**中图分类号：**

**论文编号：**



硕 士 学 位 论 文

**基于运行时间预测的并行图计算系统性能优化研究**

作者姓名 李东泽

学科专业 计算机技术

指导教师 樊文飞 教授

培养院系 计算机学院

**The Research and Implementation of Email Importance Evaluation Method based on Thread Network**

A Dissertation Submitted for the Degree of Master

**Candidate：Li Dongze**

**Supervisor：Associate Prof. Fan Wenfei**

School of Computer Science and Engineering

Beihang University, Beijing, China

**中图分类号：**

**论文编号：**

硕 士 学 位 论 文

基于运行时间预测的并行图计算系统性能优化研究

作者姓名 李东泽 申请学位级别 工学硕士

指导教师姓名 樊文飞 职 称： 教授

学科专业 计算技术 研究方向： 分布式图计算

学习时间自 年 月 日 起至 年 月 日 止

论文提交日期 年 月 日 论文答辩日期 年 月 日

学位授予单位 北京航空航天大学 学位授予日期 年 月 日

关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中作出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名： 日期： 年 月 日

学位论文使用授权书

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名： 日期： 年 月 日

指导教师签名： 日期： 年 月 日

摘 要

在社交网络服务走向多元化的今天，邮件由于其良好的隐私性和普及性仍然拥有庞大的用户群体。

**关键词：**邮件分类，特征评价，邮件会话网络，EmailRank

**Abstract**

Nowadays, with the diversification of social networking services, the email still maintains a large user base due to its good privacy and popularity.

**Key words:** email classfication, feature evaluation, email thread network, EmailRank

目 录

[摘 要 I](#_Toc24493934)

[**Abstract** II](#_Toc24493935)

[目 录 III](#_Toc24493936)

[图 目 V](#_Toc24493937)

[表 目 VI](#_Toc24493938)

[第一章 绪论 1](#_Toc24493939)

[1.1 研究背景及意义 1](#_Toc24493940)

[1.2 国内外研究现状 3](#_Toc24493941)

[1.2.1 同步计算模型 3](#_Toc24493942)

[1.2.2 异步计算模型 6](#_Toc24493943)

[1.2.3 混合计算模型 7](#_Toc24493944)

[1.2.4 自适应计算模型 8](#_Toc24493945)

[1.3 研究目标和内容 9](#_Toc24493946)

[1.4 论文组织结构 10](#_Toc24493947)

[第二章 相关理论与技术研究 11](#_Toc24493948)

[2.1 图模型与图应用算法 11](#_Toc24493949)

[2.1.1 图模型 11](#_Toc24493950)

[2.1.2 图应用算法 12](#_Toc24493951)

[2.2 分布式图计算相关技术 14](#_Toc24493952)

[2.2.1 分布式图计算执行机制 14](#_Toc24493953)

[2.2.2 图数据划分 18](#_Toc24493954)

[2.2.3 系统实现技术 19](#_Toc24493955)

[2.3 运行时间预测 20](#_Toc24493956)

[2.3.1 基于数学模型的预测 20](#_Toc24493957)

[2.3.2 基于机器学习技术的预测 20](#_Toc24493958)

[2.3.3 基于相似性的预测 21](#_Toc24493959)

[2.4 GRAPE+系统 21](#_Toc24493960)

[2.4.1 系统架构概况 21](#_Toc24493961)

[2.4.2 PIE编程模型 22](#_Toc24493962)

[2.4.3 图计算执行流程 23](#_Toc24493963)

[2.5 本章小结 24](#_Toc24493964)

[第三章 基于运行时间预测自适应计算模型 25](#_Toc24493965)

[3.1 MPAP计算模型总体结构 25](#_Toc24493966)

[3.2 MPAP计算模型的设计与实现 26](#_Toc24493967)

[3.2.1 并行模式 26](#_Toc24493968)

[3.2.2 模型执行流(好好读代码来写，从状态角度来，给出状态转换图？与伪代码) 28](#_Toc24493969)

[3.2.3 模型在系统中调度实例 28](#_Toc24493970)

[第四章 运行时间与消息达到速率预测 30](#_Toc24493971)

[4.1 邮件会话网络的改进 30](#_Toc24493972)

[4.2 邮件会话网络的生成 30](#_Toc24493973)

[4.3 邮件会话网络的存储 30](#_Toc24493974)

[4.4 实验评估 30](#_Toc24493975)

[4.4.1 实验方案 30](#_Toc24493976)

[第五章 基于会话网络的邮件重要性评价方法 31](#_Toc24493977)

[5.1 PageRank算法 31](#_Toc24493978)

[5.1.1 实验方案 31](#_Toc24493979)

[第六章 EmailRank算法在实际系统中的应用 32](#_Toc24493980)

[6.1 应用于邮件客户端 32](#_Toc24493981)

[6.1.1 需求分析 32](#_Toc24493982)

[6.2 本章小结 32](#_Toc24493983)

[总结与展望 33](#_Toc24493984)

[论文总结 33](#_Toc24493985)

[未来工作展望 33](#_Toc24493986)

[参考文献 34](#_Toc24493987)

[攻读硕士学位期间取得的学术成果 35](#_Toc24493988)

[致 谢 36](#_Toc24493989)

图 目

[图 1邮件信息过载问题是一项价值6500亿美金的经济累赘 1](#_Toc531617249)

[图 2 ERIC邮件客户端的用户界面 4](#_Toc531617250)

[图 3信任网络结构 8](#_Toc531617251)

[图 4 EMIRT算法信任分数计算原理图 9](#_Toc531617252)

[图 5重要邮件分类方法的架构 11](#_Toc531617253)

[图 6邮件会话的两种网络结构 13](#_Toc531617254)

[图 7 Webmagic系统架构 18](#_Toc531617255)

[图 8 Lucene的架构和全文检索过程 19](#_Toc531617256)

[图 9 Spring Data Neo4j的应用架构 21](#_Toc531617257)

[图 10不同分类器实验效果对比 27](#_Toc531617258)

[图 11基于特征方法与基准方法分类效果对比 28](#_Toc531617259)

[图 12邮件的两种引用方式 31](#_Toc531617260)

[图 13改进后邮件会话网络构建示意图 32](#_Toc531617261)

[图 14邮件关系匹配流程图 34](#_Toc531617262)

[图 15 Neo4j浏览器界面中的邮件会话网络 36](#_Toc531617263)

[图 16改进和未改进会话网络对比图 36](file:///C:\Users\jiang\Desktop\最终论文\毕业论文_蒋坤_SY1606301.docx#_Toc531617264)

[图 17某用户邮件关系网络中的重要节点识别 44](#_Toc531617265)

[图 18 EmailRank和基于邮件特征方法实验结果对比 45](#_Toc531617266)

[图 19原始Nylas-Mail客户端界面 48](#_Toc531617267)

[图 20插件ThreadNetwork效果图 48](#_Toc531617268)

[图 21 W3C会员影响力评价系统架构图 50](#_Toc531617269)

[图 22分布式爬虫架构图 51](#_Toc531617270)

[图 23 W3C会员影响力评价系统截图：影响力排名 53](#_Toc531617271)

[图 24 W3C会员影响力评价系统截图：影响力对比 53](#_Toc531617272)

表 目

[表 1节点中心度的度量方法 16](#_Toc531563451)

[表 2四类邮件特征列表 22](#_Toc531563452)

[表 3 SemanticDistance的算法描述 24](#_Toc531563453)

[表 4邮件重要性特征重要权值 30](#_Toc531563454)

[表 5改进和未改进会话网络下的方法性能对比 37](#_Toc531563455)

[表 6 EmailRank和基于网络拓扑方法的实验结果对比 45](#_Toc531563456)

# 绪论

## 研究背景及意义

在社交媒体、社交网络和即时通讯飞速发展的如今，随着“互联网+”以及各式互联网应用等新型产品，如抖音、猿辅导的兴起和云计算等新相关技术的飞速发展，人类获取数据的途径和规模正以前所未有的速度快速增长，于此同时，与大数据相关的技术变革于理论发展成为了当今世界的热点话题。大数据在环境科学、超级计算、生物学、互联网经济学等领域以及通讯、航天、金融等行业普遍存在，并引起人们的广泛关注。在这些领域或行业中所涉及的数据通常具有某种关联性，从而产生了大图数据。图作为计算机数据结构中一种基本且常用的数据类型，是表示实体与实体间关系的方法，一张图由若干顶点和顶点间所连接的边组成，其中根据连接边是否具有方向的不同又分为有向图与无向图。无论哪种表示方式，其在表达的语义甚至结构方面都比线性表和树更为复杂。社会或科学领域中的许多应用场景都可以通过转化成图计算模型从而得到解决，与图相关的处理和应用几乎无所不在。比如，社会安全部门需要对整个社交网络进行精准的模式匹配，从而增加寻找贩毒、犯罪集群的可能性；交通运输部门需要确定最优的运输路线；生物信息领域需要对蛋白质进行子图匹配，分析蛋白质间的相互作用从而开发出更有效的临床药物[1]。不仅如此，互联网行业中的用户账号匹配、社交网络分析、风险控制模型的训练都可以看作是大图数据的应用。因此，都大图数据的分析与计算具有很重要的现实意义。

早期传统的图数据管理与图分析计算技术通常只针对“小图”进行处理，整个计算过程中不需要过多甚至复杂的迭代过程，更不会产生大量需要信息通讯的消息，时间与空间的开销相对较低。但是，现实世界中实体规模的持续扩大，导致相应图模型的数据规模迅速增长，甚至有数十亿顶点以及上万亿条边的图数据，而这些只靠顶点和边数量组成的图数据结构还只是大图数据的冰山一角。很多应用中涉及的复杂图数据为了表达其现实语义，往往在顶点以及边上附加一定的属性信息，如表示人的节点通常含有姓名、电话、地址等相关属性，这些属性的存储与计算代价通常是点和边的数十倍乃至数百倍。而且，相比于图数据上的搜索查询，大图上的分析计算算法往往需要基于图的结构进行递归迭代操作，直至满足收敛条件或达到预设的最大迭代轮数，因此会产生巨大的通讯数据，并需要频繁的处理中间结果。此外，除了静态数据与离线计算外，很多情况下实时处理情景需要图数据是动态变化的，如淘宝“双十一”实时分析风控数据等，这种动态往往以流式载图作为扩图途径，不断改变着大图数据的规模和结构，因此面对如此大规模的静态和动态图数据，其底层存储、索引建立、统计分析、实时计算等处理的时间开销和空间开销远远超出了早期传统图处理的能力，因此如何解决大图数据的存储技术与查询计算技术，已经成为数据库领域需要迫切解决的问题，也是大数据领域极具挑战的工作。

面对上述大图数据存储与计算的诸多问题，研究者们已经不满足于仅通过传统的方法探讨基本的图管理与分析问题，一些新兴的研究热点也在不断涌现，如图查询语言、图计算框架等。对于这些研究方向，虽然早期也有一些相关的探讨，但随着计算机软硬件的以及云技术的不断发展，如何提出更准确高效的方法来解决这些问题，为图管理与分析工作带来了新的机遇与挑战。Parallel BGL[7]最为最早一代的并行图计算系统，其在经典的图应用算法基础上开发了并行版本，但此类系统大多数本身不具备一定的稳定性与容错性，并且部分图应用算法的正确性也得不到保证，为了解决稳定性的问题，一部分研究团队专门为特定的图计算应用开发了分布式图计算架构来加速图数据的并行计算，虽然针对性的架构处理可对特定图应用算法进行深度的代码优化以达到更好的处理性能，但毕竟过于局限，针对某一新的图应用算法，算法的移植甚至重开发操作需要重写大量的代码，不仅代码的复用率极低，框架的通用型、可拓展性以及可维护性也很差，对编程人员与使用人员的挑战性极大；MapReduce[8]作为一个并行的计算框架，及其开源实现Hadoop[9]为海量的数据处理提供了便利，为此，用户只需要定义Map和Reduce函数就可完成相应的逻辑计算，不需要关心并发、容错、一致性等复杂问题。但是由于MapReduce框架的自身特点，任务的执行划分为多次Map和Reduce阶段，每个阶段都需要多次的读写分布式文件系统，并且，任务的执行期间无法共享更新数据，再加上图计算的过程通常涉及多轮的迭代，各迭代涉及顶点间的复杂联系，且迭代完成时还要依赖对点的消息传递以满足图计算正确性，因此若将每轮迭代抽象成MapReduce作业不能高效的进行并行图计算，虽然Haloop[10]等处理框架对MapReduce过程进行许多改进，但是并没有从根本上解决基于分布式文件系统所带来的I/O开销问题。针对上述图计算框架的问题，为解决对大规模图数据计算的迫切需要和传统的计算平台无法满足当今的图处理需求的问题，面向大规模并行图计算框架的研究吸引了越来越多的研究者投身其中。

。

## 国内外研究现状

目前面向大规模图数据计算的分布式并行图计算模型及其相关优化研究取得了不错的结果，依据调度方式的不同，大致分为同步计算模型、异步计算模型、混合计算模型、自适应模型四类。

### 同步计算模型

为改进MapReduce计算模型在大规模图数据上的计算效率低等问题，1990年，图领奖获得者Valiant提出了基于消息传递的BSP(bulk synchronous parallel)计算模型[11]，Google也在2010年推出了基于该计算模型的大规模图数据并行计算框架Pregel[11]，这种计算模型区分于现有的MapReduce计算框架在于其适合做数据的迭代计算，其采用一种“块”同步的思想，即通过消息传递机制实现块间数据同步、块内并行计算。其核心思想是将每一次迭代称为一个超步(Superstep)，整个图计算划分为多个超步，而一个BSP作业由一系列顺序执行的并通过全局同步路障分开的超步组成，并行任务时每个节点按照各自的超步组织，接收来自上一个超步的全局消息，执行本地计算并将本次计算生成的消息发送给下一次超步，计算过程如图2.1所示，并行迭代直到全部顶点收敛，即每个顶点相邻两次迭代的变化量小于给定的常量或达到预设的最大迭代次数。

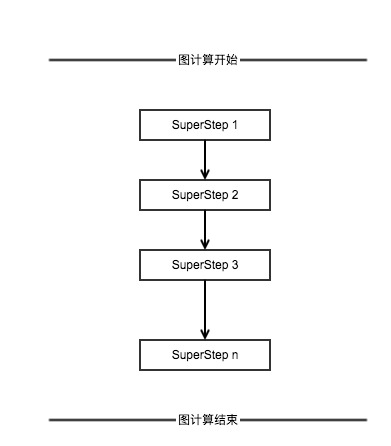


图2.1 图计算垂直结构

在每一次的超步计算中（图体现出超步的内部计算），又会分为本地计算、消息传递、全局同步三个阶段，图2.2、2.3（这两个这图感觉不太好）分别展示了每个超步过程中的计算过程及同步模型的整体计算流程。

1. 本地计算: 计算节点间相互独立，每个节点负责读取本地内存数据执行并行计算。

2. 消息传递: 本地计算阶段完成后，每个计算节点将生成消息通过通信传递给与之相关联的其他节点，以便交换彼此计算所需要的数据。

3. 全局同步: 用于数据间的整体同步，即块见同步。各个计算节点在完成本地任务后进入全局同步阶段，先完成本地任务的节点需要等待没完成任务的节点，直到最后一个节点完成本地任务，所有节点统一进入下一轮的本地计算。

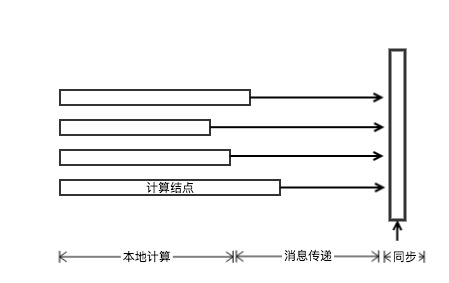


图2.2 超步内计算过程



图2.3 同步计算模型

从上图中我们可以清楚的看出，每轮计算中所有顶点只能使用上一轮迭代计算的更新消息，而通过计算得到的最新消息也只能通过同步在下一轮的计算中使用。因为全局同步的存在，使得整个图计算的调度十分简单，同时，每一轮的消息会在全局的同步下批量的传送，极大限度的利用了网络的流量。

为高效的处理大规模图数据的计算问题，在上述BSP计算模型的启发下，Google首次提出了以点为中心的计算框架Pregel[12]。在点中心的计算模型中，用户只需要从点的角度出发，重写一个与具体应用紧密相关的计算函数Compute，而不需要考虑数据的处理过程。在Compute函数内，涉及1) 处理上轮迭代的更新消息; 2) 本地计算逻辑，并更新本节点的值或节点状态信息; 3) 消息传递逻辑。 Pregel的具体执行过程如下: 首先，选定一个图分区策略[13-15]进行图分区操作，划分后的不同分区分配到不同的计算节点上，然后在每一个图节点上执行Compute函数，即用户自定义的超步处理逻辑，每一次的迭代后，通过全局同步操作保证所有计算节点统一进入下一轮的迭代计算，整个迭代过程直到没有任何消息需要同步或所有的图节点达到收敛状态时停止，输出结果。Pregel框架从容错性、一致性、同步控制等方面给出了可行的解决方案，整个计算模型简单、易用、可拓展性强。受Pregel框架的启发，相继涌现出很多基于BSP模型的大规模并行图计算系统，如GRAPE[18], Gemini、GPS[16]、Giraph[17]、Hama、Giraph++、GiraphX、Pregel+、X-Pregel、Pregellix、Mizan、MOCgraph、Blogel、Nscale、SC-BSP、LFGraph等，是对Pregel开源实现的改进与扩展，在性能与功能上不断完善了基于BSP同步计算模型。

GRAPE[18]是以子图[11]为中心基于MPI[12]消息传递的分布式并行图计算引擎，采用独创的PIE(PEval、IncEval、Assemble)编程接口，该接口易于使用并保证了并行计算过程对用户完全透明。用户只需要为特定的图应用算法编写这三个函数便可执行期望的图计算任务；Giraph作为点中心模型[12]遵循了Pregel的机制、基于Hadoop平台的开源分布式图计算引擎，其利用Zookeeper解决了单节点故障问题、采用Netty网络通信的消息传递机制将每轮超步串行化，解决了Hadoop平台本身在图计算过程中低效率问题；GPS类似于Pregel系统，拥有容错机制、扩展性强且易于编程的优势，保证其可在超大规模图数据上执行图计算任务。其于Pregel系统的不同在于1) 通过master.compute函数扩展了Pregel的API，该函数可轻松高效地实现图应用算法；2) GPS具有可选的动态重分区功能，该功能在图计算过程中根据通信状态以决定是否将顶点重新分配给不同的机器，以提高性能；3) GPS拥有LALP的优化功能可减少网络通信量；Hama[]也是基于Hadoop平台于BSP模型下的图计算系统，其专注于解决机器学习和图算法领域的大规模计算任务；Gemini[114] 作为一个分布式的图计算系统，其关注计算和通信组件的性能，采用高效的划分，积极压缩计算时间，同时隐藏通信成本，从而达到了在保持高效的同时实现可扩展，弥补了高效共享内存系统可扩展性差和可扩展分布式系统性能差的不足。

相对来说，针对需要多轮迭代处理的图计算任务而言，基于BSP的同步计算模型比MapReduce模型的大数据处理平台具有更大的性能优势，也极大的简化了图计算任务，充分利用了网络流量，但是该模型仍有以下几方面限制：

1. BSP模型具有木桶效应，因为全局同步的存在，每一轮迭代的计算时间取决于最慢的计算节点。因此，节点的计算能力不同或图数据的分区不合理都会导致计算节点间的负载不均衡，严重影响图计算效率。

2. BSP模型每轮迭代只能使用上一轮迭代的更新信息，导致迭代次数较多，收敛速度较慢。

3. 消息量大，导致通信代价昂贵。在分布式的图处理框架中，为了满足大规模图数据处理，通常需要将原始的图数据划分为多个子图，并把每个子图分配到不同的计算节点上，因为处于不同计算节点之间的顶点具有一定的关联关系，这就导致不同计算节点间需要进行消息传递以满足图计算的正确性。然而在实际的图数据中，尤其是社交网络等图应用，图中顶点的度数服从幂律分布，从而使得通信的代价成为影响系统性能的瓶颈之一。

4. 某些需要协调图中邻接点的算法无法在BSP模型下实现，例如图着色问题[19],该问题意在使用最少的颜色为图着色，保证相邻的顶点颜色不同。这类问题对于贪心的解决方案在BSP模型下不会收敛，因为图节点试图获得非邻居节点的颜色值，只能通过两者的共同邻居节点来获得，因此同步操作的过程中可能会出现这两个节点颜色不断翻转的情况，本轮迭代具有相同颜色的邻接顶点在下轮的迭代仍会选取相同的颜色，从而导致无法收敛。

### 异步计算模型

为解决上述BSP同步计算模型的诸多问题，科研工作者提出了异步计算模型AP(Asynchronous Parallel)，如图2.4所示。



图2.4 异步计算模型

异步模型中，相邻两次迭代没有明显的界限，任何计算节点随时接收来自其它节点的消息，并直接接受调度器的调度执行; 同样，本地计算生成的最新消息也随时传递给其它需要的节点。因为异步模型摒弃了全局同步的操作，使得每个计算节点不用关心其它节点的进度、状态，只要消息队列存在消息就可以进行下一轮的本地计算，因此消除了同步计算模型下木桶效应问题。经大量的理论分析和实验表明，异步计算模型在执行效率以及资源的利用率方面相对于同步计算模型具有较大的优势，尤其是在图处理系统负载不均衡时，这种优势体现的更加明显[20-22]。

目前采用异步图计算模型的系统包括GRAPE+、Maiter、PowerGraph、GraphLab、GraphX、PowerLyra。其中Maiter框架作为完全异步的大规模图数据处理框架，其采用了DAIC计算模型，避免了同步开销，加快收敛速度，极大的提高了大规模图处理效率；GraphLab主要是针对MLDM应用开发的支持同步与异步的图计算系统，其对图计算中数据一致性模型做了详细分析；PowerGraph针对图数据详从工作负载、图分割、存储、通信和计算等多方面带做了详细的分析，提供了完善的图分割数学理论支撑，证明切点法比切边法能提高一个数量级的图计算性能。其采用了以顶点为中心的GAS编程模型，增加了细粒度并发性同时支持同步和异步模型。但它不支持图的动态修改，容错机制未能充分利用顶点副本；PowerLyra从图分割方面对PowerGraph进行了改进。提出了一种混合图分割方法hybrid-cut，即出入度高的顶点采用切点法反之出入度低的顶点采用切边法，经过试验对比性能提高了至少1.24倍。

虽然异步计算模型相比于同步计算模型省去了全局同步所带来的开销，但是也同样存在着以下方面的不足之处：

1. 异步模型可以加速图计算的收敛速度，但由于没有全局同步的限制，导致迭代轮数的增多, 造成了大量的冗余计算。

2. 异步模型在分布式图计算系统的设计方面相对于同步模型来说更加复杂，在设计过程中，不仅要设计正确的调度器，还需要考虑数据的一致性问题，因为异步模型下的任何一个计算节点随时接收来自其它计算节点的消息，因此必须设计一套额外的机制来保证相同数据对象可以被不同的进程和线程互斥访问，例如GraphLab[23]采用分布式锁的形式来保证数据的一致性，而GRAPE+[24]采用了聚合函数Aggr()来保证数据的一致性。

3. 异步模型的计算过程与计算结果具有不确定性，如果算法设计的不合理，迭代计算可能无法达到预期的效果，甚至不能收敛，即使能够收敛，也同样会在迭代计算的最终收敛前，造成大量的冗余计算。不仅如此，对于采用异步模型的分布式图计算系统来说，编程难度有所增加，调试十分困难[25]。

### 混合计算模型

最近研究表明不同计算模型下，图算法的执行效率受到图应用算法、图分区方式、数据图的特性与规模、分布式集群的规模等多方面因素的影响，不仅如此，在整个迭代周期的不同阶段，可能也需要不同的计算模型以达到全局最优[26]，因此为了解决上述问题，科研工作这提出了混合计算模型。

延迟计算模型[27]作为一种轻量级的混合计算模型，简称SSP模型。其通过引入一个固定的限定值bound来保证最快与最慢的计算节点间相对的迭代轮数不会超过给定的bound值。因此，其允许计算能力强计算节点采用异步的方式执行下去，来减缓同步模型下的木桶效应; 同时通过bound值的限定强制要求计算能力强的节点需等待计算能力差的节点以减少异步模型下的冗余计算。

除此之外，混合计算模型[26]，其通过一组启发式的算法建立代价收益模型来动态预测同步模型和异步模型两种调度方式的性能，并实现了在计算过程中对两种计算模型自由的切换，如图2.5所示。



图2.5 混合计算模型

[26]的实验结果表明，采用了上述混合计算模型的PowerSwitch分布式图计算引擎可以准确预测两种模型的性能，并完成调度模式的切换，相对于使用单一的调度模式，在其上执行了大量的图算法，如PageRank算法、单源最短路径算法、图着色算法等，都在执行效率上得到了不同程度的提高。

### 自适应计算模型

自适应计算模型[24]计算模型，简称AAP(Adapt Aysc Parallel)模型。这种模型摒弃了全局同步的操作，因此从本质上来看其属于异步模型。但与异步计算模型不同的是，每个计算节点在自适应计算模型下均保留一些参数来测量与其它计算节点的相对执行进度并根据当前接收到的消息决定是否进入下一轮迭代。上述整个过程只在本计算节点下完成，不用关心其它任何节点的执行状态。该模型不同与其它计算模型主要体现在以下几个方面:

1. 同步计算模型：AAP模型通过异步的消息传递机制来消除BSP模型下木桶效应问题，摒弃了全局同步机制从而消除了这一操作所带了的额外开销。

2. 异步计算模型：AAP模型通过引入限定值Delay Stretch使得各个计算节点自己决定每一轮迭代需等待的时间，聚集更多的消息传递量从而达到执行更有效的计算，因此很大程度上减少了AP模型下的冗余计算问题。

3. 延迟计算模型：延迟计算模型下通过预设的限定值bound，使得在bound值允许的范围内，各计算节点采用要异步AP模型进行计算；一旦到到达bound限定的范围，计算快的节点就需等待，直到计算慢的节点完成当前轮数的计算，才可进行下一轮的计算。但AAP模型不同于延迟计算模型在于其通过限定值Delay Stretch动态调整各计算节点间的相对进程，当出现计算落后的节点时，其允许该节点等待一段时间以便接受更多的消息，这样可以最大化的利用其它计算节点快的优势，使得慢节点的每一轮计算更有效率。

4. 混合计算模型：相对于Hsync模型，AAP模型不需要全局的预测开关在BSP和AP模型进行切换，相反，AAP模型允许每个计算节点决定各自的计算模式，因此在图算法的执行期间，某些计算节点采用BSP模式，与此同时，其它节点采用AP模式。除此之外，由于AAP模型动态调整计算节点间的相对进程，因此其减少了模型间切换所带来的而外开销。

## 研究目标和内容

在已有的图计算模型中，同步模型存在“木桶效应”问题；异步模型存在大量的冗余计算；混合计算模型为实现同步与异步间自由切换产生了额外的内存与预测开销；而自适应模型中，每个计算节点通过引入限定值动态调整节点间相对进度，从而解决了同步、异步、混合模型下的问题，本文在已有自适应计算模型的原型下，提出了新的基于运行时间预测的自适应图计算模型，该模型可在慢机的情况下规避上述模型的弊端，加速并行图计算。本文的主要研究内容如下：

1. 提出基于运行时间预测的自适应并行图计算模型，以下简称 MPAP模型（Message Predict Async Parallel），并将对消息量的预测转化为对运行时间及消息到达速率的预测。

2. 我们将对算法运行时间的预测问题定义为机器学习中的回归问题，使用均方相对误差（MSRE）做为损失函数，最终选取机器学习中适合图计算领域的回归模型作为候选，进行训练预测，从而分析各模型在分布式图计算时间预测问题上的优缺点。

3. 我们选取包含单源最短路径（SSSP）、深度优先搜索（BFS）、网页排序（PageRank）、节点结构分析（HITS）、图采样（Graph Sampling）、图连通性（WCC WCC-HashMin）、标签传播（Label Propagation）、社区发现算法（Louvain）等9种学术届与工业界常用的图计算应用算法，并尝试将算法进行分类，针对每类算法，给出不同的特征提取方案，并展示预测效果。

4. 我们在已有的自适应图计算系统原型下，开发完成基于消息量预测的自适应并行图计算系统，并与同步、异步、及自适应原型系统进行对比，评估效果。

## 论文组织结构

本文主要分为七个部分。主要的组织结构如下：

# 

# 相关理论与技术研究

## 图模型与图应用算法

### 图模型

图作为计算机领域重要的数据结构之一，其语义与结构的表达都要比线性表和数结构更加复杂。最基础的图模型被称为简单图，这类图中，仅有顶点与连接顶点的边组成，根据边是否具有指向性又分为有向图与无向图。在图计算领域中，除了需要关注由顶点与边组成的图数据结构外，还关注顶点与边上附加的属性信息，在多数的图应用背景下，这种附加的属性信息更为重要，我们将在下一小节详细介绍本文用到的经典图应用算法，这里给出本文所讨论的简单图模型的基本定义。

给图，其中代表图中点的有限集合，代表图中边的集合，代表图中顶点或边上的附加属性信息，即，其中表示图中的顶点；，其中表示图中的边，且，表示是连接顶点与的边；表示顶点与边上属性的集合。根据边是否具有指向性，图又可以分为有向图与无向图，其中无向图中若存在顶点指向的边，那么一定也存在顶点指向的边，而在有向图中则不成立，图2.1所示为简单的有向图模型示例：

图

图2.1(a)包含n个顶点与m条边，顶点的编号为从1开始的整数序列，图中顶点没有附加属性信息，边上附有单一的属性信息且为整数 。

现实世界中，多数图数据的分布极其不均匀，即少数顶点会与周围大量的邻居相连，而多数顶点则与周围邻居交互较少。这类图数据反映到图结构即为少数顶点度数较高，而多数顶点度数较低，造成这一现象的的原因也十分直观，比如社交网络中少数明星会与大量的粉丝产生交互行为，如评论留言；淘宝订单数据中，少数用户的年均订单量明显大于其余用户，以此来鉴别是否用户具有刷单行为。这类分布不均匀的现象对图计算造成极大的挑战，容易出现负载不均衡现象，造成计算效率低甚至由于内存限制导致无法完成图计算。下图为真实的图数据中顶点度数的分布情况，我们可以从图看出，这个分布不均匀现象十分明显。

图（横坐标是点数，纵坐标是度数）

### 图应用算法

图数据处理与图数据的挖掘得到了学者们的广泛研究，不同的实际问题会抽象出不同的图应用算法[]，除了最短路径、图着色、最小生成树、连通性、可达性等基础图论算法外，还包括社区发现、标签传播、页面排序、子图匹配、图中元素分类与聚类等多种复杂的算法，本小节给出工业界与学术界常用的图应用算法。

**(1) 单源最短路径**

单源最短路径问题中，给定一个带权有向图 ，其中代表每条边上的权重且，给定某点做为源点，现要计算从源点到图中其它各顶点的最短路径长度，该长度即为到目标顶点的路径中所走边的权重之和。Dijkstra算法作为解决单源最短路径问题经典的算法之一，其按各顶点与源点间的路径长度的递增次序，生成源点到各顶点的[最短路径](https://baike.baidu.com/item/%E6%9C%80%E7%9F%AD%E8%B7%AF%E5%BE%84/6334920)的算法，既先求出长度最短的一条最短路径，再参照它求出长度次短的一条最短路径，依次类推，直到从源点到其它各顶点的最短路径全部求出为止。

**(2) 页面排序**

页面排序对于从超大量的页面文档中筛选出有价值的信息十分重要。PageRank的提出开启了链接分析研究的热潮，基于这种链接分析方法，可以客观的衡量网页质量，并且能够做到独立于内容，不需要人工干预就可以自发的挖掘互联网上有价值的资源。PageRank算法作为最早提出的链接分析算法之一，并在Google公司获得了巨大的商业成功，包含了两个基本假设：一是超文本链接包含用户对一个页面或网站的判断信息；二是某个页面被其他页面链接的数量越多，该网页的重要性越高，越能提供有价值的信息。其基本思想是如果网页存在指向网页的链接，表明网页的所有者任务网页重要，从而把网页的一部分重要性传递给网页，该部分重要的计算由网页的PageRank值与该网页的出链接数共同决定，即，而网页的PageRank值计算公式如下：

其中表示阻尼系数，其意义为用户到达某一页面后，继续沿该页面向后浏览的概率；代表所有链接到网页的页面。上述公式经过多轮迭代，系统的PageRank值逐渐收敛。

另一种对网络图结构做分析的HITS算法，是由康奈尔大学该的Jon Kleinberg博士于1997年首先提出的，该算法与PageRank算法类似，最早用于搜索，现在也常用来做社交网络结构分析。但PageRank算法不同的是，其核心思想是建立在页面链接关系的基础上，对链接结构的改进算法。HITS算法通过两个评价[权值](https://baike.baidu.com/item/%E6%9D%83%E5%80%BC)——内容权威度（Authority）和链接权威度（Hub）来对网页质量进行评估。其中Authority认为如果一个页面提供了关于某个主题的信息，那么这个页面就是有价值的，这样的页面就是Authority页面，因此如果有许多其它的页面都指向该页面，说明此页面的Authority值就高；Hub认为还有一些页面，比如Google页面，虽然它本身不提供任何主题信息，但是从这个页面出发，可以跳转到很多有价值的页面上去，这样的页面就会成为Hub页面，因此如果某个页面指向了很多Authority值高的页面，那么它的Hub值就会高。这种利用页面之间的引用链来挖掘隐含在其中的有用信息（如权威性），具有计算简单且效率高的特点。页面的网页，其链接权威度越高。

**(3) 社区发现算法**

互联网络的复杂度对计算机的计算能力提出了严峻的挑战。而随着对复杂互联网络的深入研究，发现这类复杂网络具有较强的社区结构，社区发现就是通过网络中节点间的连接关系挖掘社区结构的过程，因此对复杂互联网络社区发现算法的研究对了解网络的结构具有重要意义。

Louvain算法作为一种基于图数据与模块度优化的社区发现算法，相比于普通的模块度和模块度增益算法，具有明显的性能优势，不仅如此，该算法还能探测到层次的社区结构,不会遗漏一些小型的社区。Louvain算法分为两个阶段，初始阶段网络中每个节点独立为一类，第一阶段的目的是分类，根据划分后模块度的分值将相似的点分为一类，并做好标识；第二阶段将分类后的图重新初始化，即属于同一类别的节点合并为一个节点，继续执行第一阶段，这个迭代过程直到整张图的模块度不在发生变化。

标签传播算法作为另一种高效的处理大规模网络的社区发现算法，由于其近似线性的时间复杂度而收到广泛的关注，其基本思路是根据已标记的节点标签信息来预测该节点周围邻居中未标记的节点标签信息。每个节点标签按相似度传播给周围邻居，在传播的每一步，每个节点根据邻居节点的标签来更新自己的标签，且相似度越大，其邻居节点对其标注的影响权值越大，他们的标签越趋于一致，越容易传播。在标签传播过程中，保持已标记的标签不变，使其将标签传给未标注的数据。最终当迭代结束时，相似节点的概率分布趋于相似，可以划分到一类中。然而，该算法由于每个节点的标签依赖于周围邻居，其聚类的有效性和迭代速度对标签的更新顺序十分敏感，因此一定程度上影响了社区发现结果的稳定性和准确性。

**(4) 图采样算法**

一方面，针对社会网络中节点对的采用工作可用于大规模网络的用户兴趣识别；另一方面，随着Graph Embedding等将复杂网络映射到低维空间的机器学习技术的兴起，图采用技术吸引了越来越多的研究者投身其中。

图采样算法大体可分为两类，一类是随机游走，另一类是随机采样。其中随机采样算法的思想是随机的选择图中的点或边，其要求图中的点和边可以被自由的乱序地随机访问，因此该算法并不适用于某些注重结构的图数据；面向随机游走的算法相比于随机采样算法，只需要记录游走过程中所处的当前状态以及当前状态的邻居状态，换句话说，这种算法只需要直到局部的信息，而不以前知道全图信息，因此随机游走算法从这一角度来说更加可行。

## 分布式图计算相关技术

近些年来，云计算以及其在大规模数据处理方面的诸多优势，得到了学术界和产业界的广泛关注，依靠云计算环境对大规模图数据进行高效的处理，被证明非常具有发展潜力[书8]。尤其在分布式并行处理方面，利用云计算分布平行处理的特点，可将一个大图分割成若干个子图，把针对一个大图的处理分割为若干个针对子图的处理。正是因为云计算分布式并行计算能力，能够显著的提高对大规模图数据的处理能力。但是，面向复杂迭代计算任务的大图数据处理是一项系统性的工作，涉及诸多关键性的技术，本节将从执行机制、图数据划分、系统实现技术三个方面进行介绍。

### 分布式图计算执行机制

云环境下的大数据分布式处理为了满足不断增长的数据量，需要系统有很高的扩展性，设计有效的计算框架是基础的问题，决定分布式并行执行方式，是进行解耦处理和提高可靠性的基础。此外，依赖于云环境的大规模图处理，任务负载重，执行时间长，因此执行框架必须提供高可靠性、高灵活性、高效率和高伸缩性的执行机制，包括消息传递机制，同步控制、容错管理、任务调度、扩展性保证等关键技术为作业的高效、顺利运行提供支持。

**图查询处理的遍历模式：**在迭代计算的过程中，根据基本的计算单位，图数据读取和遍历的模式可分以点为中心、以边或路径为中心、以子图为中心三类。

对于以顶点为中心的系统，其迭代计算过程是通过遍历图中顶点完成的。对于每个图顶点，通过处理其收到的消息，完成顶点值的更新，并按照出边向其目的顶点发出新的消息。很多图应用算法可以自然的采用这种方式实现，例如PageRank算法和单源最短路径算法等，由于简单易用，目前绝大多书主流的分布式图计算系统如Pregel、Giraph、Hama、Spark[书32]、PowerGraph[书25]等均采用以顶点为中心的计算方式。此外，通常图算法在迭代过程中是逐渐收敛的，即已经收敛的顶点不必重复访问出边以发送消息，如单源最短路径计算问题。Pregel等系统采用voteToHalt机制实现该功能，即为图中每个顶点设置一个标志位，处于激活状态的顶点需要进行更新计算并发送消息，否则直接在遍历过程中跳过该顶点。

Giraph++[书24]、GoFFish[书33]、Blogel[书34]、GRAPE等部分较新的系统提出采用以子图为中心的计算方式，以加快迭代收敛速度，减少消息传递数量。这种处理方式将图数据划分为若干个Partition，每个Partition视为一个子图。在迭代过程中，同一个子图内的顶点首先进行计算，然后子图之间进行消息互换，从而减少子图内部计算产生的冗余消息，加速收敛，减少同步开销。特别的，Blogel[34]还支持以子图为中心的通讯方式，可以进一步降低消息规模。子图为中心的处理可以有效地减少整个迭代和同步的次数，但大大增加了本地节点内部冗余消息发送的概率，从而增加本地处理的开销。其虽然作为以顶点为中心处理在跨节点消息和同步方式方面的优化，但并不能完全取代以顶点为中心的处理方式，具体的性能与应用迭代情况和图数据的特点相关。

以顶点为中心的计算方式虽然简单易用且对某些算法可以避免访问非激活顶点的出边，但会引入对出边的数据，或按照出边生成的消息数据的随机访问，这个访问在磁盘环境下，引入昂贵的I/O开销，影响性能。单机磁盘处理系统GraphChip[书35]通过出边按照目的顶点的存储顺序进行排序，避免了随机磁盘访问，但同时引入较大的预处理时间。为次，X-Stream[书36]系统采用以边为中心的计算方式。具体地，以此迭代分为两个阶段：1) Scatter阶段，顺序扫描出边，获取对应的源顶点值及其激活标志位，如果源顶点处于激活状态，则产生并顺序存储消息数据，否则跳过该边；2) Gatter阶段，顺序扫描消息数据，更新对应的源顶点值。显然，以边为中心的计算方式保证了边数据和消息数据可以被顺序存取，且不需要额外的预处理开销。但同时引入了对原顶点数据的随机访问，也无法避免非激活顶点的出边数据的顺序扫描操作。对于前者，X-Stream对顶点数据进行分块操作并逐一加载到内存，以满足Scatter和Gatter阶段的数据访问需求，降低随机存取的开销。此外，以经典为计算中心和以边为计算中心的计算方式，均破坏了图本身的联通性，使得数据访问过程的局部性较差。单机磁盘系统PathGraph[书37]设计以路径为中心的计算方式，将图数据划分基于树的若干个分区，提高了图遍历过程中的数据加载的局部性。但以边和以路径为中心的遍历方式都存在着较大的应用局限，尤其缺乏在分布式环境下的成功实践。

**消息传递机制：**在图处理应用中，每一个图顶点都需要向周围邻居节点发送消息或从周围邻居节点接受消息，而图中边可以理解为收发消息的通道。对于一般的图而言，边的数目要远大于图中顶点的数目，如此大规模的消息通信，如果处理不当，很容易成为整个图计算过程的瓶颈，本小节从消息时序控制、消息交互模式、网络通信平台和顶层消息优化技术四个方面阐述消息传递的核心技术。

消息发送时序控制：根据消息发送的时序控制，可以将现有的图处理分为同步集中式与异步两种方式。其中同步集中式中计算任务与消息通信串行之行，在计算完毕后，统一进行消息传递，如图2-2(a)[书中21页]所示，控制和实现方式简单，可以最大效率的利用网络I/O，但也容易造成瞬间的网络通信阻塞，增大了发送端的消息存储开销；异步方式中，计算任务与消息通信并发执行，在计算过程中就可以发送消息，此种方式将全部的消息量分散在不同的时间段，如图2-2(b)所示，可以充分的利用空闲的网络资源，避免瞬间的网络通信阻塞，主要图处理系统GRAPE+，Maiter就是采用异步通信的方式。

消息交互模式：图处理过程中的消息交互模式可以分为Push和Pull两类。Push方式是一种以源顶点为中心的计算方式，即图计算系统遍历图中所有顶点，完成顶点的更新计算，然后按照出边广播消息给目的顶点。这种方式处理机制简单，且在内存资源充足的情况下，由于各个任务可以分别处理源顶点，因此具有很好的并发性。另一方面，考虑到出边中的目的顶点的分布具有较差的局部性，从目的顶点的角度分析，其收到的消息在时间和空间方面具有较差的局部性。而作为同步处理系统的内在约束，在第次迭代，一个顶点只有接收到了所有步的消息之后，才可以启动本地计算更新顶点的值。Push方式下较差的时空局部性，使得图处理系统必须存储所有已收到的消息，知道所有源顶点完成消息的发送，这显然增大了接收端昂贵的I/O开销。目前，Pregel、GRAPE、Giraph等系统均采用Push方式实现消息传递。其中Giraph系统同时支持内存和磁盘两种版本图计算应用。根据Zhou等人的测试结果，当内存资源紧张而采用磁盘处理时由于频繁的存取磁盘信息，Giraph的处理性能极具下降[书39]。分布式内存系统Trinity尝试在源顶点与目的顶点之间建立一个二分图，协调消息的生成顺序，使得接收端收到的消息能够及时被处理，从而减少内存开销。然而二分图的建立是一个NP-难问题，为简化该问题，Trinity忽略了出度顶点，即高出度顶点所发送的消息仍然需要一直驻留在内存。由于现实图数据具有幂律偏斜特点，因此大量消息无法被及时处理。

另一种常用的消息传递模式是以目的顶点为中的Pull模式，即各个任务遍历目的顶点，根据入边向对应的源顶点主动请求消息数据。显然，当一个目的顶点收到所有请求的消息后，即可立即执行更新计算，避免存储大量的消息数据。Chronos[书40]、Seraph[书41]等系统均支持Pull方式消息传递机制。然而，Pull方式下，在请求消息数据时，需要传送对应源顶点的ID，在分布式环境下引入额外的通信开销，尤其对于单源最短路径问题，在迭代的收敛阶段大部分目的顶点可能已经处于收敛状态，Pull方式则会引入额外的请求开销。Kylin[书42]系统根据目的顶点的状态决定是否向源顶点发送消息请求，这虽然降低了Pull的请求开销，但是可能导致部分需要发送消息的源顶点被忽略，最终影响计算结果。GraphLab[书43]则采用共享顶点状态的方式将Pull请求的网络通信开销转为本地操作，具体的地，对于某个任务，如果位于的顶点是中顶点的入度邻居，即中的请求消息，则在中维护一个顶点的共享状态。在迭代计算过程中，如果顶点的值发生变化，通过网络消息同步对应的共享状态值。而目的顶点在启动更新计算后，仅需要根据入边从本地的共享状态值中获取需要的消息数据即可，避免了传输源顶点ID所引入的额外网络通信开销。然而，根据Zhou等人的测试结果，这个方式本身也会引入较大的内存开销，尤其对顶点值规模较大的算法[书39]。

网络通信方式：这对现有分布式图计算系统的研究，网络通信的实现主要包括以下四类：1) 基于MPI(Message Passing Interface)的通信方式，典型代表有GRAPE和Maiter[书45]；2) 基于RPC(Remote Proceduce Call)的通信方式，典型代表有Hama和Giraph等；3) 机遇ActiveMQ的通信方式，如BC-BSP[书46]；4) 基于Netty的通信，如Giraph。其中部分系统同时支持多种通信方式。

顶层消息优化技术：目前，大规模图数据的计算过程中，网络通信依然是计算瓶颈之一，因此需要对通信规模和消息数据存储加以优化控制以降低消息维护的时空开销。首先利用分割技术，可以降低子图之间的联通性，使大部计算任务可在子图内完成，将网络通信变为本地通信，从根本上减少节点间的消息传递数量。良好的图划分算法可以减少网络开销，但无法降低消息存储的空间开销。其次，针对具体应用，采用聚合机制，也可以减少网络通信规模，如Pregel等系统的Combine技术，将发送给相同目的顶点的消息合并为一条消息，以达到同时降低网络通信开销与内存开销。特别的，异步通信下，由于发送端的消息被异步发送，而消息在目的顶点之间的分布具有较差的局部性，因此能够参与聚合的消息比例较少，导致通信收益降低，甚至难以抵消聚合所引入的消息遍历等额外开销，故GRAPE、Giraph等系统屏蔽了发送端的Combine功能，仅允许接收端的消息聚合，以降低存储开销。最后针对Combine技术的缺点，Zhou等人设计了一种消息在线处理技术，即MOC(Message Online Computing)技术[书39]。与Combine技术仅对消息数据可见不同，MOC技术在接收消息时，目的顶点的value也是可见的，因此接收的消息可立即参与顶点的更新计算，而不必缓存，从而节省了大量存储空间，作者在Giraph上实现了MOC技术，实验结果发现，在相同内存配置的情况下，MOC可以显著提高系统的数据处理能能力。同样使用了该技术的图计算系统还包括GRAPE。

### 图数据划分

如何针对复杂的图计算任务合理的组织数据，从而提高整个处理过程的执行效率，已经成为图计算领域的基础性研究问题。主要包括图数据的划分、图数据的存储和索引。首先实现低耦合的划分是实现大图分布式处理的基础操作，是保证负载均衡、减轻“木桶效应”的基础，特别是在处理过程中根据实时的负载变化进行动态的重划分，面临着诸多技术挑战，本小节主要介绍大图数据的划分技术。

图划分是分布式图计算系统进行分布式图数据计算的前提。由于图计算通常按照拓扑结构访问数据，所以每次迭代处理均会引入巨大的通信开销，成为制约分布式处理行性能的关键因素。因此，一个良好的划分算法保证划分后的子图在负载均衡的前提下，减少子图之间的边的数目，从而减少网络通信。另一方面，云计算资源会随着并发处理作业数目的变化和集群中计算节点的增删而动态变化，因此同一个作业在不同时刻提交时，其分布式任务数目不尽相同，这一点被称为分布式处理粒度的弹性变化，这导致图数据需要按照当前的任务数目重新划分，划分结果的不可重用性使的图划分的执行效率也成为影响总计算的重要因素。目前的图划分算法主要包括离线划分算法、和在线划分算法算法。近年来，相关研究主要可以分为两类，以METIS[书53]为代表的离线划分算法和以LDG[书63]为代表的在线划分算法。前者可以显著优化切分边的规模，降低迭代计算过程中的消息通信开销，因此受到学术界和工业界的广泛关注。然而，离线划分过程需要频繁访问图顶点，引入了昂贵的时间开销。另一方面，在线划分算法可以在图处理系统的数据加载阶段完成图划分，该过程仅扫描一次图数据。与离线划分算法相比，在线划分算法在一定程度上牺牲了分图效果来获得较高的执行效率。但是此类算法通常为集中式的，以便于维护复杂的启发式规则，保证相对较好的划分效果，其扩展性显然受到单机处理能力的限制。虽然已经存在分布式在线划分算法的相关研究工作，但是启发式规则的维护开销仍然显著影响了算法运行效率。

### 系统实现技术

无论何种图数据处理技术都需要以系统的形式作为载体呈现给用户，合理的设计系统的定位，选取适合的软硬件环境，灵活的运行相关数据结构以提高程序质量，才能将核心的关键技术高效、完整的呈现出来。近些年来，针对大图数据的分布式计算系统层出不穷，采用了不同的计算模型、数据组织等优化技术，在功能与适用性方面也不尽相同，对他们进行充分的研究可以更好地理解相关技术的实现方法。特别是大量高质量的开源图数据计算系统的出现，推动了大图分布式数据处理学术研究和产业业务的快速发展，以它们为平台实现具体算法的优化和二次开发成为了可能。

本小节对现有的主要分布式图计算系统从计算模型、消息传递机制角度进行分析，表2.1展示这些系统采用的计算模型和消息传递机制。

表3.2 分布式图计算系统计算模型对照表

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **同步计算模型** | **异步计算模型** | **其它计算模型** |
| Pregel | √ | X | X |
| Giraph | √ | X | X |
| Pregel+ | √ | X | X |
| GPS | √ | X | X |
| PowerGraph | √ | √ | X |
| PowerSwitch | X | X | √ |
| GRAPE | √ | √ | X |
| Hama | √ | X | X |
| GRAPE+ | X | X | √ |
| Gemini | √ |  |  |

表3.3 分布式图计算系统通信方式对照表

|  |  |  |
| --- | --- | --- |
| **名称** | **消息传递** | **共享内存** |
| Pregel | √ | X |
| Giraph | √ | X |
| Pregel+ | √ | X |
| GPS | √ | X |
| PowerGraph | X | √ |
| PowerSwitch | X | √ |
| GRAPE | √ | X |
| Hama | √ | X |
| GRAPE+ | √ | X |

## 运行时间预测

针对算法运行时间预测问题，现有的研究分析方法主要分为基于数学模型的预测、基于机器学习技术的预测、和基于相似性的预测。

### 基于数学模型的预测

基于数学模型的预测，主要是针对算法所使用的运行模型进行时间复杂度分析，除此之外，还可以结合运行环境对算法的运行过程进行更细致的分析。如果对所运行算法的专业领域知识认知比较深刻，并熟知算法运行过程中对运行环境对算法程序产生的影响，那么可以对算法的运行过程进行精确的建模分析。为了量化这一过程，可以首先对系统的特性进行量化，之后建立一个基于系统特性的分析模型，最后对目标机器进行特性量化，从而使用模型的出算法在目标机器上的运行概况，分析资源的消耗情况。

这类方法需要研究人员对算法所应用的专业领域具有较深刻的了解，能够理清不同输入下算法的运行逻辑，并且同时对算法运行所在的环境也具有一定的了解，能够分析算法运行时所需要的开销。因此基于数学模型的预测方法通常适用于大型专有机器上的软件开发，对跨平台性的分析较为困难。

### 基于机器学习技术的预测

Hutter[]等人提出了基于机器学习技术的算法预测方法，他们根据具体问题实例的特征，建立了算法运行时间与这些特征之间的函数关系，通过对已有数据的训练，得到具体模型参数，最终根据得到的模型用于接下来的时间预测，他们的研究结果显示，他们采用随机森林等模型相比于之前的方法，有更好的预测效果。

此外，对于预测模型的输入，Chun et al.(2010)中提出了一种对输入数据进行特征分析记录的方法，具体做法为：使用程序的插桩技术，在具有循环和分支语句结构中插入计数操作，并将技术的结果作为输入数据的特征，后续对于新的输入，需要事先运行程序的片段，以获取这些计数结果，这种方法虽然需要获取程序的源代码，但是对代码的运行逻辑以及其他程序相关知识不做任何要求。这种基于机器学习的预测方式不需要了解领域的知识背景，适合计算机专业人员进行应用，但是由于缺乏对应领域的知识，因为对模型的解释存在一定的难度，因此需要在训练模型后进行专家分析，以增强模型的可解释性。

### 基于相似性的预测

目前，很多的计算系统均基于同步计算模型设计与实现的。对于这种计算模型，程序的整个运行过程通常可以分为计算、消息同步两个阶段，迭代执行。根据这一特点，可以将一个程序可以近似的认为是多个相同阶段的多次叠加，因此可以根据某次计算中已知阶段的计算时间来预测接下来的未知阶段。这种基于不同阶段的相似性预测对程序运行的逻辑要求较高。目前同步计算模型盛行的环境下，这个假设能够较好的符合大多数程序的运行逻辑。但是并行分布式计算的发展趋势是计算与通信的相互隐藏，异步进行，因此难以使用该技术进行预测工作。

## GRAPE+系统

GRAPE+系统是Fan等人开发的基于自适应异步计算模型的大规模分布式图计算系统[11]，该系统中，首次结合了同步计算模型与异步计算模型的优点，利用每个计算节点自己维护的参数机制，实现了计算节点可独立的在同步与异步计算模式下自由切换。因为本文的研究内容基于GRAPE+系统，因此本小节该系统的体系架构、编程模型、自适应计算模型机制、和图计算执行流程方面详细地介绍该系统。

### 系统架构概况

GRAPE+体系架构如图2-4所示：

图

从图中可以看出，整个系统主要分为数据存储、GRAPE+核心组建、系统接口、图算法应用实现以及用于外围管理接口五大部分。

**数据存储层**：表示GRAPE+系统处理的源数据的存储形式与存储位置，目前仅支持源数据存储在文件系统的某些文件中。且GRAPE+支持文件格式的图数据载入功能，即文件包含图中所有的顶点，每行表示为的二元组形式分别代表顶点的ID与顶点的附加属性；文件属于图分区文件，GRAPE+支持图数据的预分区功能，该文件包含图中每个顶点所属于分区，每行表示为的二元组形式分别代表顶点ID与该顶点所属的分区ID，GRAPE+指出文件可有现有的图分区系统计算生成，如Xtrapulp[111]；文件包含图中所有的边，表示为三元组形式，分别代表边中源点ID，目的点ID以及边上的附加属性信息（可以结合小图给出文件样例）。

**核心组件层**：位于数据存储层上面的就是GRAPE+核心组件层，该层由图数据存储管理、分布式MPI消息传递机制、自适应异步控制、统计信息收集器等组件组成。其中图数据在GRAPE+中以临接表的形式存储，该存储方式可轻松的获取某顶点的一度出边，因此适合Push模式的消息交互方式；MPI消息传递负责整个系统的通信过程，包括数据的迁移、消息的收发等；自适应异步控制主要负责控制各计算节点的执行进程，即每个计算节点都会独立维护自己与其余各计算节点的执行进度参数，新一轮的迭代计算开始前，各个节点根据参数的状态以决定是否立即执行本轮迭代计算或等待一段时间后进行计算，等待的目的即为了接收更多的消息，以执行更有效的计算，从而加快收敛；统计信息收集器在GRAPE+系统中主要负责收集各节点的迭代进度，并通过MPI消息传递机制广播给所有计算节点。

**系统接口**：GRAPE+系统的接口层提供两种功能，其一是提供用户以为计算框架，基于MPI消息传递的编程接口。用户实现上述三个接口函数完成对图应用算法的处理，即可在GRAPE+系统上运行期望的图应用任务，整个计算过程并行化操作对用户透明。

**图算法应用层**：代表了目前GRAPE+系统支持的内置图应用算法，包括单源最短路径算法、PageRank算法、图若联通性算法、和协同过滤算法。

**外围管理**：GRAPE+的外围管理主要负责GRAPE+集群的环境搭建以及部署工作，并提供了一键式日志管理、系统启停等服务。

### PIE编程模型

**图符号术语**：给定某有向图或无向图，其中代表图中点的有限集合，代表图中边的集合，代表图中顶点或边上的属性；

给定某一自然数，我们采用某种分图策略将图分割为个片段，其中每个均为的子图，且有，，，，针对每个顶点 ，有，针对每条边，有。

编程模型采用边分割或点分割策略的来分割图，当使用边分割时:

(1) 代表点的集合，其中针对每个点存在一条边 ，使得，且。

(2) 代表点的集合，其中针对每个点存在一条边 ，使得，且。

我们将上述成为子图的边界点(border node)集合，本文用到上的术语见表2-1.

**PIE编程模型**：众所周知，图计算问题作为图查询问题的一类，针对某一问题实例给定问题，为回答该问题，该模型需要用户提供以下三个函数：

1. PEval (局部估值函数): 该函数将子图与问题作为输入，并在子图上计算回答该问题，得到结果，其中针对该问题的所有单机图算法均可直接使用。

2. IncEval (迭代增量函数): 该函数将已有结果 、子图增量 、与问题作为输入，并在更新后的子图上回答问题，保证有 ，其中 是在原有的基础上做的更新计算。

3. Assemble (聚合函数): 该函数用于聚合其他子图在PEval或IncEval阶段计算所产生并同步本子图的更新量，最后将聚合结果更新到已有结果。

### 图计算执行流程

图2-6是用户提交作业和后续系统处理作业的整个流程示意图。

1) 首先，用户在系统环境下提交GRAPE+作业，整个过程中用户通常以脚本命令的方式执行并添加相应的控制参数，作业提交成功后，用户可以在控制台监控作业状态，查看作业日志信息。

2) 系统接收作业提交会，根据提交参数中的进程数执行分布式图计算任务，该任务分配计算节点，同步提交参数，正式进入图计算流程。

3) 图计算流程中，每个节点首先进行图数据的加载任务，GRAPE+系统本身不支持图数据的划分功能，因此载图工作需要读取预先处理好的图分区文件，因此加载工作时只需要根据图中顶点所属的分区信息选择性的载入本地的数据管理器即可，对应图中的Load阶段。

4）图数据载入完成后，才真正进入迭代计算的计算框架。根据用户编写的接口函数，即函数，计算框架自动在每个计算节点上迭代完成上述函数，整个计算过程与消息传递过程异步进行，直到满足迭代终止条件或到达预设的最大迭代轮数，整次图计算任务结束，输出计算结果到本地文件系统。

## 本章小结

本章主要介绍了本文研究所涉及的相关技术背景，首先给出了图的定义，并介绍了工业界与学术界常用的图应用算法，包括单源最短路径算法、PageRank算法、HITS算法、联通性算法、随机游走图采样算法、标签传播算法、以及社区发现算法。然后，本章介绍了分布式图计算的相关技术，包括计算模型、图划分技术、消息传递机制，并介绍了一些常用的分布式图计算系统。接着，本章介绍了算法运行时间预测的常用技术，包括基于数学模型的预测、基于机器学习与历史数据的预测、以及基于相似度的预测。最后本章介绍了GRAPE+自适应分布式图计算系统，本文接下来的研究工作均在该系统上开展。

# 基于运行时间预测自适应计算模型

在前面章节中，我们针对现有的图计算模型进行了研究，并且分析了不同计算模型的优缺点，发现了潜在的性能提升空间(这句话前面要应和一下)。在这一分析的基础上，我们基于传统的自适应计算模型，设计了基于运行时间预测的自适应计算模型——MPAP计算模型，并在GRAPE+系统的基础上扩展了该系统的架构，实现了GRAPE++系统。

MPAP计算模型的设计与实现的关键部分在于重构分布式图计算系统的调度层，使得每个计算节点独自控制自己的调度进程。在MPAP调度模式的设计中，我们需要定义严谨且正确的调度方式，不仅如此，还要时刻维护正确的图并行计算语义。在具体的实现中，则需要确定不同时刻下的每个进程的当前状态以及状态转换过程。同时，在系统的设计层面行需要增加对运行时信息的收集模块、模型的训练与预测模块，以利于自适应计算模型的顺利进行。整个系统的实现还需要针对使用的数据进行简化，最大可能降低训练与预测引入的额外开销，以提升图计算性能。

因此，本章首先介绍MPAP模型的结构，并详细介绍该计算模型的设计与实现上的细节，接着本章给出MPAP计算模型下的工作流程，并给出单源最短路径算法在该模型下的伪代码。

## MPAP计算模型总体结构

MPAP计算模型的总体结构如图3-1所示，整个模型分为MPAP调度引擎模块、网络通信模块、数据采样模块、模型训练与预测模块、容错模块5个部分。

在MPAP计算模型中，我们将分布式图计算的异步调度实现为一个新的独立调度引擎，整个引擎的执行过程中，会与预测模块进行交互，根据交互得到的预测结果，通过参数控制，在保证兼容同步与异步两种不同的计算模型的同时，以决定是否立即进入下轮迭代或等待一段时间后进入下一轮迭代计算，该过程看起来就像是每个计算节点独立控制自己的图计算进度，不受任何其他节点影响。这一部分在接下来的小节中会详细说明。

数据采集模块主要负责采集图计算过程中的运行时信息，采集后的信息会收集到模型训练模块以提供机器学习预测模型的训练数据，因此数据采集工作主要分为离线数据采集和实时数据采集两种。离线数据采集是实现独立运行多遍图计算任务，从而得到相同图数据下不同求解问题的运行时信息，后续的模型训练工作完全可以基于这些已收集的日志数据，提供预测服务；我们可以看到，例如单源最短路径等一类算法，图中的每个顶点都可以作为图查询任务的源点，以计算该源点到其余各顶点的最短路径，因此我们可以通过离线数据采集方式以获取大量的训练数据，使我们的模型更加有效。但我们以注意到例如图连通性分析、PageRank这一类算法，往往针对固定图数据，查询任务与结果单一，因此无法像单源最短路径算法一样获取大量离线数据。所以，这类算法可以通过实时数据收集的方式，在图计算开始阶段，通过前几轮迭代以收集小部分数据，同于后续模型的训练预测，因此这需要我们的模型选取在小样本集上具有较好的预测效果。

在整个图计算系统的执行过程中，预测模块会针对现有的运行时信息，给出对运行时间的预测，因此各个计算节点会轮询的与预测模块进行交互，以控制自己的执行进度。

MPAP模型的设计业考虑到了容错模块的支持，通过下一节的介绍，可以看到，MPAP模型可以很好的模拟同步或异步计算模型，保证极端情况下图计算的顺利执行。

## MPAP计算模型的设计与实现

本小节从MPAP计算模型的核心内容，包括MPAP模型的并行模式、MPAP模型的执行流、和系统中调度实例四个方面，对该计算模型的设计与分析进行详细的介绍。

### 并行模式

**MPAP计算模型**：MPAP计算模型中，为解决同步模型下的“木桶效应”及异步模型下的冗余计算问题，每个计算节点引入了限定值，之后每轮迭代计算开始前，各计算节点需判断是否等待长的时间以积累更多的消息，我们给出基于下述函数进行动态调整：

上述函数中的各参数描述如下，表示某一fragment新一轮计算等待的时间：

(1) ：表示计算节点当前接收的消息量。直观上来看，如果 越大，则当前计算节点接收消息越多，应尽快开始下轮迭代计算。

(2) ：预测的时间与消息到达速率。

(3) ：表示fragment前一轮计算后的空闲时间，以防止无限期的等待，超参数 处于(0,1)之间，用于控制fragment等待的时间，越大，则每个fragment等待的时间越长。

**并行模型**：MPAP模型本质上仍然采用上述介绍的PIE编程模型，给定某一图计算问题实例及在该实例上的问题，MPAP模型需要用户指定上述PIE函数。首先，将输入的图数据 通过边分割或点分割策略分为多个片段 ，每个片段存储在某计算节点上，除此之外，MPAP模型需要做为coordinator节点，用于完成后续将要提到的集群终止状态检测与运行时特征的收集、训练和预测任务。因此，若 ，则多个片段会分到同一个虚拟机上并分享内存。

与传统的自适应计算(前面是不是要介绍一下)模型不同的是，MPAP计算模型还需增加以下声明，其中所有在PEval中的声明均在IncEval中共享。

1. 状态变量：PEval中每个片段都会声明并维护一个集合，该集合用于存储子图中每个点的状态信息，其中包括结果信息，在接下来的IncEval阶段中，该集合也用于根据接收到的消息，更新有用信息。

2. 聚合函数：PEval中需要指定聚合函数来解决多台机器传递的消息同时作用于相同的变量的问题，常用的聚合函数如函数、函数、函数等。

3. 特征提取：IncEval中每个片段在接收处理本轮的消息后，会向coordinator节点发送本轮的运行时日志，该日志主要包含当前该片段的状态信息与特征信息。

**消息传递**：每轮计算结束后，每个worker都会收集属于自己的fragment的更新结果，这些结果来自与每个fragment内部更新数据的改变量，并将更新结果发送给其他的worker，为实现该消息传递机制，每个worker需声明维护以下数据结构：

(1) 图中全部点与该点所属worker的映射，以便顺利将该点的更新消息发送给对应worker

(2) 一个缓冲buffer，用于接收来自其他worker发送的消息

MPAP的消息传递过与经典AAP模型相同，是点对点通信、且发生在图计算的任意时刻，即任意worker在任意时刻都可将消息发送给worker 而不需要关系此时处于状态，同样，也会在任意时刻接受来自其他worker的消息，保存在缓冲buffer中，该过程不会阻塞任何计算过程。但与AAP模型不同的是，每个fragment除发送本轮更新的消息外，还需像coordicator节点发送本轮的运行时信息，该信息用于后续的模型训练。

**终止条件**：MPAP模型的终止条件与经典AAP模型相同，即新一轮迭代计算开始前，如果当前没有任何接收到的消息，则会像coordinator发送结束命令, 当coordinator接收到所有worker的命令后，会广播各个worker 终止命令，各个worker会对此返回ack确认自己是否真的结束，如果某个worker仍有任务需要计算，则回应wait命令，继续进行下一次计算，coordicator重新进入监听等待状态。

### 模型执行流(好好读代码来写，从状态角度来，给出状态转换图？与伪代码)

在这一小节中，我们通过算法1来描述MPAP计算模型的调度流，整个执行过程包括两个线程，其中线程用于迭代执行MPAP模型下的PIE函数，线程根据MPAP模型的预测结果来控制线程的执行。

因为PEval过程不参与预测，因此初始阶段线程首先执行PEval函数，根据图分割后的子图数据执行本地计算，得到运行结果，并根据子图临接情况生成消息数据，存放在数据结构中。此时线程进入睡眠监听状态；线程会轮训的执行下述流程：1) 检测本地是否有未发送的消息，若有则通过MPI消息传递机制，发送给目标节点；2) 检测是否接收到其他计算节点数据的发送请求，若有则接收数据，并置接收标志位为1；3)

从上述流程可以看出，MPAP模型虽然每个计算节点根据预测结果独立的控制相对进度，但其本质上仍属于异步计算模型，因此所有异步模型下的语义正确性保证，都适用于MPAP计算模型。

### 模型在系统中调度实例

之前我们介绍了图计算领域中的经典算法，本小节我们以单源最短路径算法为例子，给出MPAP模型的下的调度实例。算法的输入输出形式化表述如下：

Input: 带权有向图 ，与图中某顶点

Output:

如表2-2所示，在PEval阶段中采用了众所周知的求单源最短路径的算法—Dijstra算法。但与该算法单机版本不同的是1) 我们针对该子图中的每个点定义元组，其中代表该点与原点的距离，初始化为无穷大，意味着与原点不可达；代表在本轮计算中该点是否被访问过，初始化为。该元组声明在PEval中，但也在IncEval中共享，后续也同样用于更新IncEval接受来自其它worker的消息。2) 我们采用做为聚合函数，即当该fragment中某个点接收到了来自多个worker的更新消息，我们选取其中的最小值做为最终的唯一更新消息。

上述PEval过程执行结束后，每个子图会将外部点的距离存储到预先声明的数据结构中，被异步的发送给目的节点。

IncEaval的过程如表2-3所示，首先对接收的消息执行聚合函数，并更新对应点的值，之后同样采用Dijstra算法计算本轮最短路径，并将计算后的最新值以消息的方式传递给其它worker。与传统自适应模型不同的是，IncEval过程还需将本轮计算的运行时信息发送给coordinator所在的节点，用于接下来的模型训练与预测，详细内容下节阐述。

如表2-4所示，除PEval与IncEval函数外，MPAP模型还需要Attl函数，用于动态调整各fragment间相对进程，该函数对用户透明，具体细节后续给出。此例中不需要Assemble函数，因此不做介绍。

## MPAP计算模型的容错性

## 本章小节

# 运行时间与消息达到速率预测

通过上一章对MPAP计算模型的详细介绍，我们发现该模型通过对消息量的预测，与目前已经接收到的消息量进行比较，计算可能需要等待的时长，已接收更多的消息，从而增加每轮迭代计算的质量，加快收敛。但是我们发现，分布式图计算中消息的预测是及其困难的，因为某个计算节点接收的消息量不仅取决于该计算节点与其余各节点的的连接情况，更取决于其余与之相连的各节点在上一轮迭代的计算情况，甚至网络波动等外界因素都会对消息量的预测造成致命的影响。因此，我们通过分析，完成对每轮迭代运行时间和消息到达速率的预测，并通过公式计算，已达到预测消息的目的。

本节首先分析了目前工业界与学术界常用的8种图应用算法，包括单源最短路径算法、广度优先搜索算法、PageRank算法、HITS算法、社区发现Louvain算法、标签传播算法、随机游走图采样算法、和弱联通性算法的两种实现，在PIE编程模型下的通过对上述算法运行时状态的打桩监测，我们将算法分成3类。针对每类我们给出了不同的图特征提取方案，并采用不同的机器学习模型进行训练预测。实验表明，我们我们的预测十分准确。

## 算法分类

我们以图2-1为例，介绍同种类图算法的Active Node Window的变化情况。如图所示，图包含19个节点以及27条边，假定该图采用边分割的策略分为3个子图，，,其中包含点以及与这些点所连接的边，包含点 以及与这些点所连接的边，包含点以及与这些点所连接的边。其中与有两条边相连，与有3条边相连。

图2-2展示了pagerank，sssp以及hits算法在4个超步运行下的Active Node Window情况，每个超步中，浅灰色的点代表非活跃点，深灰色点代表活跃点；其中图2-2-(a)展示了pagerank在前4轮超步下的working window，图中所有的点在每轮计算中均为活跃点，同样每个fragment均参与了每轮计算；图2-2-(b)展示了sssp在前4轮超步中下的working window，分别是,,,,每轮计算的活跃点数不恒定；图2-2-(c)代表了hits算法在前4轮超步下的working window,可以看出该算法在第一，第三轮迭代中，所有点均为活跃点，而第二，第四轮迭代中只有部分点（接收更新消息的点）处于活跃状态。

因此我们可以看出，不同的图算法在每轮迭代计算中，active nodes有不同的变化效果，接下来我们给出所有算法的现实中图数据上的active nodes变化折线图，所有算法都需要实现上述用户定义的PIE函数。

从图2-3中可以看出，pagerank，label-propagation算法在每轮计算中，所有点均处于active状态，因此我们将该类算法称为Always Active Style；而sssp、bfs、sampling 算法的折线图上working window表现为先升高，后降低(其中sampling算法在采样终止时降为最低)；而wcc、wcc-hashmin算法在折线图的表现为开始升高，之后逐渐下降，我们将上述5种算法均成为Message Dependency Style；因为每轮迭代计算活跃点的总数依赖于当前fragment接收到的消息数(后面我们针对不同种类的图算法给不同的特征提取方案)；hits 以及louvain算法在折线图上表现出一定的规律性，其中hits算法在奇数轮迭代计算中，所有的点均处于active 状态，而偶数轮的迭代计算中部分点处于active状态，这类算法典型的将一次计算流程分为多个超步，因为我们将这些算法成为Multiple Parse Style，这类算法在图计算中也十分常见，包括最小生成树MST等。

### 持续是活跃型（种类一）

该类算法中，每个fragment在每一次超步计算中，都需要遍历所有的内部点，完成规定的图计算任务，并将本轮计算生成的消息传给邻居。

表2-5 **Always Active Style IncEval**

|  |
| --- |
|  |

我们可以看出，输入代表该fragment接收到来自于上一轮迭代其它fragment发来的消息，在IncEval中，该类算法对本fragment的所有内部点进行相应计算（line 2）最后将更新后的消息发往其他的fragment（line 3）。

### 消息依赖型（种类二）

表2-6 **Message Dependency Style IncEval**

|  |
| --- |
|  |

在该风格下，通常计算开始时只有一个点或若干点处于活跃状态，而其它点是否参与本次计算取决于是否接到到消息，因此消息的传播条件决定了这类算法的活跃点数目，如表2-6所示。

### 多阶段型（种类三）

表2-7 Multiple **Phase Style IncEval**

|  |
| --- |
|  |

如表2-7中所描述，该类图算法无法在一次超步中完成一次图任务计算，例如hits算法，该算法与pagerank算法类似，都是对一个网络图做结构分析，最早用于搜索，现在也常用来做社交网络结构分析，但pagerank算法不同的是，hits算法一个(网络)节点的重要性分成了两种属性，分别为Authority于Hub，其中Authority认为如果一个页面提供了关于某个主题的信息，那么这个页面就是有价值的，这样的页面就是authority页面，因此如果有很多其它的页面都指向该页面，说明这个页面的authority值就高；Hub认为还有一些页面，比如google页面，虽然它本身不提供任何主题信息，但是从这个页面出发，可以跳转到很多有价值的页面上去，这样的页面就会成为hubs，因此如果某个页面指向了很多有authority值高的页面，那么它的hub值就会高。在PIE计算模型中，因为一轮中，每个点既要算本身的authority值，也要算hub值，并且hub值的计算要依赖于周围邻居节点的aythority值，所以无法用PIE模型一轮的IncEval同时算出所有点的authority值与hub值，因此需要拆分成两次IncEval去做，其中一次用于计算所有的Authority值，另一次根据上轮的authority值计算hub值。通常情况下，这类算法会将一次图计算任务分为PIE模型下的n次IncEval计算，这类算法还包括Louvain，MST。

## 运行时间预测

## 消息到达速率预测

## 本章小节

# 基于运行时间预测的图计算系统的设计与实现

本文的研究

## PageRank算法

PageRaEmailRank算法

在网页实验评估

### 实验方案

我们将第三本章小结

# EmailRank算法在实际系统中的应用

通过以上章

## 应用于邮件客户端

现有的邮件客户端如Gmail、Outook、Foxmail W3C会员影响力评价系统实现

### 需求分析

W3C是一个制定网络标准的全球会员组织总体架构设计

W3C会员影响力评价系统主要包括四大模块，系统模块实现

## 本章小结

本章我们介绍了提出的邮件重要性评价方法在实际系统的作用。

总结与展望

## 论文总结

在社交媒体和即时通讯飞速发展的如今，。

## 未来工作展望

本文主要研究了邮件重要性评价问题。

参考文献

1. The Radicati Group. Email-Market-2018-2022 [EB/OL]. https://www.radicati.com/wp /wp-content/uploads/2018/05/Email-Market-2018-2022-Executive-Summary.pdf, 2018-06-11/2018-11-20
2. 《Fast unfolding of communities in large networks》louvain

攻读硕士学位期间取得的学术成果

**硕士期间发表的论文**

1. **Jiang Kun**, Hu chunming, Sun Jie, Shen Qi. Email importance evaluation in mailing list discussions [C]. International Conference on Web Information Systems Engineering. 2018. (CCF推荐会议列表C类，workshop，12页)
2. **Jiang Kun**, Hu chunming.：Characterizing and predicting important emails in mailing list discussions [C]. 北京航空航天大学第十五届研究生学术论坛. 2018.

**硕士期间申请的专利**

[1] 胡春明, **蒋坤**, 何潇安. 一种基于会话网络的邮件重要性评价方法[P]. CN：201811441956.X，2018-11-29.(专利学生第一发明人)

致 谢

在北京

。