一种改进的基于 BSP 的大图计算模型

赵 翔[1),2)] 李 博[1)] 商海川[3)] 肖卫东[1),2)]

[1)](国防科学技术大学信息系统与管理学院 长沙 410073)

[2)](地球空间信息技术协同创新中心 武汉 430079)

[3)](东京大学工业科学研究院 东京 日本 153-8505)

摘 要 伴随大数据的涌现,云存储和计算技术近年得到长足发展.图数据是一种重要而普遍的大数据,在生物信息学、社会网络、化学信息学等领域都有众多应用.因此,大图计算作为大数据分析应用的典型代表,正成为云端负载的重要组成部分.目前,高可扩展性的图计算主要依赖于高性能计算解决方案,需要进行环状(或网状)计算机网络之上的高效全集合通信.然而,在通用计算集群和云计算基础设施上实现基于环状计算机网络的算法时,低效的网络通信将导致巨大的系统延迟.因此,这就要求那些基于云端的大数据计算平台和系统具备十分良好的水平可扩展性.但是,大图的幂律分布和缺乏局部性使得设计一套高度可扩展的大图计算系统变得更具挑战.为此,文中提出了一种面向通用计算集群的可扩展大图计算模型.专注于水平扩展能力,设计了一种新颖的基于分离器-合并器 BSP 的图计算方法,能够提供原生的负载平衡,仅需很低的通信开销.从而,图数据规模的增大可以通过增加计算节点数量得以解决.最后,在一个图数据通用测试集上,通过大量实验验证了所提模型和方法的有效性和高效性;结果显示,相比经典的以顶点为中心的 BSP 大图计算模型和其他主流大图计算系统,所提改进的基于 BSP 的大图计算模型能够提供更好的水平可扩展性.

关键词 BSP 模型;大图;水平扩展能力;图分割;通用集群

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2017.00223

A Revised BSP-Based Massive Graph Computation Model

ZHAO Xiang[1),2)] LI Bo[1)] SHANG Hai-Chuan[3)] XIAO Wei-dong[1),2)]

[1)](*College of Information System and Management, National University of Defense Technology, Changsha* 410073)

[2)](*Collaborative Innovation Center of Geospatial Technology, Wuhan* 430079)

[3)](*Institute of Industrial Science, The University of Tokyo, Tokyo* 153-8505, *Japan*)

Abstract Along with the emergence of big data, cloud storage and computing technology lately has been developing rapidly. Graph data is an important and pervasive type of big data, which find a wide spectrum of applications, including bio-informatics, social networks, chem-informatics, etc. Massive graph computation, as a typical representative of big data analytics and application, is becoming an important part of workloads on the cloud. Currently, scalable graph computation mainly resorts to high performance computing solutions, which requires high performance all-to-all collective communication over torus or mesh networks. However, implementing these torus or mesh-based algorithms on commodity clusters and cloud computing infrastructures may result in high latency, due to inefficient network communication. Thus, this requires those cloud-based platforms and systems to be of good scale-out capabilities. However, the vertex degree skewness and lack of locality on massive graphs further challenge the design of a highly scalable system. To

收稿日期:2015-09-21;在线出版日期:2016-05-08.本课题得到国家自然科学基金(61402494,61402498)、湖南省自然科学基金(2015JJ4009)资助.**赵 翔**,男,1986 年生,博士,讲师,中国计算机学会(CCF)会员,主要研究方向为图数据管理与挖掘、智能情报分析等.E-mail:xiangzhao@nudt.edu.cn.**李 博**,男,1994 年生,硕士研究生,主要研究方向为大数据分析.**商海川**,男,1984 年生,博士,副研究员,主要研究方向为图数据管理与分析、并行计算等.**肖卫东**,男,1968 年生,博士,教授,博士生导师,主要研究领域为情报大数据分析、社会计算等.

resolve the issues, we exploit a commodity cluster-oriented programming model for scalable graph computation. Focusing on scale-out capability, we propose a novel separator-combiner BSP-based graph computation method, which provides native load-balancing and low communication overhead. Thus, larger graphs can be addressed by adding more computing nodes. Finally, on a general testing benchmark for big graph data, extensive experiments confirm the effectiveness and efficiency of the proposed model and method. The results demonstrate that the proposed revised BSP-based massive graph computation model provides better scale-out capability, in comparison with the classic vertex-centric BSP-based graph computation model and other major massive graph commutating systems.

Keywords BSP model; massive graphs; scale-out capability; graph partition; commodity cluster

1 引 言

后信息时代，数据爆炸式增长使大数据分析应用相关的研究俨然成为业界的一个关注重点，期望给种类繁多的海量数据量身打造专有的并行计算平台与服务. 此类计算的广泛应用亦改变了可扩展并行计算研究的现状. 此前，大规模计算主要依靠高性能计算(high performance computing)技术实现大规模数据处理与分析；现在，业界则逐渐将目光转向了性价比更高的大数据计算平台. 大数据计算平台和高性能计算平台的主要区别在于：(1) 大数据计算平台依赖通用计算机群(commodity computing cluster)的存储和计算基础设施；(2) 云计算的效用模型使得计算资源可以由用户按需定义. 正是因为这两点显著区别，就要求那些基于云端的大数据计算平台和系统具备良好的水平扩展能力.

作为最流行的数据密集型应用之一，包括社会网络在内的大图计算正亟需一套高效且可扩展的解决方案. 然而，目前主流的大数据计算框架 MapReduce 以及相关技术(Pig 和 Hive 等)却无法很好地胜任大规模图数据上的迭代计算. 因此，若干大图并行计算模型应运而生，有的面向大数据计算平台，有的则基于高性能计算技术[1]. 不同的计算模型，意味着不同的通信网络要求以及不同的图划分方法. 所谓大图计算的图划分任务，即将大图 G 的顶点和边分成若干个部分，分布式存储在参与大图计算的工作者(workers)上. 根据图划分方法的不同，可将已有大图计算模型和系统分为两类：

(1) 一维划分. 根据顶点出边(out-going edges)对顶点进行划分，主要以 Pregel[2] 为代表. Pregel 是一种基于以顶点为中心(vertex-centric)的 BSP[3](bulk synchronization parallel)模型设计的大图计算框架. Pregel 计算框架的主要优势在于能够衍生一系列简单易用的编程接口，降低分布式大图算法实现所需的人力和时间，毕竟让企业 IT 部门开发优化完善的原生代码(native code)是不现实的. 但是，真实世界的图多数服从幂律分布[4-5]，即超过 80% 的绝大多数顶点只有少量的边，但极少数顶点却拥有极多的边. 在分布式处理这类网络时，负责高度数顶点的工作者相比其他工作者需要消耗更长的时间，进而延长了系统的整体响应时间. 因此，基于一维划分的并行计算模型缺乏面向计算节点数量的良好的水平扩展能力；换言之，由于处理最高度数顶点的工作者所需的处理时间决定了系统的响应时间，那么增加集群中的计算节点的数量将无法进一步降低整个系统的响应时间.

(2) 二维划分. 其中每个顶点都由一组计算节点进行管理. 每一步在同一组中的计算节点通过相互通信对接收到的消息进行聚合，然后通过出边向其他组发送消息，供下一步计算使用. 同组顶点的通信由 AllReduce 或 AllGather 方法实现[6-7]，这类操作在提供可扩展性的同时，通常要求特殊的高性能计算环境，譬如网状计算机通信网络等. Graph 500 基准测试①说明，基于高性能计算技术的并行算法可以水平扩展到 6 万余个计算节点，但这类方法的缺点亦显而易见——巨大的网络通信开销. 诸如 AllReduce 和 AllGather 等行/列集合通信(collective communication)方法的高效实现是保证高性能和高可扩展性的关键[8-9]，而它们都强烈依赖于高效但昂贵的集合通信操作. 此外，实现基于二维划分的图算法需要更多的人力和时间，这违背了大数据分

① The graph 500 list: http://www.graph500.org/

析应用所追求的简单易用的理念.

综上，当前大图计算系统可采用如下 3 种实现方案，总结如表 1 所示.

表 1　大图计算可用解决方案

	Degree-skew	Commodity	Seale-out
1D (Commodity)	×	√	√
2D (HPC)	√	×	√
2D (Commodity)	√	√	×

(1) 基于 MapReduce 或以顶点为中心的 BSP 模型，譬如 Hadoop 平台上的 Apache Giraph① 和 Spark 平台上的 GraphX②. 这种方案易于实现，但不能很好地处理真实世界中服从幂律分布的大型网络；

(2) 基于高性能计算技术. 这种方案能够提供良好的系统性能，但是需要配备高级通信网络的昂贵 HPC 集群；

(3) 在通用计算集群上实现集合通信操作，如 PowerGraph[10]. 这种方案可在小规模集群上缓解幂律分布带来的问题，但是在大规模集群上集合通信的开销问题依旧严重，可能急剧降低系统处理大型网络的性能.

通过上述综合与分析，认为面向大规模通用计算集群的大图计算模型应解决至少如下 3 个方面的挑战：(1) 幂律分布. 在处理服从幂律分布图时应尽可能保证负载均衡；(2) 通用计算基础设施. 不能依赖于昂贵的高级通信网络；(3) 水平扩展能力. 通过增加计算节点就能提升系统处理更大规模图的能力. 然而，目前暂无可用的方法能同时满足上述要求.

鉴于此，文中尝试提出一种基于分离器(separator)和组合器(combiner)的 BSP 大图计算模型，基于该模型设计的大图计算方法能够同时满足上述 3 个要求. 此外，该模型还具备下列特点：

(1) 可无缝集成到包括 Apache Hadoop 在内的所有面向通用计算集群的解决方案中；

(2) 处理方法考虑了各种大图计算任务的通用性；

(3) 针对服从幂律分布的大型网络具有良好的水平扩展能力；

(4) 通过非集合通信进行同步，降低并发网络通信开销的影响；

(5) 传承 BSP 的一对一通信模式，保证系统的可靠性和容错性.

本文第 2 节介绍相关工作和背景知识；第 3 节提出一种改进的大图计算模型，介绍其划分和计算方法；第 4 节简述系统实现；实验结果及分析在第 5 节中；结论和下一步工作计划则放在第 6 节.

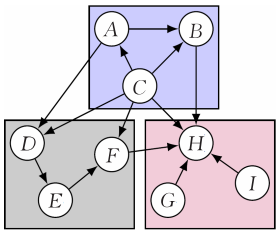
2　相关背景

为了不失通用性，文中着重讨论简单有向图，即无环无多重边的有向图. 其中，顶点可具有出边(outgoing edges)和/或入边(in-coming edges)，一个顶点的度数(出度)等于其出边的数量. 给定一个顶点，与其出边相连的顶点称为该顶点的外向邻居顶点.

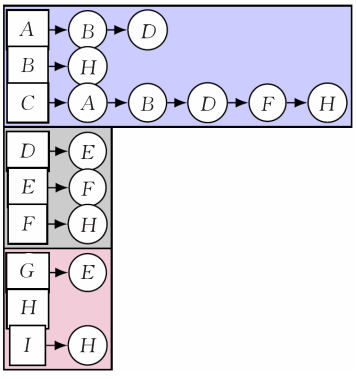
本节首先简要回顾有关一维划分和二维划分上的大图计算方面的相关工作，关于大图处理系统更加详细的综述和横向实验评价，感兴趣的读者可参考文献[11-13]. 它们都很好地总结了当前的研究成果，是了解大图计算系统的重要参考. 然后，介绍以顶点为中心的 BSP 模型，它是后续章节提出和改进模型的基础.

2.1　一维划分

一维划分是分布式环境下最流行的图划分方法. 顶点依据其出边被划分到不同的工作者上. 例如，图 1(a)和 1(b)分别描绘了一个示例图及其划分和对应的分布式邻接表. 在此例中，示例图的顶点被划分成 3 组：{“A”，“B”，“C”}，{“D”，“E”，“F”}和{“G”，“H”，“I”}③.



(a) 示例图及其划分



(b) 对应的分布式邻接表

图 1　一维划分示例

① Apache Giraph：http://giraph.apache.org/
② Spark GraphX：https://spark.apache.org/graphx/
③ 双引号加大写字母表示顶点标签，小写字母表示顶点标识符.

基于一维图划分进行并行图计算的并行BGL[14]是一个面向研究开发的通用C++库,它在Boost Graph Library之上实现了一维划分方法,使用消息传递接口(Message Parsing Interface,MPI)在BSP模型[3]中进行通信.

Pregel[2]是目前最流行的大规模图计算框架,其设计严格参照了以顶点为中心(vertex-centric)的BSP(VC-BSP)模型.在Pregel中,主机节点(master)主要用于工作者(workers)之间的协同,也参与聚合器(aggregators)的运算,实现全局通信.GPS[15]将聚合器拓展为一个"主机-工作者"的运算机制,后被Apache Giraph项目采用.在Apache Giraph设计的API中,主机节点的MasterCompute类具有*compute*()函数,用户可以通过实例化该函数来读取和改变在上一个超步中聚合器的值;此外,每个聚合器都被分派到一个工作者上,这样主机不需要进行任何聚合计算.文献[16]提出监控活动子图大小的策略,即当它变得足够小时,将其发送到主机上,串行执行剩余计算;此外,还讨论了将顶点合并为超顶点(supervertices)、按需清理边等基于Pregel计算框架的图算法优化技术.归纳来说,上述系统或框架均采用了"顶点-顶点"的消息通信模式,即"以顶点为中心"来考虑消息的传递和数据的更新.

鉴于VC-BSP模型不适合需要协同的图数据计算应用(如图着色等),最近Giraphx[17]提出将顶点分成内部点和边界点两类,并对BSP模型改进以满足前述的应用需求.这种改进包括允许直接访问工作者上的内点;对于边界点,采取用餐哲学家(dinning philosophers)和令牌环(token ring)两种协同机制实现同步等.

注意到,VC-BSP模型虽然易于实现,但牺牲了顶点访问与其在同一个工作者上的其他顶点信息的灵活性.因此,最近的研究,譬如Giraph++[18]和VB-Partitioner[19]等,利用同一工作者内部信息共享,提高本地工作者内部的通信效率.这类方法的主要思想是将同一个工作者上的顶点的计算放在一个*compute*()函数中进行.通过精心算法设计,这种方法可以取得一定程度的性能提升,但编程实现相对复杂,其易用性相比VC-BSP模型被打了折扣.

幂律图在现实世界中十分常见.在处理幂律图时,单个工作者中的高度数顶点很可能导致系统的性能瓶颈.为此,PowerGraph[10](及GraphLab[20])提出用随机化的边划分方法来平衡工作量.由于每个点的出边存储于多个工作者上,同步操作需要聚合在不同工作者上所有顶点的值.由于聚合操作是一个all-to-all的集合通信,并且聚合器的规模和顶点数是成比例的,因此这种方法针对计算节点数量的水平扩展能力是十分有限的.区别于经典BSP的同步策略,最近GiraphUC[21]提出一种BAP(Barrierless Asynchronous Parallel)同步模型,减少消息滞后和全局通信,提升整体性能.此外,其他采纳异步执行的并行方案的研究工作还包括针对动态图的SpecGraph[22]等.

另外,诸如微软的Trinity[23]和Facebook的TAO[24]等系统,与上文讨论的大图计算模型和框架也有较大不同.它们主要服务于大规模图数据上的在线查询任务,强调低读取延迟(latency)和高读取可用性(availability).此类查询任务与高吞吐量的离线处理有着迥异的差别,因此超出了本文的研究讨论范畴.

2.2 二维划分

二维划分首次是在文献[7]中提出,后被文献[6]采用.二维划分通常建立在一个网状网络之上,一组顶点的出边被网状网络中的一行(row)计算节点所共有.二维划分方案通过增加通信开销来解决载入平衡和可扩展性的问题.譬如,经典的二维BFS算法需要扩展(expand)和折叠(fold)两个步骤,其中折叠的通信开销等于一维划分上BFS算法的总通信开销,扩张的通信开销则是为载入平衡和可扩展性付出的额外开销.

由于缺乏高性能计算网络以及关键算法操作的高性能实现,在通用计算集群和云计算基础设施上部署这些算法会引发很高的通信代价,导致很低的时间效率.此外,实现基于二维划分的图算法需要很大的人力成本,这违背大数据分析应用关于易用性的理念.

注意到,在MapReduce上还有一系列关于图计算的研究(如Pegasus[25]等),它们依赖于Map和Reduce操作进行图数据处理.然而,已有研究通常认为,MapReduce计算框架并不适合处理图结构数据[23],强求并发性可能导致非最优的性能及易用性方面的问题.因此,基于MapReduce方法亦不属于本文的探讨范围.

2.3 VC-BSP模型

BSP模型使用超步(superstep)的概念使多个并行的工作者保持同步.一个典型的超步一般由3个部分组成:

(1)计算.每个工作者执行其管辖顶点上的

计算；

(2)通信. 工作者代表其管辖顶点向它们的邻居顶点发送消息；

(3)屏障(barrier). 每个工作者等待其他所有工作者完成计算和通信.

之后，算法进入下一个超步. 也就是说，一个并行算法通常包含了一系列依次执行的超步. 参照以顶点为中心的 BSP 大图计算模型，图 2 展示了一个基于图 1(a)划分的 BFS(Breadth First Search)算法执行过程.

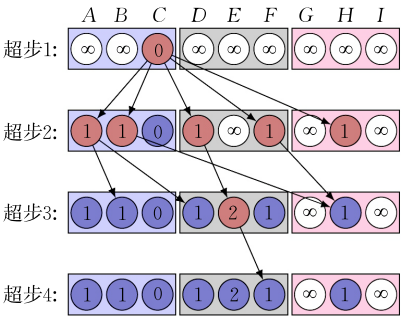


图 2　基于一维划分的 BFS 算法执行过程

假定 BFS 算法从根顶点“C”开始执行，其取值初始化为 0，即它到根顶点的距离为 0；剩余顶点取值初始化为无穷大. 在超步 1，管辖顶点“C”的工作者发送包含距离 1 的消息给“C”的所有出边指向的外向邻居顶点. 这些收到“C”的消息的顶点更新自身取值为 1，即它到根顶点的距离是 1. 然后，它们发送包含距离 2 的消息给各自的外向邻居顶点. 依此迭代，倘若一个顶点收到消息值小于其当前值，它就更新自身取值，并向其外向邻居顶点发送包含自身取值加 1 的消息. 直至全局都没有更新时，BFS 算法停止并退出.

当前，绝大多数图计算框架都采用了 VC-BSP 模型. 然而，真实世界中服从幂律分布的大型网络中，极少数顶点拥有大量的边. 通过上述 BFS 算法执行过程示例可以发现，在分布式处理这类网络时，按照 VC-BSP 模型，负责高度数顶点的工作者通常需要执行更多的顶点计算任务，处理更多的接收消息. 因此，相比其他工作者，它们需要消耗更长的时间，极易成为整个系统的瓶颈. 此外，其网络通信和内存方面的限制将直接制约系统处理更大型的图(参见第 5 节实验结果). 为解决上述问题，下一节将提出一种改进的基于 BSP 的大图计算模型，改善系统面向真实幂律网络的处理性能和水平可扩展能力.

3　SC-BSP 模型

本节提出一种改进的基于 BSP 的 SC-BSP 大图计算模型，它采用一种新的图划分和计算方法来解决前文提出的幂律分布和水平可扩展性问题，并兼容当前主流的通用计算和存储基础设施.

具体地说，首先提出了分离器(separator)的概念，分离器是面向顶点出边子集的一种容器(container)，图(边)的划分由分离器进行控制. 然后，设计了一种基于分离器-组合器(separator-combiner)的 BSP 大图计算模型，简称 SC-BSP 模型，进行图处理计算. 注意到，SC-BSP 模型和 VC-BSP 模型都是依赖 BSP 的大图计算模型，基于这些模型可以进一步设计实现相应的计算框架. 下面介绍 SC-BSP 模型的划分和计算方法的基本原理，实现相关的细节则在第 4 节中.

3.1　划　分

顶点首先由用户定义的划分函数划分到各工作者上，默认的划分函数是 $hash(ID)$ 再对 N 取模，其中 ID 是顶点的标识符，N 是所需划分的数量.

边的划分则通过一种称作分离器的容器(container)实现，一个分离器容纳了某个顶点出边的一部分出边(子集). 首先，设计了两种类型的分离器来负责顶点的出边：前分离器(pre-separator)和后分离器(post-separator). 一个顶点的所有出边要么被前分离器管理，要么由后分离器管理，且只能被一种分离器管理. 然后，给出了一种简单的基于顶点度数的选择机制，即给定一个分离器选择阈值，度数高于阈值的高度数顶点由后分离器管理，度数低于阈值的低度数顶点则由前分离器管理.

前分离器的工作原理与现有一维划分方法类似，将消息根据出边(和消息传递目的顶点)的不同进行分组. 因此，前分离器存在于管辖的出边对应的顶点所在的工作者内存中.

相对而言，后分离器稍复杂一些. 假如一个顶点 v 由后分离器管理，那么后分离器就会出现在与其相连的外向邻居顶点所在的工作者内存中. 换句话说，至多可以有 N 个不同的后分离器，分别对应 N 个工作者，其中 N 是系统中的工作者数量. 每一个后分离器控制 v 出边的一个子集，同一个后分离器控制的出边均指向位于同一个工作者上的 v 外向邻居顶点，该后分离器直接位于这些外向邻居顶点所在的工作者上. 举个例子，假如 v 是一个由后分离器

负责的顶点，其出边集为$\{vu, vw, vx, vy, vz\}$，其中v的外向邻居顶点u和w位于同一个工作者上，x、y和z位于另一个工作者上，那么需要设置两个后分离器分别管理出边子集$\{vu, vw\}$和$\{vx, vy, vz\}$，这两个后分离器分别位于u和w所在的工作者，以及x，y和z所在的工作者上. 因此，后分离器控制的出边集合之间是互不重叠的，且一条出边只能隶属于一个后分离器.

考虑图3展示的一张有向带权重具体网络，边权重标示在有向边上的尖括号内. 首先，一个用户定义的哈希函数将顶点划分为两组：{“A”，“B”，“C”}和{“D”，“E”，“F”}. 通过评估每个顶点的度数确定其分离器的类型；在此例中，假设分离器选择阈值为3，那么，顶点“C”的出度是4，应该使用后分离器，而剩余其他顶点使用前分离器. 具体地说，“C”由两个后分离器来管控，一个基于出边子集{“CA”，“CB”}，另一个负责出边子集{“CD”，“CE”}. 这种分配是由这些出边的终点及其所在的工作者所决定的，“CA”和“CB”被分配到同一组，因为这两条边的终点“A”和“B”之前被划分给同一个工作者；基于同种原理，“CD”和“CE”被分配到另一个工作者的后分离器上.

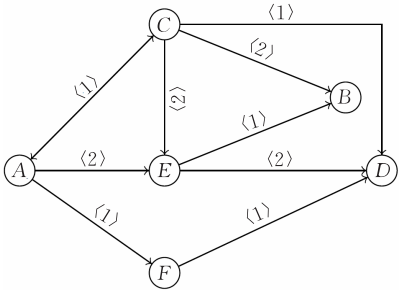


图3 带权重有向图示例

3.2 计算方法

以第3.1节的图划分为基础，下面设计相应的计算方法. 所谓组合器(combiner)，是一个聚合(aggregation)函数的实例，它聚合所有发向同一个顶点的消息.

下面通过在图3的示例网络上运行一个SSSP(Single Source Shortest Path)作业来解释计算方法的工作过程. SSSP问题在网络路由选择、城市布局设计、个人旅程规划等方面均有众多应用，是一个典型的大图计算问题.

假定SSSP算法从根顶点“A”开始. 在超步1，根顶点“A”的值被初始化为0，并发送消息给它的外向邻居“C”，“E”和“F”. 通过计算，这3个邻居顶点将它们的最短距离取值分别更新为1，2和1. 这里跳过超步1的执行细节，重点阐述超步2的执行过程，将流程图形化如图4所示. 在图4中，根据最浅灰色矩形框着色显示，当前系统中有2个工作者分别管辖顶点集{“A”，“B”，“C”}和{“D”，“E”，“F”}. 在超步2中，顶点“C”，“E”和“F”是SSSP算法的前沿顶点(front vertices)，即SSSP算法当前已经扩展到这些顶点；它们的最短距离取值在超步1中刚刚被更新了. 此时，低度数顶点“E”和“F”分别由两个前分离器管控，而高度数顶点“C”由两个后分离器管控，分别负责顶点“C”的出边子集{“CA”，“CB”}和{“CD”，“CE”}.

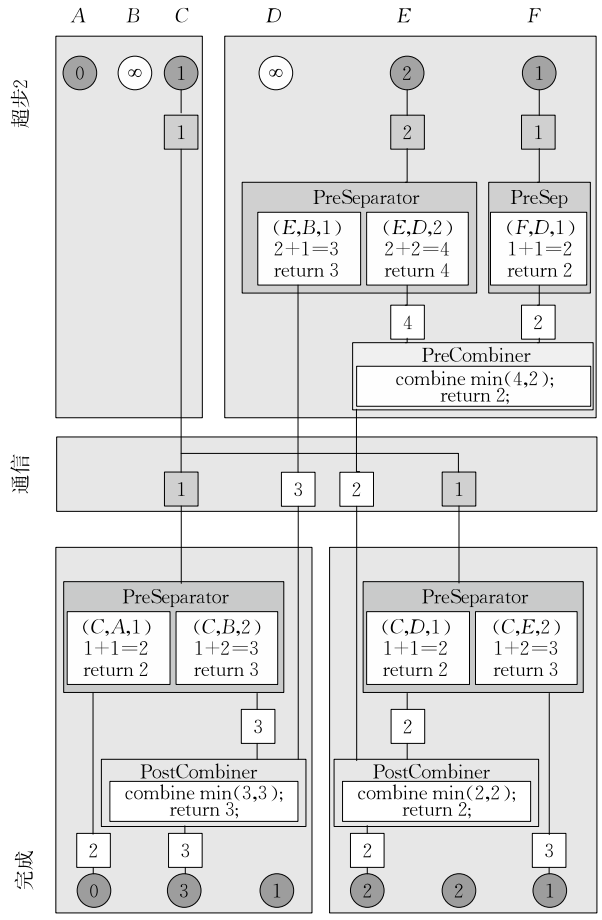


图4 SSSP算法之超步2示意

具体地说，一个超步设计包含了3个部分：前计算(pre-computation)、通信(communication)和后计算(post-computation). 在超步2的前计算部分，顶点“C”将其值1广播给所有工作者，因为顶点“C”是由后分离器来管控的. 同时，顶点“E”和“F”调用各自的前分离器执行*edgeCompute*()函数. 在SSSP算法中，*edgeCompute*()函数将边的权重与顶点当前值进行求和，并把结果发送给这条出边对应的终

点.由于“*ED*”和“*FD*”具有相同的终点“*D*”,前分离器之后调用前组合器(pre-combiner),对上述两个消息进行组合运算.在 SSSP 算法中,此功能由 MinValueCombiner 类实现,由此多个消息被合并,仅保留最小值.

然后,在超步 2 的通信部分,两个工作者之间进行 4 个消息的交换.

最后,在超步 2 的后计算部分,顶点“*C*”的后分离器遍历它的出边,通过调用 *edgeCompute*()函数对边的权重和顶点取值求和.后分离器产生的消息依据消息的目的顶点被后组合器(post-combiner)合并.之后,将合并后的消息发向目的顶点,成为下一个超步的输入.

在超步 3 的初始时,顶点“*B*”和“*D*”的最短距离取值进行了更新,成为新一轮 SSSP 算法扩展的前沿顶点.

由上述计算执行流程可以看出,基于 SC-BSP 模型的计算方法使用前分离器及前组合器对单个工作者内部的消息进行合并计算,减少了工作者之间的通信开销;同时,利用后分离器将不同目标工作者的消息均衡地分派到多个工作者上处理,从而优化了计算负载均衡.相比较而言,基于 VC-BSP 模型的计算方法则无法利用前、后分离器等对消息运算进行调度配置,因此在均衡计算负载方面的能力相对较弱.文中将在第 5 节的实验与分析中进一步验证上述结论.

SC-BSP 大图计算模型中,选择使用前分离器或后分离器的顶点度数阈值是可由用户指定的.目前的实验结果说明,该阈值取值等于系统中工作者个数时,对于大多数应用,系统性能达到最优;换句话说,对于大多数应用,此默认值可以最小化系统中的网络通信规模.

注意到,已有相关工作曾提出利用顶点度数对算法进行优化的初步想法[26],但是只是针对某一特殊问题的优化;相比较而言,文中所提模型和方法的贡献在于,系统地提出一套基于顶点度数的大图并行计算方法,它将底层负载均衡方面的设计困难和复杂性透明化,使用户可以自主便捷地研发大图数据上的分析应用.

4　实　现

模型的最终目标是在通用计算集群上实现一个易用、易维护且具备良好水平扩展能力的系统.鉴于篇幅限制,本文略去了基于 SC-BSP 模型计算框架的详细 API 介绍,感兴趣的读者可以参阅相关的设计技术报告①.这里简要讨论如下几个方面的问题:

(1)存储和索引.网络结构数据存储在多个工作者的分布式内存中.用户定义的划分函数将顶点划分到工作者上后,分布式内存里保存了前/后分离器及各自对应的 B-树索引.此外,还支持进阶用户对一个工作者上的所有顶点和前/后分离器进行遍历,进而使用户可以在本模型的框架下进一步开发“以图为中心(graph-centric)”的算法[18].

(2)后分离器.后分离器有两种可用的实现方案:① 对工作者进行索引,只有度数高的顶点才具备后分离器,可将分离器类型值 SeparatorType 发送给已经被索引的工作者;② 将高度数顶点的分离器类型值广播给所有工作者.考虑到,当系统中工作者数量增加时,高度数顶点的出边可能分布到更多的工作者上,方案(1)中的索引效率就会随之降低.为保持代码可读性和易维护性,目前实现的系统中采用了第 2 种方法.

(3)更改拓扑.部分图挖掘算法可能需要在算法执行过程中更改图的拓扑结构(graph mutation).目前,实现的系统可以支持有限的在线图拓扑更新操作.例如,Vertex 类支持 *setVertexValue*(),*addVertex*(),*removeVertex*(),*addEdge*()和 *removeEdge*()等函数,用于修改顶点和边信息;同时,EdgeHandler 类中提供 *setEdgeValue*()等函数.

(4)永久存储(persistent storage).目前,只在输入、输出及检验点(checkpoints)处访问磁盘等永久存储设备.输入和输出操作均支持通用的和特定划分的图格式:① 通用的图格式可由用户指定,目前可用的选项包括邻接表、边列表等.当通用的图格式作为输入时,需要对边随机抽样来评估每个顶点的度数,从而决定分离器的类型.评估之后,图结构被载入分布式内存中,并迅速在线构建基于内存的索引.② 特定划分的图格式把图结构组织为两部分——顶点和分离器,直接把分离器写到其所属工作者的本地磁盘上.这种格式也可用于检验点处,以满足容错与恢复的需要.

(5)容错与恢复.因为 SC-BSP 模型也是建立在经典 BSP 之上,所以容错与恢复机制都与 VC-BSP 模型类似,即基于检验点(checkpoints)的机制.根据

① https://site.google.com/xzhaounsw/sc-bsp.pdf

用户设定的间隔参数 C,在每执行 C 个超步之后执行检验点操作.在这些检验点处,修改的图数据和缓存的消息被写到永久文件系统上,以备失败时恢复使用.目前,永久文件系统采用可配置副本系数的 Hadoop 分布式文件系统(Hadoop Distributed File System,HDFS),即文件内容被复制到多个计算节点上进行冗余存储,以提高系统的整体可靠性和容错性.

5 实 验

本节汇报实验过程和结果分析,仿真实验主要在大型幂律网络上进行.首先,开展了大量的实验来比较与评价 SC-BSP 模型与 VC-BSP 模型的性能.这里以 SSSP 算法作业为例进行重点报告.作为一种图遍历(graph traversal)类算法,SSSP 算法的一个重要特征是网络通信量是变化的,即消息量逐渐增大,然后达到峰值,最后随着超步数量增加而减少.因此,SSSP 算法是用于测试系统应对动态变化通行量的极佳算法.其算法执行过程中,网络通信量逐步由少到多,使得系统瓶颈也易于被发现.然后,还引入了 GraphLab[20] 系统进行了横向比较,评测了基于 SC-BSP 模型的系统与 GraphLab 系统在同步模式执行 PageRank 算法方面的差异.PageRank 算法是一种经典的网页排序算法,在众多大图处理系统方面的论文中被多次论述和实现,便于横向比较与分析.

实验使用了一个包含 128 台物理机器的配置的 HDFS 集群,单台机器的物理配置为双核 Intel Xeon E5530 2.4 GHz 的中央处理器,24 GB 内存和 4 个 500 GB 固态硬盘.在默认参数设置中,工作者的数量固定为 512,因此,单台物理机器运行 4 个 SSSP 算法实例.

采用默认配置的高性能计算测试集 Graph 500①,该测试集采用了一个 Kronecker 生成器[27]产生服从幂律分布的大型网络.当生成图包含 10 亿个顶点时,其最大顶点出度超过 5 亿.

Apache Giraph② 是一个以 Java 语言开源实现的基于 VC-BSP 模型的大图计算系统;在此基础上,利用 Hadoop、Netty 和 Giraph 以内存系统的形式,同样采用 Java 编程语言实现了基于 SC-BSP 模型的大图计算系统.注意,实验记录的是基于 SC-BSP 和 VC-BSP 模型而实现的系统的性能,用以比较两种图计算模型;但为了方便论述,保持行文简洁,文中用"SC-BSP"和"VC-BSP"分别指代基于相应大图计算模型实现的系统.

5.1 水平扩展能力

首先,考察基于两种模型实现的系统的水平扩展能力,相关实验结果展示在图 5 中.所谓良好的水平扩展能力,是指通过往系统中增加工作者,即可提升系统处理更大规模网络的能力.图 5 给出了 3 种不同数量工作者的设置,即 SC-BSP 和 VC-BSP 两种系统分别在 32、128 和 512 个工作者上能够处理网络的最大规模(以边数计).

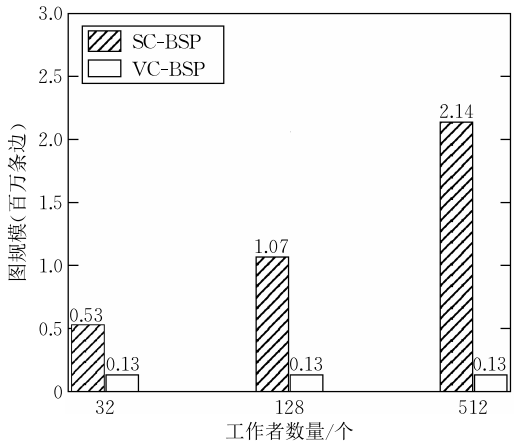


图 5 水平扩展能力实验结果

初始化时,SC-BSP 和 VC-BSP 均分配了 32 个工作者,每个工作者拥有 4 GB 内存.利用 32 个工作者,SC-BSP 可以处理包含高达 5 亿条边的图;相比之下,VC-BSP 只能处理包含至多 1 亿条边的网络.换句话说,在同等规模的 32×4 GB 分布式内存中,SC-BSP 相比 VC-BSP 可以处理更大规模的网络,这种处理规模上的优势正是源自于其原生的负载均衡技术.

当最大度数的顶点发出超过顶点总数 10% 的边时,VC-BSP 由于其以顶点为中心的特性,需要将所有边保存在单个工作者的内存中;而 SC-BSP 将后分离器作为边的容器,这样最大度数顶点的出边就由不同工作者上的后分离器分开管理,减轻了最大度数顶点所在工作者的负载,达到负载均衡的目的.

然后,通过增加工作者数量,检验系统是否可以处理更大的图.发现 VC-BSP 由于最大度数顶点消耗单个工作者的过多内存,不能处理更大的图;因此,即便增加工作者数量,单个工作者的内存限制直接制约了 VC-BSP 处理更大规模网络的能力.因而,

① The graph 500 list: http://www.graph500.org/
② Apache Giraph: http://giraph.apache.org/

VC-BSP 能够处理最大图规模的边数未能随着系统中工作者数量的增加而增加. 相比较而言，SC-BSP 在增加工作者数量后，即可处理更大的图；当增加到 512 个工作者时，SC-BSP 能够处理包含高达 21 亿条边的大图.

5.2　响应时间

接着，考察系统响应时间. 实验结果汇报的时间包括载入和计算过程实际消耗的时间（wall-clock time）. 图 6 描绘两种系统在响应时间上的实验结果，图的规模以包含的边数来衡量. 由于 VC-BSP 不能处理包含超过 1.3 亿条边的图，因此图 6 中的实验结果只记录到 1.3 亿条边为止的时间性能. 在实验可测量范围之内，观察到两种系统均展示出关于边数的良好的响应时间，并且 SC-BSP 在时间性能上总是优于 VC-BSP.

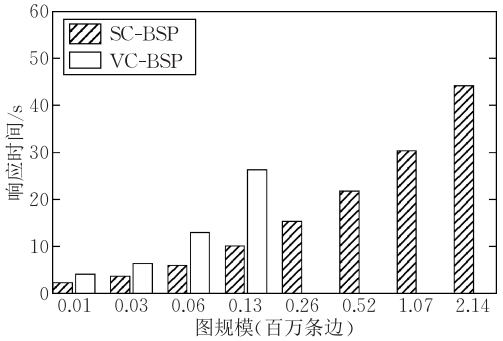


图 6　响应时间实验结果

5.3　通信开销

然后，以每个工作者产生的平均消息规模来度量系统的通信开销. 图 7 描述了两个系统在不同大小（以边数计）网络上的对比结果.

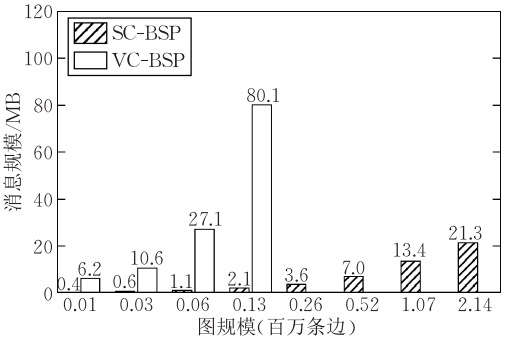


图 7　通信开销实验结果

结果显示，两者通信开销方面的差距可高达 40 倍，甚至是在相对较小的图上情况也类似. SC-BSP 模型的前分离器产生的消息数量与 VC-BSP 模型的消息数量相当；但是，后分离器极大地减少由高度数顶点产生的消息数量，因为此时 SC-BSP 模型只需要传输分离器的类型值，而且同时所有通过出边发送的消息都会被后分离器在同一个本地工作者中进行处理.

在服从幂律分布的网络中，当网络规模增大，更多的边将会连接到已有的高度数的顶点上. 在 SC-BSP 模型中，这些增加的边不会产生额外的消息，因此，它和 VC-BSP 模型之间在通信开销方面的差距随着网络规模的增加而急剧变大. 另外还观察到，在 SC-BSP 模型中，消息数量增加的趋势总是低于线性增长.

5.4　动态 SC-BSP

当图相对较小时，分配大量工作者可能引起不必要的协同开销. 为此，设计了一个简单的动态优化策略来减少这种开销，即依据图的规模决定参与计算的工作者数量. 在该组实验中，简单将图规模与工作者数量之间的比值设为每个工作者 2 百万条边. 因此，给包含 3 千万条边的图分配 16 个工作者；给包含 10.7 亿条边的图分配 512 个工作者；而包含 21.4 亿条边的图，仍然分配 512 个工作者（最大工作者数量的限制）.

图 8 描述了响应时间随图规模变化的趋势. 实验结果显示，在相对较小的图上，动态 SC-BSP 相比基本的 SC-BSP 可以减少至多 3/4 的响应时间，提高了系统的时间性能；而当图的规模不断增长时，动态 SC-BSP 策略与 SC-BSP 的差别逐渐缩小. 其原因是分配大量工作者的“不必要”开销此时成为必须，并且不占总体时间开销的主导地位.

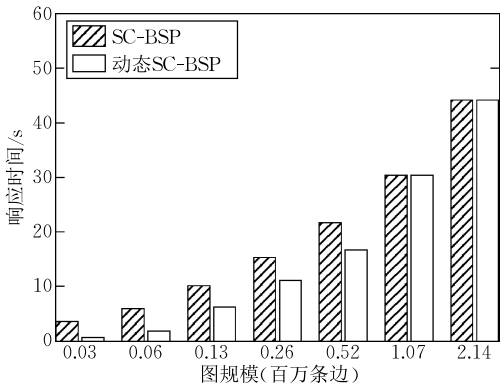


图 8　动态配置实验结果

因此，动态 SC-BSP 策略主要是为较小型网络而设计的，以避免不必要的工作者协同开销；在大规模网络上并不需要使用动态 SC-BSP 策略就能获得令人满意的系统性能.

5.5　分离器选择阈值

这一组实验通过设定不同的分离器选择阈

值——决定一个顶点是由前分离器还是后分离器管控,考察它对通信开销的影响.图 9 中描绘消息规模随选择阈值变化的趋势.

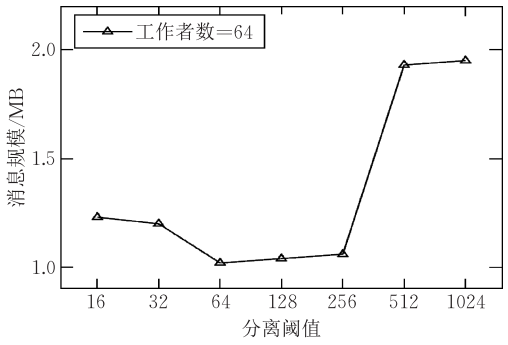


图 9 分离器选择阈值实验结果

实验结果显示,当阈值等于工作者数量时,系统中的消息规模达到最小.因此,可把选择阈值等于系统工作者数量作为默认设置;当然,用户也可以根据不同算法的特性,尝试其他参数设置,以求不同的运行效能.

小结:通过上述 5 组实验,表明基于 SC-BSP 模型实现的系统相比基于 VC-BSP 模型实现的系统具有更好的水平扩展能力,可以良好地处理服从幂律分布的大图.因此,认为 SC-BSP 模型相较于 VC-BCP 模型更适用于社交网络等巨型幂律网络.此外,SC-BSP 模型可以实现在所有类 Pregel 系统(Pregel-like systems)之上,提升系统整体性能.

诚然,和所有基于 BSP 的图计算模型一样,SC-BSP 模型不能自然地处理基于连接(join-based)的子图匹配等计算任务,也存在不支持异步执行模式方面的限制.

5.6 与 GraphLab 的对比

上述实验说明在服从幂律分布的大规模网络上,所提的 SC-BSP 模型对 VC-BSP 模型的优势显著.当前,还有一系列的大图计算系统并非以 VC-BSP 计算模型为蓝本而设计,主要代表性产品包括 GraphLab 和 GraphX 等.

其中,GraphLab 的系统设计基于 GAS(Gather-Apply-Scatter)计算模型,可执行同步和异步计算两种模式.GraphX 则同时支持 Pregel 和 GraphLab 两种图计算虚拟框架(abstraction),其主要优势在于基于 Spark 并行平台的通用性.已有研究[28]证明 GraphX 在图计算方面的性能不及 GraphLab.因此,将其排除在实验对比行列之外.本节着重对比以 SC-BSP 模型实现的系统与同步执行模式下 GraphLab 系统在运行 PageRank 算法上的差异.

PageRank 算法是 Google 用于网页排序的一种算法,其基本思想是越重要的网页会被更多的其他网页所连接.实验中使用了和以往相同[12]的衰变系数 0.85,其意义是在一个给定网络顶点上,一个随机游走者下一步可能以 0.85 的概率前往该顶点的任一外向邻居顶点,也可能以 0.15 的概率随机跳转到任一顶点.

PageRank 算法的主要特点是,所有顶点在算法执行过程中始终保持激活(active)状态——区别于前述的 SSSP 算法,顶点总是申请暂停(vote to halt),直至重新接收到新的消息才重新进入激活状态,并且消息传递量保持相对稳定.因此,已有大图计算模型和系统的研究中都描述和实现了 PageRank 算法.下面的实验通过执行 PageRank 算法,揭示 SC-BSP 和 GraphLab 在执行顶点保持活跃一类算法时性能上的特点.

首先对比了两者在响应时间上的实验结果,如图 10 所示.观察到,PageRank 作为一种常规易于实现的算法,两个系统对这种算法的执行都表现出良好的系统响应时间.其中,SC-BSP 在所有规模的图上表现均优于 GraphLab,而 GraphLab 采用的 GAS 模型也同样很好地处理 PageRank 这类顶点全部活跃的算法.当在边数达到 2.14 B 的大规模网络上,SC-BSP 所需的相应时间为 4120 s,而 GraphLab 则需要将近 5000 s 的时间;SC-BSP 能够提供接近 1.2 倍的性能提升.因此,可以认为,在同样图规模处理能力的情况下,SC-BSP 具备更优的时间效率,提供更佳的用户体验.

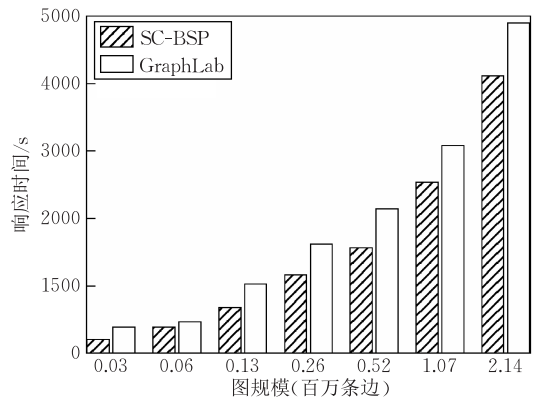


图 10 响应时间对比结果

关于通信开销方面,同样可以看到两个系统都有较低的网络通信量,即通信开销.图 11 描绘了两个系统对应的实验结果,比较结果表明,尤其是在网络较小时,两者的通信开销方面的表现旗鼓相当,规

模上差距很小. 可以看出 GAS 模型对于系统通信量相对稳定的算法具有良好的通信量优化，而 SC-BSP 模型针对消息传递和负载均衡方面的控制则更加优秀. SC-BSP 在所有网络规模输入上都优于对手，它和 GraphLab 之间的差距随着网络规模的增大而变得更加显著，当边数规模达到 2.14 B 时最大可达约 8.5 MB.

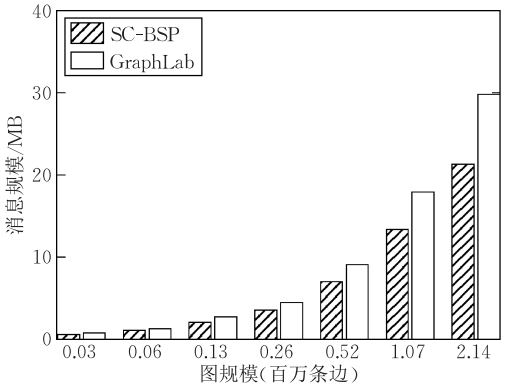


图 11　通信开销对比结果

注意到，SC-BSP 是基于 Apache Giraph 开源项目实现的，它使用的是 Java 编程语言；而 GraphLab 的开源项目是采用 C++编程实现的，且目前暂无开源可用的基于 Java 语言实现的 GraphLab 系统. 因此，上述性能差异也可能和编程实现语言本身的效率差异有关系，暂时无法完全归结到大图计算模型本身上.

6　结论和进一步工作

旨在通用计算集群和云计算基础设施上进行大规模图数据处理，本文首先分析了目前大图处理系统在横向可扩展性方面面临的技术挑战. 为应对这些困难，提出了一种基于分离器-组合器的 SC-BSP 大图计算模型. SC-BSP 模型的核心是在 BSP 的计算框架下引入分离器和组合器的概念，对图划分的边集进行合理分配和管理. 然后，基于所设计的模型，讨论了一种典型大图计算作业 SSSP 算法的设计过程和实现细节. 最后，通过大量大图数据上的实验，其结果表明所提大图计算模型的性能和水平扩展能力能够满足大规模图数据处理的需要，可为当前众多大型网络图数据分析应用提供能力支撑.

总的来说，相较于目前流行的以节点为中心的 VC-BSP 模型，SC-BSP 大图模型的主要优势在于，其设计理念面向服从幂律分布的真实世界大型网络，能改善大型网络图并行计算任务中的并行化水平和大图处理系统的水平扩展能力；其良好的易用性、容错与恢复以及可移植性使得该模型可以在现有配置了 Apache Hadoop 的通用计算集群上进行简单的配置、开发和使用.

目前的实验结果显示，当分离器选择阈值等于系统中工作者个数时，基于 SC-BSP 模型的系统性能最优. 下一步工作中，计划开展如下几个方面的探索和研究：一方面，将在实验系统上使用其他算法作业，如 HashMin 算法、星型结构模式匹配算法等，观察系统性能特点，考察多阶段算法和拓扑结构更新等方面的系统表现；另一方面，计划从理论上分析和证明所提分离器选择策略的最优性，为模型提供理论保证，提高 SC-BSP 模型适用性；此外，还可以用同一种编程语言实现包括 GraphLab 在内的大图计算系统，排除编程语言本身效率的差异，以发现系统本身的性能差异，从而追求更佳的模型和系统设计.

致　谢　感谢匿名评审和编辑对本文质量的改进提供了大量建设性的意见帮助. 同时感谢相关研究项目及小组对 Apache Giraph 和 GraphLab 相关实验代码的开源！

参　考　文　献

[1] Yu Ge, Gu Yu, Bao Yu-Bin, Wang Zhi-Gang. Large scale graph data processing on cloud computing environments. Chinese Journal of Computers, 2011, 34(10): 1753-1767(in Chinese)
(于戈，谷峪，鲍玉斌，王志刚. 云计算环境下的大规模图数据处理技术. 计算机学报，2011，34(10)：1753-1767)

[2] 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. Indianapolis, USA, 2010: 135-146

[3] Valiant L G. A bridging model for parallel computation. Communications of the ACM, 1990, 33(8): 103-111

[4] Broder A Z, Kumar R, Maghoul F, et al. Graph structure in the web. Computer Networks, 2000, 33(1-6): 309-320

[5] Faloutsos M, Faloutsos P, Faloutsos C. On power-law relationships of the internet topology//Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. Cambridge, USA, 1999: 251-262

[6] Checconi F, Petrini F. Massive data analytics: The graph 500 on IBM blue gene/q. IBM Journal of Research and Development, 2013, 57(1/2): 10

[7] Yoo A, Chow E, Henderson K W, et al. A scalable distributed parallel breadth-first search algorithm on BlueGene/L// Proceedings of the 2005 ACM/IEEE Conference on Supercomputing. Seattle, USA, 2005: 25

[8] Chan E, Heimlich M, Purkayastha A, Geijn R van de. Collective communication: Theory, practice, and experience. Concurrency and Computation: Practice and Experience, 2007, 9(13): 1749-1783

[9] Pjesivac-Grbovic J, Angskun T, Bosilca G, et al. Performance analysis of MPI collective operations. Cluster Computing, 2007, 10(2): 127-143

[10] Gonzalez J E, Low Y, Gu H, et al. Powergraph: Distributed graph-parallel computation on natural graphs//Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. Hollywood, USA, 2012: 17-30

[11] Batarfi O, Shawi R, Fayoumi A, et al. Large scale graph processing systems: survey and an experimental evaluation. Cluster Computing, 2015, 18(3): 1189-1213

[12] Lu Y, Cheng J, Yan D, Wu H. Large-scale distributed graph computing systems: An experimental evaluation. Proceedings of the VLDB Endowment, 2014, 8(3): 281-292

[13] Han M, Daudjee K, Ammar K, et al. An experimental comparison of pregel-like graph processing systems. Proceedings of the VLDB Endowment, 2014, 7(12): 1047-1058

[14] Gregor D, Lumsdaine A. Lifting sequential graph algorithms for distributed-memory parallel computation//Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications. San Diego, USA, 2005: 423-437

[15] Salihoglu S, Widom J. GPS: A graph processing system// Proceedings of the 25th International Conference on Scientific and Statistical Database Management. Baltimore, USA, 2013: 22: 1-12

[16] Salihoglu S, Widom J. Optimizing graph algorithms on pregel-like systems. Proceedings of the VLDB Endowment, 2014, 7 (7): 577-588

[17] Tasci S, Demirbas M. Giraphx: Parallel yet serializable large-scale graph processing//Proceedings of the 19th International Conference on Parallel Processing. Aachen, Germany, 2013: 458-469

[18] Tian Y, Balmin A, Corsten S A, et al. From "think like a vertex" to "think like a graph". Proceedings of the VLDB Endowment, 2013, 7(3): 193-204

[19] Lee K, Liu L. Efficient data partitioning model for heterogeneous graphs in the cloud//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. Denver, USA, 2013: 46

[20] Low Y, Gonzalez J, Kyrola A, et al. Graphlab: A new parallel framework for machine learning//Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence. Catalina Island, USA, 2010: 340-349

[21] Han M, Daudjee K. Giraph unchained: Barrierless asynchronous parallel execution in pregel-like graph processing systems. Proceedings of the VLDB Endowment, 2015, 8(9): 950-961

[22] Jing Nian-Qiang, Xue Ji-Long, Qu Zhi, et al. SpecGraph: A distributed graph processing system for dynamic result based concurrent speculative execution. Journal of Computer Research and Development, 2014, 51(Supplment): 155-160 (in Chinese)
(景年强，薛继龙，曲直等. SpecGraph：基于并发更新的分布式实时图计算模型. 计算机研究与发展，2014，51(增刊)：155-160)

[23] Shao B, Wang H, Li Y. Trinity: A distributed graph engine on a memory cloud//Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. New York, USA, 2013: 505-516

[24] Bronson N, Amsden Z, Cabrera G, et al. Tao: Facebook's distributed data store for the social graph//Proceedings of the 2013 USENIX Conference on Annual Technical Conference. San Jose, USA, 2013: 49-60

[25] Kang U, Tsourakakis C E, Faloutsos C. Pegasus: Mining peta-scale graphs. Knowledge and Information Systems, 2011, 27(2): 303-325

[26] Shang H, Kitsuregawa M. Efficient breadth-first search on large graphs with skewed degree distributions//Proceedings of the 16th International Conference on Extending Database Technology. Genoa, Italy, 2013: 311-322

[27] Chakrabarti D, Zhan Y, Faloutsos C. R-mat: A recursive model for graph mining//Proceedings of the 4th SIAM International Conference on Data Mining. Lake Buena Vista, USA, 2004: 442-446

[28] Xin R, Gonzalez J, Franklin M, Stoica I. GraphX: A resilient distributed graph system on Spark//Proceedings of the 1st International Workshop on Graph Data Management Experiences and Systems. New York, USA, 2014: 2

ZHAO Xiang, born in 1986, Ph. D., lecturer. His research interests include graph data management and mining, intelligence analytics, etc.

LI Bo, born in 1994, M. S. candidate. His research interest lies in big data management and analytics.

SHANG Hai-Chuan, born in 1984, Ph. D., associate professor. His research interests include graph data management and mining, parallel computing, etc.

XIAO Wei-Dong, born in 1968, Ph. D., professor, Ph. D. supervisor. His research interests include intelligence analytics, social computing, etc.

Background

With the rapid proliferation of huge volume of data, cloud storage and computing technology is developing in a fast manner. Massive graph computation, among various kinds of real big data, is a typical representative and important part of workloads on the cloud. There are a number of existing solution to processing large graphs, and representatives systems are Pregel by Google, Trinity by Microsoft, TAO by Facebook, Giraph, GraphX, PowerGraph, and GiraphUC, etc.

Currently, scalable graph computation mainly resorts to high performance computing solutions, which requires high performance all-to-all collective communication over torus or mesh networks. However, implementing these torus or mesh-based algorithms on commodity clusters and cloud computing infrastructures may result in high latency, due to inefficient network communication. Besides, the vertex degree skewness and lack of locality on massive graphs further challenge the design of a highly scalable system.

To resolve the aforementioned issues, in this paper, we exploit a commodity cluster-oriented programming model for scalable graph computation. Focusing on scale-out capability, we propose a novel separator-combiner BSP-based graph computation method, which provides native load-balancing and low communication overhead. Thus, larger graphs can be addressed by adding more computing nodes. Finally, extensive experimental results confirm the effectiveness and efficiency of the method.

Potential applications of the proposed techniques include social network analysis, entity-relation network management, mining heterogeneous information network, etc. Hence, this research is part of our ongoing projects in the research center of big data and social computing at the university.

This work was partially supported by the National Natural Science Foundation of China under Grant Nos. 61402494 and 61402498, and the Provincial Natural Science Foundation of Hunan under Grant No. 2015JJ4009.