**密级： 保密期限：**

xm 拷贝

**硕士学位论文**



**题目：北京邮电大学硕士论文模板**

**学 号：**

**姓 名：**

**专 业：**

**导 师：**

**学 院：**

**年 月 日**

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。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，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在年解密后适用本授权书。

非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

导师签名：日期：\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_

**基于分布式图计算的大规模网络分析系统**

**摘 要**

非常方便的模板哟！

**关键词：**模板 word

**Title**

**ABSTRACT**

With the development of intelligent automobile technology and the support of the Ministry of Education on students' practice activities, the scale of various kinds of smart car competition for students has been gradually expanded.

**KEY WORDS:** Intelligent Car Race; Image Processing Algorithm; Visual Programming

**目 录**

[第一章 绪论 1](#_Toc477870353)

[1.1 本模板发布简介 **错误!未定义书签。**](#_Toc477870354)

[1.1.1 适用范围 **错误!未定义书签。**](#_Toc477870355)

[1.1.2 模板特性 **错误!未定义书签。**](#_Toc477870356)

[1.1.3 使用技巧 **错误!未定义书签。**](#_Toc477870357)

[1.2 作者信息 **错误!未定义书签。**](#_Toc477870358)

[第二章 测试 **错误!未定义书签。**](#_Toc477870359)

[2.1 测试环境 6](#_Toc477870360)

[2.2 测试方法 **错误!未定义书签。**](#_Toc477870361)

[第三章 TEST 17](#_Toc477870362)

[参考文献 25](#_Toc477870363)

[致谢 26](#_Toc477870364)

[攻读学位期间取得的研究成果 27](#_Toc477870365)

1. 绪论
   1. 研究的背景与意义

随着信息技术的发展，互联网上的信息规模出现了爆炸式增长，如今互联网网页接近47亿，而且用户数目也突破了30亿。与此同时，互联网上的服务模式也越来越丰富，信息之间的联系越来越紧密。超连接图（hyperlink）将不同的网页连接起来，知识图谱（knowledge graph）将不同的实体（entity）联系起来，而社交网络（social network）又将不同的用户连接起来。

这些大规模，高度结构化的数据很大程度的反应了真实世界中的关系，蕴藏着巨大的研究和商用价值。因此，相关领域也出现了大量的图分析算法，他们通过计算数据的结构化特征，提取出重要的信息。其中，具有代表性的技术包括排序（ranking）技术， 社区群体（community）分析技术，话题（topic）分析技术等。

近年来，以社交网络为代表的新媒体发展势头迅猛，这些社交网络产生的数据动态性非常之强，并且持续不断的产生着海量的数据。比如Facebook，Twitter，新浪微博的日互动量高达几十亿次，并且这些数据都包含着显著的时间信息，真实的反映着当前世界发生的事。这种包含了时序信息的图被称作动态图，动态图相较于没有时间信息的静态图更有研究价值，比如关注大选的走势，谣言信息在社交媒体的传播等。动态图相较于静态图的优势主要在两个地方，一是获取最新的变化，二是可以反映图的变化过程，可以发现图的演化过程中的一些规律。

现在虽然已经有了增量图计算的方式，但是其算法适用范围受限，并且没有完整的平台系统支持图的存储和分析。因此研究出一套完整的支持大规模动态图存储和分析的系统就非常重要了。

* 1. 国内外研究现状
     1. 图存储

目前基于图的存储主要有三个主流的方向，一个是基于关系型数据库构建图数据库，这种类型主要是基于当前的RDBMS做一些的修改，仍然使用扩展度非常高的PostgreSQL作为与之对应的查询语言。因为关系型数据库有着几十年的工程积累，通过大量的实践积累了相当丰富的设计经验，所以基于这方面的扩展还是有很多支撑的，近些年也一直有论文讨论相关的话题[]。 这一类数据库的一个典型代表就是微软的GraphView。

另一派是以Neo4j为代表的称之为原生的图数据库，这种数据库摒弃了关系型数据库的设计模式。主要特点是查询一个点的边或者是边上的端点时不用再次线性查找或者重新走一遍B+树索引，而是直接用指针指向下一度的物理地址，基于关系的查找使用遍历链表的方式实现。它的双向链表结构在内存足够大或者是有SSD盘辅助的情况下查找性能还是不错的，但是在内存不够的时候，性能上会有稍微的折损，但是总的来说还是一个不错的选择。

还有一派呢，则是以JanusGraph为代表的使用了NoSQL存储的分布式图数据库。目前的产品实现方案主要是在NoSQL数据库（比如HBase， Cassandra）上封装了一层逻辑的图结构，存储和查询分离，目前的性能提升空间还有很大。

三中存储方式各有千秋，也各自有难以克服的缺点。图查询的本质难题是数据高度关联带来的大量的随机访问，传统的关系型数据库很难解决这个问题，因为他们是面向磁盘优化，最大化的利用了磁盘顺序读写的优势，对于图数据的查询却未必适用。

* + 1. 图计算

通用的大规模数据存储分析平台，大部分已经集合了并发执行，作业调度，容错管理等一系列复杂的功能，想要使用的话只要部署了相应的环境，并且调用通用的接口就可以实现大部分的算法的执行。目前比较知名的大规模分布式计算平台主要有Google提出的的MapReduce系统，微软的Cosmos、Apache的Hadoop和Spark。这些通用的计算平台当初的设计目的是为了解决大量共有的海量数据分析过程中遇到的难题，因此设计目标比较宽泛，可以支持多种多样任务的计算。如今，这些有名的计算系统已经被广泛的使用在了工业界和学术界，为信息化产业的进步贡献了很大的力量。

然而通用计算平台的设计目标宽泛，主要是为了解决业界遇到的大部分海量计算任务，并没有明确的区分不同的计算任务之间的差别，导致这些通用的计算平台在某些特定的计算任务上表现的并不是很好，特别是在图计算这个领域，这个问题尤为突出。

* + 1. 动态图处理系统

动态图是具有事件属性的图，静态图相当于是动态图在某一具体时刻上的映射。相较于静态图来说，动态图更能显示现实世界中的种种联系，比如说互联网中的网页，每时每刻都有新的网页链接加入进来。人与人之间的社交关系，也是时时刻刻都产生着新的互动。因此想要更深入准确的显示现实世界的关系，可以及时处理动态图的图计算系统就显得尤为必要了，然而现有的研究往往还在静态图计算系统的研究上，主要是动态图计算难度大，涉及方面广，很难做出一个适用广泛的可用系统来。

近些年以来，时序数据库的出现就是一个重要的标志，人们已经越来越意识到单凭关系型数据库已经很难在这个数据爆炸的时代分析各种大数据任务了。比如自动驾驶环境中汽车遇到的各种场景的收集分析，自动化交易算法持续不变的收集着交易场所的数据，智能空调实时的调整房间的温度。从FaceBook推出了beringei以来，其他的工业界巨头也都纷纷的推出了自己的时序数据库产品，包括百度天工的时序数据库，阿里的HiTSDB等。然而在图数据库方面，仍然没有一个时序图数据库的出现。虽然各大研究机构也都纷纷进行了各自的研究，但是从目前的产品来看，效果并不那么显著。其中比较著名的便是微软亚洲研究院的KineoGraph，KineoGraph从存储到计算，提出了一个方便的动态图计算框架，遗憾的是至今并没有听说在工业界使用。其他比较著名的一些关于动态图的计算还有SpecGraph、InscGraph，不论这些产品是否成功，但是在图计算系统的研究上都贡献了宝贵的经验。

* 1. 本文的研究内容

本文首先提出了使用现有的图数据库加上存储结构的优化和索引技术的加持，实现一个高效的、可用的可以存储动态图的存储系统。动态存储的基础是Neo4j，Neo4j作为一个原生的图数据库，在图查询方面有着得天独厚的优势，相对于传统的关系型数据库在深层次关系的查询方面更是有着几倍甚至十几倍的速度优势。针对动态图的时间属性，可以使用索引技术优化。这样就可以方便快速的得到整个时序图中的某个时间区段快照。

其次设计了既有主体存储与计算副本并存的存储系统。对于主体存储来说，最重要的是实现快速、精准的查询。对于快速查询的需求，主要是从图的底层存储结构，硬盘的加载以及索引技术加持，实现低延迟的查询。对于准确性的要求就需要数据库操作满足原子性、持久性、一致性和容错性等特点，也就是要支持事务，可以实现事务的提交与回滚。对于计算副本来说，最重要的就是要适合多次的迭代计算，可以保持数据与主体存储的一致。在这里主要采用Spark的RDD存储，每次迭代计算RDD都会检测和上次计算的差别，从而决定是否要进行下一轮计算。主题存储和计算副本之间采用Mq系统通信，既可以保证数据的一致性，也可以缓解网络通信的压力。

然后设计了可以用于动态图计算的增量式算法。增量式算法的重要特点是在动态图中每次变化部分的计算都可以借助上次计算的结果，实验表明，在数据的变化量在每次10%-20%的时候，增量式算法相较于非增量式算法在时间效率上大大提升，但是准确率却并没有下降多少。增量式算法的有效性主要体现在增量式算法相较于全量式算法的完整性和收敛性上面。也就是将新增部分对于原有部分上影响使用一个阈值来决定是否进行进一步传播，一旦新增的部分对原有部分的影响小于阈值立即停止传播，从而可以快速的达到收敛。

最后则是基于Neo4j，Spark，Docker，Spring Boot等技术实现了一个完整可用的图计算平台，这个计算平台主要包括存储模块，计算模块，数据导入模块，结果展示模块，状态监控模块等。

* 1. 论文结构安排

本文主要分为六个部分，每个部分按照下列结构进行组织：

第一章，绪论。本章主要有研究背景，国内外研究现状，本文研究内容等部分。研究背景主要介绍了对于本文中研究内容的选题来源以及研究的必要性。国内外研究现状主要描述了国内外的学术界和工业街在这个问题上的最新研究，主要包括图存储系统，图计算系统和动态图计算系统，同时表达了本课题的研究潜力。研究内容着重阐述了本文中的主要研究点以及最后达到的效果，最后的论文组织结构则是对全文的论文组织做了综合概述。

第二章，关键技术。关键结束方面主要介绍了本文中用到的一些前沿研究技术，以及这些技术现在的发展状况。主要包括图存储的的实现方式，分布式图存储和单机图存储，以及分布式图存储涉及到的图分割技术。图计算方面主要包括图计算的基础理论，分布式图计算和单机式图计算。

第三章，动态图存储的研究与实现。本文的主要研究点之一，动态图相对于静态图多了一个时间维度，因此实现起来的难度大大增加。图的动态存储最重要的是可以获取某个时刻整个网络的快照，在这里主要是利用了特殊的数据结构和划分时间区间的存储来实现。通过RabbitMQ系统实现主体存储和计算副本之间的相互通信，保证了主体存储和计算副本之间的一致性。

第四章，分布式图计算系统的研究与实现。本文的主要研究点之一，本文中的分布式图计算系统一个重要的特点就是可以支持增量式计算，也就是支持动态图的分析。本文中的实现方式是对现有的全量式算法进行改造，实现了增量式PageRank，增量式ShortestPath，增量式TrustRank和增量式TunkRank算法。并且在这一章的最后做了大量的实验，验证了增量式算法的有效性。

第五章，第五章是三四章的工程实现。第五章利用三四章的研究成果实现了一个完整的可用的图计算平台。这个图计算平台包括，数据导入模块，图存储模块，图计算模块和结果展示模块，并且包装了可以直接调用的接口。

第六章，总结与展望。这一章主要总结了前几章的工作成果，做了一个全面的总结。并对前几章中未解决的为题提出了思考，作为下一步待解决的问题。

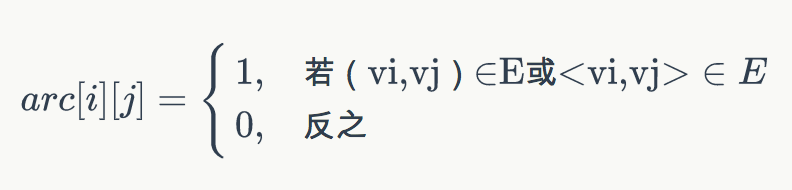
1. 相关技术介绍
   1. 图存储系统的介绍与研究

本章的重点是介绍图存储的发展历程及最新成果。先是是对图存储的数据结构、算法以及工业界的实现做了全面具体的梳理。这些问题为本文中动态图存储系统的设计提供了基本理论和优化方法。后一部分是针对于近些年大数据技术的兴起，图计算在这一方面的发展，详细的阐述了图计算的应用场景。最后是总结了现有图计算和存储方面的成果和缺陷，表名本文研究的必要性。

* + 1. 图存储的实现方式
       1. 基于邻接矩阵的存储结构

图的邻接矩阵（Adjacency Matrix）的存储方式是一种比较浅显易懂的存储方式。主要是利用一个一维数组和一个二维数组，一维数组用来存储图中的顶点信息，二维数组（也就是邻接矩阵）用来存储边之间相互联系的信息。

图主要分为点和边两种主要的结构，点（Node）上主要存储节点的属性，边（Edge）则是建立两个点之间的联系。直接将两种结构合并在一起表示还是有些困难的，因此，人们在已有的数据结构上发掘出了简单的表示图的方式。一个基本的思路是将点和边分开矩阵。一维数组存储点及点的属性，点之间的关系很难由一维数组表示，那就考虑用二维数组来表示。这就是邻接矩阵表示法的诞生。

假设图G是一个有n个顶点的图，则邻接矩阵的表示就是一个n\*n的方针，具体的定义如下：

两个节点有边连接，则arc[i][j] = 1，否则arc[i][j] = 0。邻接矩阵具有以下特点：

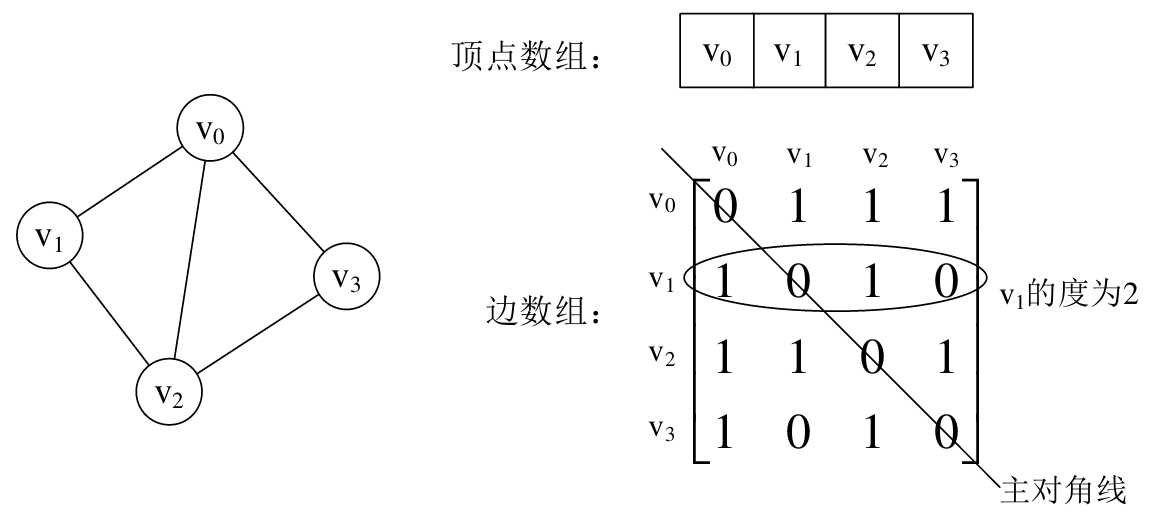
1. 每个图的邻接矩阵表示都是唯一的；
2. 无向图的邻接矩阵是一个对称矩阵。对称矩阵可以进行压缩，压缩的时候可以只存储上三角或者下三角，可以节省存储空间；
3. 对于一个无向图来说，邻接矩阵的第i行（或者是第i列）非零元素的个数正好是第i个顶点的度；
4. 对于一个有向图来说，邻接矩阵的第i行（或者是第i列）非零元素的个数正好是第i个顶点的出度（入度）；

图1-1 无向图的邻接矩阵存储

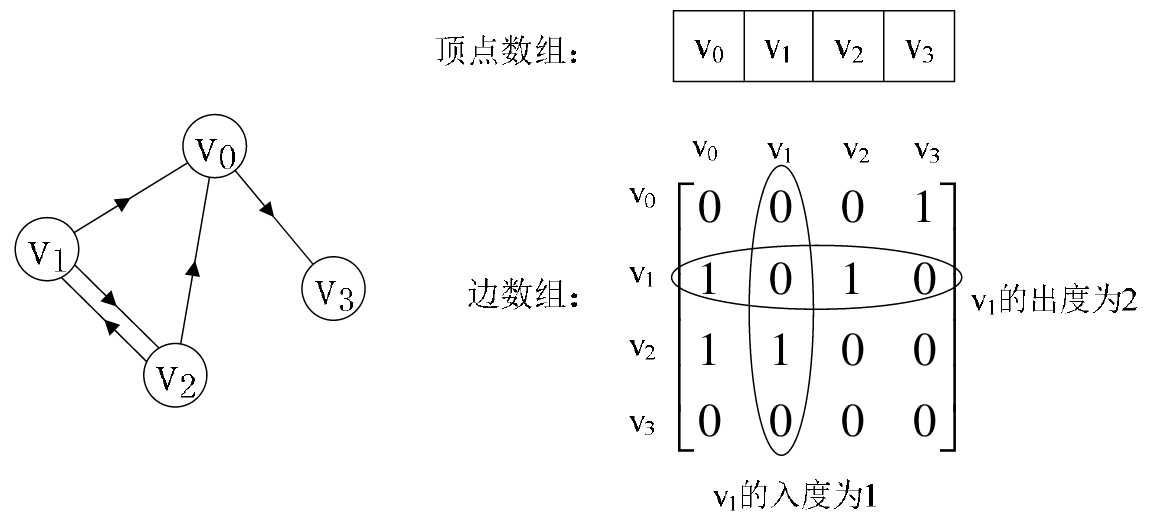
对于一个无向图，如图1-1所示，邻接矩阵的主对角线都为0，因为没有自身到自身的边。v1的度即为第二行所有非零元素的个数。并且该邻接矩阵关于主对角线对称。

图1-2 有向图的邻接矩阵存储

对于一个有向图来说，如图1-2所示，v1的出度为第二行所有非零元素的个数之和，v1的入度为第二列所有非零元素的个数之和。

邻接矩阵是一个简单易懂的存储结构，但是我们也可以很容易的看出，在有n个节点的图中，每个节点都要维护它与另外n-1的节点的关系，这在节点n的数量非常巨大，并且是稀疏图的时候会造成很大的空间浪费。针对这个缺点，又产生了适合存储稀疏矩阵的存储方法，就是下一节要介绍的邻接表。

* + - 1. 基于邻接表的存储结构

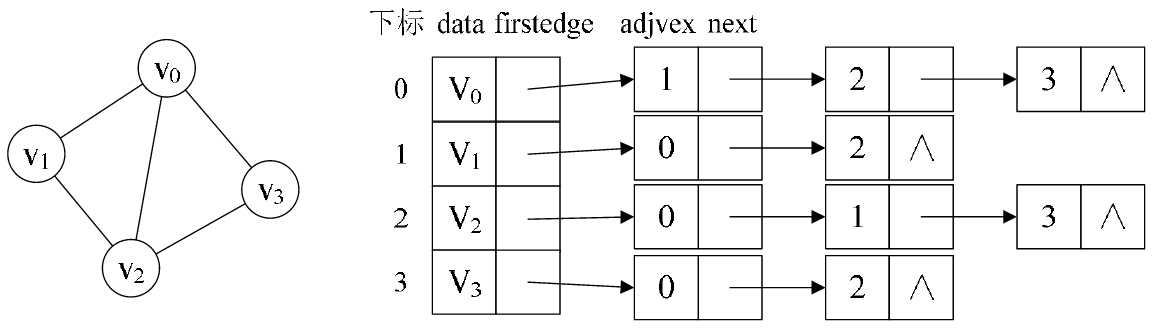
当邻接矩阵存储边比较少的图时，对于存储空间来讲，会造成极大地浪费，因此邻接矩阵不适合存储稀疏图。因为图上每个顶点并不一定和所有的其他顶点都有关系，每个顶点只需维护和他周围节点的关系即可表达整个图。因此可以考虑将每个顶点周围的顶点存储在一个线性表中。整个图就组成了一个数组和链表组成的存储结构，这就是邻接表（Ad-jacency List）的实现方式。

图1-3 无向图的邻接表存储

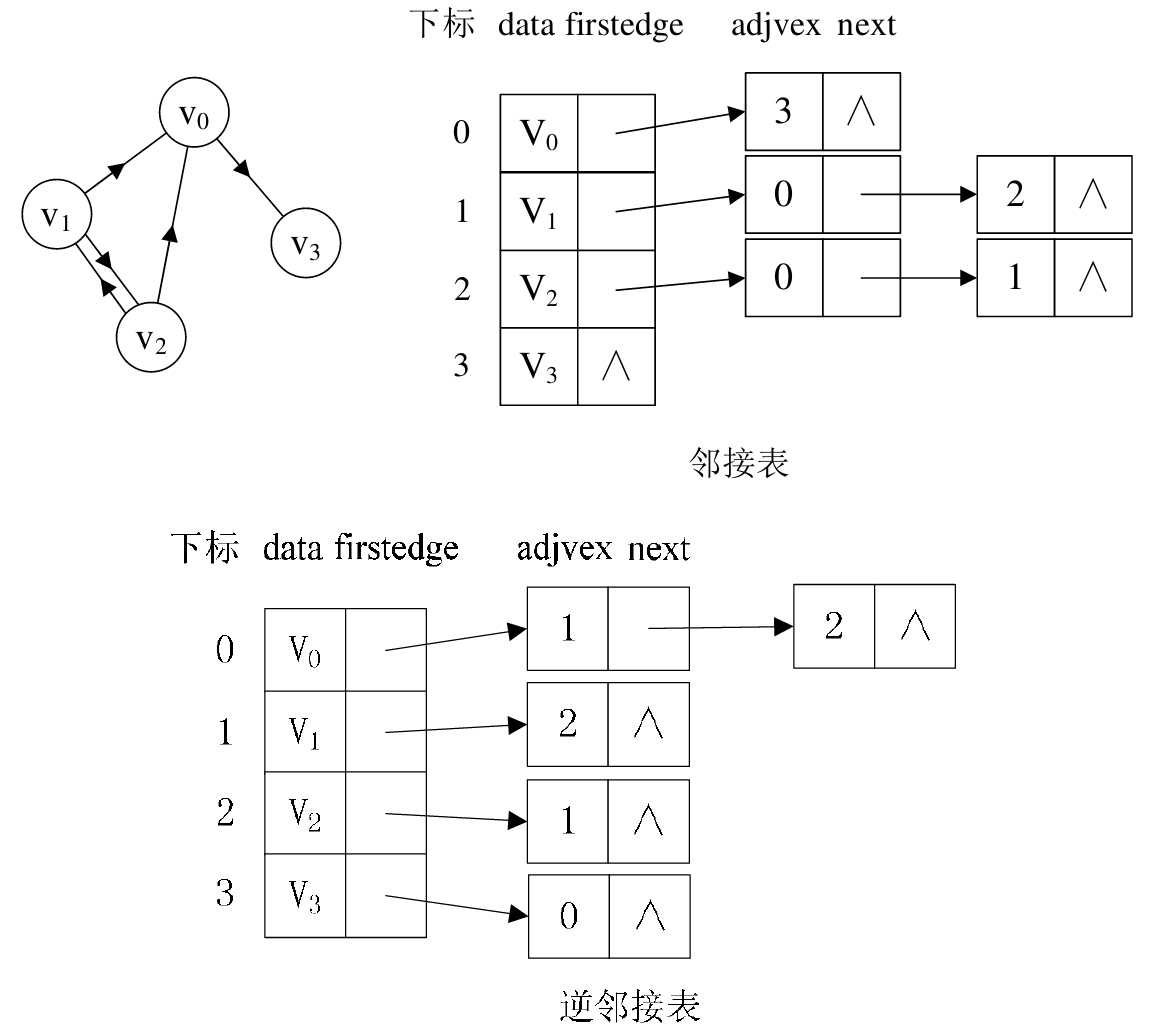
在邻接表中顶点用一维数组或者是链表来表示，不过一般使用一位数组的情况较多，因为使用数组存储可以快速查找。另外每个节点会存储其指向的第一个邻接节点的指针，要想查找该顶点的邻接顶点或者是查找该顶点的边，需要从该节点开始遍历链表。

图1-4 有向图的邻接表存储

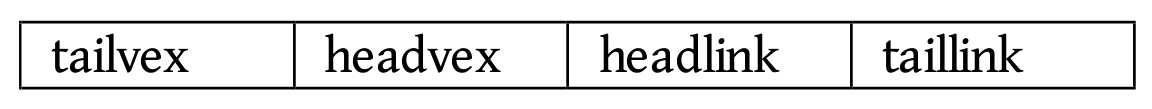
如图1-3所示，无向图的邻接存储主要有几个结构，首先是data域，用来存储节点及其节点数据。Firstedge用来存储指向第一个邻接节点的指针，adjvex是用来存储第一个邻接节点的地址，next用来存储下一个邻接节点的指针。

如图1-4所示，有向图的存储和无向图基本一致，只是无向图中每个链表只能存储单向的关系，如某个节点的出度。这样当想要计算每个节点的入度时就无从下手，因此可以建立有向图的逆邻接表，这样要想查找一个点的入度或者上游节点就可以直接查找逆邻接表。

* + - 1. 基于十字链表的存储结构

对于有向图来说，邻接表的存储结构仅仅能表现出一个方向上的链路。比如建立了邻接表，要想获取一个节点的上游节点就要遍历全表，这造成了极大的不方便。

因此有人提出将邻接表和逆邻接表结合的实现方案，这就是十字链表（Orthogonal List）。十字链表使用类似于双线链表的结构，从一个节点开始，即可以找到它的上游节点也可以找到它的下游节点。

十字链表的顶点和边分别用两种存储机构，顶点的存储结构如图1-5所示，存储边的结构如图1-6所示。

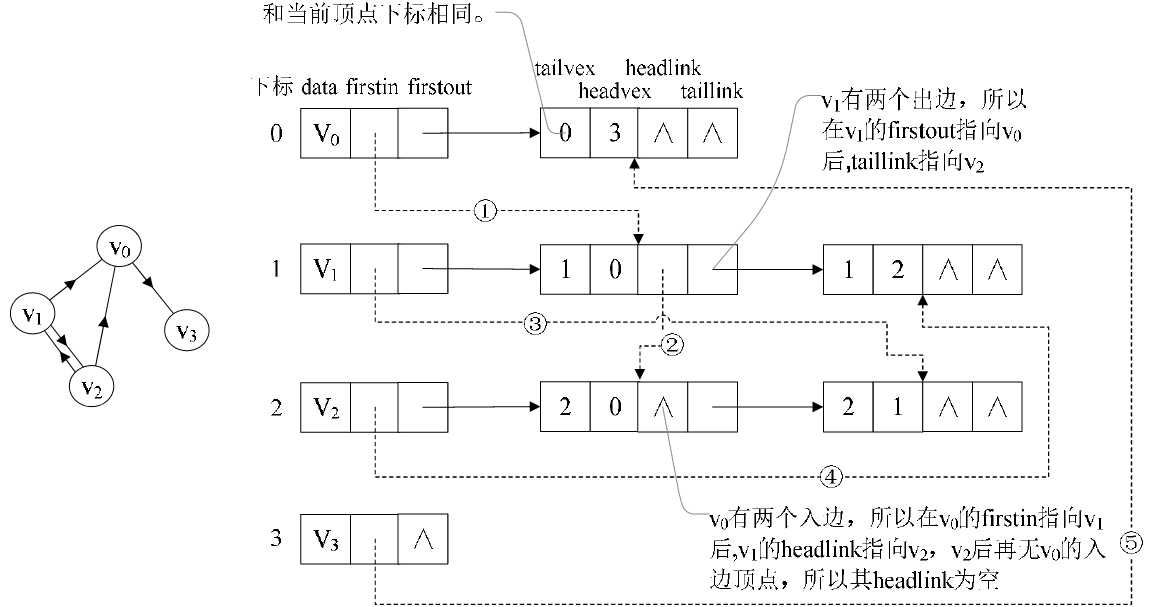
图1-6 十字链表边的存储结构

图1-7 十字链表边的存储结构

十字链表将点和边都以一种特殊的结构存储，并且维护他们之间的指向关系。点的存储结构是data存储节点数据，firstin存储指向第一条入边的指针，firstout存储指向第一条出边的指针。tailvex存储这条边的起始节点下标，headvex存储这条边的结尾节点下标。Headlink存指向下一条入边的指针，taillink存指向下一条出边的指针。

如图1-7所示，要查找v0节点的上游节点即入边起始节点，从firstin的指针入手，首先根据1所示先查找第一条入边，再从这个入边的tailvex域可以得到这条入边的起始节点为v1，同样也可以得到第二个入边起始节点为v2。查找v0的下游节点即出边结尾节点，从firstout指针入手，首先可以找到第一条出边，再从这个出边的headvex可以得到该出边的结尾节点为v3。

十字链表结合了邻接表和逆邻接表的优势，可以方便的查询一个节点的上下游节点。而且又保证了查询速度与邻接表在同一个级别。在本文中我们的主体存储结构也正是采用的这种方式。

* + 1. 单机图存储

现在图研究领域在图存储方面主要分为两个派别，单机式图存储和分布式图存储。两个派别争论的焦点便是究竟图的存储适不适合分布式存储。首先介绍单机式图存储。

* + - 1. 单机图存储的依据

在分布式技术成熟以前，长久以来存储技术一向是单机的。特别是图这种结构，由于其复杂的结构特征，它的存储一直是让人们比较头疼的问题。上一节介绍了图的存储结构，但是不论从哪一种结构上来看，分布式的实现代价都不小。而且要想分布式存储，图的最优分割点是一个NP难的问题，一旦分割的算法实现的不好，所造成的通信干扰非常之大，严重的影响到了图查询和图计算的效率。

因此支持单机式图存储的人们认为把一个复杂网络分割后存储并不是一个最好的选择，随着硬件技术的革新，完全可以通过增加硬件配置的方式完成一个大规模网络的存储，并且可以保持图的原生特性。

* + - 1. 典型实现方式

Neo4j是这派技术的典型代表产品。Neo4j利用类似于十字链表的方式，将点和边分别采用定长字段存储，每个点存储第一条入边和第一条出边的索引，每条边存储它的首尾节点索引和上一条边与下一条边的索引。每当在Neo4j中进行一次关系查询时，都会使用深度优先或者广度优先的方式从查询顶点开始遍历链表，直到找到相应的顶点或者边为止。这样的查询相较于传统的关系型数据库来说查询速度能高到一个数量级。

* + 1. 分布式图存储
       1. 分布式图存储的必要

随着近些年来分布式技术的逐渐稳定，图分割任务在现有的计算能力下看似也不是一个困难的问题了。虽然单机式图存储可以通过堆积硬件来实现，但是近些年数据的扩展规模越来越快，硬件的革新速度根不是数据的爆发速度。因此图的分布式存储还是一个特别紧迫的寻求，特别是在一些海量计算任务上面，分布式存储可以方便的与分布式计算技术结合，将会使处理效率大大提升。

分布式存储的难点是图分割技术，图分割的方式选择将会影响到之后的查询和计算效率。图分割的方式会直接决定之后在其上运行系统的效率。

衡量一个图分割算法好坏主要在三个方面：

1. 通信代价：图计算的主要依据是图遍历，而图遍历会涉及到不同节点之间的通信，加入一个遍历操作的节点分布在多个机器上，不同机器之间的网络通信将会极大地影响计算效率。因为在本机内存上查找的时间可能在纳秒级别，而不同机器间的网络通信会在毫秒级别。
2. 负载均衡：主要是要平均的使用分布式系统上每台机器的计算能力，假如图的分布规模不均匀，那会造成部分机器过载和另一部分机器空载的情况，影响分布式系统计算能力的发挥。
3. 存储冗余：对于一个分布式系统来说，为了保证容错性往往会进行复制以保证高可用。对于分割不好的图也会造成大量的存储冗余，比如分割点在比较重要的节点处。
   * + 1. Hash分割

Hash分割是最简单的一种分割，这种分割不会去考虑通信代价、负载均衡、存储冗余等问题。比较适合图比较小的时候的分割，因为对图分割也是需要时间消耗的。研究表明在图的节点规模在10亿以下时，采用Hash分割的方式完全可以满足大部分的需求，当图的节点规模在10亿以上时，采用复杂的算法分割效果更优。

* + - 1. 启发式算法分割

图的分割问题是经典的NP完全问题，因此大多数的图分割算法都是基于启发式算法来实现的。主要分两种实现方式：一种是启发式的交换点对，另一种是多层次的划分。

启发式方法的代表是KL和FM方法。对于图分割的问题，早在上世纪的八十年代便有了深入的研究，主要是科宁汉（Kernighan）和林（Lin）提出的一种非常有效的启发式算法叫做KL算法。KL算法的主要思想是在每轮的划分中都把一个图随机的进行2等分，然后交换两部分的节点，分别计算交换了点之后得到的收益，再把收益率最高的节点进行再次对换。该算法的最坏时间复杂度为O(n^2)，对于复杂的图分割问题来讲这个时间复杂度是可以忍受的。但是这种方法也有其缺点，他最多只能处理10的4次方以内的节点数量，再高规模的节点数量效果就有些不好了。

为了处理大规模的图切分任务，库玛（Kumar）等人之后提出了图层图划分算法那METIS。METIS算法的核心是先将大图进行一定方法的“粗糙化”，再对粗糙化之后的每个小图执行KL算法，这样的改进成功的克服了KL算法无法支持大规模数据划分的缺点。百万级的图分割问题基本上可以在秒级时间代价内完成。

* 1. 图计算系统的介绍与研究
     1. 图计算基础理论

随着互联网迅速的发展，大规模图分析的需求也越来越紧迫。因此近些年涌现出来了一批图计算系统，这其中具有标志性意义的有Pregel，GraphLab，VENUS和GraphChi等。他们实现了类似于Hadoop和Spark这样的框架，通过简单的更新函数就可以完成大规模图计算。现有的图计算的基本思想是使用“Think like a vertex”去表达图计算的过程，使得庞大的计算任务可以转换成顶点任务就可以完成。

顶点任务程序只用计算更新图中顶点和边的更新状态，然后传播到周围的节点。图计算中有两个重要的计算机制，一个是BSP整体同步计算模型，另一个是GraphLab的异步并行计算模型。

具体的来说就是，图计算系统会预先定义好一个顶点更新函数update()，用户要想使用该系统做自己的计算任务，只需要重载该函数即可。update函数可以使用到节点的信息，也可以使用到邻接边的信息。每次迭代过程就是每个节点都执行自己的update()函数。

* + - 1. BSP算法

BSP（Bulk Synchronous Parallel，整体同步计算模型）是一种新兴的分布式计算模型，是由哈佛大学的Viliant和牛津大学的Bill Coll提出的并行计算模型，最开始被称为“大同步模型”，又因为他们提出这个模型的初衷是改变现有的计算机体系结构，同来作为计算机体系结构和计算机语言之间的沟通桥梁，因此它还有个别名“桥模型”。

BSP由一系列的全局超步计算过程组成（每个全局超步就是一次迭代过程），每个超步主要进行三个任务：

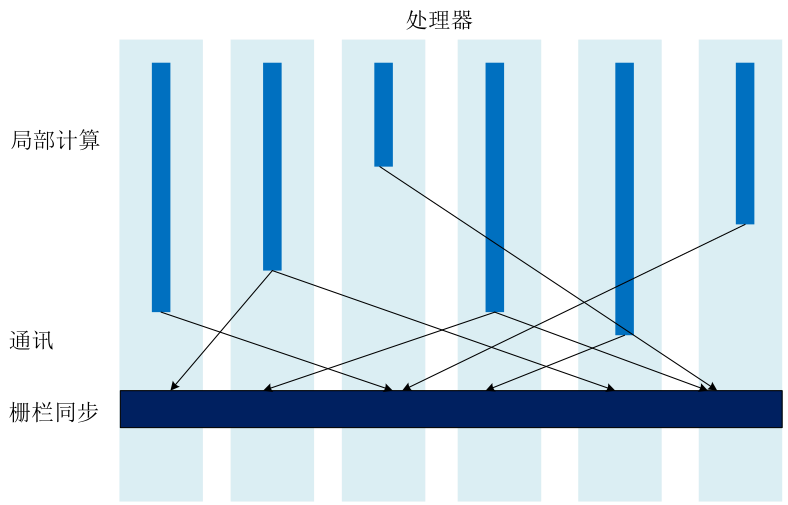
1. 局部计算：每个processer都只计算自身的计算任务，他们只读取存储在本地内存中的值，不同的计算任务之间是相互独立的并且异步执行。
2. 交换信息：每个processor计算完自身的任务之后，会将消息传递到与他相关联的processor。
3. 整体同步：所有的processor都处理完毕并且消息交换完成以后，会进行下一个超步，也就是下一轮迭代。

图2-1 BSP计算模型

* + - 1. Pregel算法

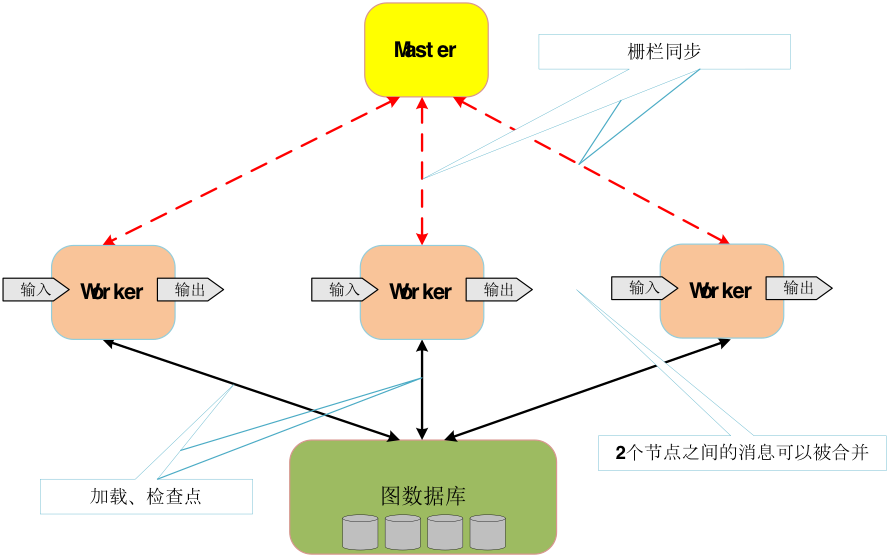
Pregel最早是由Google提出的大规模图计算模型，现在已被大量的新兴图计算框架所采纳。Pregel是基于BSP实现的并行图处理系统，可以运行在多台廉价计算机组成的集群上面。通常一个图计算任务会被分解到多台计算机上同时执行，任务执行的过程中，临时数据一般会存储在本地磁盘，而持久化数据会存储在分布式文件系统。

图2-2 Pregel体系结构

Pregel的体系结构如图2-3所示：

* 集群中多台机器一起执行计算任务，其中一台机器作为Master，其他的机器作为Worker。
* Master负责着管理其他的Worker，与Worker进行通信。每当一个计算任务来临Master将图分为多个分区，并且将分区分配给每个Worker，监控Worker的运行。
* Worker会向Master注册自己的信息。每个Worker管辖自己分区的内存状态，计算状态被保存在自身内存中。每个超步中每个Worker都会遍历自己内存中的节点，并且计算每个节点的update函数。
* Master使用定期发送消息的方式进行容错。Master定期向Worker发送消息，一旦消息在一定的时间内没有的到响应，Master会认为这个计算节点出现了错误，然后会启动恢复模式。

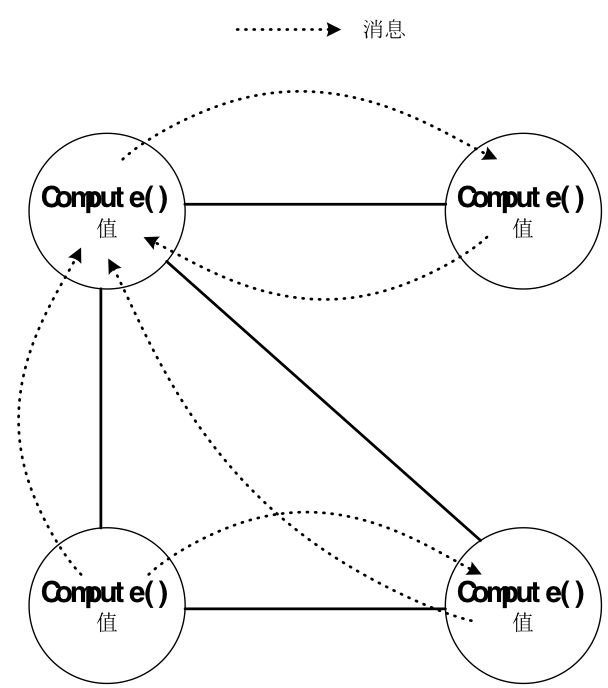
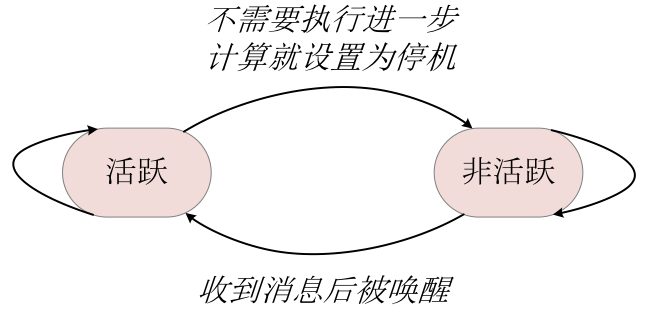
Pregel顶点间的信息传递采用纯消息传递的模式，在使用异步和批量消息传输的方式下，可以极大地提升整个系统的性能。

图2-2 消息传递模式 2-3 状态机图

图2-2是Pregel的消息传递模式图，该函数描述了一个顶点在一个超步S需要进行的操作。该函数读取前一个超步（S-1）发送过来的消息，在这个节点上做聚合，聚合以后修改本身的信息，然后沿着出边发送自身的信息。这些在超步S发送的信息会在（S+1）的超步被别的节点接收。

图2-3是Pregel执行的状态机图，每个收到消息的节点都会被设置为活跃，活跃的节点可以参与本次计算并且发送消息，非活跃的节点如果收到消息就会变为活跃。迭代算法执行到图中所有的节点都变为非活跃，就可以结束。

* + 1. 单机图计算系统

单机图计算系统的代表是华为的VENUS和伯克利大学的GraphChi。VENUS提出了以顶点为中心的流水化图计算模型，这样的做法显著地降低了基于磁盘的图IO，极大地改善了磁盘IO与计算速度不匹配的情况。VENUS使用了独有的基于外存算法的存储扩展方式，叫做分级存储的基于流水线的图计算模型，可以突破单机图计算的可扩展能力，达到了不错的计算效果。

VENUS系统使用了数据分片的方法以及独有的外存模型。在该系统中，每个数据分片被分为v-shard和g-shard两个部分。V-shard用来存储图中的点部分，数量不大，并且需要频繁的修改。G-shard用来存储图中的边数据部分，数据量巨大，但是几乎不发生改变。在图计算的执行过程中，首先磁盘按照顺序读取g-shard分片的数据，每当遇到g-shard所关联到的点的时候，就将v-shard中的点也加载到内存。处理g-shard中边对应的更新函数，此时g-shard关联的点已经在内存中了，可以直接读取计算。一旦计算完成，立马丢弃g-shard中加载过来的边数据，这样可以保证系统在加载数据的同时进行计算，极大地减少了运行的时间。

另一个基于单机的图计算系统是GraphChi，在GraphChi中一个图也会被分割成多个数据片，保证每个数据片都可以放在内存里面。每个数据片内存存储的都是一个子图，包括一系列的顶点和边。图计算过程由多个迭代过程组成，每个迭代过程都会对图中的节点进行更新计算。在GraphChi中，每次迭代系统都会一次的处理一个数据分片，并且在每个数据分片上执行更新函数。这部分的处理主要分为单个具体的步骤。首先是载入，将一个数据分片载入到内存中。然后是更新，在每个数据分片中的每个节点上执行更新操作。最后是写回，将计算好的信息重新写会到磁盘上。这种处理方法的缺点主要有二，一个是在数据加载的过程中将会产生很大的IO，而这部分IO的过程中无法同时进行计算。第二个是每个分片计算完后会传播本身的信息给相邻分片的节点，这样将会产生大量的随机IO。

* + 1. 分布式图计算系统
       1. GraphLab

GraphLab是一个基于内存的分布式图计算系统，在GraphLab中一个图被分割成多个子图，每个子图被存储在一台单独计算机的内存上，图分割采用的是顶点切分的方式。图分割之后，被切分的顶点会在每个与他有关联的自图中都存在一个备份。这样一个顶点就可能会存储在多个主机上，每次一个子图上发生计算，更新过信息的节点就会将计算结果同步到其他有该节点备份的主机上。图上的数据包括图的顶点和边，以及顶点和边上的值。这样做的好处就是不用网络传输图结构信息了，但是同步节点信息仍然会需要大量的网络通信。

* + - 1. Spark GraphX

Spark GraphX是基于Spark平台的一个分布式图计算系统，随着Spark的开源影响力的进一步增加，Spark GraphX也逐渐的被更多的人熟知。

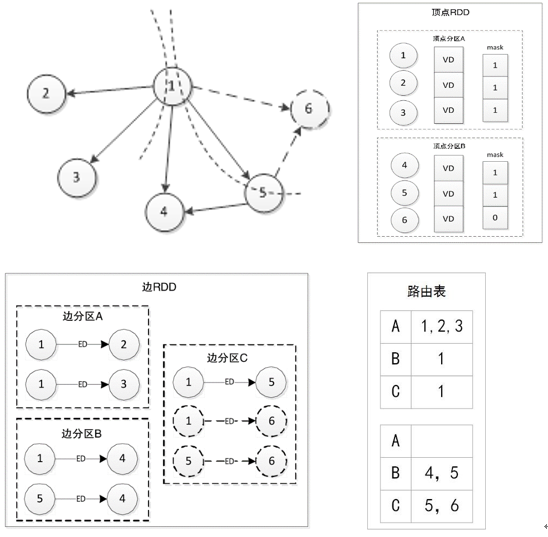
GraphX将图数据以RDD的模式分布式的存储在集群节点上，使用顶点RDD（VertexRDD）和边RDD（EdgeRDD）分别存储顶点的集合和边的集合。顶点RDD一般是按照Hash的方式分区，各个顶点以Hash的方式分布在不同的节点上。边RDD按照特定的分区策略分区（一般是使用边Id的Hash值分区），分布在不同的节点上。此外，顶点RDD还存有顶点指向边分区的路由信息，也叫做路由表。路由表一般和顶点RDD一起存储，他记录的是顶点RDD和边分区的映射关系。当进行计算的时候，顶点RDD会根据路由表信息将顶点信息数据分发到边RDD分区。具体如图2-4所示。

图2-4 SparkGraphX的存储结构

在图计算的过程中，有些边的计算需要需要点数据的支持。比如在计算PageRank的时候会需要计算出边的权值，这是顶点RDD分区要做的就是把顶点数据传送到出边所在的RDD分区。GraphX会根据路由表，生成顶点RDD与边RDD对应的重复顶点视图，因为在进行以顶点划分的分区的时候，同样的顶点会处在不同的边分区中。生成的的这个重复顶点视图与边分区的RDD个数相同，当要进行计算的时候，顶点会与边一起形成三元组。因为顶点的数据相对于边的数据非常少，每次生成重复顶点视图的代价并不大。并且随着迭代次数的增加，每次需要更新的节点数量也会越来越少，这样顶点RDD的更新代价也会越来越小，从而SparkGraphX可以快速的执行。

* 1. 本章小结

本章主要介绍了本课题中相关的关键技术。由于本课题中要研究实现一个图计算系统，涉及到的技术主要分为图的分布式存储和分布式计算。第一节介绍了现有的图存储方式以及他们之间的优劣关系，为下一章文中提出的存储结构提供依据。第二章主要介绍了现有的图计算技术，以及他们各自适用的场景，分析了他们的适用情况以及存在的不足，为本文的图计算系统做出了铺垫。

1. 动态图存储系统的研究与实现
   1. 适用于OLTP的主体存储系统

在图处理领域一个重要的目标是实现图关系的快速查询，现实中的主要场景有在社交网络中查询某个人的朋友，查询某两个人的共同朋友，推荐可能认识的朋友，查询任意两个人怎么样才能快速认识。在科研论文方面查询引用的论文，有共同作者的论文，有共同关键词的论文。互联网网页中一个页面链出的页面，链入的页面等。这些需求的共同点是查询的准确性和快速性，这就要求我们的存储系统可以快速的响应查询，同时还要有一定的安全性。

* + 1. 图的存储结构设计

第二章提到图的存储方式主要有邻接矩阵、邻接表和十字链表等。邻接矩阵存储稀疏图会造成巨大的浪费，单一邻接表无法做到双向遍历，不能满足一些复杂查询，十字链表可以双向遍历但是在范围查询方面表现的并不是很好。在这里我们采用另一种灵活的邻接表存储方式。同时为了快速查询，采用BigTable模型作为存储后端。

* + - 1. 内存中的存储结构设计

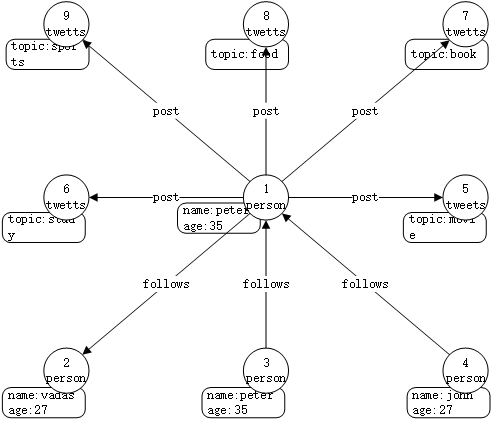
如图3-1所示，一个图可以抽象为节点（vertex）和边（edge）的集合。每个点都有自己的标签（label）和属性（property），每个边也具有自己的标签（label）和属性（property）。

图3-1 BSP计算模型

一个大图由一系列的节点和边组成，具体组成如下所示。

1. 点的集合

* 每个节点都有一个唯一的标识。
* 每个节点都有一个出边的集合。
* 每个节点都有一个入边的集合。
* 每个节点一个由键值对组成的属性的集合。

1. 边的集合
   * 每条边都有一个唯一的标识。
   * 每条边都有一个尾部节点id。
   * 每条边都有一个头部节点id。
   * 每条边都有一个由键值对组成的属性的集合。

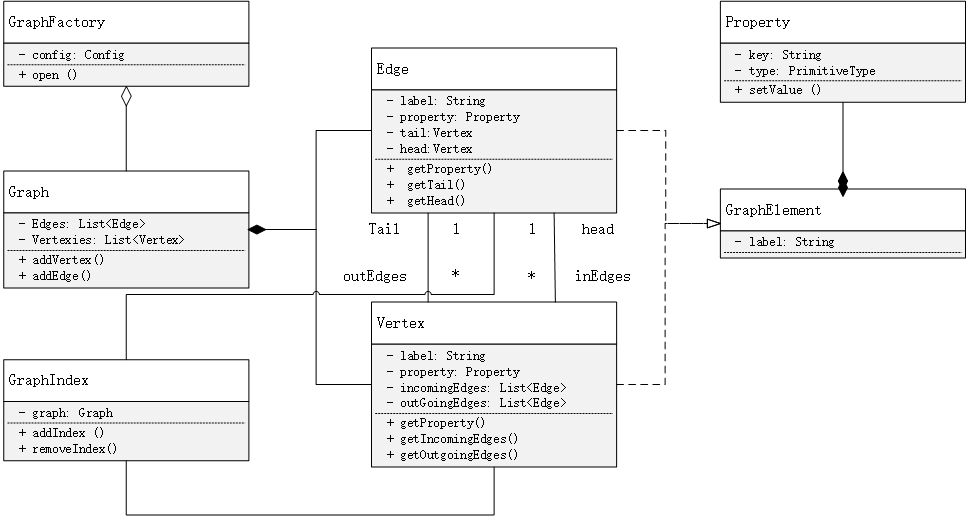
根据以上描述我们设计出几个重要的类，如图3-2所示，主要包括GraphFactory类，用来连接存储后端。Graph类，图上一系列操作的集合。Edge类，边属性和行为的集合。Vertex类，节点属性和行为的集合。Property，节点和边的属性。GraphIndex类，索引行为的集合。

图3-2 BSP计算模型

驱

* + - 1. 后端存储结构设计

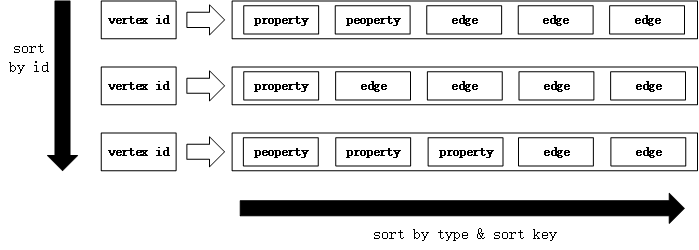
在BagTable的数据模型下，每个表都是行的集合，每一行都由一个唯一主键标识。每行由任意有限大小的单元（cell）组成，每个单元由一个列（column）和值（value）组成。在BagTable存储模型下，每行都支持大量的单元格，并且这些单元格不用像在关系数据库中那样需要预先定义。

图3-2 BSP计算模型

如图3-1所示，我们将每个节点的邻接表作为一行存储在Hbase中。顶点id是该顶点邻接表的行关键字，每条边（包括出边和入边）和属性都作为一个独立的单元存储在行中，这样可以高效的删除和插入。特别是在使用vertex id作为关键字之后，我们可以将该表按照id进行排序，从而在图切分的时候，将经常互相访问的节点存储在同一个分区中，可以极大地加快访问速度。

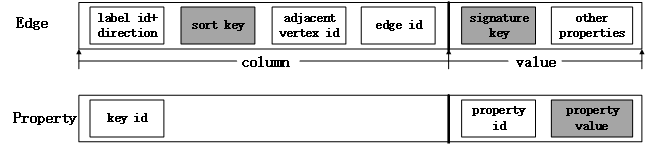
每一行的数据可以按照某个type，或者是按照我们自定义的sort key排序，

图3-2 BSP计算模型

边（Edge）和属性（Property）的存储结构如图3-2所示。每个顶点的邻接边和属性都作为一个单独的单元存储在以这个顶点id为键值的行中。每个单元都被划分为两部分，column和value。在Edge的存储单元中，column是由label id&direction，sort key，adjacent vertex id，edge id等四部分组成。Label id表示边类型，sort key表示排序字段，adjacent vertex id表示邻接节点的id，edge id代表边本身的id。

* + 1. 时序数据的存储
    2. 计算快照的生成
  1. 适用于OLAP的计算副本存储
     1. 顶点分区存储结构
     2. 边分区存储结构
  2. 存储主体与计算副本的同步
  3. 存储体系完备性讨论
  4. 本章小结

1. 分布式的图计算系统的研究与实现
   1. 图算法的并行化实现
   2. 并行化增量式算法的研究与实现
   3. 增量式算法效果评估
   4. 小结
2. 分布式图计算分析平台的设计与实现
   1. 总体设计
      1. 系统架构
   2. 详细设计与实现
   3. 小结
3. Test
   1. Test
   2. TEST

参考文献

1. 张瑞成, 陈至坤, 王福斌. 以智能汽车竞赛推动实验教学改革[J]. 实验室研究与探索, 2010, 29(8):103-105.
2. 赵书朵，周云旭，陈云生. 浅析学科竞赛与创新型人才培养—以大学生智能车竞赛为例[J]. 实验室科学，2011，14(6):4-6.
3. 卓晴，王京春，黄开胜，等. 全国大学生智能汽车竞赛的研究与实践[J]. 中国大学教学，2012(4):74-77.
4. 王瑞荣. 基于事件触发并发数据流模型的可视化编程语言研究[D]. 浙江：浙江大学，2003.
5. 王海鹰，王逢州，王乐柏. 轮式机器人图形化编程系统的研究与设计[J]. 电脑知识与技术，2013，9(27):5968-5969.
6. 刘林森. 智能汽车技术的最新进展[J]. 汽车维修，2005(1):10-11.
7. 谭黎丽. 智能交通系统(ITS)中的智能汽车技术研究[J]. 技术与市场，2016，23(4):120-120.
8. 黄敏学. 当百年汽车遇上智能智能汽车技术新发展[J]. 微型计算机， 2015(7):20-24.
9. 乐德广，郭东辉，刘瑞堂，等. 虚拟仪器结构及其可视化编程的技术进展[J]. 计算机测量与控制，2001，9(1):1-3.
10. 杨力. 基于WebKit的图形化编程软件[D]. 吉林：吉林大学, 2015.
11. 张梅. 面向机器人应用的可视化开发平台的设计与实现[D]. 哈尔滨：哈尔滨工业大学，2015.
12. 肖孟强. 混合噪声图像滤波算法在医学图像中的应用研究[D]. 兰州：兰州交通大学，2012:6-8.
13. 朱红. 数字图像处理基础[M]. 北京：科学出版社，2005.
14. 丁明月，蔡超. 医学图像处理[M]. 北京：高等教育出版社，2009.
15. 袁智. 数字图像混合去噪算法的研究[D]. 湖南：中南大学，2010.
16. 王科俊，熊新炎，任桢. 高效均值滤波算法[J]. 计算机应用研究，2010， 27(2):434-438.
17. Assa A, Janabi-Sharifi F. A Robust Vision-Based Sensor Fusion Approach for Real-Time Pose Estimation[J]. Cybernetics, IEEE Transactions on. 2014, 44(2): 217-227.

致谢

感谢CCTV!

攻读学位期间取得的研究成果