云计算环境下的大规模图数据处理技术

于 戈 谷 峪 鲍玉斌 王志刚

(东北大学信息科学与工程学院 沈阳 110819)

(医学影像计算教育部重点实验室(东北大学) 沈阳 110819)

摘 要 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.文中结合云计算的特点,从图数据管理与图数据处理机制两个方面,综述了云计算环境下进行大规模图数据处理的关键问题,包括图数据的存储方式、图索引结构、图分割策略、图计算模型、消息通信机制、容错管理、可伸缩性、图查询处理等.全面总结了当前的研究现状和进展,详细分析了存在的挑战性问题,并深入探讨了未来的研究方向.

关键词 图处理;云计算;数据管理;分布式计算

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.01753

Large Scale Graph Data Processing on Cloud Computing Environments

YU Ge GU Yu BAO Yu-Bin WANG Zhi-Gang

(*College of Information Science and Engineering*, *Northeastern University*, *Shenyang* 110819)

(*Key Laboratory of Medical Image Computing of Ministry of Education* (*Northeastern University*), *Shenyang* 110819)

Abstract With the rapid growth of emerging applications like social network analysis, semantic Web analysis, and bioinformatics network analysis, it is urgent to require the processing capability on large scale graphs with billions of vertices, which is the hot topic of the research and development in the current high performance computing field. With the features of cloud computing and from the aspects of graph management and graph processing mechanisms, this paper surveys the key issues of large scale graph processing on cloud computing environments, including graph data storage scheme, index structure of graph data, graph partitioning strategy, graph computing model, message communication mechanism, fault-tolerance management, scalability, and graph query processing. This paper summarizes the state-of-art of current research works completely, analyzes the existing challenge problems in detail, and deeply explores the research directions in future.

Keywords graph processing; cloud computing; data management; distributed computing

1 引 言

图是计算机科学中最常用的一类抽象数据结构,在结构和语义方面比线性表和树更为复杂,更具有一般性表示能力.现实世界中的许多应用场景都需要用图结构表示,与图相关的处理和应用几乎无所不在.传统应用如最优运输路线的确定、疾病爆发

收稿日期:2011-08-12;最终修改稿收到日期:2011-09-15.本课题得到国家自然科学基金(61033007,61003058)、中央高校基本科研业务费专项资金(N090104001)资助.于 戈,男,1962 年生,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为数据库理论和技术、分布与并行系统等. E-mail: yuge@mail.neu.edu.cn. 谷 峪,男,1981 年生,博士,副教授,主要研究方向为 RFID、空间数据管理、云计算. 鲍玉斌,男,1968 年生,博士,教授,主要研究领域为海量数据管理等. 王志刚,男,1987 年生,硕士研究生,主要研究方向为数据库系统与云计算.

路径的预测、科技文献的引用关系等;新兴应用如社交网络分析、语义 Web 分析、生物信息网络分析等.

虽然图的应用和处理技术已经发展了很长时间,理论也日趋完善,但是随着信息化时代的到来,各种信息以爆炸模式增长,导致图的规模日益增大,如何对大规模图进行高效处理,成为一个新的挑战.

1.1 大规模图数据处理问题

以互联网和社交网络为例,近十几年来,随着互联网的普及和 Web2.0 技术的推动,网页数量增长迅猛,据 CNNIC 统计,2010 年中国网页规模达到 600 亿,年增长率 78.6%,而基于互联网的社交网络也后来居上,如全球最大的社交网络 Facebook,已有约 7 亿用户,国内如 QQ 空间、人人网等,发展也异常迅猛.

真实世界中实体规模的扩张,导致对应的图数据规模迅速增长,动辄有数十亿个顶点和上万亿条边.本文所指的大规模强调的就是单个图的大规模性,通常包含 10 亿个以上顶点.面对这样大规模的图,对海量数据处理技术提出了巨大挑战.以搜索引擎中常用的 PageRank 计算[1] 为例,一个网页的 PageRank 得分根据网页之间相互的超链接关系计算而得到.将网页用图顶点表示,网页之间的链接关系用有向边表示,按邻接表形式存储 100 亿个图顶点和 600 亿条边,假设每个顶点及出度边的存储空间占 100 字节,那么整个图的存储空间将超过 1 TB.如此大规模的图,对其存储、更新、查找等处理的时间开销和空间开销远远超出了传统集中式图数据管理的承受能力.针对大规模图数据的高效管理,如存储、索引、更新、查找等,已经成为急需解决的问题.

1.2 采用云计算环境处理大规模图的优势

云计算是网格计算、分布式计算、并行计算、效用计算、网络存储、虚拟化等先进计算机技术和网络技术发展融合的产物,具有普遍适用性.云计算技术的发展,一直与大规模数据处理密切相关.因此,依靠云计算环境对大规模图数据进行高效处理,是一个非常有发展潜力的方向,其主要优势表现在:

(1)海量的图数据存储和维护能力.大规模图的数据量可达几百 GB 甚至 PB 级别,难以在传统文件系统或数据库中存储,而云计算环境提供分布式存储模式,可以汇聚成百上千普通计算机的存储能力和计算能力,提供高容量的存储服务,完全能够存放和处理大规模的图数据.云计算环境下的并发控制、一致性维护、数据备份和可靠性等控制策略,可以为大规模图数据的维护提供保障.

(2)强大的分布式并行处理能力.利用云计算分布平行处理的特点,可以将一个大图分割成若干子图,把针对一个大图的处理分割为若干针对子图的处理任务.云计算分布式并行运算能力,能够显著提高对大规模图的处理能力.

(3)良好的可伸缩性和灵活性.从技术角度和经济角度讲,云计算环境具有良好的可伸缩性和灵活性,非常适合处理数据量弹性变化的大规模图问题.云计算环境通常由廉价的普通计算机构成.随着图数据规模的不断增大,可以向云中动态添加节点来扩展存储容量和计算资源,而无需传统并行机模式的巨大投资.

1.3 关键技术挑战

虽然云计算环境对于大规模图数据的管理有诸多优势,但是由于云计算只是一个通用的处理框架,而且其本身也正处于发展阶段,如何在云计算环境下进行大规模图数据处理,仍有很多关键技术难题需要解决.图计算及其分布式并行处理通常涉及复杂的处理过程,需要大量的迭代和数据通信,针对联机事务处理等应用的传统技术很难直接应用到图数据处理中.云计算环境下的大规模图处理主要面临两大挑战:

(1)图计算的强耦合性.在一个图中,数据之间都是相互关联的,图的计算也是相互关联的.图计算的并行算法中对内存的访问表现出很低的局部性[2].对于几乎每一个顶点之间都是连通的图来讲,难以分割成若干完全独立的子图以进行独立的并行处理.并且,“水桶效应”问题加剧,即先完成的任务需要等待后完成的任务,处理速度最慢的任务,将成为整个系统的效率制约瓶颈.为了提高执行效率,需要采取多种优化技术.首先,在预处理阶段,进行合适的图分割时,尽可能地降低子图之间的耦合性;其次,在执行阶段,应选取合适的图计算模型,避免迭代过程中反复启动任务和读写磁盘,降低任务调度开销和 IO 开销;应充分利用迭代过程中的收敛特性进行查询优化,同时进行有效的同步控制和消息通信优化,减少通信开销,以达到降低水桶效应的目的.

(2)云计算节点的低可靠性.大规模图处理,需要相对较长的时间来完成计算任务,如 PageRank 计算需要约 30 次迭代处理,消耗大量的时间和资源.而云计算节点通常是由普通的计算机组成,在这种长时间的处理过程中,个别节点出现故障是难免的.这时,不能简单地重新计算,而应该从断点或者某个合适的位置接续执行.否则,将造成很大的浪

费，甚至一些大型的图计算根本就不能完成. 另一方面，由于图计算并行子任务之间的强耦合性，一个子任务的失败可能导致其它子任务的失败，这又增加了恢复处理的复杂性. 因此，需要考虑有效的容错管理机制，减少大规模图处理过程中的故障恢复开销，尽量避免重复计算，提高大规模图处理的运算效率和稳定性.

为了解决云计算环境下的大规模图处理问题，可从图数据管理和图处理机制两方面加以考虑. 在图数据管理上，需要解决图数据的分割、图数据的存储、图数据索引的建立、图查询处理等问题；在图处理机制上，需要解决处理过程中图计算模型选取、同步控制、消息通信、容错管理和可伸缩性等问题. 本文将针对上述内容，结合云计算的优势和存在的挑战，综述云计算环境下的大规模图处理现有技术的进展、解决方案以及今后的发展趋势和研究方向.

2　图数据模型与存储管理

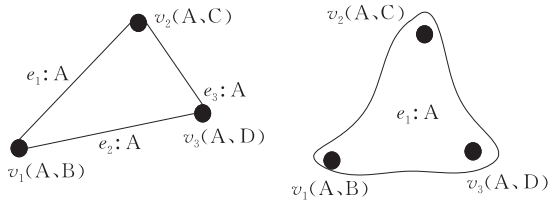
图数据的逻辑表达形式和物理存储结构是实现图处理的基础. 本节首先介绍图数据模型，然后，介绍图的存储管理以及为了提高查找效率而为图数据建立的索引结构.

2.1　图数据模型

作为数学的一个重要分支，图论以图作为研究对象，在简单图的基础上衍生出超图理论、极图理论、拓扑图论等，使图可以从多方面表达现实世界. 当前大规模图数据管理，采用的数据模型有多种，按照图中节点的复杂程度分为简单节点图模型和复杂节点图模型[3]；按照一条边可以连接的顶点数目分为简单图模型和超图模型. 不论是简单图模型、超图模型、简单节点模型还是复杂节点模型，它们的顶点和边都可以带有属性. 下面介绍简单图模型和超图模型，其它模型请参考文献[4].

(1) 简单图模型. 这里所说的简单图，并不是图论中的简单图，是相对于超图而言的. 简单图中，一条边只能连接两个顶点允许存在环路. 简单图的存储和处理都比较容易，对于一般的应用，简单图的表达能力完全可以胜任，如 PageRank 计算、最短路径查询等. Pregel、Hama 等系统均采用简单图模型来组织存储和处理大规模图数据[5-6].

(2) 超图模型. 一条边可以连接任意数目的图顶点. 此模型中图的边称为超边. 基于这种特点，超图比上述简单图的适用性更强，保留的信息更多. 例如，以图顶点代表文章，每条边代表两个顶点(文章)享有同一个作者. 现有 3 篇文章 V_1(作者 A、B)、V_2(作者 A、C)、V_3(作者 A、D)，3 篇文章的作者都有 A. 图 1(a)表示了简单图存储模式，3 条独立的边 $e_1, e_2, e_3 = \{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}$，无法直接保留作者 A 同时是 3 篇文章 V_1、V_2、V_3 的作者这一信息. 图 1(b)代表了超图存储模式，超边 $e_1 = \{v_1, v_2, v_3\}$ 直接保留了 A 是 3 篇文章 V_1、V_2、V_3 的作者这一信息. 对于具有复杂联系的应用，可以使用超图模型建模，例如社交网络、生物信息网络等. Trinity 等图数据库系统支持超图模型来管理大规模图数据[7].



(a) 简单图　　(b) 超图

图 1　简单图和超图

2.2　图数据的存储方式

在目前的大规模图数据管理应用中，主要采用简单图和超图两种数据模型，二者的组织存储格式略有不同. 这两种模型都可以处理有向图和无向图，默认情况是有向图，而无向图中的边可以看作是两条有向边，即有向图的一种. 在之后的讨论中不再强调图中边的方向.

简单图模型的常用存储结构包括邻接矩阵、邻接表、十字链表和邻接多重表等多种方式. 从大规模图处理的应用需求和维护的复杂程度考虑，邻接矩阵和邻接表是最常用的两种结构. 采用邻接矩阵表示图的拓扑结构，直观简洁，便于快速查找顶点之间的关系，但是邻接矩阵的存储代价高昂，对于大规模图数据，这个问题尤为严重. GBASE 系统以邻接矩阵的形式组织存储图，考虑到邻接矩阵的存储开销，GBASE 对矩阵进行了聚簇分割，尽量将矩阵中的非零值集中存储并采用 Zip 技术压缩编码，减少矩阵的存储代价[8]. 与邻接矩阵相比，邻接表的应用范围更加广泛. 像 PageRank 计算、最短路径计算等应用，并不需要频繁查找两个图顶点之间的连通性，邻接表完全可以满足计算需求. 邻接表的存储开销小，逻辑简单，便于分割处理，是一种比较理想的图组织方式，Pregel、Hama 和 HaLoop 等系统均采用邻接表的形式组织图数据[5-6,9]. 超图模型的组织方式主要使用关系矩阵[10]. 从形式上讲，关系矩阵和邻接

矩阵较为相似，但是矩阵的行和列分别表示图顶点编号和超边的编号.

大规模的图数据存储需要依赖云计算环境的分布式存储系统. 云计算环境的存储系统分为两种：一种是以 GFS[11]、HDFS[12] 为代表的分布式文件系统，对于邻接矩阵、邻接表等结构，可以直接存放；另一种是以 BigTable[13]、Hbase[12] 为代表的 NoSQL (Not Only SQL)分布式数据库.

NoSQL 数据库采用的数据模型主要有文档存储(Document Store)模型、列族存储(Column Family Store)模型、Key-Value 存储模型、图存储模型等几大类[14]. 文档存储模型在存储格式方面十分灵活，比较适合存储系统日志等非结构化数据，CouchDB 和 MongoDB 是采用这种存储模型的典型系统[15-16]. 但是，文档存储模型不太适合以邻接矩阵或邻接表组织的图数据. 此外，文档存储模型为支持灵活性所导致的处理效率的降低也会成为大规模图数据管理的性能瓶颈. 列族存储模型比较适合对某一列进行随机查询处理，但是对于穷举式遍历，反而不如传统的面向行的存储模式. 采用该存储模型的典型系统有 BigTable、Hbase、Cassandra 等[2,13,17]. 图存储模型的相关研究目前还不完善，只有少数分布式图数据库，如 Neo4j[18] 等采用这种模型存储图数据.

与上述 3 种存储模型相比，Key-Value 存储模型较为适合存储大规模图数据. Key-Value 存储模型的存储模式简单，支持海量数据存储和高并发查询操作，非常适合通过主键进行查询或遍历，但对复杂的条件查询支持度不佳. 采用该模型的典型系统有 Dynamo 和 SimpleDB[19-20]. 从图处理的角度出发，像 PageRank 计算等，并不需要复杂查询，Key-Value 模型完全可以胜任. 若图数据采用邻接表组织，可以将图的源顶点作为 Key，将源顶点值、出边及边信息作为 Value. 文献[21]结合语义 Web 和传统的 Key-Value 存储模型，提出 Key-Key-Value 存储模型. 以社交网络为例，Key-Key-Value 模型将 *Alice* 和 *Bob* 之间的好友关系组织为一个三元组 $\langle Alice, Bob, FriendShip\rangle$. 该模型存储的信息比传统的 Key-Value 模型更加丰富，可以据此进行数据迁移和合并，以提高空间局部性，使得在查询处理时能减少远程读取数据的次数，因而可以提高数据读取效率.

此外，对于分布式图数据库，当图数据更新时，需要提供事务功能，解决在分布式环境下的一致性控制问题. HyperGraphDB 和 Trinity 等人都宣称自己支持事务机制和一致性控制[7,22]. 如果图数据存储在 HDFS 类型的分布式文件系统上，因其不支持更新和随机插入操作[12]，也就不存在一致性维护问题.

上面讨论了 NoSQL 数据库的 4 种主要存储模型. 文献[23]从管理数据的规模和模型的复杂性两个维度比较了这 4 种基本存储模型，见图 2.

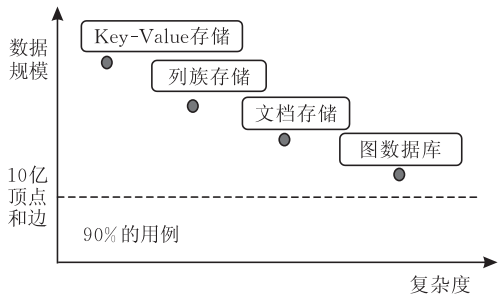


图 2 NoSQL 数据库的 4 种主要存储模型的比较

从图 2 可以看出 Key-Value 存储的复杂性最低，存储数据的规模可以很高，而基于图存储模型的图数据库的复杂性最高，且存储图数据的规模较低，不过也可以管理 10 亿个以上的顶点及其对应的边.

2.3 图数据的索引结构

索引是传统关系数据库中的关键技术，包括 B+树索引、Hash 索引、位图索引等，技术较为成熟，可以提高数据查询处理效率，尤其是在查询结果的数据量远小于原始数据的情况下. 对于一个大规模图，云计算的分布式并行处理机制，可以根据查询条件遍历所有的子图数据，如果查询结果的数据量较大，这种处理方式的性能是比较好的，但是如果查询结果的数量很小，则会访问很多无用的数据，造成计算资源的浪费和查询效率的低下，而通过建立合适的索引，可以有效解决这一问题.

在云计算环境下，大规模图的原始数据保留在分布式存储系统上，建立的索引也必然是分布式的. 如果图的原始数据规模很大，那么它的索引文件也会很大. 另外，分布式环境和数据更新的延迟，也加剧了索引维护的难度. 因此，云计算环境下的图数据的索引，无论是存储还是维护都是十分棘手的问题.

从使用目的和实际效果的角度，索引可分为两大类：一种是为支持普通查询而在云计算环境下建立索引，有助于提高数据查找效率，主要在分布式图数据库中使用；另一种是为加快计算处理而建立的索引，主要在图的计算处理应用中使用，如最短路径计算、PageRank 计算、聚类分析等.

目前用于云计算环境下的索引技术，很少有专

门针对图数据的.但是,这些索引技术大都是可以被图数据存储所利用.目前的云环境下用于数据管理的索引结构可以分为适用于 P2P 网络结构的索引[24-26]以及适用于 Shared-nothing 集群结构的索引[27-28].文献[24]针对云计算环境下的大规模数据查询处理,提出了二级索引技术 CG-index.它首先在每一个数据分片上建立本地 B+tree 形成索引分片,然后将计算节点组织成 Overlay 结构,接下来基于 Overlay 的路由协议把各个计算节点上 B+tree 分片发布到 Overlay 上,建立全局索引 CG-index.这种索引具有自适应性和可扩展性.文献[25]建立了多维索引机制 RT-CAN,集成 CAN 协议和 R 树的特点,在云计算环境下提供高效查询服务.文献[26]也针对 P2P 环境,在分布式 KD-tree 基础上提出多维索引 MIDAS,用以支持多维查询、范围查询和 k 最近邻查询等应用.文献[27]提出了一个通用的、灵活的、容错的、且可扩展的分布式 B-tree 索引结构.文献[28]将 R-tree 和 KD-tree 结合起来组织数据记录,提出了用于云数据管理的多维索引 EMINC,可以提供快速的查询处理和有效的索引维护.

云计算环境下的通用索引机制,没有考虑图结构的特点,在图查询处理方面,效果不明显.而分布式图数据库,无论是数据存储还是索引结构,都针对图数据进行了优化.Neo4j 的索引分为两类[18]:数据库本身就是一个树形结构的索引,可用于提高查询效率,此外,还可以使用独立的 Lucene 索引,提供全文索引和索引命中率排序功能.Neo4j 可以对图顶点和边分别建立索引,通过对索引的缓存,可进一步加快查找速度.索引的维护操作(如删除、更新)则必须在事务管理机制下进行.对于更新,必须先删除旧的索引值,然后才能添加新索引值,代价较高.InfiniteGraph[29]与 Neo4j 类似,也提供内建索引和 Lucene 索引.在 HyperGraphDB 的两层存储架构[22]中,索引是存储层必备的组成部分,一个 Key 可对应多个已排序的 Value,并支持 Value 共享;在模型层,Key 采用 UUID 编号并排序,在 Key 上建立索引.索引文件以 B-tree 格式存储并具有缓存功能,可以为查询等操作提供持久化的元数据信息.Trinity 数据库提供 Trie 树和 Hash 两种索引结构来访问图顶点、边的名字以及相关的其它信息,可减少有公共前缀的字符串的匹配次数[7].

支持图计算处理的索引,主要是在云计算平台中体现.文献[30]针对 Hadoop 系统中的 Shuffle 过程和 Reduce 过程进行了改进,采用动态增量式 Hash 索引和缓存技术降低 Shuffle 过程的磁盘 IO 代价,提高 CPU 利用率.HaLoop 则对 Map 任务的输入、Reduce 任务的输入和输出进行缓存并建立索引,在一定程度上提高了 Hadoop 迭代处理的效率[9].文献[31]针对 MapReduce 的连接操作,提出 Trojan 索引技术,通过对分片信息和 Key 值的重新构造,建立索引,使需要连接的数据位于同一个分片上,减少连接操作的网络通信量.由于 MapReduce 模型是一个通用计算模型,Hadoop 处理平台的索引也是通用性的,对于大规模图的迭代处理的针对性并不强.

3　图数据的分割策略

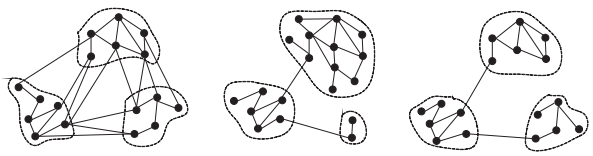
在云计算环境下,对于一个大规模图的处理,必须进行分布式并行处理.由于图数据本身固有的连通性和图计算表现出强耦合性的特点,为了实现高效的并行处理,尽可能降低分布式处理的各子图之间的耦合度是非常重要的.有效的图分割就是实现解耦的重要手段.这首先要将一个逻辑上完整的大图分割成若干部分,分别放置到分布式存储系统的各工作节点上.其后对图的处理,就是针对已经分布式存放的每一个子图,启动一个计算任务,进行相同的处理操作,当所有子图处理结束,则完成整个大图的一次处理了.

3.1　图分割原则

虽然并行处理可以提高效率,但是由于子图之间的连通性,在任务执行过程中或执行结束时,各任务之间需要进行消息通信,这个通信处理的代价很大,是制约图处理效率的瓶颈.如果在大图存储或处理之前,利用良好的图分割算法,将一个大图分割成若干个适当大小的子图,且子图内部具有较高的连通性,而子图之间的连通性较低,那么相当一部分消息就不需要通过网络跨节点传输,而是直接本地处理,可以极大地减少通信代价,提高整个系统的运行效率.

在云计算环境下,实现大图分割并取得较好的分割效果,是一项挑战性很大的工作.将一个大图分割为若干子图,有两个主要原则:一是提高子图内部的连通性,降低子图之间的连通性,这种特点尤其适合云计算的分布式并行处理机制;二是考虑子图规模的均衡性,尽量保证各子图的数据规模均衡,不要出现较大的偏斜,从数据规模方面防止各并行任务的执行时间相差过大,降低任务同步控制过程中“水

桶效应”的影响.当然,大图分割的时间复杂度必须控制在可以忍受的范围内.图 3 对比了 3 种不同的图分割效果,如果单纯考虑子图的数据均衡或子图之间的连通性,其效果均不理想,只有同时考虑着两个因素,才能显著提高分布并行式处理效率.



(a) 没考虑子图关联性 (b) 没考虑子图均衡性 (c) 理想情况

图 3 3 种图分割效果

3.2 单指标分割技术

如果只考虑数据负载均衡这一单项指标,最简单的图分割技术,就是 Hash 方式,即在设定了分片数目之后,对图顶点 ID 进行 Hash,将数据划分成给定数目的分片.这种分割方法效率很高,时间复杂度为 $O(n)$,可以在图数据的载入过程中或图处理之前完成分片操作.在一定条件下,设计良好的 Hash 方式可以避免数据偏斜,使各子图的数据规模接近相同.但是 Hash 方式没有考虑图数据的局部性,甚至连原始数据的局部性也无法得到保留.Hash 方式虽然在云计算环境下容易实现,但是,负责各子图处理的任务之间消息通信频繁,会造成较大的网络通信开销.

如果只考虑子图内敛性这一单项指标,即增大子图内部的关联性,降低子图之间的关联性,可采用聚类技术,效果十分明显.关于在云计算环境下实现分布式聚类的方法,在 Apache 开源项目 Mahout 中有详细介绍[32].但是,聚类操作一般都是一个迭代处理的过程,时间开销不容忽视.另外,聚类技术一般不考虑各聚簇之间的数据规模偏斜问题,很可能导致分割后的各子图的数据规模相差较大,增大了图处理过程中的“水桶效应”.在改善子图分割的时间复杂度方面,Yahoo 研究院发现,即便图数据的原始规模很大,最终得到的聚簇仍然要小得多[33].基于这一发现,Yahoo 研究院开发出“Local Partition”算法[34],该算法的运行时间与最终输出结果的聚簇大小成正比,而与图的原始输入数据规模无关,从而可以对更大规模的图进行分割处理.

此外,还有很多研究者从其它方面对分布式环境下的图分割技术进行了探索.文献[35]采用随机森林(SF)方法,进行图分割.文献[36]提出了确定性和非确定性的分布式图分割算法 Sync_Part、Fast_Part 和 Elect_Part.文献[37]在使用 Hadoop 进行图的迭代处理时,通过设置 Partitioner 函数来不断调整图的分布.将一个 Reduce 任务处理的图数据视为一个子图,通过 Map 阶段和 Reduce 阶段的 Shuffle 处理,使连通性较强的图顶点分布到同一个 Reduce 任务并作为输出结果存储为一个单独的文件,在下一个 Map 阶段中加以利用,以减少通信量.以 PageRank 应用为例,可以针对顶点 URL 的内容,设计 Hash 函数,使得关联较大的图顶点能够分配到同一个 Reduce 任务中.后两种方法虽然实现了子图之间的有效解耦,但由于子图的个数不确定和大小不均匀,导致并行处理的负载不平衡.

3.3 多指标分割技术

同时考虑子图数据规模均衡和子图内敛性等多项指标,也有很多研究者进行了尝试.文献[38]针对分布式 P2P 网络,提出了一种基于图顶点度序列和广度优先搜索的 k-图分割技术,能够将一个大型 P2P 网络分割成 k 个子网络并且能够做到各子网络的任务负载均衡.文献[39]通过 3 步处理来实现大规模图的分割:(1)建立带权重的深度优先搜索树;(2)将大图分割成若干个均衡的子图;(3)迭代处理,尽量减少子图之间的关联.GBASE 系统[8]利用现有的 METIS、Disco 等划分算法,对存储图数据的邻接矩阵进行聚类,将行和列重新排序,把一个大矩阵聚集为多个均匀区域,形成分块,保证块内的子图联系紧密,块间联系松散,将若干个块作为一个网格,分给一个任务进行处理,在一定程度上解决了数据均衡问题.

Kernighan-Lin 算法[40]既考虑了聚类技术的特点,同时又可以保证分割后的子图在数据规模上的均衡性,主要用于网络节点的分割.其主要思想是首先将一个网络图分割成两个大小相等的顶点集合 A 和 B,在集合 A 和 B 中的顶点,除了和本集合内的顶点有边连接外,还可能和另外一个集合内的顶点有边连接.对于后者,用 T 表示所有这种连接边的权重之和,作为衡量集合 A 和 B 之间连通性的指标.K-L 算法在执行过程中,不断调整集合 A 和 B 内的顶点,直到 T 值最小.但是,K-L 算法的时间复杂度过高,是 $O(n^2\log n)$,其中 n 是图顶点的数量,随着图顶点数据规模的增大,将超出目前的计算处理能力.为降低时间复杂度.文献[41]提出基于 Quick_Cut 技术的 K-L 算法,利用“邻居搜索”的特点,避免了不必要图顶点的遍历,达到时间复杂度为 $O(\max(ed, e\log n))$,其中,e 是边的数量,d 是图顶点的最大出度值.

此外，Horton 系统[42]中提出，对于需要长期存储的图数据，采用静态处理和动态处理相结合的技术实现图分割. 在图数据载入分布式存储系统的过程中，采用静态处理方法，使用“Evolving Sets”技术[43]，将大图分割存储. 新的图数据加入或更新时，采用动态方法，使用基于消息通信的 Affinity propagation 算法[44]对已有的子图进行增量式分割和维护. 这种分割技术既能降低子图计算之间的耦合性，又能保证负载平衡.

4　图计算模型与典型系统结构

本节介绍典型的云计算环境下大规模图数据处理的计算模型和一些典型的图处理系统. 计算模型决定了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.

4.1　MapReduce 模型和 BSP 模型

在云计算环境中，最广泛使用的就是 MapReduce 模型[45]. 一个并行处理作业由多个 map 任务和多个 reduce 任务组成. 作业的执行分为 Map 阶段和 Reduce 阶段. 在 Map 阶段，每个 map 任务对分配给它的数据（通常是本地的数据）进行计算，然后按照 map 的输出 *key* 值将结果数据映射到对应的 reduce 任务中；在 Reduce 阶段，每个 reduce 任务对接收到的数据做进一步聚集处理，得到输出结果. 数据通常保存在分布式文件系统中，如 HDFS.

BSP（Bulk Synchronous Parallel model）模型是 2010 年图灵奖得主 Valiant 在 1990 年提出来的一种基于消息通信的并行执行模型[46]. 一个 BSP 作业由若干个顺序执行的超步（super step）组成：S_1，S_2，…，S_n，对应于 n 次迭代处理. 并行任务按照超步组织，在超步 S_i 内，各任务异步接受来自 S_{i-1} 的消息，执行本地计算并发送消息给下一个超步 S_{i+1}. 在超步之间，通过显式地同步控制，确保所有任务均已完成超步 S_i 的工作. 这种同步方式可避免死锁和数据竞争问题.

在云环境下实现大规模图的处理，主要采用这两种模型，下面将对比它们在图处理方面的特点.

在执行机制方面，MapReduce 是一个数据流模型，每个任务只是对输入数据进行处理，产生的输出数据作为另一个任务的输入数据，并行任务之间独立地进行，串行任务之间以磁盘和数据复制作为交换介质和接口. 而 BSP 是一个状态模型，各个子任务在本地的子图数据上进行计算、通信、修改图的状态等操作. 并行任务之间通过消息通信交流中间计算结果，不需要像 MapReduce 那样对全体数据进行复制.

在迭代处理方面，MapReduce 模型理论上需要连续启动若干作业才可以完成图的迭代处理，相邻作业之间通过分布式文件系统交换全部数据. BSP 模型仅需启动一个作业，利用多个超步就可以完成迭代处理，两次迭代之间通过消息传递中间计算结果. 由于减少了作业启动、调度开销和磁盘存取开销，BSP 模型的迭代执行效率较高.

在数据分割方面，基于 BSP 的图处理模型，需要对加载后的图数据进行一次再分布的过程，以确定消息通信时的路由地址. 例如，各任务并行加载数据过程中，根据一定的映射策略，将读入的数据重新分发到对应的计算任务上（通常是存放在内存中），既有磁盘 IO 又有网络通信，开销很大. 但是一个 BSP 作业仅需一次数据分割，在之后的迭代计算过程中除了消息通信之外，不再需要进行数据的迁移. 而基于 MapReduce 的图处理模型，一般情况下，不需要专门的数据分割处理. 但是 Map 阶段和 Reduce 阶段存在中间结果的 Shuffle 过程，增加了磁盘 IO 和网络通信开销.

MapReduce 模型的商业化应用已经开始推广，其良好的可伸缩性和容错管理能力受到了业界推崇，在大规模数据处理方面的表现也值得称赞. 但是作为通用计算模型，在图处理方面，连续的作业调度和任务分配，代价较高，对于图拓扑结构信息的反复磁盘读取，尤其是从分布式文件系统上读取，也增大了 IO 开销. 此外，在迭代处理方面，需要用户编程控制，较为繁琐. 相对而言，BSP 模型是一个比较适合迭代处理的计算模型，为用户提供了简单易用的编程接口，Googel 的 Pregel[5]、Yahoo! 的 Giraph[47] 和开源的 Hama 系统[6]，都是基于 BSP 模型开发的.

从原理上讲，BSP 模型避免了 MapReduce 模型在多次迭代时的数据反复迁移和作业连续调度，其特有的超步和全局同步机制，使迭代处理的控制更加灵活，在大规模图处理方面很有开发前景. 但目前上述系统还处于研究开发阶段，所处理的数据放置于内存，未考虑索引问题，数据处理规模也受到极大的制约，需要进一步开发基于磁盘的系统并对 I/O 操作进行优化. 此外，BSP 模型中各任务之间的消息通信也是难以消除的效率瓶颈，而在容错管理等方面，尚无完善的理论和方法.

4.2 典型系统结构

目前,关于云计算环境下的大规模图数据管理系统,大致可以分为 3 类:基于 MapReduce 模型的分布式并行处理系统、基于 BSP 模型的分布式并行处理系统和分布式图数据库系统.

基于 MapReduce 模型的分布式并行处理系统,大部分是通用处理平台,如 Hadoop 以及改进版本 HOP 系统[48],可以应用于各种大规模数据处理,为了适应需要多次迭代的图处理应用,很多研究者对 Hadoop 原有处理平台进行了优化改进,如 HaLoop、Twister、Prlter[9,49-50],HaLoop 使用缓存、索引技术来减少不必要的磁盘 IO,改进原有的任务调度模块使连续作业的调度和迭代条件的控制变得较为容易,具备一定的实用价值. Twister 系统对 Hadoop 进行了较大的改动,全部处理数据驻留内存,采用第 3 方消息通信机制,使用任务池来避免多次作业调度. 但是驻留内存的限制使其难以实用,目前只是供研究使用. Prlter 是在 Hadoop 和 HOP 基础上开发的,支持带优先级的迭代计算,可以确保迭代处理的快速收敛,尤其适合在线查询,如 top-k 查询.

基于 BSP 模型的分布式并行处理系统,最著名的就是 Google 提出的 Pregel 平台. Pregel 对于图的分割、计算处理、消息通信优化、同步控制和容错管理都提出了可行的解决方案,是目前较为完善的专门针对大规模图处理应用的系统[5]. Hadoop 的开发商 Yahoo!提出了开源项目 Giraph. Giraph 可以视为在 Hadoop 平台上运行的一个大规模图算法库,在原有 MapReduce 模型基础上,只启动 map 任务,在 map 任务里面参考 Pregel 的设计,嵌套了 BSP 模型,实现多次循环迭代,以支持大规模图处理应用[47]. 开源项目 Hama 同 Pregel 一样,也是一个独立的分布式并行处理系统,适合需要多次迭代的图处理. 但是 Hama 目前很不完善,尚无可稳定运行的发布版本[6].

无论是 MapReduce 模型还是 BSP 模型,上述提及的处理平台都是分布式并行处理系统,它们的优势是完成复杂的图处理任务,如 PageRank 计算、最短路径查询、社交网络分析和图挖掘等,但是对于图数据的一般性存储、更新维护等,则不如分布式图数据库系统.

分布式图数据库系统集数据存储、维护、查询于一体,继承了传统数据库的事务、一致性控制等特点,有的甚至支持较为复杂的管理. HyperGraphDB 是一种基于 Key-Value 模型的分布式 P2P 数据库,采用超图作为数据模型,利用 UUID 技术在分布式环境下实现 Key 编号的唯一,支持海量图数据的高速存储;在查询方面,依靠索引的帮助支持快速图遍历和集合查询,而基于 SPARQL 语言的模式查询正在开发中[22]. Trinity 是微软研究院开发的基于内存的分布式图数据库系统,该系统采用超图作为数据模型,支持满足 ACI 特性的事务机制、一致性控制和索引,能满足高并发查询请求. Trinity 提供良好的图分割算法,以减少查询时的网络延迟,支持同步、异步两种模式的批处理计算[7]. 其它著名的分布式图数据库系统还有 Neo4j 和 InfiniteGraph 等[18-29].

此外,也有很多研究团队开发了自己独特的图处理平台和图管理系统. 如微软针对云计算环境开发的 Dryad、DryadLINQ 分布式执行引擎[51-52],提供完善的输入输出、任务调度、容错管理机制,支持 SQL 查询;Orleans 处理平台[53]支持异步消息通信和索引,采用新的消息机制,避免 RPC 通信的应答阻塞,采用随机分配方法实现负载均衡,支持任务迁移;Horton[42]则支持对大规模图的在线查询优化. GBASE 系统是一个可伸缩的通用图数据管理系统,具有完整的图数据分块、压缩、索引和存储机制以及一系列能够支撑复杂图挖掘应用的原语操作. GBASE 底层采用邻接矩阵存储图数据,所有的图处理操作最终都转化为 Hadoop 作业执行[8].

5 图数据处理的执行机制

本节介绍实现云计算环境下大规模图数据处理的基本执行机制,包括消息通信、同步控制、容错管理,并讨论可伸缩性问题. 其中消息通信和同步控制是针对图计算强耦合性进行优化处理的重要内容,容错管理旨在解决可靠性方面的挑战,可伸缩性是云计算灵活性的重要体现.

5.1 消息通信

在图处理应用中,每一个图顶点都需要向邻居节点发送消息或从邻居节点接收消息,而图的边,可以理解为消息收发的通道. 对于一般的图而言,边的数目要远大于图顶点的数目. 当一个图的顶点数达到百亿级别后,边的数据规模更为巨大,如此大规模的消息通信,如果处理不当,很容易成为整个图处理过程的瓶颈.

图处理的消息,主要产生在图顶点的计算过程中. 但是消息发送方式,则可以根据不同的通信策略

分为异步式和集中式.

对于异步式通信，图顶点的计算处理与消息通信并发执行，在计算过程中就可以发送消息，将大规模消息的发送分散在不同的时间段，避免瞬时网络通信阻塞，但是接收端需要额外空间，存储临时接收到的消息，相当于用空间换取时间. 目前，Pregel、HOP 系统等采用异步通信方式[5,48].

对于集中式通信，图顶点的计算处理与消息通信串行进行，在计算完毕后，统一发送消息，控制和实现方式简单，可在发送端对消息进行最大程度优化，但容易造成瞬间的网络通信阻塞以及增加发送端的消息存储开销.

鉴于大规模图数据处理过程中的网络通信瓶颈，需要对通信次数和通信的数据量加以优化控制以降低耦合代价. 利用图分割，可以降低子图之间的连通性，使大部分消息的目的图顶点均位于同一个任务的处理范围中，将网络通信变为本地通信，从根本上减少任务之间的消息发送；针对具体应用，采用消息合并机制，也可以减少网络通信量和存储量，如 Pregel[5] 和 Hadoop 系统[12]；此外，通过消息缓存和批量发送机制，可以减少网络通信的次数，降低通信链接的维护开销. 至于消息通信的实现方式，Hadoop、Hama 和 Giraph 等采用基于 Http 协议的 RPC 通信机制[6,12,47]. 作为 Hadoop 的改进版本，Twister 系统直接使用第 3 方消息通信管理插件来完成通信控制，Twister 系统目前支持 NaradaBroking 和 ActiveMQ 等基于发布—订阅架构的通信插件[49].

5.2　同步控制

同步控制是所有分布式计算处理框架都必须面对的问题，只不过有的框架显式地提供同步控制，如采用 BSP 模型的 Pregel 系统、Hama 系统[5-6]；有的处理框架提供隐式的同步过程，如采用 MapReduce 模型的 Hadoop 系统，在 Map 阶段和 Reduce 阶段存在隐式的同步控制. 如果使用 MapReduce 模型进行大规模图迭代处理，相邻作业之间也存在同步控制的过程. 在需要多次迭代的图处理应用中，同步控制还应该提供图中间状态信息统计查询功能和收敛条件判断功能. 同步控制的优化可以减少图计算强耦合性带来的影响.

目前，同步控制的设计方案有两种：主从式控制和分散式协同控制. 前者由主控节点统一协调各任务的同步，完成收敛条件判断以及中间状态信息统计查询功能，便于集中管理，结构清晰，可维护性好，不容易产生死锁. 但是当数据量较大、任务数量很多时，主控节点会成为处理瓶颈，多作业并发运行以及图处理应用的多次迭代，更加剧了这种瓶颈效应. 后者的同步过程由各任务自己协调，无主控节点，避免了单点处理瓶颈，可伸缩性很好. 但是不便于集中管理，一旦各任务开始运行，就难以在迭代过程中加以人工控制，灵活性差.

在同步控制中，由于任务处理速度不一致，当各任务负责处理的数据规模或数据内部的复杂程度不同时，会导致任务处理速度相差很大，因此造成了水桶效应. 为降低水桶效应，Hadoop 系统采用“任务推测式执行方式”[12]，希望“纠正”执行缓慢的任务，降低 Map 阶段和 Reduce 阶段的水桶效应. 文献[30]提出动态增量 Hash 技术来弱化 Map 阶段和 Reduce 阶段之间的同步，实现计算过程的部分重叠，减少 Reduce 任务等待时间. 另外，在图处理应用中，传统 Hadoop 平台难以解决相邻作业之间的水桶效应，HOP 系统的“pipeline”技术，可以在一定程度上缓解该问题[48].

5.3　容错管理

对于一个大规模图的处理，任务的执行时间会很长. 而云计算平台通常由普通廉价计算机构成，故障率很高，在大规模图处理过程中，出现不可预知的故障导致作业无法继续运行，是十分常见的现象. 对于图处理这种需要多次迭代的应用，如果每次作业失败，都重新启动，会导致昂贵的重复处理代价，甚至作业根本无法正常结束.

在云计算领域，当前容错管理的主流设计思想是通过硬盘读写和冗余备份来提供保障. 容错管理需要考虑的内容主要包括：冗余备份的写入时机、冗余备份的存放位置、故障侦测、故障恢复等. 其中故障的侦测，目前均是采用“心跳”报告的方法完成.

针对图处理一般是多次迭代的特点，备份的写入时机应该在两次相邻迭代之间，但这又提出了备份生成频率的问题：迭代多少次进行一次备份是合适的. 较高的备份频率会导致作业运行速度缓慢，较低的备份频率又会导致故障恢复时重复处理的代价增高. 目前对于这个问题，并没有定论. 一种可能的解决方案是统计特定云计算节点的故障率，根据不同图处理作业的迭代步数来动态设定备份频率.

借鉴 HOP 的容错思想[48]，可以在一个 map 任务的中间增加备份同时记录原始数据的处理偏移量，当故障发生后，重启的 map 任务直接从偏移量处开始计算. 也可以在一次图处理迭代过程中，设置

中间备份并记录处理偏移量,这可以减少故障恢复时的重复处理,提高恢复效率,但是增大了备份生成频率和磁盘开销,也增大了容错管理的复杂度.

故障恢复时,如果各并行任务之间完全独立,则重启故障任务即可,Hadoop 系统就采用这种恢复策略[12].在图处理过程中,可以直接利用 Hadoop 系统自身的容错机制,但是由于 Hadoop 的容错是以 MapReduce 作业为单位,而一个迭代的图处理作业一般需要多个 MapReduce 作业,多个 MapReduce 作业间的容错管理就不是 Hadoop 所能解决的了,需要用户自己编码实现,较为繁琐.如果各并行任务之间耦合度很高,如基于 BSP 模型开发的 Pregel 系统,就需要使所有任务回归到同一个检查点.作为改进,Pregel 提出"检查点加消息记录"的容错管理方案[5],将图顶点状态备份后,还需要将每个超步内各任务间收发的消息写入磁盘.在故障发生时,仅需恢复故障任务,不必全部回滚,减少因任务耦合度过高导致的高昂的恢复代价,但是对消息的记录增大了磁盘存储开销,在一定程度上也影响了作业的正常运行效率.

5.4 可伸缩性

云计算环境下的可伸缩性,应该从两个方面考虑:硬件方面,即动态添加、删除节点来实现云平台处理能力的伸缩性;软件方面,系统处理框架应该尽量避免单点处理瓶颈.

从硬件方面考虑,应该允许在运行期间,动态添加物理机器以扩充整个云计算平台的可用资源.云计算平台可用资源的伸缩性,一方面是指新提交的作业可以利用新添加的资源,其实现比较容易,不同云计算系统的实现方式较为统一,都是通过注册方式将新机器添加到可用工作节点集合;另一方面也包括正在运行的作业可以利用新添加的资源,不同的处理框架对其实现方式是不同的,而且对于大规模图处理应用,更有意义.假设目前正在运行一个大规模图处理作业,由于云平台处理资源的限制导致运行缓慢,此时考虑动态添加一批工作节点,如果正在运行的作业能够利用新添加的计算资源,就可以加快处理速度.

Hadoop 系统中,由于任务之间是完全独立的,通过"任务推测式执行"技术[12],可以轻松利用新加入资源.但是新启动的任务必须从头开始运行,除非原任务所在的物理机器负载很重导致运行速度极其缓慢,否则新启动任务的完成时间通常晚于正在运行的任务.因而,导致这种任务的"推测式执行",在很多情况下是一种资源的浪费,并不适用.Pregel 系统和 Hama 系统目前还不支持正在处理的作业可以利用新添加的计算资源.由于 BSP 模型以超步实现大规模图数据的迭代处理,每个超步中,各任务耦合度很高,所以不能像 MapReduce 模型那样,通过"任务推测式执行"来利用新资源.对此,一种可能的方案是"任务迁移",通过计算任务的迁移代价,决定是否将导致整个作业处理缓慢的任务迁移到新工作节点上运行.

从数据存储能力方面考虑,基于 BSP 模型的图处理框架,具有较高的内存资源要求,最理想方法是将所有的图数据都驻留在内存中,这样不需要进行内外存交换,否则计算速度将显著下降.但这提高了对硬件配置的要求,在一定程度上也制约了数据处理的规模.基于 MapReduce 的图数据处理系统,只要计算的中间结果能够存储在磁盘上,系统就可以运行,而对节点的配置没有过高的要求.

从理论上讲,云计算环境的伸缩能力应该是没有上限的,即加入的物理机器越多,平台中可用资源越多,处理性能越好.但是,从实际来看,并不是这样的.以 Hadoop 为例,Yahoo 发现,当计算节点的规模达到 4000 台时,Hadoop 系统遭遇到伸缩性壁垒[54],新加入的计算资源不能被云平台充分利用.造成这种问题的根源,是由于目前的云计算环境主要依赖于主从式控制模式,存在单点处理瓶颈,当整个云平台规模过大,主控节点的处理能力成为提高系统性能的制约瓶颈.

6 图查询处理

图数据管理的最终目的是支持查询处理,这里的查询是指广义的查询,既包括简单的查询,如好友关系查询,也包括复杂的图计算和图挖掘,如 PageRank 计算、社交网络分析等.从查询语义的角度考虑,将大规模图查询分为两大类:一类是基本的图查询计算,如特定图顶点或边的查询、好友关系查询等;另一类是复杂查询计算与图挖掘,如 RWR(Random Walk with Restart)计算、子图挖掘等.

对于一个大规模图,由于数据的海量性,必须考虑查询处理的效率,采用云计算环境作为处理平台,就是希望利用分布式并行处理机制提高效率.此外,还可以根据不同类型的查询,设计有针对性的查询优化策略,本节将围绕查询语义,着重介绍两类图查询处理的研究进展.

6.1　基本的图查询计算

图的简单查询，一般不需要多次迭代，用户可以对大规模图进行查找，查询自己感兴趣的信息. 查找过程中，对于某些应用，通过建立合适的索引、调整查询顺序和查询复用等技术，可以避免对整个图顶点的遍历，有效提高查找效率. 从所处理的查询请求和优化技术方面考虑，此类图查询类似于普通的数据库查询. 传统集中式数据库系统，不仅为用户提供了良好的 SQL 查询语言接口，还通过索引组织和查询优化，提供高效的查询服务. 云计算环境下，对于大规模图的简单查询，在考虑分布式环境和图结构特点的同时，也应该尽量提供类似的功能. 这里简单介绍 Horton[42] 和 HyperGraphDB[22]. Horton 是微软正在设计的一款专门针对大规模图数据提供高效在线查询服务的系统. Horton 为减少查询时的网络通信代价，对图数据进行专门分割，以提高局部性. 从查询语言和处理机制方面，Horton 将用户的具体查询请求分解为若干原语，采用有限自动机方法，确定底层的广度优先搜索的执行顺序. 至于查询优化方面，在具体应用中，如好友关系和照片关注关系，研究发现，以不同的顺序执行有限自动机，查询代价相差很大. 如图 4 所示的查询，如果按照图中边的序号进行广度优先搜索，则方案 1 的代价（即处理步骤数）为 8，而方案 2 的代价为 4，优于方案 1. Horton 采用基于图顶点出度统计信息的预处理优化技术，能够以较低的预处理代价获得较优的自动机执行顺序. 在高并发环境下，Horton 还可以复用不同查询请求的原语. 此外，Horton 可以根据查询的历史统计信息，在常用查询顶点之间建立“衍生边”并持久化存储，以空间换时间，可以减少后续查询过程中图顶点的遍历开销. 如图 4 所示，对李四朋友关注的照片建立“衍生边”，在方案 3 中，可以直接定位找到查询结果，代价比方案 2 更优. HypergraphDB 是一个分布式图数据库系统，支持超图模型，底层以 Key-Value 格式存储，可以建立 B-tree 索引，为用户提供查询原语接口以表达具体的查询请求，如图顶点和边条件限定查询、比较操作、连接操作等，可以通过广度优先搜索或深度优先搜索返回符合条件的信息集合. 总体来看，图数据库所支持的查询处理都比较简单，无法完成类似最短路径计算等基本图计算功能（但微软的 Trinity 系统集成了图数据库和图处理平台功能，可以完成高级图处理应用[7]）.

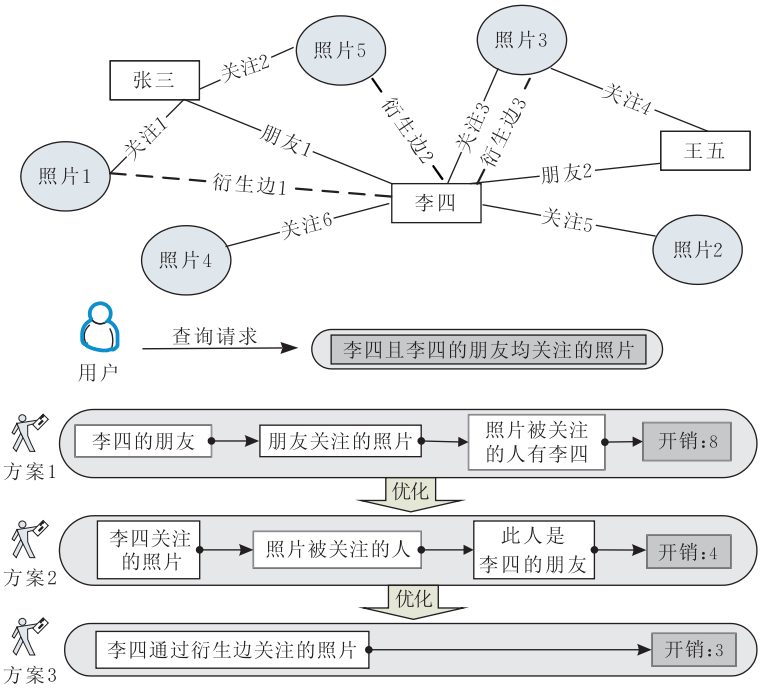


图 4　Horton 系统查询处理示意图

关于图应用中的基本计算，Pregel、Giraph 和 Hama 等系统或图算法库，有较大的发展前景，它们均具备一定的通用性. 目前，基于 Pregel 和 Giraph 图计算处理，需要用户重写系统提供的图顶点基类来表达自己的查询处理请求. 对于具体的图计算应用，可以进行针对性的优化，获得较好的性能指标. 以最短路径计算为例，可以利用预计算的索引来加快响应时间，如文献[55]提出的 HEPV 和文献[56]

提出的 HiTi 就可以视为多级索引，文献[57]则提出使用 2-HOP 索引为每个顶点分配两个顶点标签集合，通过标签集合的交集来实现两个图顶点之间的最短路径查询. 文献[58]对云计算环境下，基于邻居隐私保护的最短路径计算问题，提出了采用贪心算法将原始大图分为本地存储的连接图 G_l 和位于云端的外包图集合 G_0（outsourced graphs），以 1-neighborhood-d-radius 模型保证邻居隐私，通过 G_l 和 G_0 联合对外提供最短路径查询服务.

6.2 复杂查询与图挖掘

相比于基本图计算，复杂图计算与挖掘属于高级图处理应用，如模式匹配、RWR 计算等. Pregel 系统给出了对于 Bipartite Matching 和 Semi-Clustering 算法的 BSP 模型实现方式[5]. PEGASUS 算法库提供了大量基于 Hadoop 的图挖掘算法[59]. PEGASUS 系统使用 GIM-V（Generalized Iterative Matrix-Vector multiplication）原语算子，将图挖掘的迭代处理过程转换为图邻接矩阵的迭代相乘. 以大图中连通域查找为例，PEGASUS 提出了一个新算法 HCC：对于图中的每一个顶点 v_i，都维护一个分组 id，即 c_i^h，它是在 h 跳内能够到达 v_i 的最小顶点 id. 初始时，v_i 的 c_i^h 被设为它自己的顶点 id，即 $c_i^0 = i$. 在每次迭代过程中，每个顶点都将它自己的 c_i^h 发送给它的邻居. 然后，顶点 v_i 的下一步的分组 id，即 c_i^{h+1}，就被设为它当前分组 id 与它接收到的所有邻居发送来的分组 id 中的最小值. 而最关键的一步就是，这种在邻居之间的通信过程可以用 GIM-V 的形式来表达：$c_i^{h+1} = M \times_G c^h$. 通过反复迭代这个过程，在一个分组中的顶点的分组 id 值就会收敛顶点 id 值. 迭代地进行这种乘法，直到分组 id 收敛. 由于实际应用中的图一般具有较小的直径，因此，HCC 算法通常可以较少的迭代次数结束.

目前比较通用的图查询处理系统，主要包括基于 BSP 模型的 Pregel 和基于 Hadoop 的 GBASE. Pregel 对于最短路径计算、PageRank 计算、聚类等操作都可以支持. 但是，Pregel 仅支持稀疏图，且全部数据驻留内存，需要进一步的改进以支持更复杂的应用. 在 PEGASUS 算法库基础上开发的 GBASE 是一个功能比较强大的图数据管理系统，所支持的查询处理也比较全，将所有的大图查询分为两类：全局查询（global query）和定向查询（targeted query）. 前者需要遍历整个图结构，如 PEGASUS 算法库中包含的 RWR、连通域查询等应用. 后者仅需要访问图的一部分，例如 GBASE 目前支持 neighborhoods、induced subgraphs、egonets、K-core 和 cross-edges 等多种不同的定向查询. GBASE 以邻接矩阵存储图数据，将所有的查询处理转化为矩阵迭代操作，底层计算由 Hadoop 完成. 当然对于大规模图，邻接矩阵过大的存储开销和分割存放的复杂性，对于某些以图顶点为核心的应用，如 PageRank 计算而言，是有些得不偿失的.

从大规模图应用的角度看，目前各种图处理系统，尚不能很好地支持子图挖掘、图模式匹配等功能. 在图的模式匹配查询方面，针对 RDF 开发的 SPARQL，可以完成 Triple Pattern、Optional Pattern 等查询[60]，具有很高的参考价值，Hypergraph-DB 正计划采用该语言实现对超图的模式查询[22].

6.3 图查询处理的实现模式

虽然图查询处理的种类很多，但在云计算的分布式环境下，支持图查询处理的底层图遍历操作主要有两种驱动模式：一种是以图顶点驱动的主动遍历模式；另一种是以消息驱动的被动遍历模式. 这两种遍历模式仅是驱动不同，但是遍历操作的对象仍为图顶点，即处理操作的核心始终是图顶点.

在云计算环境下，以图顶点驱动的主动遍历计算模式，需要每个任务在计算时，主动遍历其所负责的全部图顶点，对于每一个图顶点，进行计算处理和消息收发. 这种处理模式适用性强，可以支持所有的图应用场景，Hama 系统就采用这种遍历模式. 但是对于某些特定的应用，可能造成资源浪费. 以 PageRank 计算为例，图顶点状态的更新依赖于消息的接收，即只有邻居顶点所存储的 PageRank 值发生变化时，其出度顶点的 PageRank 值才可能变化. 如果某顶点没有接收的消息，那么它的值就不会更新，也不会对其邻居发送消息，此时可以将该顶点视为达到稳定状态，如果所有图顶点均达到稳定状态，那么 PageRank 的计算任务就完成了. 由于 PageRank 计算是一个收敛算法，随着迭代处理的进行，越来越多的图顶点将达到稳定状态，消息发送量越来越少. 如果以图顶点为驱动，相当一部分图顶点的遍历和处理是没有实际意义的. 作为改进，在以消息驱动的被动遍历模式中，各任务仅需对有消息需要处理的图顶点，进行遍历计算. 这样，对于 PageRank 这种具有收敛性的大规模图迭代处理应用，可以有效减少不必要的顶点调用处理开销.

Pregel 系统采用的遍历模式介于两者之间，通过“投票”机制，图顶点可以将自己置于非活跃状态，除非该顶点接收到新的消息，否则不会在后续的迭

代处理中被计算.但是各任务仍需要遍历图顶点来检查图顶点是否处于活跃状态.Pregel 的优化之处在于,对于非活跃状态的图顶点,不需调用处理函数,节省了处理函数调用的开销.

7　结束语

综上所述,在云计算环境下进行大规模图数据的处理,涉及到图算法以及云计算领域的多个方面.目前的研究重点主要集中在以下 5 个方面:

(1)大规模图分割.云计算环境下的大图分割,需要提高子图内部的连通性,降低子图之间的连通性,维持子图之间数据规模和图拓扑结构的均衡性,同时应该有较小的时间复杂度.良好的图分割算法,是在云计算环境下降低图并行计算强耦合性的基础,但是目前的图分割技术,难以在连通性、均衡性和时间复杂度等方面同时达到较好的性能.

(2)大规模图索引结构.大规模图的数据管理,虽然依靠云计算环境的分布式并行处理机制,可以提高效率,但是索引的加入,无疑将使管理效率有一个大幅度的提升.目前关于分布式图数据的索引机制,已经有成型的产品,但是仍处在不断探索和研究过程中.而针对图处理的索引,只有部分图处理进行了云计算环境下的索引研究,如最短路径计算等,相当一部分图处理方法尚未考虑索引机制.

(3)查询处理与磁盘存储.在云计算环境下的大规模图查询处理,正处于开发阶段,还有很大的性能提升空间,而且像子图挖掘、图模式匹配查询等复杂应用,尚没有得到很好的解决方案.此外,基于 BSP 模型 Pregel、Hama 和 Giraph 等大规模图处理系统,目前均基于内存,限制了数据处理规模,将硬盘存储融入 BSP 模型并对磁盘 IO 进行有针对性的优化,是急需解决的问题.

(4)消息通信优化.在云计算环境下进行大规模图处理时,制约系统效率的重要瓶颈就是消息通信,特别是由于存在任务间的耦合性,大量的网络通信使云计算的效率大打折扣.Google、Yahoo、微软等开发机构对于网络通信的优化都进行了探索并取得了一定的效果,但目前仍只是缓解而没有从根本上解决瓶颈问题.对图处理应用,结合图分割技术以及图数据结构的特点,若能进一步降低网络通信开销,甚至解决瓶颈问题,会极大地提高大规模图处理效率.

(5)容错管理.虽然云计算系统本身拥有较为完备的容错管理机制,但是针对需要多次迭代、运行时间过长的大规模图处理而言,故障恢复的代价仍然十分昂贵.而且,冗余备份与系统处理效率之间的矛盾,没有得到明确的解决.

参　考　文　献

[1] Brin Sergey, Page Larry. The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems, 1998, 30(1-7): 107-117

[2] Lumsdaine Andrew, Gregor Douglas, Hendrickson Bruce, Berry Jonathan. Challenges in parallel graph processing. Parallele Processing Letters, 2007, 17(1): 5-20

[3] Soussi Rania, Aufaure Marie-Aude, Baazaoui Hajer. Graph atabase for Collaborative Communities//Pardede Eric ed. Community-Built Databases: Research and Development, Berlin, Heidelberg: Springer, 2011: 205-234

[4] Angles R, Gutierrez C. Survey of graph database models. ACM Computing Surveys, 2008, 40(1): 1-39

[5] Malewicz Grzegorz, Austern Matthew H, Bik Aart J C et al. Pregel: A system for large-scale graph processing//Proceedings of the SIGMOD. Indianapolis, Indiana, USA, 2010: 135-145

[6] Welcome to Hama Project. http://incubator.apache.org/hama/, 2011-7-13

[7] Trinity-Microsoft Research. http://research.microsoft.com/en-us/projects/trinity/, 2011-7-13

[8] Kang U, Tong H, Sun J et al. GBASE: A scalable and general graph management system//Proceedings of the KDD. San Diego, California, USA, 2011: 1091-1099

[9] Bu Ying-Yi, Howe Bill, Balazinska Magdalena et al. HaLoop: Efficient iterative data processing on large clusters//Proceedings of the VLDB. Singapore, 2010, 3: 285-296

[10] Berge Claude. Hypergraphs: Combinatorics of Finite Sets. New York: Elsevier Science & Technology Books, 1989

[11] Ghemawat Sanjay, Gobioff Howard, Leung Shun-Tak. The Google file system//Proceedings of the 19th Symposium on Operating Systems Principles. Bolton Landing, New York, USA, 2003: 29-43

[12] Wbite Tom. Hadoop: The Definitive Guide. O'Reilly: Yahoo! Press, 2009

[13] Chang Fay, Dean Jeffrey, Ghemawat Sanjay et al. Bigtable: A distributed storage system for structured data//Proceedings of the 7th Symposium on Operating System Design and Implementation (OSDI). Seattle, WA, USA, 2006: 205-218

[14] Tiwari Shashank. NoSQL data modelling: Concepts and cases//Proceedings of the SDEC. Delta Winnipeg, Winnipeg, Canada, 2011

[15] Apache Software Foundation. The Apache CouchDB Project. http://couchdb.apache.org/, 2011-8-10

[16] 10gen, Inc., mongoDB. http://www.mongodb.org/, 2011-8-10

[17] Apache Software Foundation, Cassandra. http://cassandra.apache.org/, 2011-8-10

[18] neo4j open source nosql graph database. http://neo4j.org/, 2011-7-14

[19] DeCandia Giuseppe, Hastorun Deniz, Jampani Madan et al. Dynamo: Amazon's highly available key-value store//Proceedings of the SOSP. Stevenson, Washington, USA, 2007: 205-220

[20] Amazon SimpleDB. http://aws.amazon.com/simpledb/, 2011-8-10

[21] Connor Alexander G, Chrysanthis Panos K, Labrinidis Alexandros. Key-key-value stores for efficiently processing graph data in the cloud//Proceedings of the GDM. Hannover, Germany, 2011: 88-93

[22] Lordanov Borislav. HyperGraphDB: A generalized graph database//Proceedings of the IWGD. JiuZhai Valley, China, 2010: 25-36

[23] Eifrem Emil. NOSQL: Scaling to size and scaling to complexity. http://blogs.neotechnology.com/emil/2009/11/nosql-scaling-to-size-and-scaling-to-complexity.html, 2009-1-15

[24] Wu Sai, Jiang Da-Wei, Ooi Beng Chin et al. Efficient B-tree based indexing for cloud data processing//Proceedings of the VLDB. Singapore, 2010: 1207-1218

[25] Wang Jin-Bao, Wu Sai, Gao Hong et al. Indexing multi-dimensional data in a cloud system//Proceedings of the SIGMOD. Indianapolis, Indiana, USA, 2010: 591-602

[26] Tsatsanifos George, Sacharidis Dimitris, Sellis Timos et al. MIDAS: Multi-attribute indexing for distributed architecture systems//Proceedings of the SSTD. Minneapolis, MN, USA, 2011: 168-185

[27] Aguilera M K, Golab W, Shah M A. A practical scalable distributed B-tree//Proceedings of the VLDB. Auckland, New Zealand, 2008: 598-609

[28] Zhang Xiang-Yu, Ai Jing, Wang Zhong-Yuan, Lu Jia-Heng et al. An efficient multi-dimensional index for cloud data management//Proceedings of the CloudDB. Hong Kong, China, 2009: 17-24

[29] InfiniteGraph, the Distributed Graph Database. http://www.infinitegraph.com/, 2011-7-29

[30] Li Bo-Duo, Mazur Edward, Yan Lei et al. A platform for scalable one-pass analytics using MapReduce//Proceedings of the SIGMOD. Athens, Greece, 2011: 985-996

[31] Dittrich Jens, Quiane-Ruiz Jorge-Arnulfo, Jindal Alekh et al. Hadoop++: Making a yellow elephant run like a Cheetah (without it even noticing)//Proceedings of the VLDB. Singapore, 2010, 3: 515-529

[32] Apach Mahout: Scalable machine learning and data mining. http://mahout.apache.org/, 2011-7-14

[33] Leskovec Jure, Lang Kevin J, Dasgupta Anirban et al. Statistical properties of community structure in large social and information networks//Proceedings of the 17th WWW. Beijing, China, 2008: 695-704

[34] Andersen Reid, Chung Fan, Lang Kevin et al. Local graph partitioning using PageRank vectors//Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science. Berkekey, California, USA, 2006: 475-486

[35] Metivier Y, Robson J M, Saheb N et al. About randomised distributed graph colouring and graph partition algorithms. Information and Compution, 2010, 208(11): 1296-1304

[36] Derbel Bilel, Mosbah Mohamed, Zemmari Akka. Fast distributed graph partition and application//Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS). Rhodes Island, Greece, 2006: 125-125

[37] Lin Jimmy, Schatz Michael. Design patterns for efficient graph algorithms in MapReduce//Proceedings of the 8th Workshop on Mining and Learning with Graphs. Washington, DC, USA, 2010: 78-85

[38] Wu Chun-Jiang, Zhou Shi-Jie, Wei Lin-Na et al. A new k-graph partition algorithm for distributed P2P simulation systems. Lecture Notes in Computer Science. Hangzhou, China, 2007, 4494: 391-402

[39] Bi T, Ni Y, Shen C M et al. An efficient graph partition method for fault section estimation in large-scale power network//Proceedings of the IEEE Power Engineering Society Winter Meeting. Ohio, USA, 2001: 1335-1340

[40] Kernighan-Lin algorithm. http://en.wikipedia.org/wiki/KernighanLin_algorithm, 2011-7-13

[41] Dutt Shantanu. New faster kernighan-lin-type graph-partitioning algorithms//Proceedings of the IEEE/ACM International Conference on Computer-Aided Design. Santa Clara, CA, USA, 1993: 370-377

[42] Elnikety Sameh. Horton: Online query execution on large distributed graphs//Proceedings of the GDM. Hannover, Germany, 2011

[43] Reid Marlow Andersen, Yuval Peres. Local Graph Partitioning Using Evolving Sets: USA, 20100205126[P]. Aug. 12, 2010

[44] Frey Brendan J, Dueck Delbert. Clustering by passing messages between data points. Science, 2007, 315(5814): 972-976

[45] Dean Jeffrey, Ghemawat Sanjay. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008, 51(1): 107-113

[46] Valiant Leslie G. A bridging model for parallel computation. Communication of the ACM, 1990, 33(3): 103-111

[47] Ching Avery. Giraph: Large-scale graph processing infrastructure on Hadoop//Proceedings of the Hadoop Summit. Santa Clara, 2011

[48] Condie Tyson, Conway Neil, Alvaro Peter et al. MapReduce Online//Proceedings of the NSDI. San, Jose, California, USA, 2010: 33-48

[49] Ekanayake Jaliya, Li Hui, Zhang Bing-Jing et al. Twister: A runtime for iterative MapReduce//Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. Chicago, Illinois, USA, 2010: 810-818

[50] Prlter-Distributed Computing Framework for Prioritized Iteration. http://code.google.com/p/priter/, 2011-7-13

[51] Isard Michael, Budiu Mihai, Yu Yuan et al. Dryad: Distributed data-parallel programs from sequential building blocks//Proceedings of the Eurosys Conference. Lisbon, Portugal, 2007: 59-72

[52] Yu Yuan, Isard Michael, Fetterly Dennis et al. DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language//Proceedings of the OSDI. San Diego, California, USA, 2008: 1-14

[53] Orleans: A Platform for Cloud Computing. http://research.microsoft.com/en-us/projects/orleans/, 2011-9-25

[54] Yahoo! plans to reconstruct Hadoop-MapReduce. http://cloud.csdn.net/a/20110224/292508.html, 2011-7-13

[55] Jing N, Huang Y, Rundensteiner E A. Hierarchical encoded path views for path query processing: An optimal model and its performance evaluation. IEEE Transactions on Knowledge and Data Engineering, 1998, 10(3): 409-432

[56] Jung S, Pramanik S. An efficient path computation model for hierarchically structured topographical road maps. IEEE Transactions on Knowledge and Data Engineering, 2002, 14(5): 1029-1046

[57] Cohen E, Halperin E, Kaplan H et al. Reachability and distance queries via 2-hop labels//Proceedings of the SODA. San Francisco, CA, USA, 2002: 937-946

[58] Gao Jun, Xu Jeffery Yu, Jin Ruo-Ming et al. Neighborhood-privacy protected shortest distance computing in cloud//Proceedings of the SIGMOD. Athens, Greece, 2011: 409-420

[59] Kang U, Tsourakakis Charalampos E, Faloutsos Christos et al. PEGASUS: A peta-scale graph mining system-implementation and observations//Proceedings of the ICDM. Miami, Florida, USA, 2009: 229-238

[60] SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/#bgpExtend, 2011-7-29

YU Ge, born in 1962, Ph. D., professor, Ph. D. supervisor. His major research interests include database theory and technology, distributed and parallel systems, etc.

GU Yu, born in 1981, Ph. D., associate professor. His major research interests include RFID data management, spatio data management and cloud computing.

BAO Yu-Bin, born in 1968, Ph. D., professor. His major research interests include massive data management, cloud computing, etc.

WANG Zhi-Gang, born in 1987, M. S. candidate. His major research interests include database system and cloud computing.

Background

Graph is one of the most powerful and popular representation for data with complex structures. It can be said the graph processing and its applications are almost anywhere. With the rapid growth of emerging applications like social network analysis, semantic Web analysis, and bio-informatics network analysis, it is urgently required to provide the processing capability on a large scale graphs of billions of vertices, which becomes the hot topic in the current high performance computing field. Recently, there has been much exploration work on processing large scale graphs with clouding computing technique, such as HaLoop, Twister, Pregel, Giraph et al.

This paper analyzes the features of clouding computing and processing on large scale graphs. Two challenge problems must be solved: (1) tightly coupling of graph computing algorithms, which makes it hard to parallelize the graph processing. (2) low reliability of cloud computing nodes, which makes it hard to finish a long running and heavy graph computing task.

To build a high performance large scale graph processing system on cloud computing environments, this paper introduces the key issues from the aspect of graph data management and graph data processing mechanism, including graph data storage scheme, index structure of graph data, graph partitioning strategy, graph computing model, message communication mechanism, fault-tolerance control, scalability, and graph query processing.

This paper summarizes the state-of-art of current research works to solve the problem on the tightly coupling problem and the low reliability problem, analyzes the existing challenge problems remaining to be solved, and explores the research directions in future, including large scale graph partitioning, large scale graph indexing, large scale graph query processing and disk storage, message communication optimization, and fault-tolerance management.

This work was supported by the National Natural Science Foundation of China under grant Nos. 61033007, 61003058, and Fundamental Research Funds for the Central University under grant (N090104001).