综述

大规模图数据划分算法综述 *

许金凤,董一鸿,王诗懿,何贤芒,陈华辉
(宁波大学信息科学与工程学院 宁波 315211)

摘 要:对大规模图数据划分算法进行了总结,介绍了并行环境下图计算模型,详述了大规模静态图划分算法和动态图划分算法,归纳了这些算法的优缺点以及适应性。最后,指出了关于大图划分尚未探索的有意义的研究课题。

关键词:大数据;大图;分布式图划分;负载均衡;BSP;MapReduce;动态图
doi: 10.3969/j.issn.1000-0801.2014.07.016

Summary of Large-Scale Graph Partitioning Algorithms

Xu Jinfeng, Dong Yihong, Wang Shiyi, He Xianmang, Chen Huahui
(College of Information Science and Engineering, Ningbo University, Ningbo 315211, China)

Abstract: The large-scale graph partitioning algorithms were summarized and graph computing models in the distributed environment were introduced. Firstly the large-scale static graph partitioning algorithms and the dynamic graph partitioning algorithms were discussed. Then the advantages and disadvantages of these algorithms and its adaptability conscientiously were sumed up. Finally, some meaningful research subjects about the distributed graph partition, which have not been explored were pointed out.

Key words: big data, large-scale graph, distributed graph partitioning, load balancing, bulk synchronous parallel model, MapReduce, dynamic graphs

1 引言

近年来,随着互联网的普及,网络用户数快速增加。据CNNIC统计,全球最大的社交网络Facebook目前已有近10亿用户。如果将用户看作图中的顶点,而用户与用户之间的关系看作图中的边,那么整个网络就可看作一张网络图。随着网络中用户规模的不断扩大,与之对应的网络图动辄有数十亿个顶点和上万亿条边,普通的计算机由于内存的限制无法正常处理,这给常见的图计算(如寻找连通分量、计算三角形和pagerank)带来了巨大挑战。解决这一问题的最好方法就是分布式计算,即将大规模图数据划分成多个子图装载到分区中,利用大型的分布式系统进行处理。为了提高不同分区间的并行速度需要使这些子图的规模均衡,同时减少通信开销,所以不同分区之间相连的边数应当足够小。基于这个因素,图划分的工作就显得非常迫切和必要。

国内外对图划分及其相关问题进行了广泛深入的研究,主要包括2个方面。

(1)集中式图划分算法

集中式图划分算法已经研究了相当长一段时间,它可以处理顶点数和边数不是很多的图,已有的算法包括局部改进图划分算法和全局图划分算法,其中局部改进图划分算法比较经典的是KL(Kernighan-Lin)算法[1]和FM(Fiduccia-Mattheyses)算法[2];全局图划分算法比较经典的是Laplace图特征值谱二分法[3]和多层图划分算法[4]。这些算法具有较高的时间复杂度,无法处理顶点数为百万以上的图,因此这些算法不适用于现实生活中大规模的图处理。

* 国家自然科学基金资助项目(No. 61202007),宁波市自然科学基金资助项目(No. 2013A610063)

(2) 分布式图划分算法

分布式图划分算法是针对近些年出现的大规模网络图而研究的,本文将已有的算法分为静态图划分算法和动态图划分算法,其中静态图划分算法的工作比较多,主要包括散列划分、BHP 算法[5]、静态 Mizan 算法[6]、BLP 算法[7]等;动态图划分算法经典的主要包括动态 Mizan 算法[8]和 xDGP 算法[9]等。

本文将详细阐述分布式图划分算法,分析它们的性能特点并做出比较,从而总结出各个算法的优势与不足,同时展望图划分算法未来的发展方向,使读者能系统而全面地了解图划分领域的研究状况与发展趋势。

2 图划分定义和评价指标

定义 1 交互边。一对互为邻居的顶点,它们被划分到不同的子图中,它们的边称为交互边。

定义 2 图划分。给定图 $G(V,E)$,正整数 k,将顶点集 V 划分为互不相交的 k 个集合 $V_1,V_2,\cdots,V_k$。

图划分方法需要满足两个主要原则:子图与子图之间相连的边数尽量小,即交互边数少;二是子图与子图的规模应当相差不大,即负载均衡。其满足以下条件。

(1) 对于每个 V_i,$|V_i|\approx\frac{|V|}{k}$, $i=1,\cdots,k$。

(2) 交互边数尽量少。该图划分问题的优化策略是在保证分区负载均衡的前提下,最小化交互边总数。

针对以上两个原则,定义了评价函数[10]:

$$\text{mimimize}_{P=(V_1,\cdots,V_k)}|\partial e(P)| \text{ and } |V_i|\leqslant\lambda\frac{n}{k},\forall i\in\{1,\cdots,k\}$$

其中,$P=(V_1,\cdots,V_k)$,表示顶点集的一个划分,$\partial e(P)$表示所有分区间总的交互边,λ 是一个大于或等于 1 的常量,n 为总负载。根据原则(1)(负载均衡),理想情况下应是$|V_i|=n/k$,考虑到实际情况应加一个参数 λ。根据原则(2)(交叉边最少),即 $\text{mimimize}_{P=(V_1,\cdots,V_k)}|\partial e(P)$。

3 大规模图数据的图划分

随着互联网的普及,图数据的规模日趋庞大,如 Web 图数据至少有 1 万亿的链接,Twitter 有超过 4 000 万的用户和 15 亿的社交链接等。这些不可预测的大规模图数据给图计算带来了严峻的挑战。解决这问题的最好方法就是分布式计算,即将大规模图数据划分成多个子图装载到分区中,然后利用大型的分布式系统来处理它们。

3.1 大规模图数据划分算法的难点

大规模图数据划分的不平衡因素主要有 3 个源头:图结构、算法本身和动态图。图结构分为两类,幂律图和非幂律图。幂律图是小部分顶点的度很高,大部分顶点的度很低。图 1 显示了 3 种最常见的划分算法(基于散列划分、基于范围划分、最小割划分[11])在不同图数据上的运行结果,结果表示没有任何一种算法适用于所有图。所以要设计一个针对所有图都有效的划分算法并不容易。

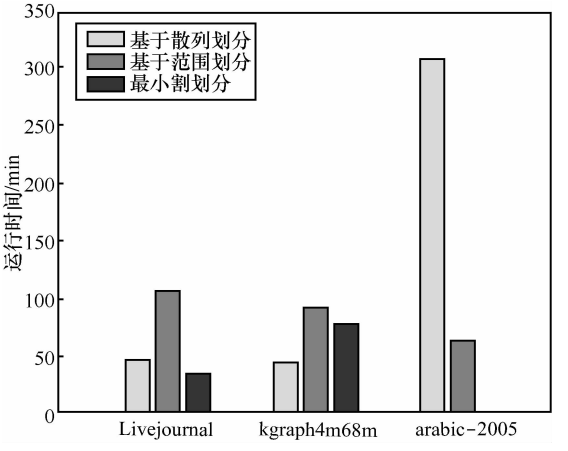


图 1 运行时间[8]

第二种不平衡因素是算法本身,一部分算法会在运行过程中影响计算节点间的负载均衡。参考文献[10]根据超步间通信特点将算法分为动态算法和静态算法。在静态算法中,每个超步的活动节点接收和发送的消息是相同的,既没有增加或减少边,输入和输出也都是相同的地址(图的结构不变),例如 pagerank、直径评估、寻找连通分量等。动态算法是输出消息的大小和目标地址一直在变,如分布式最小生成树(DMST)、图查询(如 top$k$ 问题)、广告推荐等,在每个超步中,这些变化都会产生负载不平衡影响系统效率。图 2 显示了不同算法运行时接收消息的变化趋势,*Total* 为总的消息数,*Max* 是此刻最大的消息数。由此可见,要设计一个针对所有算法都有效的划分算法是非常困难。

第三个不平衡的来源是动态图。很多在线应用如 Facebook、Skype 和 Twitter 等需要实时响应。它们每时每刻都会产生新用户或者删除旧用户,图的拓扑结构就会随时改变,这种图称为动态图。如何有效地处理动态图是一个具有挑战性的新问题。如图 3 所示,其中 HSH 是散列取模算法,DGT[12]是一种图流算法,ADP[9]是动态图划分算法。随着图拓扑结构一直变化,可看出静态或者图流算法的划分质量一直在恶化,而动态图划分算法性能变化不大,因此需

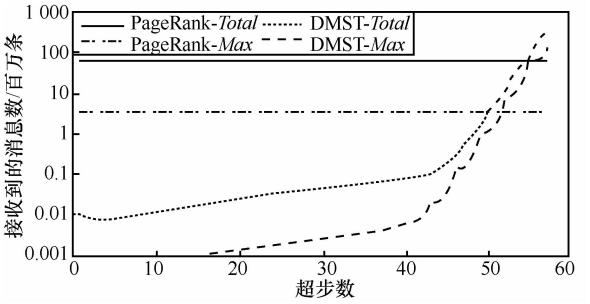


图 2　消息数量[8]

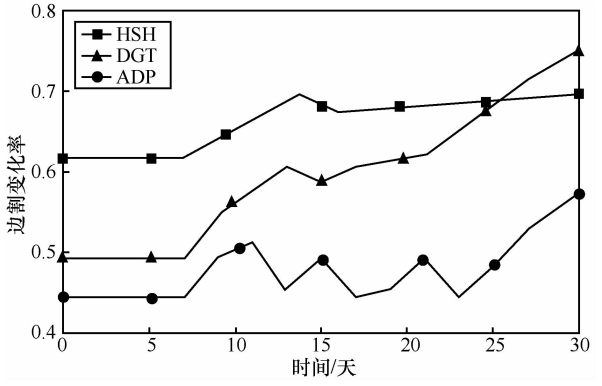


图 3　不同算法随着时间的推移的边割变化[9]

要设计一个好的动态图算法来解决以上 3 个不平衡来源。

3.2　大规模图的计算模型(MapReduce 模型和 BSP 模型)

常见的高性能分布式的计算模型主要有 MapReduce 模型和 BSP 模型。

MapReduce[13]是目前大规模数据环境中最广泛使用的模型。MapReduce 使用经典的“分而治之”策略,将对大规模数据集的操作,分发给一个主节点管理下的各个分节点共同完成, 然后将各个分节点的中间结果融合在一起,得到最后结果。Hadoop[14]和 HOP 系统[15]都是基于 MapReduce 模型的分布式并行处理系统,它可以应用于各种大规模数据处理,由于 Hadoop 不适用于迭代,很多研究者优化改进了 Hadoop 平台,如 HaLoop[16]、Twister[17]、Prlter[18]。BSP(bulk synchronous parallel model)[19]是 Valiant 早在 1990 年就提出来的一种基于消息传递的并行执行模型。计算由一系列超步组成,每个超步的最后均有一个全局同步机制,它的优点就是可以避免死锁和数据竞争问题。基于 BSP 模型最著名的就是 Pregel[20]平台。它是 Google 为了满足图迭代计算而提出来的分布式并行处理系统。由于 Pregel 不开源,Yahoo 提出了 Pregel 的开源版本:Giraph[21]。

大数据环境下主要采用以上两种模型实现大规模图处理,MapReduce 主要针对大数据的分布式处理, 而图数据具有一些独有的特点,如点与点之间的关系。BSP 模型是基于消息传递的,它处理图计算有着如下优点。

- 极少的磁盘读写,因为只有最终结果才会写到磁盘上。而 MapReduce 的串行任务之间以磁盘和数据复制作为交换介质和接口, 所以需要频繁地读写磁盘,导致 I/O 开销很大。
- 用消息传递机制代替了迭代计算模型, 每次迭代每个顶点独立地计算状态, 只传递更新状态需要的消息,而 MapReduce 需要对全体数据进行复制传递。
- 改善了数据局部性,顶点的所有信息基本上在同一分区里。

BSP 模型避免了 MapReduce 模型频繁地读写磁盘和数据混乱,其独有的全局同步机制,使迭代处理更加方便灵活,更适用于大规模图处理。

3.3　大规模图数据的静态图划分方法

大规模图划分的静态算法典型的有散列算法、BHP 算法、静态 Mizan 算法和 BLP 算法,这些算法的特点是图的结构不变,没有顶点添加和删除,不需要实时响应。

3.3.1　散列划分

最经典的大规模图划分算法是散列划分,即每个顶点首先赋予唯一的 ID 号,将图的顶点散列划分到相应的分区中。采用散列方法进行图划分的优势在于简单且易于实现,不需要额外的开销,负载是均衡的。但是散列方法没有考虑到图的内部结构,顶点会被随机地划分到分区中,这样分区与分区之间的交互边会很大,会产生巨大的通信开销。

3.3.2　BHP 算法

BHP 算法保留了传统散列算法的优点,将实际分区划分成多个虚拟桶, 通过重组虚拟桶来减少分区间边割。BHP 算法首先确定虚拟桶的数量 t(t 是分区数的倍数),然后将 t 个虚拟桶重组为 k 个,方法是:如果虚拟桶到某个实际分区的出边数与该虚拟桶总边数的比值超过一定的阈值,则将该虚拟桶归属到这个实际分区。在重组过程中通过贪婪算法保证每个分区的数据量均衡。由于一个分区只能分配给一个任务, 根据该实际分区上的数据来自哪个任务最多,就将这个实际分区交由这个任务执行。数据的本地化一定程度上降低了通信开销。BHP 算法和散列算法差不多,都没有考虑图的内部结构,没有挖掘图内部“团”结构。

3.3.3　静态 Mizan 算法

针对 Mapreduce 框架效率低的问题, 静态 Mizan 算法采用了类似 Pregel 的框架,分析了 Pregel 框架不考虑图的

结构性的缺陷。将图分为幂律图和非幂律图。对于幂律图，实现了 Mizan-α。对于非幂律图，实现了 Mizan-γ。

(1)Mizan-α

对于幂律图，图中小部分顶点的度很高，大部分顶点的度很低，肯定不适合散列划分，因为散列划分没有考虑到边的局部性，即图的内部结构，而最小割算法有很好的划分，可以最小化分区间的边数。但是计算最小割是一个 NP 难问题，所以使用并行图划分工具 ParMETIS。对于幂律图虽然最小割的开销很大，但是它带来的效益好。

(2)Mizan-γ

最小割划分不适用于非幂律图，因为它带来的开销比带来的效益大。对于非幂律图使用任意随机划分，这样预处理没有开销，但是分区间的边割数会很多。如果继续使用点对点消息传递，那么通信开销将会很大，针对非幂律图的特点，消息传递采用虚拟覆盖环。如分区 PE_1，PE_2，…，PE_m，将它们组成虚拟环：$PE_1 \rightarrow PE_2 \rightarrow \cdots PE_{m-1} \rightarrow PE_m \rightarrow PE_1$，每个 PE 从它的前驱接收消息，发送给它的后继。每个消息 M 都要进入环，它们没有具体的目的地 PE，需要此消息的分区会接收它。虚拟覆盖环从以下两个方面降低了开销：减少了物理链路上消息传递的数量；每个分区 PE 仅需要两条链路，前驱和后继，而点对点需要多个链路导致高 CPU 开销。

静态 Mizan 算法对幂律图采用 parMetis 划分，虽然考虑了边的局部性，但是最小割划分的时间开销很大，不适用于很大的图。非幂律图使用随机划分和虚拟覆盖环来传递消息，虚拟覆盖环消息传递机制会带来时延，因为很多消息到达目的地之前需要传递整个环。

3.3.4 BLP 算法

BLP 算法是将标签传递算法应用到图划分上，将凹优化问题转化为线性规划问题，在保证分区均衡的情况下，很好地减少了分区间交互边。

首先将图顶点随机划分到分区中。由于某些分区的访问很频繁，所以通过顶点转移再定位，转移那些最有增益的节点。比如顶点 u 初始化到分区 i，由于它在分区 j 上的邻居比本地多，根据规则将会被分到分区 j，顶点 u 在本地的邻居数就会增加，形成顶点的增益。每次迭代将要转移的 k 个顶点按照增益进行降序排序，表示从分区 i 转到分区 j 的第 k 个顶点的增益。$f_{ij}(x)=\sum_{k=1}^{x} u_{ij}(k)$ 表示从分区 i 转移 x 个顶点到分区 j 的总的增益函数，$u_{ij}(k) \geqslant 0, u_{ij}(k) \geqslant u_{ij}(k+1)$。由于增益是降序的，所以增益函数是递增的凹函数。将这个问题转化为线性规划问题：

$$\max_x \sum_{i,j} f_{ij}(x_{ij}) \quad (1)$$

x_{ij} 是指从分区 i 转移到分区 j 的顶点数。假设图 G 有一个固定的度，目标函数 $f(x)=\sum_{i,j} f_{ij}(x_{ij})$ 是一个线性分段凹函数，根据参考文献[22]将其转化为线性规划问题。算法迭代过程如下。

(1)确定每个顶点是否进行转移，并确定每个顶点的增益。

(2)对顶点增益进行排序。

(3)解线性规划，决定分区间应该转移多少顶点。

(4)转移这些顶点。

BLP 算法解决了标签传递算法应用到图划分上的难题(分区大小约束)，它将一个最大凹优化问题转化为线性规划问题，既保证了分区平衡，又保证了边的局部性。但线性规划的时间复杂度高，每次迭代都需要解线性规划问题。BLP 算法适用于静态图分区，因为顶点的变化，就需要重新设计和计算线性规划函数。

3.4 大规模图数据的动态图划分算法

很多场合需要实时响应，例如在线应用 Facebook、Skype 和 Twitter。它们每时每刻都会产生新用户或者删除用户，图的拓扑结构就会随时改变，这种图称为动态图。如何有效地处理动态图是一个具有挑战的新问题。通常一段时间过后，其他静态算法和图流算法的划分性能都快速降低，这样就需要重划分整个图，导致很大的计算开销。下面介绍的两个动态算法都是通过转移顶点来进行图划分的，动态 Mizan 算法主要是为了负载均衡，而 xDGP 算法主要是为了减少分区间边割，类似的还有 GPS 算法[23]和 x-pregel[24]算法。

3.4.1 动态 Mizan 算法

由于算法本身具有动态性，频繁改变图结构，通信需求不可预测。动态 Mizan 算法监视顶点运行时的特点，检查计算节点负载是否均衡，如果不均衡，每次迭代通过转移顶点来平衡计算节点间的负载。

动态 Mizan 算法是一个基于 BSP 模型的图处理系统，采用散列对图进行初始划分，在所有计算节点间动态实现负载均衡。该算法监视每个顶点的 3 个主要信息。

- 顶点输出远程消息数，即顶点向不同计算节点的邻居发送的消息数。
- 顶点输入消息。

· 顶点执行时间,顶点开始处理输入消息到结束的时间段。每个计算节点保存一个高度浓缩这些信息的版本,简称高缩信息。

顶点的 3 个主要信息中任何一个值很高都可能表示顶点有很差的布局,这就需要寻找造成负载不平衡的原因并转移部分顶点。转移时间是在一个超步结束之后,下一个超步开始之前,如图 4 所示。转移计划分为以下 5 步。

(1)根据高缩信息,确定不平衡的来源,即确定此超步中哪个计算节点不均衡。

(2)利用关联度公式选择转移对象。

(3)每个计算节点根据第(2)步确定的转移对象创建一个有序列表, 将负载大的计算节点 A 与负载小的计算节点 B 匹配,顶点转移时就将节点 A 上的顶点转移到顶点 B 上。

(4)选择顶点转移,转移顶点的数量是由被匹配的计算节点的转移对象的差距决定的。

(5)计算节点将选择的顶点转移到目的节点,每个顶点编码成流(顶点 ID、状态、边信息、接收的信息),转移顶点(在 BSP 同步之后又指定转移同步)。

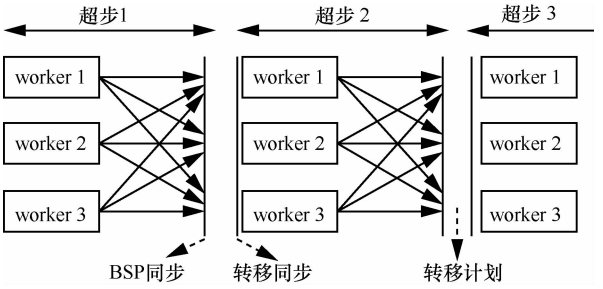


图 4　Mizan 的 BSP 转移模型[10]

实现分布式顶点转移有三大难题。

· 维护顶点所有权,使得顶点自由地在计算节点间转换,提出了 DHT(分布式散列表)来实现分布式查找顶点位置。DHT 存储的是键值对(*key*,*value*),*key* 代表的是顶点 ID,*value* 代表的是顶点当前物理位置,存放在固有节点上(每个顶点有唯一一个节点存放它的最新位置)。

· 当顶点转移后要快速更新顶点所有权信息,当顶点转移后,目标计算节点将该顶点的新位置发送给该顶点的固有节点。

· 通过延迟顶点转移来最小化顶点转移开销。

3.4.2 xDGP 算法

由于动态图结构一直变化, 如果分区不更新的话,额外的通信开销和不平衡的分区使得性能不断降低。为了适应图结构变化,xDGP 算法根据局部信息提出了迭代顶点转移算法,分区间的顶点转移目的是最小化边割,同时也需要在结构变化的情况下分区容量不溢出。如图 3 所示,其中 HSH 是散列取模算法,DGT 是一种图流算法,ADP 是 xDGP 算法。随着图结构一直变化,静态或者图流算法的划分质量一直在恶化。

(1)贪婪顶点转移

采用散列初始化分区,再使用贪婪启发式方法进行重划分。比如一个顶点添加后,根据散列取模将其划分到一个分区,启发式方法将该顶点转移到它的邻居数最多的那个分区上,即每次迭代顶点将决定去具有邻居数最多的那个分区。顶点只要知道它的邻居位置就可以选择分区,在实际系统中这个信息是可知的,因为每次迭代的时候顶点向它的邻居发送消息。

(2)分区容量限制

对于一个分区来说,每次迭代有很多顶点从不同分区转移进来,容量很可能超过分区原本的容量。算法为每个分区设置一个容量限制,每次迭代分区的使用容量不得超过分区的最大容量,将分区的剩余容量均分。这种方法减少了转移的顶点数,在实际的系统中有一定的影响,转移顶点的开销很大,顶点数转移过多会造成瓶颈,因此一定量的顶点转移会保证每次迭代的额外开销。

(3)确保收敛

转移决策的独立性会影响启发式方法的收敛,局部对称性会导致无效的转移,一对互为邻居的顶点,在同一次迭代中,两顶点可能相互转移到对方的分区中。该算法采用一个随机函数来决定是否转移,每次迭代时每个顶点转移的概率为 $s(0<s<1)$,s 很小的时候,转移顶点少,收敛时间长,s 很大的时候,会产生无效的顶点转移,执行时间增加,时延收敛。

3.5　性能归纳

综上所述,大规模图数据的划分算法中,散列算法的优点是负载均衡,算法简单易实现且易确定顶点位置。但是它打破了图的内部结构,导致分区间的边割较多,通信开销很大,一般用作初始划分。BHP 和静态 Mizan 可以作为图的预处理,其中 BHP 算法仅考虑负载均衡,没有考虑到图的拓扑结构,通信开销没有优化,静态 Mizan 算法是针对不同的图,采用不同的划分策略。对幂率图使用最小割划分,最小割的代价很大,一般图划分不使用它;对非幂率图使用虚拟覆盖环来传递消息,但会带来时延,因为很

表 1 算法比较

算法	优点	缺点	适用范围
散列	简单易实现,负载均衡	没有考虑图的结构性,边割多	静态图划分,初始划分
BHP	负载均衡,边割相对散列少	没有考虑图的结构性	静态图划分,初始划分
静态 Mizan	将图分类进行分别处理	幂律图划分开销大非幂律图有时延	静态图划分
BLP	负载均衡,边割相对少	时间复杂度大	静态图划分
动态 Mizan	负载均衡	没有考虑图的结构性,边割没有限制	动态图
xDGP	考虑图的结构性,边割少	负载均衡没有很好地控制	动态图

多消息在遇到它的目的地之前需要传递整个环,不利于图的扩展性。BLP 算法解决了标签传递算法应用到图划分上的难题(分区大小约束),它将一个最大凹优化问题转化为线性规划问题,既保证了分区平衡,又保证了边的局部性。但线性规划的时间复杂度高,每次迭代都需要解线性规划问题。BLP 算法适用于静态图分区,因为新注入一些顶点,就需要重新设计和计算线性规划函数。动态 Mizan 算法和 xDGP 算法主要为了解决动态图拓扑结构随时变化而设计的。性能方面,xDGP 算法优于动态 Mizan 算法,因为动态 Mizan 算法仅仅平衡了各计算节点间的负载而没有考虑到节点间边的局部性,所以通信开销没有很好地解决,xDGP 则充分考虑了边的局部性, 所以边割可以达到局部最优,但是它达不到全局最优,且 xDGP 算法只考虑到不超过每个分区的总容量而没有考虑到严格意义上的负载均衡。综上所述,目前静态图算法最好的是 BLP 算法,动态图算法最好的是 xDGP 算法。

表 1 对静态和动态大规模图划分算法的性能进行了归纳总结。

4 结束语

大数据时代的到来,不仅带来了机遇,也迎来一系列的困难和挑战。在社交网络日益发展的今天,集中式环境下的图划分算法已经难以适应当前应用的需求,分布式并行环境下大规模图数据的划分算法的研究日益迫切,具有重大的现实意义。大规模图数据划分的研究刚刚起步,未来的研究将在动态图、数据流图、有向图以及带权图和概率图的并行环境下进行图的划分算法的研究, 运用图论、机器学习、统计分析和散列等各种技术展开,具有极大的挑战性和现实意义。

参考文献

1 Dutt S. New faster kernighan-lin-type graph-partitioning algorithms. IEEE/ACM International Conference on IEEE, Santa Clara, CA, USA , 1993: 370~377

2 Fiduccia C M, Mattheyses R M. A linear-time heuristic for improving network partitions. 19th Conference on IEEE, Las Vegas, NV, USA, 1982: 175~181

3 Pothen A, Simon H D, Liou K P. Partitioning sparse matrices with eigenvectors of graphs. SIAM Journal on Matrix Analysis and Applications, 1990, 11(3): 430~452

4 Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on scientific Computing, 1998, 20(1): 359~392

5 周爽, 鲍玉斌, 王志刚等. BHP: 面向 BSP 模型的负载均衡 Hash 图数据划分. 计算机科学与探索, 2014, 8(1): 40~50

6 Kalnis P, Khayyat Z, Awara K, et al. Mizan: optimizing graph mining in large parallel systems. http://www.academia.edu/2601716/Mizan_Optimizing_Graph_Mining_in_Large_Parallel_ Systems

7 Ugander J, Backstrom L. Balanced label propagation for partitioning massive graphs. Proceedings of the sixth ACM International Conference on Web Search and Data Mining. ACM, Rome, Italy, 2013: 507~516

8 Khayyat Z, Awara K, Alonazi A, *et al*. Mizan: a system for dynamic load balancing in large-scale graph processing. Proceedings of the 8th ACM European Conference on Computer Systems, Prague, Czech Republic, 2013

9 Vaquero L, Cuadrado F, Logothetis D, *et al*. xdgp: a dynamic graph processing system with adaptive partitioning. http://www.researchgate.net/publication/256423219_xDGP_A_Dynamic_Graph_Processing_ System_ with_Adaptive_Partitioning

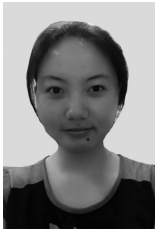
10 Tsourakakis C E, Gkantsidis C, Radunovic B, *et al*. Fennel: Streaming Graph Partitioning for Massive Scale Graphs. Microsoft Technical Report MSR-TR-2012-113, 2012

11 Karypis G, Kumar V. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.376

12 Stanton I, Kliot G. Streaming graph partitioning for large distributed graphs. Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, China, 2012

13 Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Communications of the ACM, 2008, 51(1): 107~113
14 White T. Hadoop: The Definitive Guide. O'Reilly Media, Inc, 2012
15 Condie T, Conway N, Alvaro P, *et al*. MapReduce Online. NSDI 2010, San Jose, USA, 2010
16 Bu Y, Howe B, Balazinska M, *et al*. HaLoop: efficient iterative data processing on large clusters. Proceedings of the VLDB Endowment, 2010, 3(1-2): 285~296
17 Ekanayake J, Li H, Zhang B, *et al*. Twister: a runtime for iterative MapReduce. Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, New York, NY, USA, 2010
18 Zhang Y, Gao Q, Gao L, *et al*. Priter: a distributed framework for prioritized iterative computations. Proceedings of the 2nd ACM Symposium on Cloud Computing, San Jose, USA, 2011
19 Valiant L G. A bridging model for parallel computation. Communications of the ACM, 1990, 33(8): 103~111
20 Malewicz G, Austern M H, Bik A J C, *et al*. Pregel: a system for large-scale graph processing. Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, Snowbird, UT, USA, 2010: 135~146
21 Avery C. Giraph: large-scale graph processing infrastructure on Hadoop. Proceedings of the Hadoop Summit, Santa Clara, USA, 2011
22 Boyd S P, Vandenberghe L. Convex Optimization. Cambridge: Cambridge University Press, 2004
23 Salihoglu S, Widom J. GPS: A Graph Processing System. Technical Report, Stanford University
24 Bao N T, Suzumura T. Towards highly scalable pregel-based graph processing platform with x10. Proceedings of the 22nd International Conference on World Wide Web Companion, International World Wide Web Conferences Steering Committee, Seoul, Korea, 2013: 501~508

[作者简介]



许金凤,女,宁波大学硕士生,主要研究方向为大数据、数据挖掘。

董一鸿,男,博士,宁波大学教授,主要研究方向为大数据、数据挖掘和人工智能。

王诗懿,女,宁波大学硕士生,主要研究方向为大数据、数据挖掘。

何贤芒,男,宁波大学讲师,主要研究方向为大数据、数据挖掘、隐私保护。

陈华辉,男,博士,宁波大学教授,主要研究方向为数据流与数据挖掘。

(收稿日期:2014-04-02)

·简讯·

2014年6月通信业主要指标完成情况(一)

指标名称	单位	本年本月止累计到达	比上年同期累计(±%)	本月
电信营业收入	亿元	6 867.7	5.5	1 133.0
其中:电信主营业务收入	亿元	5 957.3	5.6	978.4
电信固定资产投资完成额	亿元	1 361.1	5.0	454.7
固定本地电话通话时长合计	万分钟	13 149 879.0	-13.6	2 200 427.8
区间电话通话时长	万分钟	1 311 541.3	-17.6	222 132.8
区内电话及拨号上网通话时长	万分钟	11 838 337.7	-13.1	1 978 295.0
固定长途电话通话时长合计	万分钟	2 653 475.4	-9.5	448 731.1
国内长途电话通话时长	万分钟	2 602 487.8	-9.1	440 334.3
国际电话通话时长	万分钟	29 629.8	-31.2	4 754.5
港澳台电话通话时长	万分钟	21 357.8	-22.1	3 642.3
移动电话通话时长合计(含本地)	万分钟	145 986 729.5	2.2	24 634 356.9
移动电话国内长途通话时长	万分钟	33 937 721.0	3.5	5 815 792.1
移动电话国际电话通话时长	万分钟	64 105.5	4.3	10 312.8
移动电话港澳台电话通话时长	万分钟	45 792.5	-4.8	7 325.6
移动短信业务量	万条	37 889 716.6	-18.0	6 426 259.6
移动互联网接入流量	万GB	86 621.2	52.1	15959.5

注:(1)收入增长率按可比口径计算。(2)固定长途电话通话时长和移动电话通话时长均包含相应的IP电话通话时长。(3)通话时长各项指标均为去话通话时长。

(来源:工业和信息化部运行监测协调局)