副本顶点计算模型 |
| 顶点： | 唯一存在 | 以多个副本顶点形式在多个机器中存在 |
| 边： | 两份分别存在于两个端点所在机器 | 唯一存在 |
| 计算粒度： | 图顶点 | 副本顶点 |
| 通信： | 发生在邻接顶点之间，通信量等于边被切割的数量 | 发生在顶点的多个副本顶点之间，通信量等于顶点被切割的数量 |

### 3.1.2 MR-DAIC模型

MR-DAIC（Multi-Replicas-DAIC）模型是在DAIC模型的理论基础上，结合多副本顶点的图计算模型的思想而成一个高效的图计算模型。从计算效率的角度来看，相对于原有的DAIC模型，图顶点计算具有很好的并行性，细化了优先级计算调度的粒度，提升了优先级计算的性能。从消息流通性的角度来看，MR-DAIC模型结合有效的Vertex-Cut图划分方法，将会极大的减少通信量和提高消息被应用的速率。在处理图顶点度扭曲分布的图数据时，MR-DAIC模型相对于原有的DAIC模型提升的性能会更加明显。

（1）计算并行性

从计算单元来看，相对于DAIC模型中图顶点的基本计算单元，MR-DAIC模型将基本的计算单元细化到了图顶点的副本顶点，这使得优先级计算策略在MR-DAIC模型中更加的高效。

各个副本顶点，独立的执行Receive操作和Update操作，完成对消息的接收、消息的应用和新产生消息的传递。例如在图3.1（b）中，左边分区的副本顶点A\_master接收来自于顶点B和顶点C的消息，并对接收到的消息立即应用，产生的新消息又可以被立即的传递给顶点B、C。右边分区的副本顶点A\_mirror同A\_master一样，同样执行Vertex-Program完成对本地消息的接收、应用、传递。这两个副本顶点执行期间不需要任何的同步，只需要进行异步的消息交换来保证算法的正确性。这使得副本顶点成为MR-DAIC模型中的最小计算单元。

在实际应用中产生的自然图中图顶点的度的分布，一般不是随机的、或者均衡的，而是往往服从扭曲分布（如幂率分布），导致各个顶点计算负载存在着很大的差异，这使得顶点计算的并行性会很差，在进行优先级计算时并不能快速的完成对度高的顶点的计算，降低了优先级计算效果。而MR-DAIC模型中，度高的顶点通常会被分割成若干个副本顶点，分布在不同机器上的副本顶点就可以并行的执行度高的图顶点计算任务，这就使得在进行优先级计算时，对度高的顶点的计算可以被快速完成。例如，在图3.1（b）中，图中度最高的顶点A被优先级调度器选择进行计算，分布在两个机器上的两个A的副本顶点会并行的执行，从而提高了算法的收敛速度。

（2）通信量

从通信量角度来看，在DAIC模型中通信量产生于对边的切割，而在MR-DAIC模型中通信量产生于对顶点的切割，在PowerGraph中给出了理论上的证明，通过切割顶点的划分方法相对于切割边的划分方法，具有很低的通信量。

在顶点度扭曲分布的自然图中，相对于切割边的方式，通过切割顶点的方式更加容易实现对整个图的划分。例如在图3.1中，图3.1(b)表示的是通过切割顶点A来实现图的划分，进行多副本顶点计算的模型，可以看到MR-DAIC模型相对于DAIC模型具有更低的通信量。

（3）本地消息流通量

由于MR-DAIC模型中每个数据分区都是一个完整的子图，邻接顶点之间通过边来传递的消息，其目的顶点在本地机器必然存在一个副本顶点，本文中称这类消息为“本地消息”。而副本顶点之间消息的传递是需要网络进行传输，本文中称这类消息为“远程消息”，该类消息的数量就对应着上文中的通信量。在2.3.1小节对DAIC模型的介绍中指出，DAIC模型实现了以边为粒度的异步计算，这使得在MR-DAIC模型中副本顶点只要收到消息，就可以对接收到的消息立即应用。本地消息消息没有经过网路的传输，就可以被“部分”的应用，很大程度上提高了本地消息被应用的速率，加快了算法的收敛速度。

为了让读者更好的理解本地消息，展示本地消息在MR-DAIC模型是如何被快速的应用，下面将以图3.1为例运行pagerank算法，阻尼系数d=0.8，分别采用DAIC模型和MR-DAIC模型来展示在不进行网络通信的情况下，本地消息的流通量。

表3.2 DAIC模型运行pagerank算法

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | T0 | | T1 | | T2 | | T3 | | T4 | | T5 | |
|  | v |  | v |  | v |  | v |  | v |  | v |  |
| 机器1 | | | | | | | | | | | | |
| A | 0 | 0.2 | **0.2** | **0** | 0.2 | 0 | 0.2 | 0.192 | **0.584** | **0** | 0.584 | 0 |
| B | 0 | 0.2 | 0 | 0.2 | **0.24** | **0** | 0.24 | 0 | 0.24 | 0 | **0.3168** | **0** |
| D | 0 | 0.2 | 0 | 0.2 | 0 | 0.24 | **0.24** | **0** | 0.24 | 0 | 0.24 | 0.0768 |
| 机器2 | | | | | | | | | | | | |
| C | 0 | 0.2 | **0.2** | **0** | 0.2 | 0 | **0.2** | **0** | 0.2 | 0 | **0.2** | **0** |
| E | 0 | 0.2 | 0 | 0.2 | **0.2** | **0** | 0.2 | 0 | **0.2** | **0** | 0.2 | 0 |
| Sum | 0 |  | 0.4 |  | 0.84 |  | 1.08 |  | 1.464 |  | 1.5408 |  |

表3.3 MR-DAIC模型运行pagerank算法

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | T0 | | T1 | | T2 | | T3 | | T4 | | T5 | |
|  | v |  | v |  | v |  | v |  | v |  | v |  |
| 机器1 | | | | | | | | | | | | |
| A | 0 | 0.2 | **0.2** | **0** | 0.2 | 0 | 0.2 | 0.192 | **0.584** | **0** | 0.584 | 0 |
| B | 0 | 0.2 | 0 | 0.2 | **0.24** | **0** | 0.24 | 0 | 0.24 | 0 | **0.3168** | **0** |
| D | 0 | 0.2 | 0 | 0.2 | 0 | 0.24 | **0.24** | **0** | 0.24 | 0 | 0.24 | 0.0768 |
| 机器2 | | | | | | | | | | | | |
| A | 0 | 0.2 | **0** | **0** | 0.2 | 0 | 0 | 0.192 | **0** | **0** | 0 | 0 |
| C | 0 | 0.2 | 0 | 0.2 | **0.24** | **0** | 0.24 | 0 | 0.24 | 0 | **0.3168** | **0** |
| E | 0 | 0.2 | 0 | 0.2 | 0 | 0.24 | **0.24** | **0** | 0.24 | 0 | 0.24 | 0.0768 |
| Sum | 0 |  | 0.2 |  | 0.68 |  | 1.08 |  | 1.464 |  | 1.6176 |  |

表3.2表明DAIC模型运行pagerank算法在T6时刻的总收敛值为1.5408，表3.3表明MR-DAIC模型运行pagerank算法在T6时刻的总收敛值为1.6176。MR-DAIC计算模型中计算单元是6个而DAIC中是5个，所以在一开始的时候收敛是慢于DAIC模型，这也是MR-DAIC的一个缺陷，对这个缺陷的解决将通过本文第四章图划分方法来保证副本顶点的数量不会增加太多。从T5时刻后，本地消息流通性好的MR-DAIC模型的收敛速度明显快于DAIC模型。

综上来看，MR-DAIC模型相对于DAIC模型，提高了优先级计算性能、减少了通信量、提高了本地消息流通量，特别是在处理真实世界中的自然图时MR-DAIC模型表现的更加突出。

MR-DAIC模型在计算的本质上还是利用DAIC模型差值累积的思想，但是在计算形式上发生了变化，MR-DAIC模型的更新函数如公式3.1所示。

上式中是顶点j的在第k轮迭代计算后的结果，是顶点第k-1轮迭代计算后的结果，是顶点j的副本顶点所在的机器集合，是顶点j在第r机器上副本顶点在第k轮的变化量，的值通过上一轮的计算结果，累积上顶点j的各个副本顶点的变化量进行更新。是顶点j在机器r上邻接顶点队列，是机器r上的副本顶点i传递给副本顶点j的消息，顶点j在机器r上的副本顶点的变化量是通过累积本机器上邻接副本顶点的变化量得到。

同DAIC模型一样，MR-DAIC模型对迭代计算也是抽象为Receive和Update两个操作，用户面向副本顶点编程实现两个操作，是一个以副本顶点为中心的图计算模型。下面是以顶点j在机器p上的副本顶点为例，描述MR-DAIC模型大致的计算过程，如下式3.2和3.3所示。

在机器p上，每当副本顶点j接收到其他副本顶点发送来的消息，就会将其累积到中。在迭代计算中，副本顶点首先更新自身的值，将变化量直接累积到本地的邻近副本顶点，同时也将变化量发送到其他的副本顶点完成数据的同步。最后设置为默认值。图中的每个副本顶点同上述描述一样，都执行vertex-program，完成receive操作和update操作，实现对整个的图的计算。对于MR-DAIC模型的更新函数的推导、正确性证明将会在3.2小节中介绍。

### 3.1.3 MR-DAIC模型与GAS多副本顶点模型的比较

在对大规模图数据的不断研究中发现，现实世界中自然图的顶点的度分布是扭曲，这就造成了在迭代计算中图顶点之间的并行性差，因此卡内基梅隆大学的课题组提出了一个面向图顶点度服从幂率分布数据的图处理系统PowerGraph，之后又有一些面向自然图特性的图处理系统如HDRF、PowerLyra等，这些图处理系统都是采用了GAS的多副本顶点计算模型。所以，本小节将对在多副本顶点模型中主流的GAS模型进行介绍，然后分析本文中提出的MR-DAIC模型相对于GAS模型的优势。

在GAS计算模型将计算分为三个阶段，即Gather、Apply和Scatter。Gather阶段进行消息的收集，分布在多个机器上的副本顶点并行执行；Apply阶段是对收集到的消息进行应用，根据收到的消息更新自身值，并产生新的消息；Scatter阶段就是将Apply阶段产生的新消息发送到目的顶点。其中，在执行Apply操作时需要进行一次数据的同步操作，master顶点收集完各个副本顶点所收集到的本地消息之后，在进行对消息的应用，从而保证算法的正确性。从上述模型中，可以看出GAS模型中的多副本顶点主要是实现对消息的并行收集和发送，从而降低迭代计算过程中的通信量，这也是GAS模型被提出的初衷。

在MR-DAIC模型中，则实现了副本顶点对消息收集、消息应用、消息发送三个操作的并行执行，这相对于GAS模型图顶点计算具有更好的并行性。在迭代计算中，进行有效的调度计算和尽量使用最新的消息是加快算法收敛速度的两个最有效的技术，接下来将从这两个方面来分析MR-DAIC模型相对于GAS模型的优势。

从计算的并行性来看，图顶点的计算负载与其关联边的数量呈线性关系，而在自然图中顶点度的扭曲分布就造成了顶点之间很大负载差异。下图3.2是对图计算中计算单位负载的一个抽象，顶点的大小代表着该顶点负载的大小。在Pregel中，图顶点直接来源于图数据中的顶点，各个顶点负载的差异很大，同时Pregel是按照随机分配顶点（Edege-Cut）的方式进行图划分，这就造成了Pregel框架不管是在机器间的并行性和顶点之间的并行性都很差。在PowerGraph中，度高的顶点被分割成若干个副本顶点，这些副本顶点并行的执行消息的收集和发送的任务，但是消息应用时需要一次副本顶点的同步，然后消息在master顶点被应用，这种模式相对于Pregel系统度的顶点被并行的执行，提高了消息的并行性，但是各个副本顶点之间又不是完全独立存在，它们之间的一次同步操作会在一定程度上降低顶点计算的并行性。在MR-Maiter（MR-DAIC模型的系统实现）中，度高的顶点被分割成若干个独立的副本顶点，所有的副本顶点具有几乎差不多的负载，它们之间无任何的同步操作，这使得MR-Maiter具有非常好的并行性。同时，在优先级计算中，顶点的并行将直接影响优先级计算对算法收敛速度提升的效果。

Pregel

PowerGraph

MR-Maiter

图3.2 多种图计算模型的负载分布图

Fig. 3.2

从消息的流通性来看，相对于GAS模型具有更好的并行的MR-DAIC模型，可以实现对消息更快的应用。下图3.3是对GAS模型和MR-DAIC模型的数据流分析，图中以顶点C向顶点A发送的消息m为例进行描述。在（a）中副本顶点A\_mirror只能执行消息的收集（Gather）和消息的发送（Scatter），在当收到消息m时，其并不能将消息进行应用（Apply）而是将消息发送到A\_master（存在于另一个机器），A\_master对消息m应用后，再将新消息g(m)和更新后的结果发送给A\_mirror，之后A\_mirror再将g(m)传递给顶点C和顶点E。在这个过程中顶点E和顶点C要被消息m影响到就必须要经过两次网络传输，而在分布式计算环境网络往往是瓶颈，这就很大程度上降低了图处理框架的性能。而在MR-DAIC模型中，每一个副本顶点都是一个独立的计算单元，可以并行的执行G、A、S三个阶段的操作，当收到消息m后，就能立即对其应用，并把新消息g(m)传递给顶点C和顶点E。另一方面从通信量来看，在GAS模型中完成副本顶点之间的信息交换需要5次通信，而在MR-DAIC模型中则只需要两次通信，因此不管是本地消息还是远程消息的流通性，MR-DAIC模型都要优于GAS模型。

g(m)

m

g(m)

g(m)

m

m

g(m)

g(m)

m=2

Gather

Apply

Scatter

Gather

Apply

Scatter

（b）MR-DAIC模型

B

D

A

A

C

E

sample graph

B

C

A

D

E

mirror

A

A

master

Gather

Scatter

Gather

Apply

Scatter

m=5

（a）GAS模型

B

D

A

A

C

E

图3.3 GAS模型和MR-DAIC模型数据流图

Fig. 3.3

0.6

## 3.2 MR-DAIC模型正确性证明

MR-DAIC模型是一个非常灵活的图数据计算模型，具有着计算并行性好、优先级计算调度精度高，通信量低、本地消息流通性好等突出优点。对计算模型的高效性研究一直以来都是大家重点关注的领域，但计算模型正确性是一个模型具有存在意义的前提条件，因此本节内容将对本文中提出的MR-DAIC模型的正确性给予证明。

（1）前提条件

MR-DAIC模型同DAIC模型的思想是一样的，都是对“变化量”累积的迭代计算，区别在于计算形式的不同。所以，MR-DAIC模型应用的前提条件同样采用DAIC模型的四个前提条件。

条件1：累积性表示

迭代计算中的更新函数一般形式：

可以被转化成，如公式2.3所示：

条件2：传递函数g()的满足分配率：

条件3：运算符具有交换律和结合律：

(3.6）

条件4：满足等同律，即存在抽象值满足：

同DAIC模型一样，上述四个前提条件对于MR-DAIC模型也是充分条件而非充要条件。

（2）MR-DAIC模型推导

“变化量”是MR-DAIC模型中重要概念，首先给出对变化量的以形式化定义，如公式3.8所示。

是顶点j的副本顶点所在的机器集合，是顶点j在第r机器上副本顶点在第k轮的变化量，上式表示顶点j在第k轮的结果等于其在第k-1轮中的结果累积上各个副本顶点在第k轮的变化量。

前提条件1保证了算法更新函数可以被表示成：

将变化量的定义公式3.8带入上式得：

根据前提条件2，传递函数的分配率，上式转化为：

p是数据分区个数，表示顶点1在数据分区1上的副本顶点。上式中每一行表示一个图顶点对图顶点j的影响，图数据中一共n个顶点。

根据前提条件3，运算符的结合律和交换律，上式可转化为：

上式中第一行表示图顶点在第k-2轮的结果对顶点j的影响，从第二行开始到最后，每一行都代表着一个分区中所有副本顶点在第k-1轮的变化量对顶点j的影响，那么根据前提条件1，上式可以被进一步转化为：

对公式3.9进行进一步的推导，再根据四个前提条件，就能得到完全异步性执行的计算形式。我们可以观察到计算中的变化量并不需要必须是两次迭代的差值，而是可以是任意的粒度，不存在迭代轮数的概念，这就使得MR-DAIC模型成为一个完全异步化的迭代计算模型。

（3）MR-DAIC模型的收敛性及正确性证明

在同步计算过程中，消息是一跳一跳的进行传递。在异步计算过程中，各个顶点执行的速度是不一样的，现在我们将异步计算过程中消息被传递最多跳的步数k作为异步计算中“轮数”。

在同步系统中，在进行到第k轮时，每个顶点会收到所有距离它小于k步距离顶点所发送的消息，而在异步执行中，只会收到距离它小于k步距离顶点所发送的部分消息。让k1趋近与无穷大，那么同步计算中的值会收敛到一个值。同时，异步计算中其值会小于异步计算中的值，所以其值也会收敛。现在让k2=k1+，那么总会出现异步中的收集的消息会多余同步系统k1轮后所收到的消息。在异步计算中，k1步会收敛到一个值，k2也会收敛到一个值，这两个值是几乎相等的，且这两个值一个小于同步计算k1之后的值，另一个大于。因此根据夹逼准则，得到新的计算模型计算结果收敛，且正确。

## 3.3 计算引擎的实现

MR-DAIC模型的实现是在开源框架Maiter（DAIC模型的实现）的基础上，改写了数据结构和计算引擎等模块，实现了一个支持多副本顶点计算的图数据处理框架MR-Maiter。本节将会对MR-Maiter框架的数据结构、副本顶点之间消息的同步机制和针对不同类型副本顶点的通信模型三个方面的内容进行介绍。

### 3.3.1 存储结构设计

MR-Maiter框架中计算和存储的基本单位都是副本顶点，其数据结构如下图3.4所示。一个副本顶点元数据中一共包括10个属性，Key是顶点的键值，一般用图顶点的id表示；value是要图顶点的属性值；是用来累积接收到的消息中属性值变化量；用于累积某一mirror顶点发送给master顶点属性值的变化量，此数据项设计的具体作用将在下一小节中介绍；Master数据项记录的是该顶点的master副本顶点所在的机器编号；Mirrors数据项记录的是该顶点的mirror顶点所在机器编号；Length记录的是该顶点的出度值；sub-neis存储顶点的部分邻接顶点；Pir数据项存储该副本顶点在优先级计算中的优先权；is\_use数据项记录该存储单元是否被占用。

Key

Value

Mirrors

Sub-neiss

Length

use

Pir

Master

图3.4 顶点对象存储结构

Fig.3.4 Storage structure of vertex

MR-Maiter框架对数据的存储结构采用内存key-value形式，每一个副本顶点作为一条记录，副本顶点元数据中的数据项key作为hash表中的键值，其余数据项作为hash表中的value值，每个机器worker上的数据存储和数据流如下图3.5所示。



图3.5 worker数据结构图

Fig.3.5 data structure of worker

### 3.3.2 副本顶点数据同步的实现

本文前边已经介绍过，在MR-DAIC模型中副本顶点独立的执行vertex-program，在适当的时间进行副本顶点之间的信息交换，本小节将会副本顶点之间的数据交换机制进行介绍。

在MR-DAIC模型中，副本顶点分为master顶点和mirror顶点，master顶点收集各个mirror顶点的变化量，之后在将整个顶点的变化量同步到各个副本顶点。例如在图3.6中，图顶点A被切分成三个副本顶点A1、A2、A3， mirror顶点A2、A3向master顶点A1发送消息m1和m2，A1收集各个mirror顶点的消息后，再将其发送给mirror顶点，向A2发送消息{m2,m3}，向A3发送消息{m1,m2}。另外在上一节数据存储结构对副本顶点元数据结构介绍中提到的数据项就是在这里起作用，A1在接收到A2的消息m2后，将其累积到队列中，当其向A2发送消息时，将总变化量{m1,m2,m3}抵消掉队列中m2后在发送给A2，以此来保证数据流的正确性。



图3.6 副本顶点数据流图

Fig.3.6

在迭代计算过程中，Vertex-Program运行在每个副本顶点上，完成对整个图的计算。下面本文将会对MR-DAIC模型中的master顶点和mirror顶点的执行过程用伪代码的形式进行说明，如Receive操作如算法3.1所示，Update操作如算法3.2所示。

|  |
| --- |
| **算法3.1** MR-Maiter的Vertex-Program  输入：图*G*  输出：求解的属性集合 |
| Receive： |
| 1. while receiveing do |
|  |
| 1. if ismaster then |
|  |
| 1. done |
| Update： |
| 1. for any h in sub\_neis(j) do |
|  |
| 1. if is mirror then |
| 1. send to master of |
| 1. done |
| 1. if is master then |
|  |
| 1. for any p in mirrors do |
| 1. send () to on machinep |
| 1. done |
|  |
|  |
| 1. done |

### 3.3.3 副本顶点通信模型的优化

=0,

为了减少迭代计算过程中的通信量，本文中将mirror顶点细分为compue mirror和communicate mirror，本文针对这两种mirror顶点的特性设计了两种通信模型，如下图3.7所示。

图3.7 两种副本顶点通信模型

Fig.3.7

（1）master与cp\_mirror通信模型、

compue mirror是既有入边又有出边的副本顶点，所以该类mirror顶点会执行消息的收集、消息的应用和消息的发散三个操作。该顶点向master顶点发送消息后，用于master顶点的计算，同时master也需要将其接收到的消息发送给compue mirror顶点，用于compue mirror顶点对消息的应用和发散。因此在迭代计算过程中master顶点与compue mirror顶点之间的进行两次通信。

（2）master与ct\_mirror通信模型

communicate mirror是指只有入边没有出边的mirror顶点，因此只执行消息接收的操作，所以在迭代计算过程中，其只需要向master顶点发送消息即可，只需要进行一次通信。

通过细化这两种通信模型，减少了迭代计算中的通信量，提高了系统的性能。

### 3.3.4 MR-Maiter API

MR-Maiter框架的API同Maiter框架类似，也分为两部分，一部分API用于完成Receive和Update操作，另一部分用于控制计算何时结束。第一部分如下所示：

* + *void read\_data(string& line, K& k, vector<K>& neibs, vector<pair<int > &mirrors, int& length) ;*
  + *void init(const K& k, V& delta, V& value, vector<V>& offset);*
  + *V default\_v();*
  + *void accumulate(V\* a, const V& b);*
  + *void priority(V& pri, const V& value, const V& delta) ;*
  + *void g\_func**(const V& delta, const vector<K>& neibs, vector<pair<K, V> >& output，int length);*
  + *void* *counteract (V& a, const V& b);*

*read\_data*()用于每个Worker节点读取图数据分片的信息，并用这些信息来初始化自己状态表中的*key*域、sub\_*neis*域、length域和mirrors域；inti()用于初始化状态表中的*value*域、delta域和offset域；*default\_v*()用于设置默认值。*accumulate*()用于设置Maiter运算模型中的⊕运算；*priority*()用来设置Worker状态表中每一项的计算优先级，在Maiter框架运行时，会选择优先级比较高的状态表中的项优先进行计算，这有助于快速收敛；*g\_func*()则是用*value*域中的信息来运算的到，并将算出的其他节点的信息发送给相应的节点；*counteract()*则是定义抵消操作运算符。

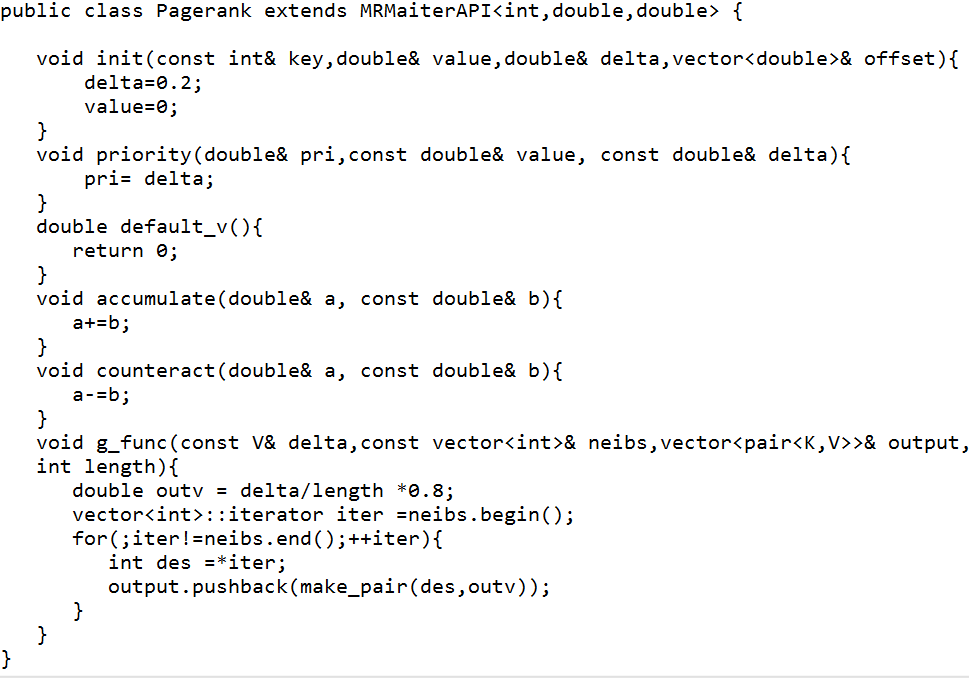


图3.8 PageRank在MR-Maiter中实现

Fig.3.8

控制程序终止的API相对于Maiter框架并没有改变，如果未指定系统会使用默认的函数实现。详细的说明请参照2.4小节。

下面是PageRank算法在MR-Maiter上的实现，其接口实现如图3.8所示。

## 3.4 本章小结

本章对MR-DAIC模型进行了详细的介绍，分析了该模型相对于DAIC模型和GAS模型的优势，然后对MR-DAIC的正确性给予了证明，最后介绍基于开源框架Maiter，对MR-DAIC模型的系统实现MR-Maiter。

# 第4章 面向异步图计算模型的图划分方法PAGraph

在第三章中介绍的MR-DAIC是一种基于多副本顶点的计算模型，这就需要对Maiter系统现有的图分区算法（随机分配顶点的）进行重新实现。同时，最近对现实世界中图数据的研究中发现，图顶点度的分布是扭曲的，而分布是迭代计算中的负载均衡是跟边的数量相关的，这就导致了现有的以点为中心的图分区（Edge-Cut）划分的分区之间负载的不均衡和高通信量。

本章内容首先对目前具有代表性的图分区算法进行介绍，并分析它们存在的不足、总结图划分中要优化的关键目标，然后对异步迭代计算模型和同步迭代计算模型在消息传递方面的差异进行介绍，并提出了异步模型中消息流通性的概念，实现了一个面向异步迭代计算的图分区算法PAGraph。为进一步改进和优化现有的图分区算法，介绍本文提出的点边混合切割（Edge/Vertex-Cut）的图分区算法EV-PAGraph。

## 4.1 自然图特性及图分区方法介绍

### 4.1.1 自然图特性介绍

自然图即现实世界中自然存在或者由人类活动而产生的图，如web网络图、社交网络图、文献引用关系图等，这些图往往服从扭曲的分布，例如幂率分布（power-law）分布。在服从幂率分布的自然图有着如下的特征，一大部分顶点（低度顶点）往往关联着少部分的邻居顶点，另外一小部分的顶点（高度顶点）却关联着大部分的邻居顶点。在幂率分布中，给定一个正常数，一个度为d的顶点出现的概率表达式为：

上式中正常数称为自然图的扭曲因子，控制着幂率分布的扭曲程度。当较大是，意味着顶点自然图有较小的密度（图中边与顶点的数量比例），有着较小的扭曲度（少数顶点关联边占边数量的比例），随着减小，自然图的密度和扭曲度都会增大。大部分自然图的扭曲因子。为更好的描述自然图幂率分布的特性，对twitter数据集进行了分析，在图4.1中，左边的图展示了不同度下顶点个数的分布，右边的图展示了不同度的顶点拥有边个数的分布。

X：度

Y：边数量

Name：边分布

X：度

Y：顶点数量

Name：顶点分布

图4.1 twitter中顶点出度分布

Fig.4.1 The out-degree distributions of twitter

从上图可以看出，随着顶点度的增加，顶点的个数在减少，而顶点关联的边的个数在增加，1%具有最高度的顶点大约关联着50%的边。自然图这种扭曲分布的特性对现有的图处理系统带来严重的挑战。

（1）计算节点负载不均衡

在GAS计算模型中，在gather和scatter阶段中，存储量、通信量、计算量都跟顶点的度成线性关系；同样在MR-DAIC模型中，Receive操作和Update操作的负载都跟边的数量呈线性关系，因此在幂率分布的自然图中，顶点运行时间会有很大的差别。那么从机器节点的高度来看负载情况，节点的负载就跟分配到该节点上边的数量成线性关系。但在，GraphLab和Maiter中均采用随机哈希顶点的方式来进行数据划分，只能保证各个机器节点上顶点的数量大致相等，但是无法保证节点上边的数量大致相等，因此会出现集群中各个机器节点负载不均衡的情况。

（2）顶点间并行性差

现有的图数据处理系统，都是采用以顶点为中心的计算方式，多个顶点可以并行的执行vertex-program，但是均不支持对单个顶点的并行执行。在服从幂率分布的自然图中的高度顶点计算负载明显要高于低度顶点，而在优先级调度中高度顶点往往具有着很高的优先权，这将会降低系统进行优先级计算的性能。

（3）通信量大且不均衡

图计算的一般形式是，顶点之间通过相连接的边相互作用，交替影响最终使算法收敛。在分布式图迭代计算中，顶点向与其有边连接的顶点发送消息，所以说边的数量就对应着该机器节点的通信量。在幂率分布的自然图中，由于各个机器节点上边数量的不对称性；同时通过Edge-Cut的方式实现图的划分，会造成大量的边被切割，这就导致机器节点之间通信量的不对称性和通信大等问题。

（4）存储量不均衡

在图数据中，边的数量通常是顶点数量的十倍甚至百倍以上，因此只有当边的数量在各个机器节点上分配均衡时，系统存储才是均衡。而自然图中度的扭曲分布，使得现有的图数据处理系统中边的数量分配是很不均衡的。现有的主流图处理系统往往都是采用内存作为存储介质，因此如果一个机器节点上出现较多度很高的顶点，就会导致分配到该节点上的数据量超过其单机内存的容量。

由于图幂率分布特性与框架性能之间的紧密相关性，而现有主流的图处理系统的在设计时并没有考虑这种特性，因此在对自然图进行处理时，其性能往往并不是高效。

### 4.1.2 点切法在自然图划分中的应用研究

自然图的一个重要特性就是高度顶点关联着大量的边，这个

1. 随机切割
2. 贪婪切割

在分布式环境下，对海量的图数据进行处理，首先面临的问题就是如何将数据比较均匀地分配到各个机器节上。对于非图数据来说，数据中的数据记录相互独立，这个问题的解决就非常的直观，只需将数据均衡的分配到各个机器上就可以了，数据切分算法不用考虑其他的约束条件。对于图数据来说，数据记录之间往往有着很强的耦合性。因此，对于图数据分片的问题，不仅仅要考虑机器之间任务量分配是否均衡的问题，还要考虑机器节点之间通信量的问题。在对图数据分片时，判定数据分片是否合理主要考虑两个因素：机器负载和网络通信总量。遗憾的是到目前为止，图数据分片问题尚未得到很好的解决。

从切分图数据的不同角度来看，图数据分片算法主要分为两类：切边法（Edge-Cut）和点切法（Vertex-Cut）[16]。

**（1）切边法：**切边法是图数据分片中最常见的一种思路，它通过切割图中的边将一个大的图分割成若干个子图，每个顶点仅出现在一个子图中。在分割过程中，要尽量将每个子图中分配到数量大致相等的顶点，也要尽量减少切割的边数，即在分割过程中要考虑负载和通信量两个因素。对于切边法的优化目标定义如下式（1）所示：

参数说明：表示边e是否被切割（切割为1，为切割为0），表示机器A上分配顶点的数量，表示不平衡因子（）。

**（2）切点法：**通过切割图中的顶点将一个大的图分割成若干个子图，每个边仅出现在一个子图中，一个顶点会出现在多个子图中。在分割过程中，要尽量保证每个子图中分配到数量大致相等的边，也要尽量减少切割的顶点数（顶点的副本数少），即在分割过程中要考虑负载和通信量两个因素。对于切点法的优化目标定义如下：

参数说明：表示顶点A副本个数，表示机器A上分配边的数量，表示不平衡因子（）。

## 4.2异步计算模型中消息的流通性

### 4.2.1不同计算模型中数据流分析

六个模型数据流图

从纵向来看，同步模型、传统异步模型、DAIC异步模型同步障碍粒度的大小（消息的阻塞性），同步模型是整个为粒度，传统异步模型是图顶点，DAIC异步模型是边。

从横向来看，单顶点模型中消息的产生来源于图划分中边被切割的数量，多副本顶点模型中消息的产生来源于副本顶点的数量，可以看出通信量是不同的。

### 4.2.2数据流通性和通信量之间的关系

迭代计算中，下一轮迭代计算的开始前，上一轮迭代的计算必须完成（映射为并行性），消息必须传递完成（映射为通信量），所以在同步模型中数据流通性等价于通信量。而在异步特别是DAIC模型中，不用保证所有的消息均被传递后采用利用，而是到达一个消息理论上就可以被马上应用，因此在异步模型中消息的流通性不等价于通信量。

例子说明在相同通信量下，流通性是不同的。

### 4.2.3 消息流通量

提出消息流通量的定义

公式

图的迭代计算过程也是消息不断传递的过程，消息分为本地消息和远程消息。本文中，本地消息的流通性用边的流通量（定义1）来描述，流通量越大，流通性越好；远程消息的流通性用通信量来描述，通信量越低，流通性越好。

s

d

… …

…m …

m1

n1

**定义1：**边的流通量. 给定边e(s,d)，源顶点s的入度表示为dsin，目的顶点d的出度表示为ddout，m1表示在该分区中指向顶点s的顶点个数，n1表示在该分区中顶点d指向的顶点个数，边e(s,d)的流通量通过式1计算得到。

 (1)

在实现中，dsin和ddout需要对数据进行一次遍历才能得到，为了避免预处理的开销，本文使用已经分配的数据获取顶点的局部度数，来近似等于顶点真正的度数，这种策略不会导致很大的性能差异[3]。c是一个很小的常数。

## 4.3 一种面向异步图计算模型的图分区方法PAGraph

基于消息流通量提出了PAGraph

### 4.3.1 PAGraph分区算法

启发式公式

### 4.3.2 算法有效性证明

分析四种case，同时分析相对于HDRF的提升

### 4.3.3 算法实现

（）

说明（四种case）

算法实现

在迭代计算过程中，边是消息流通的基本单位，同时边的数量跟负载呈线性关系，因此本文提出了以边为粒度的分区方法PAGraph。

在PAGrap中将边的流通量、通信量、负载均衡作为优化目标，并定义了启发式函数计算边e(vi,vj)在每个分区p∈P的得分Score(vi,vj,p)，然后将边e分配Score最大的分区p\*。Score(vi,vj,p)（式2）包含三部分，ScoreTUR(vi,vj,p)（式1），ScoreCOM(vi,vj,p)（式3）和ScoreBAL(p)（式4）。各计算公式如下：

 (2)

系数是一个常数，用于控制分区负载均衡的倾斜度。

 (3)

****

表示顶点的出度，在实现中用2.1节中局部出度来代替，表示当前顶点所在分区的集合，。

 (4)

是所有分区中的负载最大值，是所有分区中的负载最小值，c是一个很小的常数。

图6

对边e(vi,vj)∈E进行分区时，有以下四种情形，下面将通过这四种情形来说明启发式公式的作用过程。

1）如果，，那么ScoreTUR(vi,vj,p)和ScoreCOM(vi,vj,p)都为零，将通过ScoreBAL(p)来选择目标分区，即从P中选择负载最轻的分区作为目标分区p\*。

2）如果或者，（假设）此情况如图6(a)所示。如果，那么ScoreCOM(vi,vj,p)和ScoreTUR(vi,vj,p)值都为零；如果，所有的ScoreCOM(vi,vj,p)是一个相等的值，ScoreTUR(vi,vj,p)则会因vi在p中边的数量不同而不同，边的数量越大ScoreTUR值就越大。在图6(a)中，假设vi从它的每条入边接收一个value，若将边e分配到p1，那么将会有本地消息直接传递给vj；若将边e分配到p2，那会有本地消息直接传递给vj。ScoreTUR(vi,vj,p1)> ScoreTUR(vi,vj,p2)且ScoreCOM(vi,vj,p1)= ScoreCOM(vi,vj,p2)，在p1和p2负载差别不大的情况下将会选择p1作为目标分区。

3）如果且，此情况如图6(b)所示。边e分配到p1和p2对于本地消息的流通性是没有影响的，都需要一次网络传递才能将消息继续传递下去，在各分区负载倾斜度不大的情况，将会从（dsin<ddou）中去选择负载最小的分区作为目标分区。

4）如果且，此情况如图6(c)所示。在各分区负载倾斜度不大的情况下，从中选择流通量最大分区作为目标分区，在图6(c)中ScoreTUR(vi,vj,p1)> ScoreTUR(vi,vj,p2)，所以选择p1作为目标分区。

从以上分析中看到，在1）、3）情况下，ScoreCOM项不起作用，这两种情况下跟现有的分区方法（PowerGraph和HDRF）是一致的，而在情形2）、4）中，ScoreCOM项发挥作用，并提高了消息在分区中的流通性。

## 4.4 改进的图划分方法VE-PAGraph

4.3节中的问题引起本节内容

### 4.4.1 VE-PAGraph分区算法

### 4.4.2 参数设置

### 4.4.3 算法实现

问题->Edge-Cut和Vertex-Cut结合（powerlyra）

启发式公式及有效性说明

实现

研究发现，根据副本顶点的职能不同，可以将其细分为两类顶点，其定义如下：

**定义2**. 计算顶点，副本顶点中具有邻接链表的顶点，它既要负责相关的计算任务，也负责消息的收集和发送。

**定义3**. 通信顶点：副本顶点中没有邻接链表的顶点，它只负责消息的收集和转发。

图7

在实验中发现，PAGrahp以及HDRF、PowerGraph的分区方法都会产生大量的计算顶点，增加大量的计算量和通信量，如图7(a)所示。在点边结合的分区方法Ginger中，计算顶点数量得到了很大程度的降低，但是通信顶点数量得到了增加，如图7(b)。该方法将顶点的度作为判断该顶点是否被切割的依据，但这是一个很抽象的判断依据，并不总是准确的，只能近似的从整体上给分区方法提供一个优化。

本文中，综合计算量、通信量提出了一个通过定量计算来判断顶点是否该需要切割的方法，数据按照广度优先遍历的顺序输入，一个顶点v0及其邻接链表={v1,v2,v3…}，首先设定一个阈值threshhold，定义一个计算公式计算顶点及其邻接Score(v0,,p)在每个分区p∈P上的得分，具体的分区过程如算法1所示。

**算法1.** 优化的PAGraph.

阶段1：边的预分区。根据2.2小节中的计算公式(2)将边e(v0, vi)vi∈预分配到相应分区中，并将预分区信息保留在向量中，()里面存储着预分配到分区p中的边。

阶段2：实际分区。根据预分区的结果

计算每个分区p∈P的得分：

 (5)

 (6)

(7)

for 

if 



else

p\*=select\_loadmin(P)



end if

end for

算法中的threshhold，通过实验验证，一般设定在[5,8]的区间内比较合适。按照优化后的PAGrahp对图7(a)，图7(b)进行分区，分区结果如图7(c)所示。

//从所有的分区中选择负载最小的分区作为目标分区

## 4.5 本章小结

# 第5章 实验及评价

## 5.1 实验的软硬件环境

### 5.1.1 实验环境

### 5.1.2 实验数据集

## 5.2 MR-DAIC性能分析

实验一：Maiter和MR-Maiter速度比较（Maiter的buffer设为1），验证MR-DAIC在通信量和消息利用速率方面的优势

实验二：Maiter和MR-Maiter速度比较（Maiter的buffer正常），验证MR-DAIC在消息利用速率方面的优势

实验三：优先级计算的速度比较，验证MR-DAIC在提高计算并行性和提高调度精度方面的优势

## 5.3 图分区算法性能验证

### 5.3.1 PAGraph性能验证

实验四：各个分区算法副本因子的比较---反映通信量（PAGraph应该和hdrf近似）

实验五：不进行消息的传递，看下最后的收敛值---验证消息流通性

实验六：速度比较—验证PAGraph性能

模拟网络环境不同，分别采用1次调度/一次通信、2次调度/一次通信、3次调度/一次通信。。。。

实验七：分区个数--扩展性

### 5.3.2 EV-PAGraph性能验证

实验八：各分区的副本因子和副本数量— （要有一个综合两者的得分）

实验九：速度比较——-验证EV- PAGraph性能

实验十：分区个数--扩展性

## 5.4 本章小结

# 第6章 总结及展望

## 6.1总结

目前，对大规模图数据进行相关的分析和数据挖掘是大数据处理的一个重要领域，而要实现对大规模的图数据进行快速和复杂的处理，就需要高效的图数据处理框架。同时，图数据处理领域的图算法往往具有迭代计算的特性，因此异步型的图数据处理系统相对于同步图处理系统具有更好的性能。因此，本文的研究工作围绕着完全异步化的图数据处理系统Maiter展开。

本次硕士论文首先介绍了本次课题的研究背景、当前研究现状和研究意义，确定了本次研究工作的可行性和具有的价值性。为了便于读者更好的理解本次论文的研究内容，本文对DAIC模型和图处理系统Maiter进行详细的介绍。接下来，开始对本文的主要研究内容多副本计算模型（MR-DAIC）和面向异步迭代计算模型的图分区算法进行介绍，并通过实验对研究内容进行了验证和分析。下面将对本文的两个主要研究内容进行详细的介绍。

首先，本文提出了支持多副本顶点计算的计算模型MR-DAIC，并基于Maiter框架将其实现。MR-DAIC模型将顶点分割成若干个副本顶点并分配到不同的机器上，计算粒度的减小使得Maiter框架具有更好的并行性，并有助于进一步提升优先级调度计算的效果。同时，MR-DAIC模型中副本顶点分布在不同的机器上，这使得副本顶点可以直接对本地接收到的消息直接利用，这提高了消息的流通性，加快算法收敛速度，

另一方面，本文对图划分方法进行了深入的研究，提出了一种面向异步迭代计算模型的图分区算法PAGraph和对数据适用性更强的图分区算法EV-PAGraph。多副本顶点计算的前提是将图顶点按照一定的策略切割成多个副本顶点，所以本部分研究内容是对第一部分研究内容的有力支撑。PAGraph是基于DAIC计算模型中消息流通无阻塞性的特性，实现优化各个分区负载均衡和消息流通性两个目标。Vertex-Cut类型的图划分方法在处理顶点度扭曲分布的图数据时，相对于Edge-Cut类型的图划分方法具有很好的性能，但是也存在着不足，Vertex-Cut会创建大量的副本顶点从而给系统带来了额外的计算开销，当时图数据中边比较稀疏时，这个缺陷将会被扩大，为了应对这一挑战，本文又提出了一种Edge/Vertex-Cut混合切割的图分区算法，该算法既能持有Vertex-Cut的优点，同时又避免大量副本顶点的创建。

总的来说，在DAIC模型的迭代计算过程中，图顶点的值通过不断累积消息，最终达到收敛值，因此对DAIC模型性能的研究其实就是如何加快对消息应用（提高消息被应用速率）和优先调度累积变化量大的顶点进行计算。基于DAIC模型的异步性达到了边粒度这一特性，本次论文中MR-DAIC模型和分区算法PAGraph是提高消息的被应用速率，MR-DAIC模型和优先级计算策略是优先具有高性价比的顶点，EV-PAGraph是提高对不同数据的适用范围。

## 6.2 展望

本次硕士论文具有一定的创新性，也通过实验对这些创新点进行了验证和分析，基本上取得了预期的效果。但是由于时间和精力的限制，仍然存在一些需要研究内容和在研究过程中发现需要进一步研究的问题需要继续探究。

（1）Maiter系统的研究。本文在对Maiter框架的优化方面主要是从系统核心的计算引擎进行优化，通过多副本顶点计算来提高计算的并行性和提升优先级迭代计算的性能。其实Maiter框架还有很多需要研究的内容，例如，如何从系统的角度实时的控制消息发送缓冲池的buffer，优化Maiter的通信量；利用Maiter差值累积计算的特性，更好的支持迭代计算。

（2）图分区算法的研究。本文的另一主要工作就是，面向异步迭代计算模型提出了两个高效的图分区算法。在研究中发现，如果对流数据进行分区后，再加上时间段或者实时的调整分区的策略应该会对最终的分区结果有所提升。同时根据Maiter优先级计算的特性，如果将优先级高的顶点提取出来组成一个“核心子图”，那么将会避开分布式环境通信开销的影响，迭代计算过中消息的传递将会更加流畅，加快分布式算法收敛速度。

# 参考文献

[11] 周晓峰，王志坚。 移动Agent计算研究[j]. 计算机时代，2006，25（7）：156-158.

[12]金培弘. 分布式系统概念与设计[M]. 北京：机械出版社，2004.

[13] 中国分布式计算总站[EB/OL]. [2011-09-25]. <http://www.equn.com/>

[14]程学旗,靳小龙,王元卓,etal.大数据系统和分析技术综述.软件学报,2014,25(9).

[15]潘巍,李战怀.大数据环境下并行计算模型的研究进展.华东师范大学学报(自然科学版),2014,5:43-54.

[16] Scalability. http://en.wikipedia.org/wiki/Scalability.

[17] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Communications of the ACM, 2008, 51(1):107-113.

[18] Dean J, Ghemawat S. MapReduce: a flexible data processing tool. Communications of the ACM, 2010,53(1):72-77.

[19] Welcome to Apache Hadoop! https://hadoop.apache.org/.

[20] Isard M, Budiu M,Yu Y,et al. Dryad: distributed data-parallel programs from sequential building blocks. ACM SIGOPS Operating Systems Review, 2007,41(3):59-72.

[21] Storm,distributed and fault-tolerant realtime computation, <https://storm.apache.org/.>

[22] Zheng S. Data freeway and puma: Realtime data streams and analytics. Proceedings of Hadoop in China, 2011..

[23] Thusoo A,Sarma J S,Jain N,et al. Hive-a petabyte scale data warehouse using hadoop. Data Engineering (ICDE), 2010 IEEE 26th International Conference on. IEEE, 2010. 996-1005.

[24] Thusoo A,Sartna J S,Jain N,et al. Hive: a warehousing solution over a map-reduce framework. Proceedings of the VLDB Endowment, 2009,2(2):1626-1629.

[25] Melnik S, Gubarev A, Long J J,et al. Dremel: interactive analysis of web-scale datasets. Proceedings of the VLDB Endowment, 2010, 3(1-2):330-339.

[26] Malewicz G, Austern M H, Bik A J, et al. Pregel: a system for large-scale graph processing.Proceedings of the 2010 ACM SIGMOD International Conference on Management of data.ACM, 2010. 135-146.

[27] Yucheng L, Joseph G, Aapo K, et al. Graphlab: A new framework for parallel machine learning. Proc. Int’ 1 Conf. Uncertainty in Artificial Intelligence (UAI’ 10),2010.

[28] Low Y’ Bickson D, Gonzalez J, et al. Distributed GraphLab: a framework for machine learning and data mining in the cloud. Proceedings of the VLDB Endowment,2012,5(8):716-727.

[29] Yucheng L, Joseph G, Aapo K, et al. Graphlab: A new framework for parallel machine learning. Proc. Int

[30] Kyrola A, Blelloch G E,Guestrin C. GraphChi: Large-Scale Graph Computation on Just a PC. OSDI, volume 12, 2012. 31-46.

[31] Avery C. Giraph: Large-scale graph processing infrastructure on hadoop. Proceedings of the Hadoop Summit. Santa Clara, 2011..

[32][EB/OL]. Maiter Project, http://code.google.com/p/maiter, 2013-07-15.

[33]Zhang. Y, Gao. Q. Accelerate large-scale iterative computation through asynchronous accumulative updates[C]//Proc of the 3rd Workshop on Scientific Cloud Computing. New York: ACM, 2012:13-22.

[34]PowerGraph  
[35] Xin R S, Gonzalez J E,Franklin M J,et al. Graphx: A resilient distributed graph system on spark. First International Workshop on Graph Data Management Experiences and Systems.ACM, 2013. 2.

[36] Gonzalez J E, Xin R S,Dave A, et al. Graphx: Graph processing in a distributed dataflow framework. Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2014.

[37]Powerlyra

# 致 谢

# 攻读硕士学位期间的论文和项目情况

参与项目

2014.06-2016.12国家自然科学基金项目：云环境下基于BSP模型的大规模不动点迭代计算研究（61300023）

2015.10-2016.12国家自然科学基金项目“支持海量数据处理的多粒度异步并行计算模型的研究”（61528203）

实践证明，底层曾用通用型的模型来封装上层具体领域的应用框架具有易于同现有的框架相融合的优点，但是在性能上是逊色于从底层就针对于具体领域而设计实现框架。

现有分布式图计算框架的引擎依据控制流和数据流模型的不同可W分为同步和异步两种。控制流模型决定了顶点计算如何调度,而数据流模型决定了邻接顶点数据值何时被使用。

随着社交网络分析、语义Web分析、生物信息网络分析等新兴应用的快速增长,对亿万个顶点级别大规

模图的处理能力的需求愈加迫切,这是当前高性能计算领域的研究和开发热点[王志刚]单机已无法处理如此海量的数据和日益增长的处理需求，由于“云计算”计算的高效性和可扩展性已成为目前应对海量数据处理这一挑战的主流途径。近年来，云计算技术在学术界、工业界的联合推动下获得了巨大的发展，大量的云计算系统被投入使用，根据应用领域的不同可以将其分为批处理系统、流处理系统、机器学习系统、图处理系统等。。