异构计算环境中图划分算法的研究

李 琪[1] 李虎雄[1] 钟 将[2] 英昌甜[1] 李 青[2]

[1](绍兴文理学院计算机科学与工程系 浙江 绍兴 312000)

[2](重庆大学计算机学院 重庆 400030)

摘 要 复杂网络的研究已经广泛地应用到生物、计算机等各个学科领域.如今,网络规模十分巨大,如何对这些大规模图数据进行有效率的挖掘计算,是研究复杂网络的首要任务.并行计算技术是现在最成熟、应用最广、最可行的计算加速技术之一.而图划分技术是提高并行计算性能的有效手段.图划分问题的研究是随着实际应用的需求而驱动.针对异构计算环境下的分布式集群,本文提出了一种异构感知的流式图划分算法.该方法既考虑到集群中网络带宽及节点计算能力的不同,同时又考虑到了以 InfiniBand 为代表的高速网络环境下核之间的共享资源的竞争.实验以图算法 BFS、SSSP 和 PageRank 为例,相对于未考虑异构环境的流算法,图计算效率分别平均提高了 38%、45.7%、61.8%.同时针对流式图划分过程中邻点缓存查找效率低下问题,本文又设计了一种邻边结构的缓存查找算法,在相同条件下,图划分的效率平均提高了 13.4%.仿真实验结果表明,本文设计的异构感知图划分算法实现了异构集群环境下图计算效率的提升.

关键词 异构计算;图划分;云计算;复杂网络;图计算

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2021.01751

Research on Graph Partitioning in Heterogeneous Computing Environment

LI Qi[1] LI Hu-Xiong[1] ZHONG Jiang[2] YING Chang-Tian[1] LI Qing[2]

[1](*Department of Computer Science and Engineering, Shaoxing University, Shaoxing, Zhejiang* 312000)

[2](*Department of Computer Science, Chongqing University, Chongqing* 400030)

Abstract Large graph datasets are becoming increasingly popular nowadays. For example, graphs like Web Graphs, Biological Networks, and Social Networks, are often at the scale of hundreds of billions or even a trillion edges, and they are continuously growing. How to mine and calculate these large-scale graph data efficiently is the primary task of studying complex network. Parallel computing technology is one of the most mature, widely used and feasible computing acceleration technologies. Graph partitioning is an effective way to improve the performance of parallel computing. The increasing popularity and ubiquity of various large graph datasets have caused renewed interest for graph partitioning. Existing graph partitioners either scale poorly against large graphs or disregard the impact of the underlying hardware topology. A few solutions have shown that the nonuniform network communication costs may affect the performance greatly. Since the cost of partitioning the entire graph is strictly prohibitive, there are some recent tentative works towards streaming graph partitioning which run faster, are easily parallelized, and can be incrementally updated. Most of the existing works on streaming partitioning assume that worker nodes within a cluster are homogeneous in nature. Unfortunately, this

收稿日期:2019-11-27;在线发布日期:2020-05-10.本课题得到国家自然科学基金青年科学基金项目(62002226)、国家自然科学基金专项项目(6194100039)、浙江省自然科学基金(LHQ20F020001)、浙江省基础公益研究计划项目(LGG18F030003)和绍兴文理学院校级科研项目研究成果(2019LG1004)资助.**李 琪**,博士研究生,主要研究方向为图挖掘、图计算.E-mail:liqi0713@foxmail.com.**李虎雄**(通信作者),博士,教授,硕士生导师,主要研究方向为网络化系统分析与控制、图像识别与理解.E-mai:jsj_lhx@126.com.**钟 将**,博士,教授,博士生导师,主要研究领域为数据挖掘、并行计算、自然语言处理.**英昌甜**,博士研究生,主要研究方向为内存计算.**李 青**,博士研究生,主要研究方向为自然语言处理.

assumption does not always hold. Experiments show that these homogeneous algorithms suffer a significant performance degradation when running at heterogeneous environment. The research of graph partitioning is driven by the demand of practical application. Aiming at the distributed cluster in heterogeneous computing environment, we propose a streaming graph partitioning algorithm based on heterogeneous aware. The method not only considers the difference of network bandwidth and node compute ability in the cluster, but also considers the competition for shared resources between cores in high-speed network environment represented by InfiniBand. Taking BFS, SSSP and PageRank as examples, compared with the streaming algorithm without considering the heterogeneous environment, the efficiency of graph computing is improved by 38%, 45.7% and 61.8%, respectively. At the same time, in the process of streaming graph partitioning, aiming at the low efficiency of searching neighbor vertices in the cache, we design a cache searching algorithm with adjacent edge structure, which improves the efficiency of graph partitioning by 13.4% on average under the same conditions. Extensive experiments are conducted on a moderate sized computing cluster with real-world web and social network graphs. The results demonstrate that the proposed approach achieves significant improvement compared with the state-of-the-art solutions.

Keywords heterogeneous computing; graph partition; cloud computing; complex network; graph computing

1 引 言

如果把大脑中的神经元看作顶点,神经元之间互连的树突看作边,那么整个网络将包含 890 亿个顶点及 100 万亿条边[1],通过搜索引擎可以抓取约 1 万亿的网页链接关系图,据估计未来网页规模将超过十万亿[2].全球最大的社交网络 Facebook 目前拥有约 10 亿的用户[2],与之相对应的是数百亿的关系链接.普通的单计算节点由于内存容量的限制无法对这些大图正常处理,这给常见的图计算带来了严峻挑战(如寻找连通分量[3]、计算三角形[4]和 PageRank[5]).一个标准的解决方案是将图数据划分为多个子图装载到多个计算节点进行分布式计算.为此,Spark①、Pregel②、Giraph③ 和 Trinity④ 等分布式系统框架相继的被开发出来.这些系统通过丰富的 API 接口,简化了用户的分布式编程工作,实现了大图的有效处理.它们主要根据节点 ID 利用伪随机哈希函数将任务分发到每个分区.这种方式简单易于实现,且不需要系统维护一张巨大的路由表来保存节点的分区信息,但是在划分过程中,由于其没有考虑图的拓扑特性,完全打破了图的内在结构,导致运算过程中通信代价过大,因此设计一种划分效果优异的快速图分割算法,已经成为现有大图处理系统亟待解决的问题[6].

图的 k-划分是 NP 难问题[7],广泛应用于图像分割[8]、数据挖掘[9]、VLSI 设计[10]等领域.从 20 世纪 90 年代初期至今,国内外研究者不断对图划分及其相关问题进行深入研究,提出了许多性能较好的图划分算法[1].目前图划分研究主要分为 3 大类:离线划分[12]、流式划分[13]以及动态重划分[14].随着图数据规模的不断增大,基于传统的启发式的划分算法在划分效率上明显的降低[15-16].例如,在文献[13]中,作者对 Twitter 图($|V|=41652230$, $|E|=1468365182$)采用 METIS 划分[17],总耗时需要 8.5 h 以上,划分效率低下.针对此问题,近几年相继有算法被提出用来解决大规模图数据的划分效率问题,其中最经典的是流式划分算法[18-20],由于其优越的划分性能,逐渐被研究者们所关注.

然而,传统图划分方法都是以最小割边数为优化目标,而很少考虑到集群的结构对分布式图性计算性能的影响,都是以假定集群的同质为先决条件[11].当图分析系统建立在公共云环境[11](例如,亚马逊弹性计算云(Amazon Elastic Compute cloud、EC2[21]),阿里云(Alibaba Cloud Computing Co. Ltd[22])或公

① http://spark.apache.org/
② http://hama.apache.org/
③ http://giraph.apache.org/
④ https://github.com/trinityrnaseq/

司内私有数据中心[23]时，这些同质性假设并不总是成立.例如在文献[11]中，作者测量了 128 个 EC2 集群实例的网络带宽，节点之间最高带宽达到了 500 MB/s 以上，而最低的带宽只有 37.5 MB/s.集群环境的异构在当前的云计算中是普遍存在的，主要是由以下三个原因造成的：

(1) 硬件异质.在私有云中通常拥有多代硬件，与上一代硬件相比，新硬件可能配备带宽更高的网络适配器或计算性能更佳的 CPU，从而导致计算能力和通信带宽的异构性[22-23].

(2) 虚拟化.为了有效利用硬件资源，云系统一般都会使用虚拟技术[24].一对节点之间的带宽可能取决于实例(虚拟机)的分配方式.当两个实例被分配到同一个物理节点时，数据可以在它们之间高速传输，而当两个实例被分配到不同的节点甚至跨不同路由器节点时，它们之间的数据传输会慢得多.因此，虚拟化会导致通信带宽的异构[25].图 1 和图 2 显示了由 12 个物理节点组成的分布式集群中节点之间的带宽和节点计算能力差异.如图 1 所示，节点对之间的网络带宽差异很大.据测量统计，12 个计算单元之间通信的平均带宽为 105 MB/s，最高速度达到 195 MB/s，最低速度仅为 9.5 MB/s.图 2 说明了单计算节点计算 100 万位圆周率(PI)的运行时间，可以看出集群中计算节点的运算能力也不同.

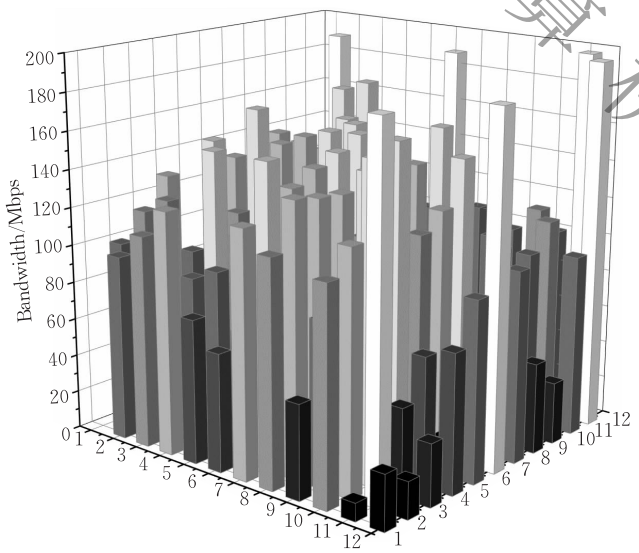


图 1　通信带宽的不均衡

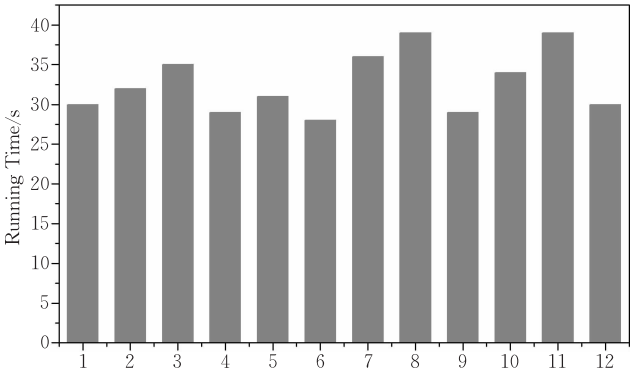


图 2　计算能力的不均衡

(3) 共享资源竞争.随着通信技术的发展，网络的带宽接近甚至超过了内存带宽[26-27].例如，每个内存通道的 DDR3 带宽目前介于 6.25 GB/s(DDR3-800)和 16.6 GB/s(DDR3-2133)之间[28]，而每个网络接口控制器端口的 InfiniBand 带宽范围为 1.7 GB/s(FDR 1X)到 37.5 GB/s(EDR 12X)[29].具有 4 通道 DDR3-1600 内存机器的内存带宽大致可以由 4 个双端口 FDR 4X NIC 提供.因此，现有的网络不再是直接内存访问(Remote Direct Memory Access，RDMA)技术的瓶颈[26].在高速网络的多核集群中，节点内核与核之间需要通信时，为了保持数据的一致性，会在共享缓存中拷贝多次，导致共享资源的争用问题.而相对于高速 RDMA 零拷贝数据通信，节点内的核与核之间的通信反而会降低分布式图计算的效率.

因此，在构建图划分模型时应考虑到这些因素对分布式图计算的影响.目前传统图划分方法的目标是如何实现最小割的数量，例如 Metis/ParMetis[12]、LDG[30]、Fennel[13]、Grapes[31]、JA-BE-JA[32]、Powerlyra[33]、Phylofactorization[34]等.但是这些仅关注最小化割边数的图划分不足以实现可伸缩的性能，因为基于最小割边数的解决方案无法保证割边是如何跨节点分布.它们最终可能被分配到通信成本高的节点之间，导致通信量的加大，尤其在高速网络环境下，会进一步加剧多核计算节点的内存子系统的争用.

这些实现不能充分利用集群结构信息来指导划分策略.近年来，也出现了针对集群的异构作业处理研究.Chen 等人[35]研究了 MapReduce 框架中集群的异质性，改进 Hadoop 上的应用程序，但没有考虑图应用的异构性.Wang 等人[31]提出了一种考虑云环境下网络带宽差异的多级图划分框架.Catalyurek 等人[36]考虑到集群中节点计算能力的差别，设计了动态负载均衡图划分算法.Dathathri 等人[37]和 Xue 等人[38]试图通过避免在具有较高网络通信成本的分区之间切割任何边来解决这种通信异构问题，然而，这些图划分方法都是建立在现有的静态图划分算法之上，具有很差的可扩展性，且只能够处理静态小规模图，无法处理大规模动态图.

事实上，现实世界的网络本质上是动态的[39]，即随着时间的推移，节点或连边随着时间的推移不断的被添加或删除. 例如，在无线传感网络中，设备连接到路由器或断开与路由器的连接. 在社交网络中，新用户和现有用户之间的友谊会随着时间的推移而产生变化. 尽管 Moulitsas 等人[40]提出了一种轻量级的架构感知图划分，但是该划分可能导致动态图计算的次优性能[41].

Zheng[14]和 Busse[41]等人已经发现，现代多核计算机的存储子系统(例如，末级缓存、内存控制器和前端总线)上共享硬件资源的争夺会极大地影响分布式工作负载的性能. 具体来说，他们主要研究了 MPI 工作负载的争用问题. 而本文的工作主要是体系结构感知(计算力和通信异构以及子系统的资源争用)图划分应用，旨在避免分布式图计算的异构和争用问题.

综上，已有关于图划分的工作重点往往是最小化割边数，而很少考虑到集群体系结构对分布式图计算效率的影响，包括节点计算力和网络通信的异构，以及高速网络下多核节点内的共享资源争用. 本文针对异构并行环境下的图划分难题，改进分布式图计算应用程序的通信模式到底层硬件拓扑的映射，提出一种异构环境感知的流式图划分算法(HaSGP). 本文的主要贡献如下：

(1) 考虑到分布式集群中节点的计算力与节点间通信带宽的不同，以及子系统内共享资源的竞争. 本文对异构环境进行了形式化建模，实现了利用底层体系结构指导大规模动态图的划分，提升了异构环境下分布式图计算的效率.

(2) 考虑到划分过程中，集群中的“主节点”存在着大量的查找添加等操作. 本文提出了基于邻边结构的缓存数据管理方式，该结构可以有效地提升缓存中邻点的操作效率，在有效利用内存空间的提前下提升图划分算法的性能.

(3) 为了评估 HaSGP 算法的有效性，实验模拟了不同的异构环境，并与已有算法进行对比分析. 评估结果表明，本文所提出的异构环境感知图划分方法能够有效地“平衡”集群的工作负载，显著提高作业执行时间.

本文第 2 节对异构环境下的图划分问题进行详细地描述；第 3 节对异构环境下的各种异构因素进行形式化建模；第 4 节阐述异构计算环境图划分算法的具体实现；第 5 节给出具体的实验结果；最后，在第 6 节对本文的研究工作进行总结.

2 问题定义

$P_k=\{S_1,S_2,\cdots,S_k\}$ 为图数据 $G(V,E)$ 划分完之后的 k 个子区. P_k 为图 G 的一种划分. k 为分区的数目. S_i 表示某个子区. V_i 指某个子区 S_i 中顶点的集合，用公式描述为

$$P=\{P_i:\bigcup_{i=1}^{k}S_i=V \text{ and } S_i\cap S_j=\varnothing \text{ for any } i\neq j\} \quad (1)$$

异构感知图划分算法旨在将图数据划分到 k 个分区，同时尽量降低分布式图计算过程中分区之间的通信量. 首先对分区 P 的通信成本进行定义，用符号 $comm(G,P)$ 表示，如下所示：

$$comm(G,P)=\sum_{e=\{u,v\}\text{ and }u\in V_i\text{ and }v\in V_j\text{ and }i\neq j}\frac{w(e)}{c(S_i,S_j)} \quad (2)$$

其中，$c(S_i,S_j)$ 定义为通信的带宽. 带宽越低，通信成本越高，带宽越高，则通信成本越低. $W(e)$ 为边 (u,v) 的权重. 在同构环境下，通常假定 $c(S_i,S_j)=1$，即网络带宽相等. 但是这种假设不能反映现代多核高性能计算(multicore High Performance Computing, HPC)基础设施的特点. 因此，在异构集群环境中，为了最小化 $comm(G,P)$，划分算法应该尽量减少通信成本较高的节点之间的通信边.

一个好的划分算法应该使集群节点在执行图计算任务时都能同时完成任务，而不应该使某些节点提前进入空闲状态等待其它节点. 需要注意的是，异构环境下的“负载均衡”与同构环境下的负载均衡的概念不同，异构环境下的“负载均衡”实际上也是一种不均衡分配，是要求每个集群节点根据自身的计算能力来分配任务. 计算能力低的节点分配的任务较少，计算能力高的节点分配的任务相对较多，以此达到同时完成任务的目的，具体节点分配的任务量在第 3 节进行详细说明.

3 集群异构因素的形式化建模

本节首先介绍采用流式图划分作为切入点的动机，说明了流式图划分所存在的问题. 最后，以流式图划分为基础，对各种异构环境进行形式化建模.

3.1 流式图划分

假设 $S^t=\{S_1^t,S_2^t,\cdots,S_k^t\}$ 表示在 t 时刻 k 路划分的状态，其中 V_i^t 表示在 t 时刻子区 S_i 中的点集合. 流式图划分就是将图中顶点按照某种规则排序(如广度优先、深度优先等)，根据此时的划分状态

S^t及当前点的邻居点$N(v)$，依次分配队列中的顶点.不同流式划分方法分配队列中的顶点所采用的启发式规则不同，例如最小非邻居节点流划分算法(Non-Neighbors，NN)，将点v分配到子区S_i的启发式规则是最小化$|S_i \backslash N(v)|$.确定性贪心流划分算法(Deterministic Greedy，DG)，启发式规则是最大化$|N(v) \cap S_i|$.指数权重确定性贪婪流划分算法(Exponentially Weighted Deterministic Greedy，EDG)，启发式规则是最大化$|N(v) \cap S_i|(1-\exp(|S_i|-n/k)$.流式划分依据不完整的局部信息，随着已分配完成的顶点数量增多，计算当前点可利用的信息量也在不断的增加.

Stanton 和 Kliot[30]分析了一系列的启发式流算法的性能，在这些流方法中，性能最好的是线性权重贪婪流算法(Linear Weighted Deterministic Greedy，LDG)，式(3)介绍了 LDG 算法将点v分配到子区S_{ind}的启发式规则.

$$S_{ind}=\arg\max_{i\in\{1,\cdots,k\}}\{|N(v)\cap V_i^t|w(t,i)\},$$
$$w(t,i)=1-|V_i^t|/(n/k) \tag{3}$$

从式(3)可以看出，每个顶点只计算一次.方程前半部分函数表示在t时刻，子区S_i中含有点v的邻居点数量.为了使子区负载均衡，在方程后面乘以惩罚函数$w(t,i)$，惩罚拥有点过多的分区.选择函数值最大的子区，将点v分配到此子区中.

流式划分由于高效的划分管理，近年来得到了不断的发展[42-44].但是流方法存在两个问题：(1)原方法没有考虑到集群计算环境的异构性.当集群存在异构时，已有流式划分方法所优化的目标可能会导致并行计算性能的下降，因此已有流式划分不适用异构集群环境下任务分配的方法；(2)划分过程中的邻点缓存数据结构不利于查找，对于查找(主要指查找当前顶点已分配完成的邻居点及邻居点所在的子区信息)效率低下.

针对以上两个问题，本文提出了异构环境下的流式图划分算法，该算法能够根据实际集群体系结构具体硬件配置，产生合理的划分策略，以提升异构环境下分布式图计算效率.同时针对流算法出现的问题(2)，本文设计了邻边缓存结构来提高划分过程中的操作效率，以提升流式图划分算法的性能.具体过程在以下小节中详细介绍.

3.2　计算能力

计算能力是以某个计算节点在单位时间内执行的任务量来度量.处理器性能的强弱是影响计算能力的一个重要因素.对于由不同计算能力的节点所组成的集群，如果负载相同，必然造成计算能力低的节点处理时间较长.计算能力强的物理节点处理时间较短，整个分布式计算任务时间是由计算时间最慢的节点所决定，严重影响了整个分布式计算任务的效率.而理想的情况是集群节点能够同时完成任务.所以需要按照节点的计算能力分配任务量，任务量与计算能力成正比.

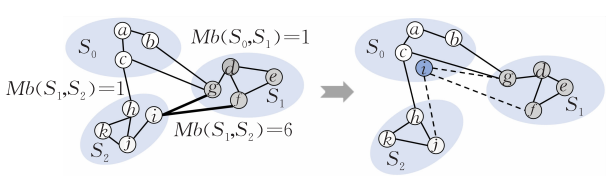
为了量化节点的计算能力，本文采用文献[45]中的方法.符号Cb_i表示节点$Worker_i$的计算能力.本文以浮点运算作为节点计算能力的指标.具体操作为，节点$Worker_i$随机生成两个浮点数并对这两个浮点数进行相乘，重复10^6次操作，记录总响应时间并计算一次浮点运算的操作(符号表示为$TexTime_i$)时间.然而，$TexTime_i$是一个非常小的浮点数，为了便于计算，我们对$TexTime_i$进行标准化.如果某个节点$Worker_{\max}$的浮点计算的时间最长，那么可以根据以下公式计算得出集群中任意一个就是计算节点$Worker_i$的计算能力，值越大表明该节点的计算能力越强.

$$Cb_i=\frac{TexTime_{\max}}{TexTime_i} \tag{4}$$

3.3　通信带宽

通信带宽是指物理节点之间网络传输数据的速率，具体是指单位时间内通过其链路层的数据量.本文使用 64 位数据作为通信单元.在异构环境中，不可避免的会出现网络带宽的不同.传输相同的数据量，带宽低的网络传输的时间较长，反之，带宽高的网络传输时间较短.

图 3 为异构网络环境下的不同划分方法的比较，图中也列出了不同节点之间的通信成本.通信成本与通信带宽相反，成本越高带宽越低.S_2与S_1之间的通信成本较高.如果按照已有k路均衡划分的方法(Old partitioning)对图进行划分，会使割边数最少，割边数为 5.但是总的通信代价却为 15.如果使用本文的异构感知的划分方法，会综合考虑通信代价及割边数(Best partitioning).如图 3 的右图所



(a) Common partitioning　　(b) Optimal partitioning

图 3　异构网络环境下的不同划分方法的比较

示，虽然割边数提高了，但是总的通信成本却降低了，降低为 6，明显提高了通信效率. 因此，异构网络环境下应该考虑带宽的不同，对于低带宽网络之间的两节点应减少分配相关联的任务量.

为了量化集群中节点之间的通信成本，本文同样采用文献[45]中的方法. 符号 $Mb(i,j)$ 表示计算节点 $Worker_i$ 和 $Worker_j$ 之间的通信成本. 假定任何一对计算节点之间的往返通信成本是相同的，即 $Mb(i,j)=Mb(j,i)$. 本文记录一个数据块从节点 $Worker_i$ 到节点 $Worker_j$ 的时间来测量其通信能力. 实验中采用的是全双工通信，与上一节的计算能力相同，对通信带宽进行标准化. 如果某节点对之间的通信带宽最大，用 $TcTime_{\max}$ 表示，则集群中其它节点对 $Worker_i$ 和 $Worker_j$ 之间的通信成本 $Mb(i,j)$ 用以下公式求出.

$$Mb(i,j)=\frac{TcTime(i,j)}{TcTime_{\max}} \tag{5}$$

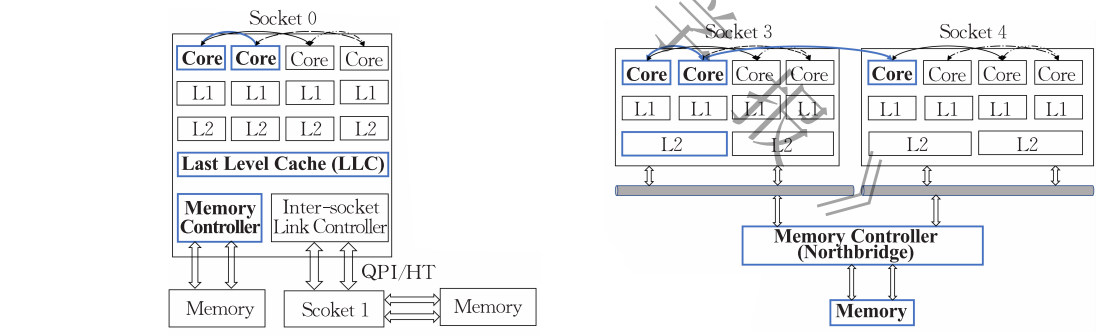
3.4 共享资源竞争

对于通过高速网络（如 InfiniBand）连接的集群，网络上的数据传输速度几乎与将数据从内存移动到 CPU 一样快. 启用 RDMA 技术的网络允许计算节点从另一个计算节点的内存中直接读取数据，而不涉及任何节点的处理器、缓存或操作系统，从而实现真正的零拷贝数据通信. 但是当多核计算节点内部的核与核之间进行通信时，需要在最后一级共享缓存（Last Level cache，LLC）拷贝多次数据. 鉴于此，在高速网络环境下，多核计算节点内的核与核之间的通信会影响分布式图计算的性能.

在对节点内共享资源竞争问题建模之前，首先需要罗列出节点内可能出现的所有核之间通信情况. 按照核的位置进行归类. 如表 1 所示，包括两核位于同一插槽共享最后一级缓存（CL1），两核位于同一插槽使用不同的最后一级缓存（CL2）以及两核位于不同的插槽（CL3）. UMA 表示非统一内存访问架构，NUMA 表示统一内存访问架构，如图 4 所示. 通信的两核所处的位置不同，导致共享及竞争的资源也不同. 所以模型的建立应该考虑共享资源竞争的程度，合理构建通信成本. 避免将邻点分配到竞争较大的两核中.

表 1 节点内共享资源竞争

		UMA			NUMA	
		CL1	CL2	CL3	CL1	CL2
共享	Socket	√	√		√	
	LLC	√			√	
竞争	LLC	√			√	
	FSB/QPI(HT)	√	√			√
	MC	√	√	√	√	



(a) Non-uniform Memory Access (NUMA) Architecture (b) Uniform Memory Access (UMA) Architecture

图 4 非统一内存访问架构和统一内存访问架构

S_i 和 S_j 代表了两个核（或者两个插槽，两个计算节点，但是当在高速网络下的多核计算节点内存在共享资源竞争时，S_i 和 S_j 为核）. $intera\_worker(S_i,S_j)$ 表示 S_i 和 S_j 之间实际的通信成本，值等于 $Mb(S_i,S_j)$. $interamax\_worker(S_i,S_j)$ 表示所有计算节点之间通信成本的最大值. $intermax\_socket(S_i,S_j)$ 表示节点内部所有插槽与插槽之间通信成本的最大值. 节点内两核之间的通信成本 $c(S_i,S_j)$ 定义如下：

$$\begin{aligned}c(S_i,S_j)=&intera\_worker(S_i,S_j)+\\&\alpha\times intermax\_worker(S_i,S_j)+\\&\beta\times intermax\_socket(S_i,S_j)\end{aligned} \tag{6}$$

其中，α 和 β 是介于 0 到 1 之间的参数，表示竞争程度. 当集群环境为高速网络时，当两核处于同一插槽时，此时竞争最大，因此 α 和 β 都不为零，目的增加其通信成本；当两核位于不同的插槽时，此时竞争相对小一些，因此 β 为零，α 不为零；当两核分别位于不同的节点，此时不存在竞争，只考虑通信异构性，β 和 α 都设为零.

4 异构环境下的图划分算法

本节主要介绍所提出的异构环境下的图划分

算法——异构感知的流式划分算法，简称为 HaSGP. 然后介绍针对流式划分的邻边缓存结构，该结构可以进一步提升流式图划分的效率.

4.1　异构感知流划分

LDG 算法的目标是根据集群节点的负载将顶点分配到具有最大邻居数的分区中. 本文将 LDG 应用在异构并行环境中原因是其算法简单，易于实现，且在一系列的流算法中划分效果优异. 考虑到集群环境的异构性，HaSGP 通过启发式方法将当前顶点 u_i 放置入分区 S_{ind}，最大化目标函数(8)，从而将非均匀的网络通信成本、节点的计算能力以及内存子系统的资源竞争考虑在内.

$$ind=\arg\max_{i\in[1,k]}\frac{1}{\sum_{e=(u,v)\in E \text{ and } u\in S_i \text{ and } i\neq j} w(e)\times c(S_i,S_j)}\times\left(1-\frac{w(S_i)}{\frac{Cb_i}{\sum_{x=1}^{k}Cb_x}|V|}\right) \tag{7}$$

其中，$w(e)$ 表示边权重. 对于无权重网络，$w(e)$ 为 S_i 与 S_j 之间的总割边数. 对于有权重网络，$w(e)$ 为 S_i 与 S_j 之间边的总权重. 其中式(7)前半部分为计算当前点分配到其中任意一个分区(S_i)与其它分区(S_j)之间总通信量的倒数. 后半部分为惩罚函数，惩罚负载过多的子区. 结合式(7)的前后部分，计算得出该方程的最大值时，i 的取值就为当前顶点所属的分区.

在 3.4 节中，我们对分区之间的通信成本 $c(S_i,S_j)$ 进行了说明. 其中，$\alpha\in[0,1]$，$\beta\in[0,1]$. 当网络的带宽很低时，节点内部的共享资源竞争可以忽略不计，这时影响分布式图计算性能的主要是节点的计算能力与节点之间的通信能力，在此环境下参数设置为 $\alpha=0$，$\beta=0$，图划分的优化目标是尽量将两邻居点分配到同一计算节点中；当网络带宽很高时，由于采用 RDMA 技术，节点之间的通信不涉及共享内存的复制等操作. 影响分布式图计算性能的主要因素是节点的计算能力与资源竞争，参数值设置为 $\alpha\in(0,1]$，$\beta\in(0,1]$. 考虑到资源争用和通信异构性的影响是高度依赖于应用程序和硬件的，用户需要在实际计算环境中对目标应用程序进行具体分析，以确定参数数值的理想情况. 在本文中，α 与 β 的值都设置为 1.

4.2　邻边结构

流式划分过程中的邻点缓存数据结构不利于查找，对于查找(主要指查找当前顶点已分配完成的邻居点及邻居点所在的子区信息)效率较低. 因此本节设计了邻边缓存结构来提高查找效率.

为了更直观地说明原始流算法中的邻点结构，我们以线性权重贪婪流算法划分为例，介绍基于邻边结构的流算法划分过程. 图 5 为示例图 $G(V,E)$ 划分为 3 个子区(S_1,S_2,S_3). 算法首先将图中的顶点随机排成队列，根据顶点的先后依次进行分配. 分配每一顶点，将此点及对应的分区信息($v\in S_i$)保存在内存中. 计算过程如表 2 所示，在 T 时刻分配完 ID 为 1 的顶点，将顶点 v_1 分配到子区 S_1，随后在缓存中保存点 v_1 及对应的分区信息 S_1. 在 $T+1$ 时刻计算 ID 为 3 的顶点(v_3)的所属子区，首先要在缓存中查找已经分配完成的此点邻居点信息(v_3)的邻居点只有 v_1 完成分配，所以只能依据点 v_1 的所属子区信息分配 v_3). 再根据邻点分区信息($v_1\in S_1$)计算当前的顶点归属子区，按照同样的规则分配后续的顶点，直到图中所有的点都分配完成，算法结束.

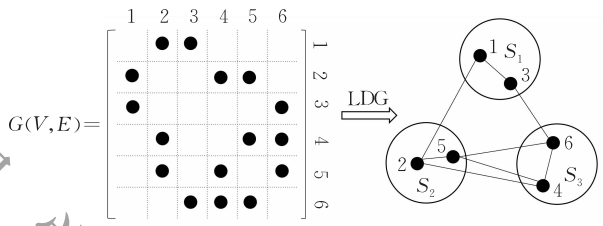


图 5　示例图 G 划分为 3 个子区

表 2　从时刻 T 到时刻 $T+5$ 对图 G 进行流划分过程中动态缓存区中的数据变化(邻点结构)

时刻	对应时刻处理的顶点 ID	处理之后缓存数据内容
T	1	$v_1\rightarrow S_1$
$T+1$	3	$v_1\rightarrow S_1$，$v_3\rightarrow S_1$
$T+2$	2	$v_1\rightarrow S_1$，$v_3\rightarrow S_1$，$v_2\rightarrow S_2$
$T+3$	6	$v_1\rightarrow S_1$，$v_3\rightarrow S_1$，$v_2\rightarrow S_2$ $v_6\rightarrow S_3$
$T+4$	5	$v_1\rightarrow S_1$，$v_3\rightarrow S_1$，$v_2\rightarrow S_2$ $v_6\rightarrow S_3$，$v_5\rightarrow S_2$
$T+5$	4	$v_1\rightarrow S_1$，$v_3\rightarrow S_1$，$v_2\rightarrow S_2$ $v_6\rightarrow S_3$，$v_5\rightarrow S_2$，$v_4\rightarrow S_3$

通过对已有流算法过程的分析，可以发现，输入图是以邻接矩阵的形式载入内存. 基于邻点结构的流算法分配当前的顶点，由于从邻接矩阵中只能得出此点的邻居点，无法判断出哪些邻居点已经分配完成，所以要在动态缓存中寻找，由邻接矩阵与动态缓存内容共同确认已经分配完成的邻居点. 据此，我们将缓存数据的结构转换为邻边形式. 分配当前的顶点，只需查找此点为键的字典条目，通过键查找到值，值对应着此点已经分配完成的所有邻点分区信息，通过值可以直接计算出此点所属的子区. 计算过

程如表 3 所示，在 T 时刻处理 ID 为 1 的顶点(v_1)，分配完成之后，将此点的分区信息作为值，邻点做为键($v_2:S_1$，$v_3:S_1$)保存入动态缓存中. 在 $T+1$ 时刻，计算顶点 v_3，由于在动态缓存中已经保存了点 v_3 的邻点分区情况($v_3:S_1$)，所以根据此信息直接计算出此点所属的分区，按照同样的规则分配后续的顶点，直到所有的顶点分配完成，算法结束.

表 3 从时刻 T 到时刻 $T+5$ 对图 G 进行流划分过程中动态缓存区中的数据变化(邻边结构)

时刻	对应时刻处理的顶点 ID	处理之后缓存数据内容
T	1	$\{v_2:S_1;v_3:S_1\}$
$T+1$	3	$\{v_6:S_1;v_2:S_1\}$
$T+2$	2	$\{v_6:S_1;v_5:S_2;v_4:S_2\}$
$T+3$	6	$\{v_5:S_2,S_3;v_4:S_2,S_3\}$
$T+4$	5	$\{v_4:S_2,S_3,S_2\}$
$T+5$	4	NULL

基于以上过程分析，当分配当前点的所属子区时，基于邻点的流式划分每次需要多次查找其已经分配完成的邻居点. 而基于邻边的流式划分只需查找一次，明显提高了划分效率. 在实验 5.4 节，我们也用实验证明了邻边结构相对于邻点结构的优越性.

4.3 更新过程

算法执行过程中，对于动态缓存中的数据，HaSGP 策略提供了两种操作，分别是添加操作及删除操作.

添加操作 $appenddate(N(v),P(v))$：每次点 v 分配完毕，要将此点的邻居点 $N(v)$ 作为键，点 v 的分区信息 $P(v)$ 作为值以字典条目的形式保存在缓存中. 如果动态缓存中已经存在点 v 的某一邻居点 v_1 为键的条目，则直接在其值域追加点 v 的分区信息 $P(v)$，如果不存在，则新建条目 $item(v_1)$.

删除操作 $deletedate(item(v))$：每次点 v 分配完毕，动态缓存中以此点为键的条目已变成冗余数据则直接将其删除. 例如在表 3 中 $T+1$ 时刻，处理完 ID 为 3 的顶点之后，动态缓存中以此点为键的条目($v_3:S_1$)对于后续顶点的计算已无价值，所以将其删除，主要是为了提升内存空间的利用率.

相应的伪代码如算法 1 所示，其中 $HaSGPalgorithm(v,N(v))$ 为线性权重贪婪流划分算法函数名，输入的参数为点及其邻居点. 函数 $rs(G(V,E))$ 作用是将图 G 中的顶点按照随机序列排序，函数 $f(S^t)$ 为异构感知的流式划分模型，输入参数 S^t 为 t 时刻划分的状态. 函数 $dycachepool()$ 表示动态缓存中的数据集合，随着不断执行更新与删除操作，数据也是动态变化的.

算法 1. 异构感知流划分 $HaSGPalgorithm(v,N(v))$.

输入：图 $G(V,E)$，子区数 k

输出：划分结果 $P(V)$

```
1.  FOR v IN rs(G(V,E))
2.    S_ind → f(S^t)
3.    appenddate(N(v),P(v))
4.    IF item(v) in dycachepool()
5.      deletedate(item(v))
6.    ELSE
7.      CONTINUE
8.    END IF
END FOR
```

4.4 复杂度分析

假设某顶点 v_i 的度为 $Deg(v_i)$，算法在某时刻分配点 v_i 时，其已经分配完成的此点邻居点数量用符号 $Deg'(v_i)$ 表示(例如表 3 中 $T+1$ 时刻，在处理 ID 为 3 的顶点时，此点的邻居点 v_1 已经处理完毕，所以已经分配完成的 v_3 邻居点数量为 1，即 $Deg'(v_1)=1$)，分区数为 k. 采用基于邻点结构的流算法分配当前顶点，首先查找已分配完成的此点邻居点，此步骤时间复杂度为 $Deg(v_i)$，再根据这些邻居点所属的分区信息进行划分，此步骤复杂度为 $k\times Deg'(v_i)$，分配完图中所有顶点所需要的时间复杂度为 $\sum_{i=1}^{n}Deg(v_i)+k\sum_{i=1}^{n}Deg'(v_i)$，即 $2m+k\sum_{i=1}^{n}Deg'(v_i)$. 引入邻边结构的流算法分配当前的顶点，只需要查找此点为键的条目，每次查找的时间复杂度为 1，再依据此条目中的值直接进行划分，分配完图中所有点所需要的时间复杂度为 $n+k\sum_{i=1}^{n}Deg'(v_i)$，其中 $(2m\gg n)$，时间复杂度明显降低.

本节通过对采用邻边结构的流算法时间复杂度进行了理论分析，并与基于邻点结构的已有流算法进行了对比，在第 5.4 节会通过具体实验衡量划分时间.

5 实验评估

实验使用三种常见的图算法：网页排名(PageRank)[44]、单源最短路径(SSSP)[45] 和宽度优先搜索(Breadth First Search，BFS)[46]. 从不同的异构环境(CPU 异构，网络通信异构，共享资源竞争，图计

算)分析算法的有效性.

实验所选用的是真实世界的图数据和理论图数据,且综合平衡了图的规模和种类,包括社交网、时序网、引文网络等.表 4 列出了本文所用到的实验图数据,所有的图数据来自斯坦福大学网络分析项目[47].本文主要考虑的是连通图.为了防止零切割边的出现,重边,自环,度为零的点都已经被移去.

表 4　图数据集

Dataset	Vertices	Edges	Type
Com-orkut	3072627	234370166	Social
Synthetic	5000000	100000000	Synthetic
Twitter	41652230	1468365182	Social
Friendster	65608366	1806067135	Social

5.1　评估指标

已有研究的图划分算法都是用割边率(Fraction of edges cut)来评价划分结果的优劣.割边率表示图数据被划分之后分区之间的割边数与总边数的比值.在同构环境中这种指标在一定程度上能够反映划分的效果.但是在异构环境中,由于网络通信的异质性,割边率并不能真正意义上用来评价通信量.例如,网络异构集群环境中由于网络带宽的不同,即使割边数最少,如果较多的割边被分配在延迟较高节点之间,反而会降低图计算的性能.因此,本文直接采用执行分布式图算法过程中总的通信量(Communication Volume,CV)作为衡量图划分性能的指标.

同时,图划分目的是提升分布式图计算的性能,因此,为了更准确的对异构环境下的图划分性能进行评估,我们主要使用作业执行时间(Job Execution Time,JET)作为算法划分结果的评估方式.值得注意的是,我们只记录了图计算的执行时间,不包括图数据的加载时间.

5.2　计算能力异构

本节主要验证在计算力异构的集群环境中,HaSGP 算法对图计算效率的影响.实验设置网络带宽相同,但是计算节点的 CPU 的性能各不相同.我们选用三种网络拓扑方案(T_1,T_2,T_3).网络拓扑 T_1 由 2 台 1.8GHz CPU 和 2 台 2.6GHz 计算节点组成,节点之间的带宽为 500 Mbps;网络拓扑 T_2 由 1 台 0.8GHz CPU 和 3 台 2.6GHz 计算节点组成,节点之间的带宽为 500 Mbps;网络拓扑 T_3 由 4 台 1.8GHz CPU 的机器组成,节点之间的带宽为 500 Mbps;对比算法是同构环境下的 LDG 算法.详细拓扑信息在表 5 中列出.

表 5　拓扑信息

Topo	CPU(Number)	Network(Number)
T_1	1.8GHz (2), 2.6GHz (2)	500 Mbps (4)
T_2	0.8GHz (1), 2.6GHz (3)	500 Mbps (4)
T_3	1.8GHz (4)	500 Mbps (4)

图 6 显示了两种算法在合成数据集 Synthetic 上执行 PageRank 算法迭代十次所运行的时间.由于网络拓扑 T_1 及 T_2 的节点计算力不同,所以 PageRank 算法在这两种划分结果之上的运行时间有了明显的差异.使用 HaSGP 划分之后的图计算执行时间平均要比原 LDG 算法减少约 8.5%.值得注意的是,虽然 T_2 相对于 T_1 只有一台计算节点的频率低于 T_2,其它都高于 T_1.但是不管是 LDG 还是 HaSGP,在拓扑 T_2 上运行图计算的时间都多于 T_1.因为在 LDG 算法中,每个节点所分配的负载相同,所以 0.8GHz 的节点需要运行更多的时间才能完成任务,而并行计算的总任务运行时间是由最慢的节点所决定,所以降低了整个图计算的效率.在 T_2 中,虽然使用 HaSGP 算法相对于 LDG 提高了图计算效率,但是总运行时间还是低于 T_1,主要是因为 0.8GHz 的性能相对较低,当使用 HaSGP 算法时,该节点所分配的任务相对较少,而其它节点分配较多的任务,虽然能够同时完成任务且没有空闲时间,但是总的计算时间还是增加.而在网络拓扑 T_3 中,由于 4 个节点的计算力相同,所以即使使用 HaSGP 算法,每个节点被分配的任务量也是相同的,这与原 LDG 算法原理相同,所以两种方法划分的结果一致,使图计算迭代的时间也相同,进一步说明了 HaSGP 算法考虑节点计算力异构的准确性.

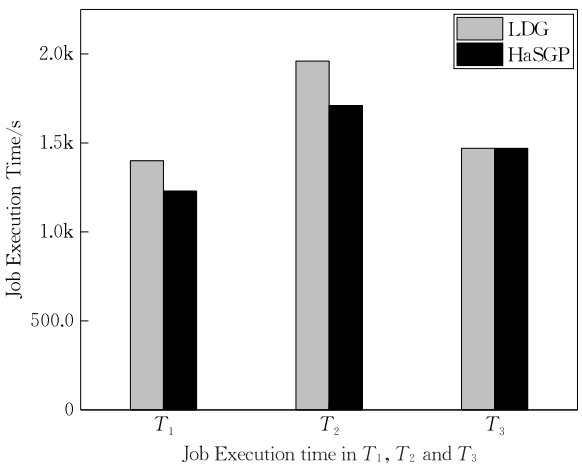


图 6　两种划分对不同拓扑网络的影响

5.3　网络通信异构

本节主要验证低速网络并行环境中(不考虑节

点内多核之间的资源竞争),通信带宽对图计算效率的影响.实验设置每个节点的计算力相同(即使用相同的 CPU),但是计算节点之间的通信带宽各不相同.实验选用三种网络拓扑方案(T_3,T_4,T_5).网络拓扑 T_3 由 4 台 1.8GHz CPU 的计算机组成,节点之间的带宽为 500 Mbps.网络拓扑 T_4 由 4 台 1.8GHz CPU 的计算机组成,其中两节点之间的带宽为 500 Mbps,另外两节点之间的带宽为 1 Gbps;网络拓扑 T_5 由 4 台 1.8GHz CPU 的计算机组成,其中两节点之间的带宽为 100 Mbps,另外两节点之间的带宽为 300 Mbps;详细拓扑信息在表 6 中列出.

表 6 拓扑信息

Topo	CPU(Number)	Network(Number)
T_3	1.8GHz (4)	500 Mbps(4)
T_4	1.8GHz (4)	500 Mbps(2), 1Gbps(2)
T_5	1.8GHz(4)	100 Mbps(2), 500 Mbps(2)

图 7 显示了两种算法在合成数据集 Twitter 上执行 PageRank 算法迭代十次所运行的时间.由于三种网络拓扑的网络带宽存在着差异,所以在这三种网络拓扑上执行 PageRank 的时间也不同.T_4 与 T_5 为异构网络,使用 HaSGP 算法划分之后的图计算运行时间平均要比原 LDG 算法减少约 6.2%.主要是因为原 LDG 算法的主要优化目标为最小化割边数,没有考虑到网络带宽的不同.这样可能会导致将割边都分配到了带宽较低的节点对之间.对于较低的网络带宽,即使传输相同的数据也会消耗较多的时间.

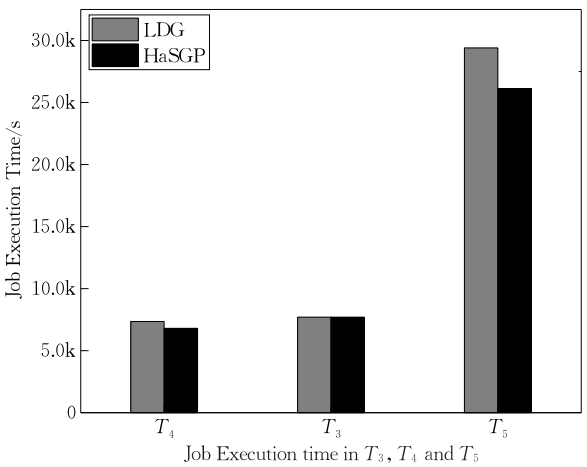


图 7 两种划分对不同拓扑网络的影响

由于 HaSGP 算法在划分过程中,考虑到了网络带宽的不同,尽量减少带宽低的节点对之间割边数的分配,将更多的割边分配到了带宽较高的节点对之间,必然提高了图计算效率.在网络拓扑 T_3 中,4 个节点之间的网络带宽相同,即使使用 HaSGP 算法,由于网络带宽的通信代价相同,使与 LDG 划分之后的图计算时间也相同.进一步说明了 HaSGP 算法考虑网络带宽异构的准确性.

同时我们也观察到在低速网络中通信成本对图计算效率的影响,虽然执行相同的任务,但是 T_3、T_4 与 T_5 存在明显的差异,主要由于 T_5 的网络带宽相对于 T_4 及 T_3 都很低,造成了数据传输的延迟,说明了提高网络带宽也能够提升整个图计算的效率.

5.4 共享资源竞争

本节主要验证高速网络并行环境中,HaSGP 算法能够有效减少分布式计算过程中节点内核与核之间的共享资源竞争.实验是由匹兹堡大学研究计算中心提供的由 32 个节点组成的集群接口.该集群具有扁平网络拓扑结构,所有计算节点通过 56 Gbps FDR Infiniband 连接到单个交换机.表 7 具体描述了集群的节点配置.

表 7 拓扑信息

Socket (2 intel Haswell Sockets)		Memory		
Cores/Socket	Clock speed	L3 cache	Capacity	Bandwidth
10	2.6GHz	25MB	128GB	65GB/s

实验测试了三种算法,分别是 LDG、HaSGP-L 和 HaSGP,其中 HaSGP-L 表示 HaSGP 不考虑节点内核之间的共享资源竞争,只考虑节点的计算能力与网络的带宽,即 $\alpha=0$,$\beta=0$.三种算法分别对图数据 com-orkut 划分.划分完毕,分别执行三种图算法(BFS、SSSP 和 PageRank),分别记录执行 BFS 算法从 10 个源点出发,遍历完整个图所需要的时间;执行 SSSP 算法分别从 10 个源点出发到图中其它节点的最短距离所运行的时间.执行 PageRank 算法迭代十次所消耗的时间.同时,实验过程中分别设置传输消息分组的大小分别为 64、128、256.

表 8 列出了运行时间结果.随着传输消息分组逐渐增大,三种划分方式分图计算执行的时间都在减小,主要由于随着消息分组的增加,减少了消息的分装同时也减少了缓存的失效率(表 9).HaSGP 在所有情况下的工作负载执行时间都是最少的.与 LDG、HaSGP-L 相比,HaSGP 方法分别加速了 BFS 执行时间的 1.95 倍和 4.21 倍,SSSP 执行时间的 2.73 倍和 5.41 倍,PageRank 的执行时间的 3.01 倍和 5.25 倍.HaSGP-L 在所有情况下表现最差,由于 HaSGP-L 没有考虑到节点内核之间的资源竞争,而且由于核之间的通信带宽高于节点之间的通信带

宽，所以按照其优化的目标是尽可能地将相邻顶点分配到一个计算节点内的不同核上. 这必然导致节点内数据通信量的增加，从而加剧内存子系统的争用. 然而，随着消息分组大小的增加，系统交换的消息次数逐渐变少，导致内存子系统上的正用减少，所以 HaSGP-L 和 LDG、HaSGP 之间的差距逐渐缩小.

表 8　使用不同分组大小所执行三种图算法(BFS、SSSP 和 PageRank)的时间　　(单位：s)

Method	BFS(10 个源点)			SSSP(10 个源点)			PageRank(10 次迭代)		
	64	128	256	64	128	256	64	128	256
LDG	272.30	67.00	21.00	6011	1051	146.00	2251.38	331.00	103.00
HaSGP-L	623.76	83.16	19.82	9528	2074	306.80	4179.00	361.67	67.09
HaSGP	**149.49**	**42.74**	**13.98**	**3126**	**412**	**104.48**	**813.67**	**143.91**	**35.73**

表 9　使用不同分组大小所执行三种图算法(BFS、SSSP 和 PageRank)的 LLC 失效率

Method	BFS(10 个源点)			SSSP(10 个源点)			PageRank(10 次迭代)		
	64	128	256	64	128	256	64	128	256
LDG	388	56.00	45.70	60187	3012	119	9215	143.00	87.12
HaSGP-L	3501	75.20	46.07	103548	16346	1171	34731	1526.00	71.30
HaSGP	**71**	56.95	**42.26**	**17421**	**319**	**101**	**298**	**95.54**	73.00

实验同时还记录了执行工作负载时最后一级缓存(Last Level Cache，LLC)的每百万数据包的失效情况. 如表 9 所示，LLC 未命中结果与表 8 计时结果基本一致. 由此我们可以得出两个结论：(1) HaSGP 在几乎所有情况下的 LLC 未命中率都是最低的，而 HaSGP-L 在大多数情况下的 LLC 未命中率都是最高的，同时也说明了在高速网络中，多核节点内共享资源竞争的存在确实降低了图计算的执行效率；(2) 随着消息分组逐渐增大，未命中率越低，三种图计算的时间就越短.

为了进一步确定资源争用的减少确实是由于节点内数据通信的减少造成的，实验分别统计了每个工作负载(BFS、SSSP、PageRank)执行时的通信量. 如图 8～图 10. *Intra-Socket*、*Inter-Socket* 和 *Inter-Worker* 分别表示同一插槽内部核之间的通信量、位于不同插槽但处于同一计算节点上的分

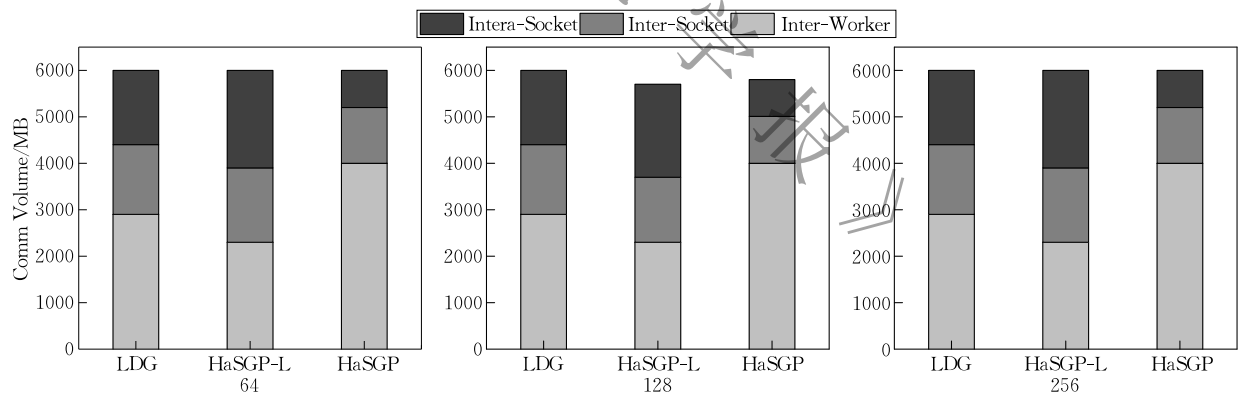


图 8　执行 BFS 任务中的通信数据量

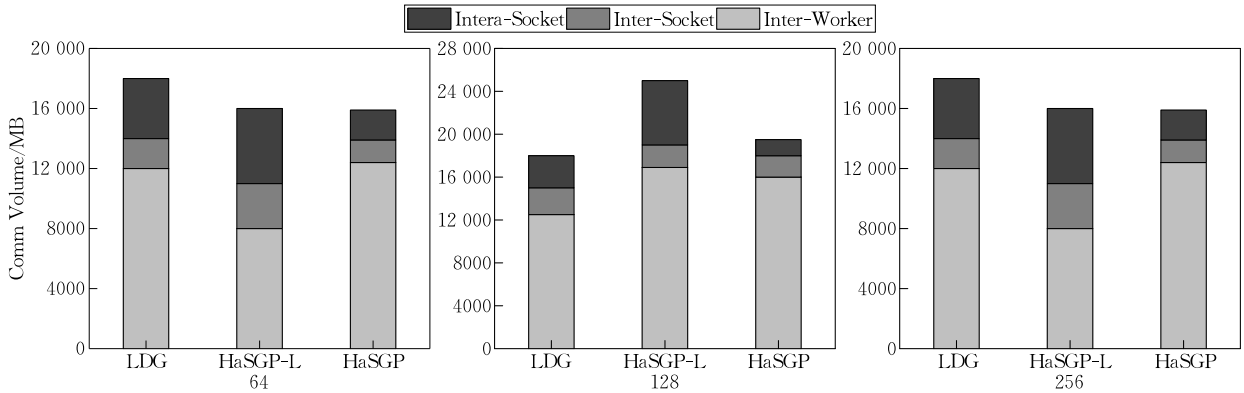


图 9　执行 SSSP 任务中的通信数据量

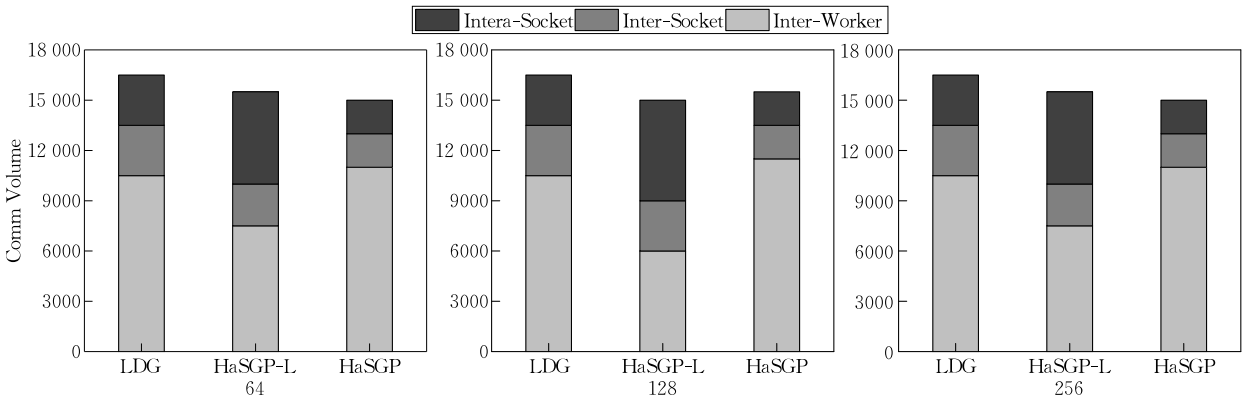


图 10 执行 PageRank 任务中的通信数据量

区之间的通信量以及不同计算节点之间的通信量. 在所有情况下,执行 HaSGP 算法的节点内数据通信量都是最低的,而 HaSGP-L 的节点内产生的数据通信量最高. 这也同样说明了 HaSGP-L 算法没有考虑到节点内部核之间的资源竞争,导致节点内部通信量的增加. 与 HaSGP-L 和 LDG 相比,HaSGP 方法分别将 BFS、SSSP 和 PageRank 算法在插槽内部产生的数据通信量分别降低了 70%和 40%、70%和 50%、70%和 50%. 所有这些结果都与表 8 的运行时间和表 9 的缓存失效相匹配.

综上所述,在高速网络中,由多核节点组成的集群环境中,节点内部核之间会导致资源竞争,影响分布式图计算的效率. 所以在研究异构环境下的图划分时,必须将子系统内的资源竞争因素考虑在内.

5.5 邻边结构的有效性

在本节中,主要验证邻边结构对流式划分算法执行效率的影响. 为了说明邻边结构的优越性,将邻边结构与邻点结构分别嵌入到 HaSGP 算法中,对 Twitter 图数据进行划分,对比不同结构所消耗的时间. 所划分的分区数分别设置为 80、100、120、140、160、180、200.

如表 10 所示,HaSGP(AVS)为基于邻点结构的流式 HaSGP 方法,HaSGP(AES)为基于邻边结构的流式 HaSGP 方法. 由于 HaSGP(AVS)算法主要是采用邻点结构的缓存方式,这种结构对于查找效率低下,每次分配当前顶点都需要与此点的邻点数量成正比的查找次数. 但是当采用邻边结构的流式划分 HaSGP(AES)时,划分时间明显降低,相较于 HaSGP(AVS),HaSGP(AES)方法划分时间平均减少了 13.4%. 说明了基于邻边结构的异构流式划分方法的优越性. 值得注意的是,没有特殊说明,本文所有其它实验所用的都是基于邻边结构的异构流式划分方法,即 HaSGP(AES),简写为 HaSGP.

表 10 在 Twitter 数据集上划分时间 (单位:min)

分区	HaSGP(AVS)	HaSGP(AES)
80	3.75	3.47
100	4.89	4.06
120	5.92	5.14
140	7.98	7.05
160	8.90	8.21
180	9.87	8.53
200	12.48	10.92

5.6 图计算性能

优化图划分算法的目标是提升分布式图计算的效率. 在本节中,采用不同图划分算法分别对图 Friendster 和 Twitter 进行划分,分析图划分对分布式图计算的影响.

实验配置了基于 InfiBand Ⅲ的高性能集群服务系统,包含 32 个节点,其中 10 核 Haswell CPU (Intel Xeon E5-2660 v3 2.6 GHz)12 个节点,4 核 Haswel-EP CPU(Intel Xeon E5-2630 v3 2.4 GHz) 20 个节点. 分区数分别设置为 80、100、120、140、160、180、200. 分别与文献[45]中的异构算法 Combined(CB)、Balanced Min-Increased (BMI)、Min-Increased(MI)以及 Min-Workload(MW)算法进行对比. 同时为了说明体系结构对图计算的影响,同样引入了 LDG 算法进行对比. 其中 HaSGP 模型中的参数 α 和 β 都设置为 1.

表 11~表 13 为在图 Friendster 上分别执行 BFS、SSSP 以及 PageRank 算法所运行的时间. 表 14~表 16 为在图 Twitter 上分别执行 BFS、SSSP 以及 PageRank 算法所运行的时间. 由表中的结果可以看出,在 HaSGP 划分之上执行的分布式图算法所运行的时间都是最短,虽然 CB、MBI、MI 以及 MW 算法考虑到了集群的异构性,但是它们忽略了高速网络环境下的子系统争用问题,而子系统的争用问题是高速网络环境下需要考虑的重要因素之一,极大

影响了分布式图计算的效率. 而 LDG 算法没有考虑到体系结构对分布式图计算的影响，所以在异构环境中图算法执行时间总是最长.

表 11　在图 Friendster 上执行 BFS 算法的时间（单位：s）

分区	BFS(5 Source Vertices)					
	LDG	CB	BMI	MI	MW	HaSGP
80	161.25	124.90	142.09	153.1	98.25	**79.10**
100	71.00	63.01	65.70	69.0	56.40	**37.72**
120	42.71	38.79	40.00	38.7	40.10	**19.16**
140	43.94	44.80	39.48	41.0	37.29	**23.00**
160	32.10	29.00	28.18	26.8	25.60	**21.69**
180	26.73	29.31	25.00	29.0	26.74	**19.50**
200	21.00	23.80	21.70	20.6	21.03	**18.47**

表 12　在图 Friendster 上执行 SSSP 算法的时间（单位：s）

分区	SSSP(5 Source Vertices)					
	LDG	CB	BMI	MI	MW	HaSGP
80	3249	3107	3201	2940	2748	**1691**
100	1817	1591	1507	1395	1207	**517**
120	891	671	612	793	519	**235**
140	374	280	317	334	274	**167**
160	269	231	230	246	240	**174**
180	217	189	191	208	147	**78**
200	182	148	149	176	132	**119**

表 13　在图 Friendster 上执行 PageRank 算法的时间（单位：s）

分区	PageRank(15 Iterations)					
	LDG	CB	BMI	MI	MW	HaSGP
80	758	691	672	598	517	**362**
100	362	307	348	271	209	**180**
120	250	218	215	223	210	**143**
140	172	153	150	147	134	**84**
160	141	110	107	129	107	**67**
180	83	74	71	71	70	**53**
200	61	54	54	59	57	**50**

表 14　在图 Twitter 上执行 BFS 算法的时间（单位：s）

分区	BFS(5 Source Vertices)					
	LDG	CB	BMI	MI	MW	HaSGP
80	284.66	214.25	208.60	200.90	197.00	**101.05**
100	126.10	113.08	107.00	109.10	105.70	**65.01**
120	86.08	76.92	75.80	72.18	60.70	**35.69**
140	78.10	67.08	65.17	60.90	57.80	**21.72**
160	38.01	31.00	29.07	28.00	28.07	**19.85**
180	25.00	25.07	24.89	23.61	24.00	**21.02**
200	29.60	30.91	29.70	27.00	26.41	**20.00**

表 15　在图 Twitter 上执行 SSSP 算法的时间（单位：s）

分区	SSSP(5 Source Vertices)					
	LDG	CB	BMI	MI	MW	HaSGP
80	6780	5971	5704	5591	5163	**4380**
100	3941	3180	3397	3507	2807	**1947**
120	2397	1983	1901	1678	1593	**834**
140	957	847	716	708	674	**390**
160	494	425	418	399	320	**163**
180	298	217	207	210	209	**147**
200	287	209	231	189	199	**129**

表 16　在图 Twitter 上执行 PageRank 算法的时间（单位：s）

分区	PageRank(15 Iterations)					
	LDG	CB	BMI	MI	MW	HaSGP
80	2309	2187	2109	2007	1907	**601**
100	677	597	580	559	551	**342**
120	490	391	407	381	307	**237**
140	271	217	204	201	208	**110**
160	188	156	163	139	134	**105**
180	129	106	102	99	98	**59**
200	127	113	110	105	109	**58**

综上所述，本文所提出的基于邻边结构的异构环境下的流式划分 HaSGP 能够有效地提升异构环境下图计算的效率.

5.7　参数设置

本节主要对模型中的参数 α 和 β 进行分析. 实验配置了基于 InfiBand Ⅲ的高性能集群服务系统，包含 32 个节点，10 核 Haswell CPU(Intel Xeon E5-2660 v3 2.6GHz)12 个节点，4 核 Haswel-EP CPU(Intel Xeon E5-2630 v3 2.4GHz)20 个节点. 对图 Friendster 划分之后分别执行 BFS、SSSP 以及 PageRank 算法，记录所运行的时间. 分区数设置为 80. 其中，β 设置为 0.1、0.2、0.3、0.4、0.5、0.6、0.7、0.8、0.9、1，相对应的 α 值设置为 0.1、0.3、0.5、0.7、1. 实验分析图划分模型中的不同 α 和 β 的值对分布式图计算性能的影响.

实验结果如图 11～图 13. 每一个 β 值所对应的不同 α 值，随着 α 值的不断增大，图计算所花费的时间越少. 主要是因为随着惩罚系数值的加大，可以更加有效地避免将相邻两点划分到竞争较大的两核中. 尽量将相邻两点划分到不同节点中. 这与低速网络环境的分配准则相反，在带宽较低的网络环境中，由于节点内核与核之间的通信所消耗的时间远小于

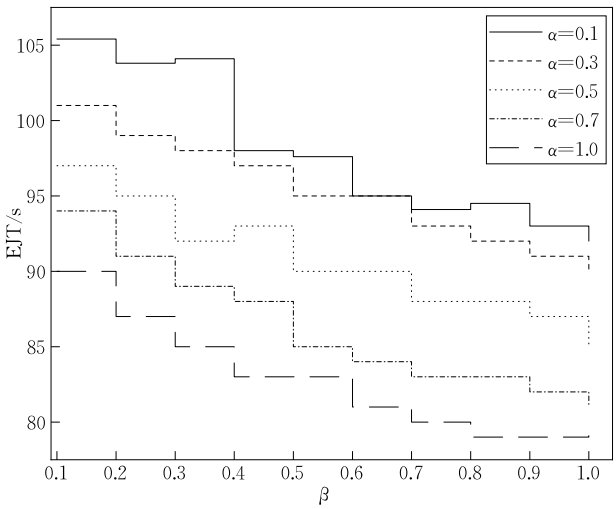


图 11　不同 α 和 β 值情况下，BFS 任务运行时间

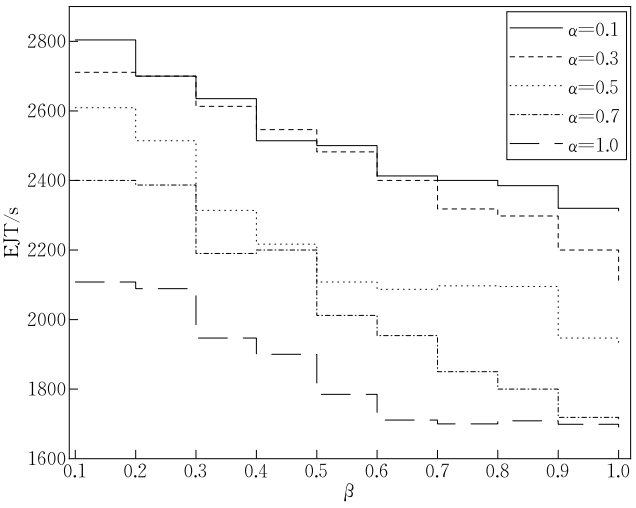


图 12 不同 α 和 β 值情况下,SSSP 任务运行时间

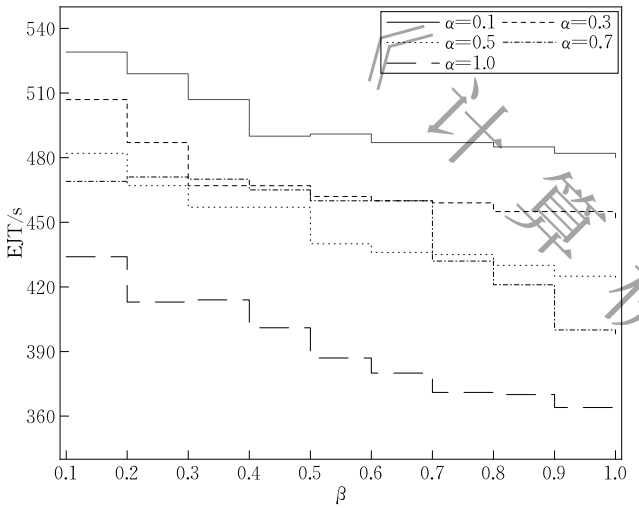


图 13 不同 α 和 β 值情况下,PageRank 任务运行时间

节点之间的通信时间,因此在低速网络环境中,尽量将邻点分配到同一节点中.

任何 $\alpha\in(0,1]$,$\beta\in(0,1]$中的值,都会考虑争用和通信异构性.当 α 和 β 都为 1 时,意味着内存子系统争用是最大的瓶颈,此时应该优先考虑通信的异构.考虑到资源争用和通信异构性的影响是高度依赖与应用程序和硬件的,用户需要在实际计算环境中对目标应用程序进行分析,以确定它们的最佳状态.通常情况下,对于高速网络的多核集群,α 和 β 都设置为最大值.

6 结 论

随着云计算技术的发展,集群环境中不可避免的会出现异构性.由于已有大部分图划分工作仅适用于同构环境,如果将这些算法应用在异构集群环境中任务的分配,可能会降低分布式图计算的效率.因此,开发针对异构环境的图划分对提升异构环境的图计算效率显得尤为重要.

本文针对此问题提出了一种异构感知的图划分算法 HaSGP,该算法不仅考虑了计算节点的运算力,网络通信的带宽,而且也考虑了以 InfiniBand 为代表的现代高速网络组建的多核集群中存在着核之间共享资源竞争的问题.最后实验从不同方面证明了 HaSGP 算法在异构环境下的适用性与优越性.

参 考 文 献

[1] Zhou S, Kannan R, Prasanna V K, et al. HitGraph: High-throughput graph processing framework on FPGA. IEEE Transactions on Parallel & Distributed Systems, 2019, 30(10): 2249-2264

[2] Kumar D, Raj A, Dharanipragada J, et al. GraphSteal: Dynamic re-partitioning for efficient graph processing in heterogeneous clusters//Proceedings of the International Conference on Cloud Computing. Honolulu, USA, 2017: 439-446

[3] Xu T, Wang G. Finding strongly connected components of simple digraphs based on generalized rough sets theory. Knowledge Based Systems, 2018, 148: 88-98

[4] Harris D G. Some results on chromatic number as a function of triangle count. SIAM Journal on Discrete Mathematics, 2019, 33(1): 546-563

[5] Li Q, Zhong J, Zheng L J, et al. Streaming graph partitioning for large graphs with limited memory//Proceedings of the International Symposium on Parallel and Distributed Processing with Applications. Guangzhou, China, 2017: 1269-1271

[6] Wang Tong-Tong, Rong Chui-Tian, Lu Wei, et al. Survey on technologies of distributed graph processing systems. Journal of Software, 2018, 29(3): 569-586(in Chinese)
(王童童, 荣垂田, 卢卫等. 分布式图处理系统技术综述. 软件学报, 2018, 29(3): 569-586)

[7] Andreev K, Racke H. Balanced graph partitioning//Proceedings of the ACM Symposium on Parallel Algorithms and Architectures. New York, USA, 2004: 120-124

[8] Masood S, Sheng B, Li P, et al. Automatic choroid layer segmentation using normalized graph cut. IET Image Processing, 2018, 12(1): 53-59

[9] Abdelhamid E, Canim M, Sadoghi M, et al. Incremental frequent subgraph mining on large evolving graphs. IEEE Transactions on Knowledge and Data Engineering, 2017, 29(12): 2710-2723

[10] Cannon J W, Thurston W P. Group invariant Peano curves. Geometry & Topology, 2007, 11(3): 1315-1355

[11] Preen R J, Smith J. Evolutionary n-level hypergraph partitioning with adaptive coarsening. IEEE Transactions on Evolutionary Computation, 2019, 23(6): 962-971

[12] 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

[13] Tsourakakis C E, Gkantsidis C, Radunovic B, et al. Fennel: Streaming graph partitioning for massive scale graphs// Proceedings of the Web Search and Data Mining. New York, USA, 2014: 333-342

[14] Zheng A, Labrinidis A, Chrysanthis P K. Planar: Parallel lightweight architecture-aware adaptive graph repartitioning //Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering. Helsinki, Finland, 2016: 121-132

[15] Barvinok A, Soberon P. Computing the partition function for graph homomorphisms. Combinatorica, 2017, 37(4): 633-650

[16] Yezerska O, Pajouh F M, Veremyev A, et al. Exact algorithms for the minimum s-club partitioning problem. Annals of Operations Research, 2019, 276(14): 1-25

[17] Soper A J, Walshaw C, Cross M. A combined evolutionary search and multilevel optimisation approach to graph-partitioning. Journal of Global Optimization, 2004, 29(2): 225-241

[18] Onizuka M, Fujimori T, Shiokawa H, et al. Graph partitioning for distributed graph processing. Data Science and Engineering, 2017, 2(1): 94-105

[19] Zhou S, Lakhotia K, Singapura S G, et al. Design and implementation of parallel PageRank on multicore platforms //Proceedings of the IEEE High Performance Extreme Computing Conference. Waltham, USA, 2017: 1-6

[20] Mcgregor A. Graph stream algorithms: A survey. SIGMOD Record, 2014, 43(1): 9-20

[21] Wang C, Liang Q, Urgaonkar B, et al. An empirical analysis of Amazon EC2 spot instance features affecting cost-effective resource procurement. ACM Transactions on Modeling and Performance Evaluation of Computing, 2018, 3(2): 1-24

[22] Ren Shen-He, Zheng Kou-Quan, Guan Dong-Dong, et al. The dynamic load balancing method of cloud computing network based on intuitionistic fuzzy time series. Systems Engineering-Theory & Practice, 2019, 39(5): 1298-1307(in Chinese)
(任神河，郑寇全，关冬冬等. 基于IFTS的云计算网络动态负载均衡方法. 系统工程理论与实践，2019，39(5)：1298-1307)

[23] Hu J, Huang J, Lv W, et al. CAPS: Coding-based adaptive packet spraying to reduce flow completion time in data center. IEEE/ACM Transactions on Networking, 2019, 27(6): 2338-2353

[24] Liu S, Huang J, Zhou Y, et al. Task-aware TCP in data center networks. IEEE/ACM Transactions on Networking, 2019, 27(1): 389-404

[25] Sotiriadis S, Bessis N, Anjum A, et al. An inter-cloud meta-scheduling (ICMS) simulation framework: Architecture and evaluation. IEEE Transactions on Services Computing, 2018, 11(1): 5-19

[26] Liu Ying-Wen, Tang Yu-Hua, Yi Xiao-Dong. Low latency InfiniBand interface for distributed memory access. Journal of Computer Research and Development, 2014, 51(S1): 123-129(in Chinese)
(刘英文，唐玉华，易晓东. 面向分布式内存访问的低延迟InfiniBand接口. 计算机研究与发展，2014，51(S1)：123-129)

[27] Wang Z, Gu Y, Bao Y, et al. Hybrid pulling/pushing for I/O-efficient distributed and iterative graph computing// Proceedings of the International Conference on Management of Data. San Francisco, USA, 2016: 479-494

[28] Yu W, Wang Y, Que X, et al. Design and evaluation of network-levitated merge for hadoop acceleration. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(3): 602-611

[29] Shudler S, Berens Y, Calotoiu A, et al. Engineering algorithms for scalability through continuous validation of performance expectations. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(8): 1768-1785

[30] Stanton I, Kliot G. Streaming graph partitioning for large distributed graphs//Proceedings of the Knowledge Discovery and Data Mining. Beijing, China, 2012: 1222-1230

[31] Wang W, Du S, Guo Z, et al. Polygonal clustering analysis using multilevel graph partition. Transactions in GIS, 2015, 19(5): 716-736

[32] Rahimian F, Payberah A H, Girdzijauskas S, et al. JA-BE-JA: A distributed algorithm for balanced graph partitioning// Proceedings of the International Conference on Self-adaptive and Self-organizing Systems. Philadelphia, USA, 2013: 51-60

[33] Chen R, Shi J, Chen Y, et al. PowerLyra: Differentiated graph computation and partitioning on skewed graphs. ACM Transactions on Parallel Computing, 2019, 5(3): 1-39

[34] Washburne A D, Silverman J D, Morton J T, et al. Phylofactorization: A graph partitioning algorithm to identify phylogenetic scales of ecological data. Ecological Monographs, 2019, 89(2): e01353

[35] Chen Q, Yao J, LI B, et al. PISCES: Optimizing multi-job application execution in MapReduce. IEEE Transactions on Cloud Computing, 2019, 7(1): 273-286

[36] Catalyurek U V, Boman E G, Devine K D, et al. A repartitioning hypergraph model for dynamic load balancing. Journal of Parallel and Distributed Computing, 2009, 69(8): 711-724

[37] Dathathri R, Gill G, et al. Gluon: A communication-optimizing substrate for distributed heterogeneous graph analytics. Association for Computing Machinery, 2018, 53(4): 752-768

[38] Xue J, Yang Z, Hou S, et al. When computing meets heterogeneous cluster: Workload assignment in graph computation//Proceedings of the International Conference on Big Data. Santa Clara, USA, 2015: 154-163

[39] Il'yasov B G, Saitova G A. Stability analysis of dynamic systems in the polynomial vector-matrix representation. Journal of Computer & Systems Sciences International, 2018, 57(2): 171-178

[39] Moulitsas I, Karypis G. Architecture aware partitioning algorithms//Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing. Agia Napa, Cyprus, 2008: 42-53

[40] Kalavri V, Vlassov V, Haridi S, et al. High-level programming abstractions for distributed graph processing. IEEE Transactions on Knowledge and Data Engineering, 2018, 30(2): 305-324

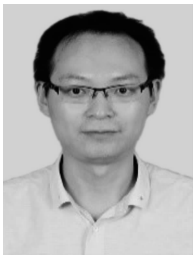
[41] Busse A, Schonherr J H, Diener M, et al. Analyzing resource interdependencies in multi-core architectures to improve scheduling decisions//Proceedings of the ACM Symposium on Applied Computing. Coimbra, Portugal, 2013: 1595-1602

[42] Sajjad H P, Payberah A H, Rahimian F, et al. Boosting vertex-cut partitioning for streaming graphs//Proceedings of the IEEE Big Data Congress. San Francisco, USA, 2016: 1-8

[43] Abbas Z, Kalavri V, Carbone P, et al. Streaming graph partitioning: An experimental study. Proceedings of the Very Large Data Bases Endowment, 2018, 11(11): 1590-1603

[44] Hua Q S, Li Y, Yu D, et al. Quasi-streaming graph partitioning: A game theoretical approach. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(7): 1643-1656

[45] Xu N, Cui B, Chen L, et al. Heterogeneous environment aware streaming graph partitioning. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(6): 1560-1572



LI Qi, Ph. D., lecturer. His current research includes include graph data mining and graph calculation.

LI Hu-Xiong, Ph. D., professor. His research interests include network system analysis and control, image recognition and understanding.

ZHONG Jiang, Ph. D., professor, Ph. D. supervisor. His research interests include data mining, parallel computing and natural language processing.

YING Chang-Tian, Ph. D., lecturer. His research interests include memory computing.

Li QING, Ph. D. His research interests include natural language processing.

Background

Graph partitioning is becoming increasingly challenging as graphs grow in size. The graphs consist of terabytes of compressed data when stored on disks and are all far too large for a single commodity type machine to efficiently perform computations. Currently, the main use of distributed cluster systems to deal with large graph partitioning, although distributed computing resources have become more accessible, but the development of distributed partitioning algorithms still has challenges, especially for non-experts.

The streaming partitioning has been applied in graph partitioning in recent years because it is more efficient than offline partitioning. Because of efficient partition management, the streaming partition has been developed continuously in recent years. However, there are two problems in streaming method: (1) The original method does not consider the heterogeneity of the cluster computing environment. When there is heterogeneity in the cluster, the optimized goal of the existing streaming partition method may lead to the degradation of parallel computing performance, Therefore, the existing streaming partition method is not suitable for task allocation in heterogeneous cluster environment. (2) The cache data structure of neighbor vertices in the process of partitioning is not well for searching, and it is inefficient for searching (mainly referring to searching the information of allocated neighbor vertices and their subsets).

In view of the above two problems, we propose that the streaming partition algorithm in the heterogeneous environment can adapt to the heterogeneity of the parallel environment. At the same time, aiming at the problem of the streaming algorithm (2), we also design the adjacent edge cache structure to improve the search efficiency. The specific process is described in detail in the following sections.

This work is supported by the Youth Program of National Natural Science Foundation of China (62002226), the National Natural Science Foundation of China (6194100039), the Natural Science Foundation of Zhejiang Province (LHQ20F020001), the Basic Public Welfare Research Project of Zhejiang Province (LGG18F030003) and the Research Results of University Level Scientific Research Projects of Shaoxing University of Arts and Sciences (2019LG1004).