DOI：10.3969/j.issn.1009-6868.2016.02.009
网络出版地址：http://www.cnki.net/kcms/detail/34.1228.TN.20160222.1019.004.html

BC-BSP：一个基于BSP的高可扩展并行迭代图处理系统

BC-BSP: A BSP-Based High Scalable Parallel Iterative Graph Processing System

刘恩孚/LIU Enfu
冷芳玲/LENG Fangling
鲍玉斌/BAO Yubin
（东北大学 计算机科学与工程学院，辽宁 沈阳 110819）
(School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China)

中图分类号：TP393 文献标志码：A 文章编号：1009-6868 (2016) 02-0038-006

摘要： 提出了一个基于整体同步并行计算（BSP）模型的、具有磁盘暂存功能的大规模图处理系统——BC-BSP。该系统通过提供应用程序接口（API）实现系统配置和有关策略的可扩展性，通过优化的图数据磁盘存储实现了数据处理规模的高可扩展性以及高性能的容错方案，并且可以处理普通数据集的聚类和分类等需要迭代计算的数据挖掘算法。通过实验验证了该系统的可扩展性，其在真实数据集上性能优于Giraph1.0.0，在模拟数据集上稍逊于Giraph的内存版。

关键词： BSP；大规模图处理；迭代计算；磁盘缓存

Abstract: We describe a bulk synchronous parallel (BSP)-based parallel iterative processing system for graph data with disk caching assist. This system is called BC-BSP. The system can achieve the scalability of system configuration and policy by providing APIs, high scalability of the data scale processed, and high performance of fault-tolerant scheme by disk storage optimization to graph data. It can also execute some data mining algorithms with iterative processing, such as clustering and classification on non-graph data sets. The experimental results show that the scalability and performance of the proposed system are better than that of Giraph1.0.0 on the real data set, but it is lightly poorer than the memory version of Giraph.

Keywords: BSP; large-scale graph processing; iterative computing; disk cache

图是计算机科学中最常用的一类抽象数据结构，更具有一般性的表示能力。现实世界中的许多应用场景都可以很自然地使用图结构表示。例如，交通运输网络、社交网络中的资源对象之间的关系以及生物信息网络等。在大数据时代，需要分析的图规模越来越大。以互联网和社交网络为例，随着互联网的深入使用和Web 2.0技术的推动，网页数量增长迅猛，据中国互联网络信息中心（CNNIC）统计：截止2014年12月中国网页规模达到1 899亿个，年增长率26.6%；而基于互联网的社交网络更是如此，如全球最大的社交网络Facebook，2014年7月已有约22亿用户，其中月活跃用户数13亿人。在中国，如QQ空间、微博、开心网等，发展也异常迅猛。因此，实际应用中图的顶点可达10亿，而边就会更多，对应的数据文件会更大。对如此大规模图数据的存储和分析处理的时间和空间开销远远超出了传统集中式图数据处理的承受能力。因此，对大规模图的有效处理成为了一个新的挑战。

MapReduce计算模型可以实现对大规模（图）数据的处理，并且具有很好的容错性和可扩展性。但是由于图数据分析（如网页的PageRank[1]计算、最短路径计算、聚类分析）都需要多次迭代才能完成。每次迭代需要一个或多个开销较大的MapReduce作业完成。为解决迭代计算的时间性能问题，谷歌公司开发了基于整体同步并行计算（BSP）模型的Pregel[2]系统，之后Apache的两个开源项目Hama和Giraph也开展了基于BSP的迭代计算系统的开发。它们都是在内存中做数据处理，因此能够处理的图的规模有限。文中，我们设计开发了基于BSP模型的、能够处理大规模

收稿时间：2016-01-08
网络出版时间：2016-02-22
基金项目：国家自然科学基金重点项目（61433008）；国家自然科学基金（61173028）；教育部-中国移动科研基金（MCM20122051）

（图）数据的并行迭代计算系统——BC-BSP。该系统主要特色在于：（1）实现了具有磁盘辅助的基于BSP的大规模图数据并行迭代处理系统，该系统在内存受限的情况下具有很好的数据处理能力，即在可用的节点规模和内存配置的情况下，可以处理的数据规模较大；（2）系统多方面考虑负载均衡，在充分考虑数据本地化的前提下考虑了各个节点的负载均衡问题，并且结点的负载均衡优先于数据本地化。我们做了大量的实验，比较了基于BSP的大规模图处理系统的性能和扩展性。

1 BSP模型和相关工作

BSP是一种“块”同步模型[3]，即通过消息传递机制，实现块内异步并行，块间显式同步。一个基于BSP的计算系统是由具有处理机和存储器的多个自治的计算服务器组成的集群，并且这个集群采用主/从结构。主节点用于协调整个集群，包括接收用户的作业提交、作业调度、故障监控等功能，从节点（也称为工作节点）用于存储和处理数据。

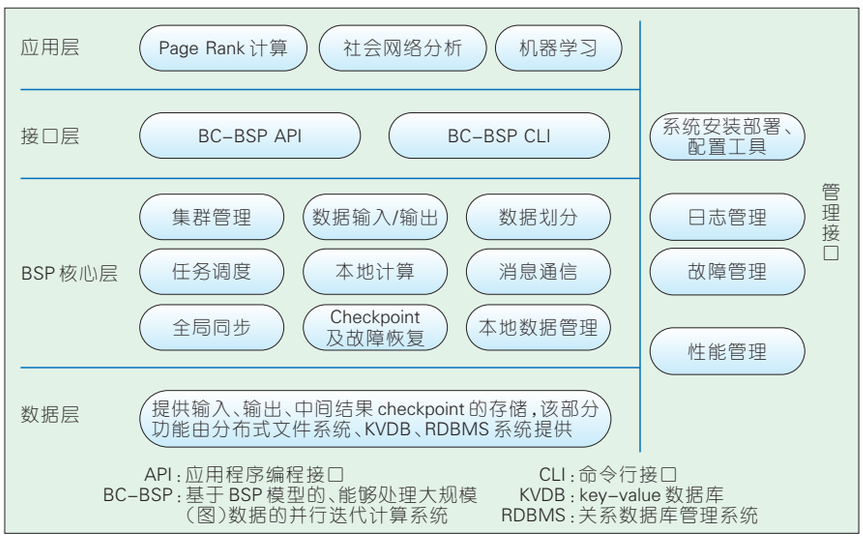
谷歌公司开发的基于BSP模型的分布式图计算框架Pregel主要是为了处理大规模图数据，如网页的PageRank计算、最短路径等。Pregel假设处理的数据都在内存中，因此在一定的节点规模下，它能够处理的数据规模是有限制的。基于Pregel的思想，许多基于BSP的大规模图处理系统被开发出来。例如，Apache推出了基于Java的开源项目Hama[4]，它是一个纯粹的基于BSP的用于大规模科学计算（如矩阵计算、图和网络算法）的计算框架，同样它的早期版本没有考虑磁盘辅助的问题，而是假设所有数据全部位于内存中，最新的版本也在添加磁盘辅助功能，但是很不完善；而Apache的另一个开源项目Giraph，是建立在Hadoop基础之上的Pregel的开源实现[5]，可以认为它是MapReduce模型和BSP模型的结合体，即它利用MapReduce作业的Map任务实现了基于BSP模型的迭代计算，而不需要Reduce任务，整个图处理过程只需要启动一次MapRedcue作业，但是一旦出现故障，整个作业需要重新启动；GraphLab是卡内基梅隆大学提出的面向大规模数据挖掘和图计算的分布式内存计算框架[6]。更多的基于BSP模型的类Pregel的大规模数据分布式并行处理系统和框架请见文献[7]。

2 BC-BSP概述

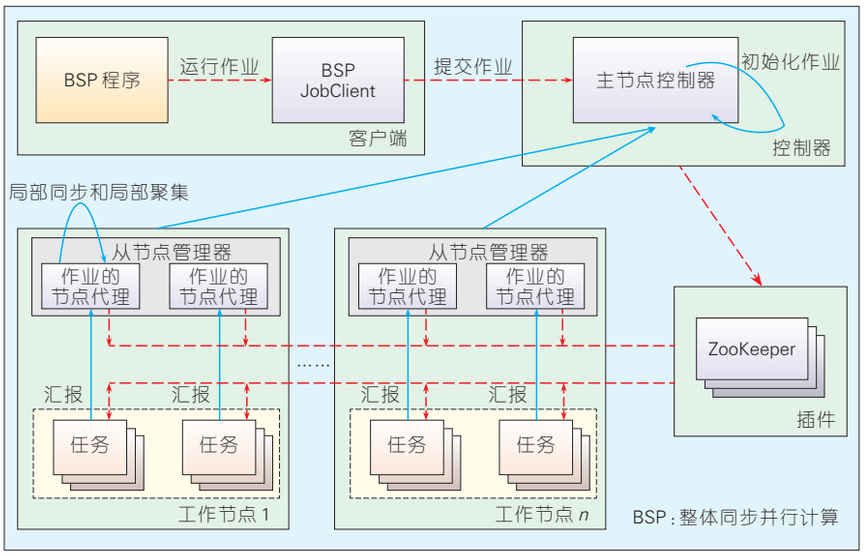
图1给出了BC-BSP系统的整体结构，主要包括BSP核心层、管理接口层和接口层。BC-BSP实现了对Hadoop分布式文件系统（HDFS）、HBase、MySQL等底层存储系统的支持，包括数据的输入和输出。BC-BSP系统内部核心层主要包括客户端作业提交和数据划分，主节点端的作业调度和集群监控，从节点端的本地计算处理、全局同步、消息通信和容错控制；接口层主要包括应用编程接口（API）和命令行接口（CLI）；管理接口层主要包括集群管理、系统自动化安装部署、日志管理、性能管理和故障管理等工具。

从系统实现的角度，BC-BSP系统是一个主从式结构，主要分为客户端、主控节点、工作节点、任务模块、全局同步模块。图2给出了BC-BSP的运行控制机制以及系统中客户端、主控节点、工作节点、任务模块、全局同步模块之间的协作关系。

在BC-BSP系统中，客户端主要根据用户指定的输入路径进行数据分片，调整分区数目，检查作业运行的可行性，向主控节点申请作业并将作业打包提交给BSP主控节点，当作业开始运行后，负责及时反馈作业运行状态；主控节点端管理集群工作节点的注册、心跳信息和状态信息收集等，并作为容错控制的控制中心，提供各种状态查询接口，并以作业为单位，负责作业的初始化、调度和同步控制等；工作节点端主要负责工作节点本地的任务管理和局部同步控制以及局部聚集计算等；任务模块端是任务运行的实体，主要负责执行用户的业务处理逻辑和数据输入输出处理等；全局同步负责同一作业的所有任务在各个超步之间的全局同步工作，超步路障同步由主节点端、工作节点端及任务模块端共同完成，在同步过程中，可以完成聚集计算，系统中的同步主要通过第三方组件Zookeeper实现；消息通信主要在每一个超步的本地计算执行过程中，负责异步地发送和接收消息，并将接收的



▲图1 BC-BSP系统的功能组件



▲图2 BC-BSP的运行控制机制

消息暂存到本地的接收消息队列中，当内存空间不足时，支持磁盘辅助存储，这里主要是通过远程过程调用协议（RPC）机制实现消息传递；容错控制模块负责容错备份、故障检测和故障恢复等功能，以写检查点机制作为主要的容错方案，支持手动备份和自动周期备份功能；管理工具主要通过Web界面或命令行的方式为用户提供可视化的系统管理和监控功能；接口模块主要为用户提供本地计算、消息发送/接收等的应用编程接口，以及为用户提供启动和关闭系统服务、作业提交等命令行接口。

3 BC-BSP提供的API

系统给用户提供了与作业建立相关的API，用于编写针对图处理或科学计算的处理程序。另外，系统还提供了用于系统功能扩展的接口。下面我们简单介绍这些接口。

（1）消息管理接口负责消息的发送/接收功能，在每一个超步的本地计算执行过程中，并行地发送和接收消息，并将接收的消息缓存到本地的接收消息队列中，在发送消息队列达到一定规模的时候，执行Combine操作，然后再将消息发送给目的节点。

（2）分区数据管理接口负责在进行图数据处理之前将待处理的图数据按照一定的原则划分给各个任务。本系统实现了基于Hash的划分方法和基于Hash的均衡划分方法。

（3）图顶点上下文接口负责在任务处理的一个超步中，处理每个图顶点时获取正在处理的图顶点的相关属性信息和方法。

（4）消息合并接口在图处理过程中，通常以顶点为中心进行处理，该接口为了减少在网络上传送的消息数量，在发送端对发给同一个顶点的消息进行合并。

（5）聚集计算接口许多图处理/机器学习算法中需要聚集计算，实现该接口可进行超步间的聚集值计算。

（6）数据输入输出接口包括输入接口和输出接口，用于实现将数据从指定数据存储系统中读入和写出。

4 BC-BSP系统的实现

本节介绍BC-BSP系统在实现上的一些主要策略和细节，主要包括图数据的表示、主节点控制器、从节点管理器、本地计算与消息通信、图数据划分以及故障恢复等的实现。

4.1 主节点控制器

主控节点是整个BC-BSP集群的控制中心，负责管理所有的工作节点，监控整个集群的工作状态，接收各工作节点的心跳信息并加以处理，完成整个作业的全局同步控制，并提供统一的信息查询接口和作业提交接口。当集群启动后，主控节点接收各工作节点的注册信息，形成统一的集群资源信息，在运行过程中通过心跳信息不断更新集群资源信息，例如，可用任务槽数量。当客户端请求提交作业时，将其放入作业等待队列，作业调度器按照优先级加先入先出队列（FIFO）的策略调度作业；而完成一个作业的具体任务的调度则是按照负载均衡和数据本地化的原则。因为本系统中一个作业的所有任务需要同时运行，所以系统中的任务调度是采用由BSP主节点控制器根据上述原则将任务依次不断下推给各个节点。

4.2 从节点管理器

工作节点是硬件上的计算单元，系统启动后，BC-BSP集群的各个节点上启动一个从节点管理器（WM）进程，负责完成具体的任务启动和消息通信。每个工作节点启动后，都首先向主控节点注册，使自己成为BC-BSP集群中的一员；之后，工作节点定期向主控节点发送心跳信息，汇报自己的状态；当有新任务下达时，工作节点根据新任务的指令，到HDFS上读取作业信息并下载到本地文件系统；然后创建任务控制对象和对应的执行进程，接着运行任务。WM为在本节点上运行的每个作业建立一个WorkerAgent对象，用于收集该作业在本节点上的各个任务的心跳信息、工作状态信息等。这样全局同步采用两级同步方式，即一个工作节点上的属于同一个作业的各个任务在本节点上实现局部同步，然后再以节点为单位向Zookeeper注册实现全局同步。工作节点以作业为单位维护在本节点上运行的隶属于同一个作业的所有任务，进行统一管理，完成

各种局部操作，例如本地聚集计算。

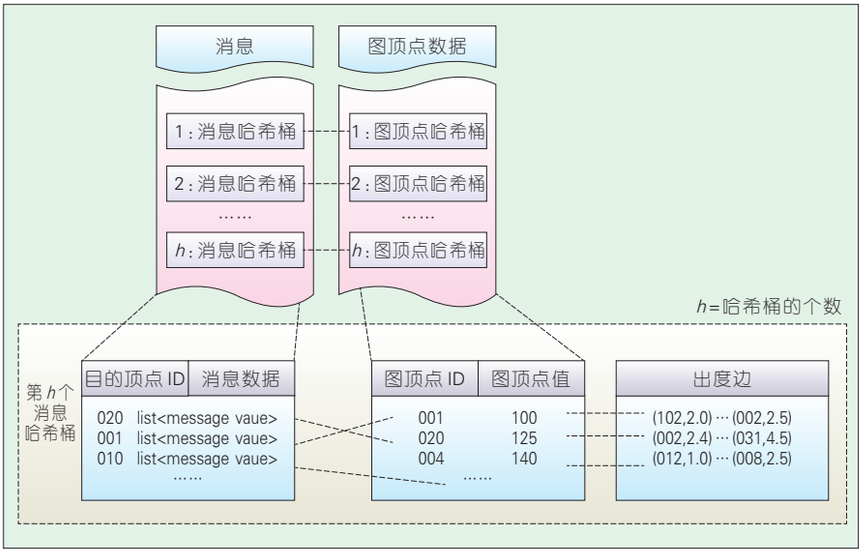
4.3 磁盘辅助的本地计算和消息通信

任务模块是逻辑上的计算处理单元，称为一个任务。BSP主节点控制器中的任务调度器根据负载均衡和数据本地化原则将任务分配到具体的工作节点上，由WM创建该任务模块进程。任务模块启动后，首先完成数据加载，将需要处理的数据分片从存储介质上按照指定的输入格式读入本地，并进行数据划分。计算过程中会定期地向WM的WorkerAgent对象发送心跳信息，报告任务的状态等信息。

在Pregel系统以及基于它思想的各种实现中，都假设集群的处理节点足够，使得待处理的数据等够完全存放在内存中。但是实际情况却不是这样的：一方面对于一个给定的待处理数据集，用户很难确定需要几个工作节点才能使得各个任务处理的数据能够存放在内存中；另一方面，当集群规模有限时，也希望能够处理相对较大规模的数据。对于系统中发送（或接收）的消息也是如此。鉴于以上原因，本系统中使用了磁盘临时存储数据和消息（也称之为磁盘暂存），以便能够处理较大规模的数据。

对于消息数据，将消息数据的内存占用比例按照用户指定的静态划分参数确定，系统运行时处理各种类型的消息时内存的使用单独分配处理，每种类型的消息内存占用都具有一个独立的阈值控制。

对于任务处理的数据而言，在迭代计算过程中常驻磁盘。对于出边表不变的计算情况，即不增加也不删除边的情形，将顶点的出边表与顶点的其他在计算中变化的部分，例如顶点的值或标签等信息，分开存放，但是同样使用记录的ID的Hash映射进行划分，如图3所示。将图数据分开处理的好处在于：每次迭代结束只需将本次迭代过程中变化的数据写回本地磁盘文件即可，不变的静态部分不需要写回磁盘，同时也为容错控制提供了方便。



▲图3 磁盘暂存的Hash索引示意

4.4 图的顶点类

一个图是由顶点集合和边集合构成，因此有顶点类和边类。本系统中使用邻接表的方式组织图数据。这样一个顶点类中除了顶点本身的属性之外，还有与之相连的出边信息，同时提供了对顶点和边进行操作的方法（见图4）。

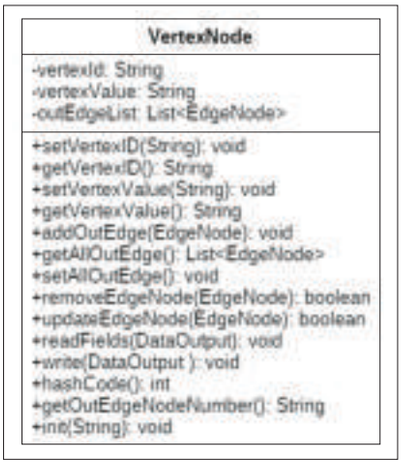
4.5 数据划分

数据划分是BSP计算与MapReduce计算不同的地方。前者需要在迭代计算中能够定位消息发送的目的地在哪里。因此，数据划分是将各个任务与之绑定的数据分片的数据从数据源读入，然后利用一定的数据划分原则，例如Hash划分，将图数据分配给某个任务，以便形成超步迭代计算时的数据分区。

一个作业的各个数据分区大小是否均匀直接影响系统的负载均衡，但是Hash函数很难保证各个分区大小的均衡。为此，我们采用了多Hash桶合并的划分方法，以实现数据的近似均衡划分。合并的原则可以是各个桶中的对象数据尽可能均衡，还可以考虑数据的本地性。本系统目前是按照各个桶中数据对象近似均衡为主兼顾本地性的原则进行合并。

4.6 容错机制

容错是本分布式处理系统必须考虑的问题。BC-BSP系统中考虑两类故障：一类是任务故障，例如任务进程宕掉；另一类是工作节点故障，例如一个Worker出现网络断开故障或者磁盘读写故障。系统中各个任务通过心跳机制向所在Worker的WM汇报自己的工作状态，而各个工作节点也是通过心跳机制定期向BSP主节点控制器汇报工作状态。



▲图4 图顶点类结构

本模块包括写检查点、故障检测和故障诊断以及故障恢复等功能。写检查点是定期或者人工控制方式将某个时刻的作业运行快照保存到分布式文件系统，如HDFS；故障检测与故障诊断是完成故障信息的收集与故障类型的判断，不同阶段的不同类型的故障，采用不同的恢复机制。BC-BSP系统实现了基本的基于检查点的故障恢复策略和面向磁盘驻留的多级容错处理策略。

所谓的面向磁盘驻留的多级容错处理策略，是利用了本系统的磁盘辅助机制的一些措施，即将图数据分成不变的常驻磁盘的静态部分（例如图顶点的出边表）和每次迭代计算几乎都会变化的需要写回磁盘的动态部分。因此在进行系统快照备份时，实现增量备份，即对图数据的静态部分只需要备份一次即可，而每次迭代计算时只需增量地备份动态变化部分。当然每次备份时需要备份本次收到的所有消息。

5 BC-BSP系统应用示例

本节讨论使用本系统进行图数据的PageRank计算和多维数值型数据集的k-means聚类分析的示例。在k-means示例中，可以论证BC-BSP系统也可以有效地处理非图数据的数据挖掘算法。

5.1 PageRank

使用BC-BSP系统实现PageRank计算中，首先将一个顶点的PageRank值按照一定的规则（如各个出边顶点平分），通过发送消息的方式发送给出边顶点，同时获得来自入边顶点的消息；之后按照PageRank算法的PageRank值计算公式，将一个顶点的消息值（即PageRank贡献值）累加，计算当前顶点新的PageRank值。因此用户可以提供combine方法实现消息发送前的合并，再基于顶点的新PageRank值重复上面的计算过程，直到满足收敛条件结束计算，并按预先的用户配置输出计算结果。

5.2 多维数值型数据集的k-means聚类

使用BC-BSP系统对多维数值型数据集进行k-means聚类，不需要进行顶点间的消息传递，但是需要利用聚集器计算新的聚类中心，可以通过各个聚簇的所有数据点的累计和与累计数据点计数两种聚集器实现。因此，用户可以实现BC-BSP系统提供的staffStartup接口，完成整个聚类作业开始之前的聚类中心初始化工作，例如读取预先设定好的存储在分布式文件中的初始聚类中心，利用系统提供的聚集器接口实现聚簇内数据点累计和与累计计数计算新的聚类中心，这样就需要每个任务计算自己任务内的局部累计和与累计计数，然后在BSP主节点控制器计算各个类的总累计和以及总类内数据点数，在新的超步开始时计算聚集中心。

当k-means聚类的k值较小（例如几十个）时，这种利用聚集器的方法是可行的。然而，实验中我们发现：当k值上百或更大时，就会出现异常。这是因为需要向Zookeeper写的内容太多。因为系统框架中聚集器的实现利用了Zookeeper，所以在实现k-means聚类时，使用了分布式文件暂存各个任务的局部聚集结果。在执行超步计算前读取这些临时文件，计算新的聚类中心，可以解决k值较大时引起的异常问题。

6 BC-BSP系统的实验

选择同样基于BSP模型的Hama[4]和Giraph[5]作为参照比较系统，并且使用它们的API实现了PageRank算法。实验软硬件配置是：30个工作节点，一个作为控制节点，29个用作存储和计算的工作节点，Java虚拟机（JVM）的内存设置为2 GB。每个节点的配置如下：Intel Core i3-2100双核中央处理器（CPU）、8 GB双倍速率同步动态随机存储器（DDR）3内存、500 G/7200 RPM磁盘，安装了Red Hat Centos 6.0操作系统、JDK1.6.0-30、Hadoop-0.20.2和Zookeeper-3.3.2。统计了运行PageRank 10次迭代的运行时间开销。

测试数据采用不同规模的真实数据和人工合成数据；人工合成数据集由数据生成器生成。实验中我们选择了定点规模不同的5个真实数据集[8]，它们的统计信息见表1。

6.1 真实数据集测试结果

利用表1中描述的5个真实数据集，在Giraph1.0.0的内存版（Giraph 1.0.0_MEM）和磁盘版（Giraph 1.0.0_HDD）、Hama 0.6.4和BC-BSP 2.0系统上分别运行了PageRank算法，得到了图5所示的结果。

由图5展示的结果可得出：BC-BSP2.0的性能优于另外3个对比系统，总体上比Giraph1.0.0的内存版的性能好。

6.2 虚拟数据集测试结果

通过测试虚拟数据集进行系统可扩展性的对比，我们可知：数据从1 000万顶点至11 000万顶点，主要用于测试系统的可扩展性和计算性能，平均出度规模为11.5。

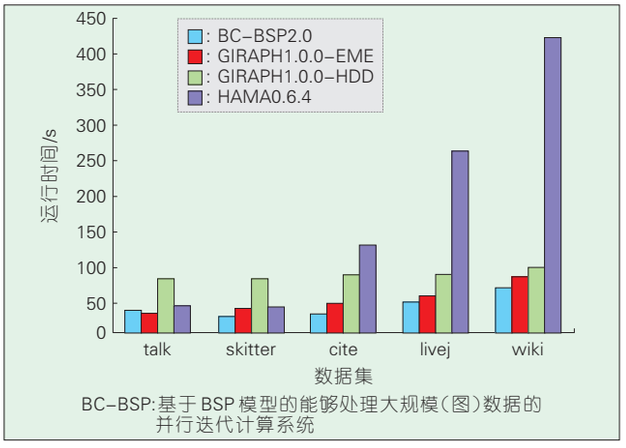
由图6展示的结果可得出：图数据的顶点从1 000万到11 000万，BC-BSP 2.0在数据吞吐量以及在相同数据集的处理效率上都要优于HAMA-0.6.4，并优于GIRAPH-1.0.0_HDD，效率略低于GIRAPH-1.0.0_MEM，但可扩展性更好。

7 结束语

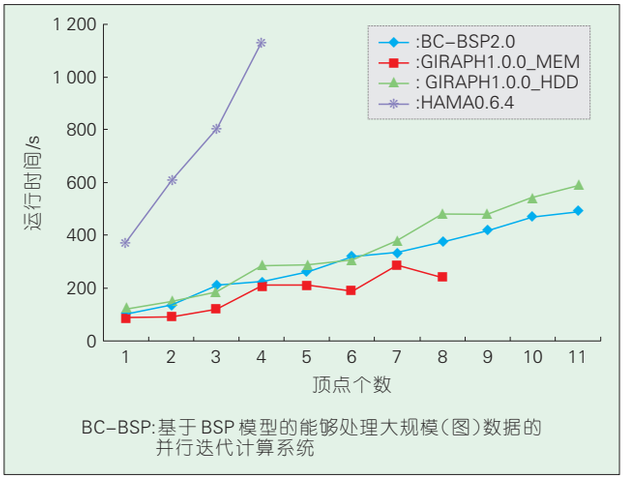
文章描述了在Java语言环境下基于BSP模型实现的用于大规模图数据迭代处理的系统BC-BSP。该系统在Pregel思想的基础上，实现了它的基本功能，同时增加了若干优化策略，包括增加了均衡的数据划分策略，使得每个任务处理的节点数量尽可能相近，图数据处理和消息通信过

▼表1 测试用真实数据集信息

名称	顶点数目	边数目	平均出度	磁盘文件规模/ MB
1: skitter	1、696、414	12、791、712	7.54	99.29
2: talk	2、393、819	7、415、229	3.098	67.42
3: cite-pr	6、009、555	22、528、503	3.7488	215
4: livej-pr	4、847、571	73、841、344	15.233	553.56
5: wiki-pr	5、716、808	135、877、199	23.768	1 047.1



◀图5 真实数据集PageRank测试结果



◀图6 模拟数据集PageRank测试结果

程中的磁盘暂存使得在计算节点及其内存资源有限的情况下可以处理较大的数据，具有更高的可扩展性。

尽管在系统开发过程中已经做了大量的优化工作，但是系统还有可优化的地方。例如，关于图数据结构的优化与改进：(1)目前不论是图顶点对象还是边对象都采用字符串方式存储，可以改成支持泛型的实现；(2)系统利用写检查点机制实现了故障恢复，但是对于故障类型的捕获和诊断还有待进一步加强；(3)在系统实现中发现Java环境对内存的开销巨大，因此对数据结构的设计以及使用需要仔细地斟酌。

致谢

本研究得到东北大学于戈教授和谷峪副教授的帮助，以及中国移动(苏州)研发中心钱岭博士的支持，谨致谢意！

本系统开发工作是由东北大学计算机软件所王志刚博士研究生以及许多已经毕业的研究生共同完成，对他们谨致谢意！

参考文献

[1] SERGEY B, LARRY P. The Anatomy of a Large-Scale Hypertextual Web Search Engine [J]. Computer Networks and ISDN Systems, 1998, 30(98): 1-7
[2] GUERON M, LLIA R, MARGULIS G. Pregel: A System for Large-Scale Graph Processing [J]. American Journal of Emergency Medicine, 2009, 18(18):135-146
[3] VALIANT L G. Bulk-Synchrony: A Bridging Model for Parallel Computation [J]. Communications of the ACM, 1990, 33(8): 103-111
[4] Welcome to Hama Project [EB/OL].[2011-07-13] . http://incubator.apache.org/hama/
[5] AVERY C, CHRISTAN K. Giraph: Large-Scalegraph Processing Infrastructure on Hadoop [EB/OL]. [2011-06-29]. Hadoop Summit 2011, https://github.com/aching/Giraph
[6] LOW Y, BICKSON D, GONZALEZ J, GUESTRIN C, et al. Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud [J]. Proceedings of the VLDB Endowment, 2012, 5(8): 716-727
[7] MAMOU H. An Experimental Comparison of Pregel-Like Graph Processing Systems [C]// Proceedings of Vldb Endowment. USA: ACM 2014: 7(12):1047-1058
[8] Using the Stanford Large Network Dataset Collection [EB/OL], https://snap.stanford.edu/data/index.html

作者简介

刘恩孚，东北大学计算机科学与工程学院在读硕士研究生；主要研究方向为数据挖掘、图数据管理。

冷芳玲，东北大学计算机科学与工程学院讲师，中国计算机学会高级成员；主要研究方向为数据仓库和联机分析处理等。

鲍玉斌，东北大学计算机科学与工程学院教授，中国计算机学会高级成员；主要研究方向为联机分析处理、云计算、图数据管理等。