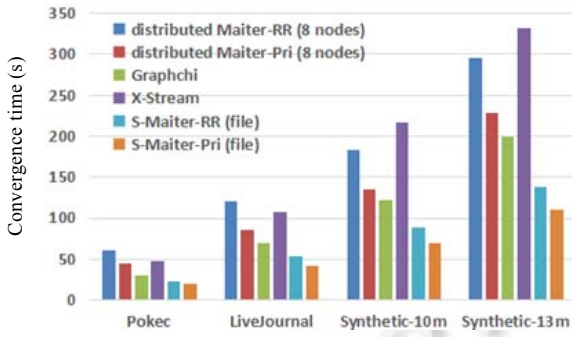

Fig.5 Convergence time of WCC on different datasets

图 5 WCC 算法在不同数据集下的收敛时间

4.3 内存配额对性能的影响

图 6 显示了随着内存预算大小从 0.5GB~3GB,PageRank(运行 Synthetic-10m 数据集)在 GraphChi[7]、X-Stream[13]、S-Maiter-RR(file)、S-Maiter-Pri(file)下的收敛时间.从图中可以看到:S-Maiter-RR(file)的性能相对 GraphChi[7]提高了 1.5 倍,相对 X-Stream[13]提高了 1.4 倍;S-Maiter-Pri(file)性能相对 GraphChi[7]提高了 1.7 倍,相对 X-Stream[13]提高了 1.6 倍.例如:在内存预算大小为 3GB 时,GraphChi[7]需要 820.658s,X-Stream[13]需要 777.22s,S-Maiter-RR(file)只需要 559.156s,S-Maiter-Pri(file)只需要 474.156s.

S-Maiter 相对于 Graphchi[7]会产生更少的 shard,如图 7 所示.在内存预算大小从 0.5GB~3GB 的情况下,Graphchi[7]产生了 5 个~23 个 shard;随着 shard 的增多,在滑动窗口的过程中会带来更多额外的随机的 I/O.而 S-Maiter 只产生 4 个 shard,这是因为 S-Maiter 分片只是为了支持基于 shard 的优先级调度,在 S-Maiter 中,即使 shard 增多,也不会产生随机的 I/O.

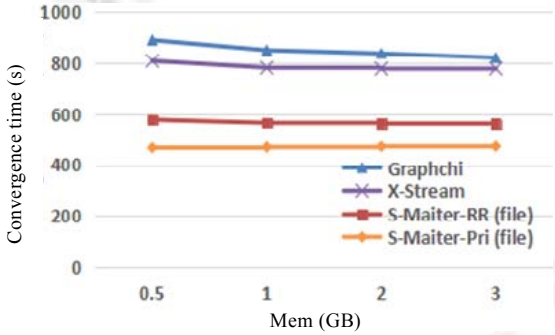

Fig.6 Convergence time comparison with different memory quotas (Synthetic-10m)

图 6 不同内存配额下的收敛时间对比(Synthetic-10m)

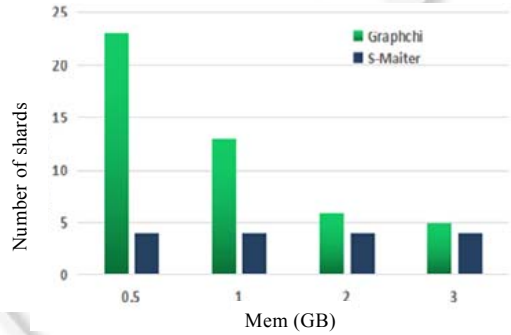

Fig.7 Number of shards comparison with different memory quotas (Synthetic-10m)

图 7 不同内存配额下的 shard 数量对比(Synthetic-10m)

4.4 读写数据量对比

图 8 和图 9 分别显示了 Graphchi[7]、X-Stream[13]、S-Maiter-RR(file)、S-Maiter-Pri(file)的写/读数据量.结果显示,S-Maiter 写/读数据量比 Graphchi[7]和 X-Stream[13]要小得多.具体来说:在整个计算过程中,在内存预算大小从 0.5GB~3GB,无论内存预算是多少,Graphchi[7]都要将 86.5GB 的数据量写入磁盘,并从磁盘读取相同大小的数据量到内存.X-Stream[13]将 56.438GB 的数据量写入磁盘.S-Maiter-RR(file)和 S-Maiter-Pri(file)只将最终收敛结果写回磁盘,所以只写了 0.17G 的数据量.在读取数据量方面,X-Stream[13]读取了 112.65GB 的数据量,

S-Maiter-RR 读取 41.7GB 数据量,S-Maiter-Pri(file)只读取 36GB 的数据量.这是因为 S-Maiter-Pri(file)使用了 shard 级的优先级调度,加快了收敛速度,所以读取的数据量相对 S-Maiter-RR(file)会少一些.S-Maiter 在数据访问中的优越性主要是由于将不变的结构数据与可变的值数据分离,并将可变的值数据缓存在内存中,对值数据的更新和重复访问可以在内存中完成,最小化了对图数据访问产生的 I/O 开销.

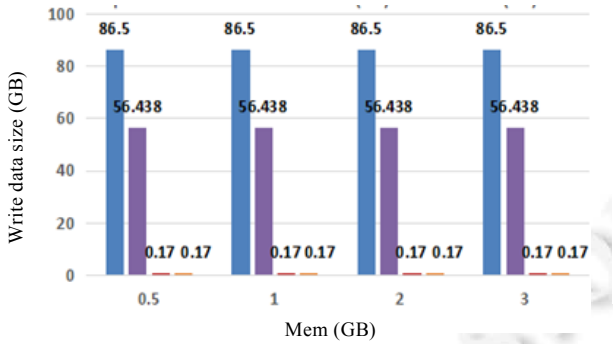

Fig.8 Write data size comparison (Synthetic-10m)

图 8 写入数据量对比(Synthetic-10m)

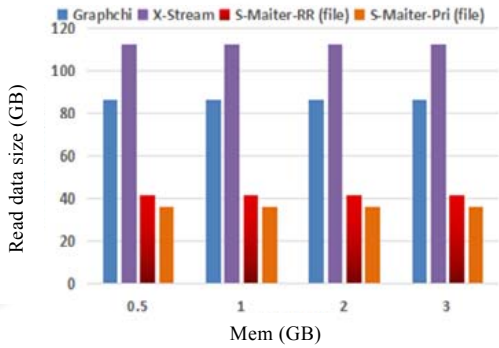

Fig.9 Read data size comparison (Synthetic-10m)

图 9 读入数据量对比(Synthetic-10m)

4.5 与分布式Maiter的性能对比

本节对 S-Maiter 和分布式 Maiter 进行了性能对比实验,内存预算大小设置为 4GB.S-Maiter 的内存模式相比于外存模式性能提升很大.图 10 所示为 PageRank(运行 Pokec 数据集)算法在 S-Maiter-RR(file)、S-Maiter-Pri(file)、S-Maiter-RR(men)和 S-Maiter-Pri(men)下的收敛时间.可以看出:内存模式相对外存模式性能提高了大约 1.5 倍,这是因为内存模式并不产生 I/O 开销.

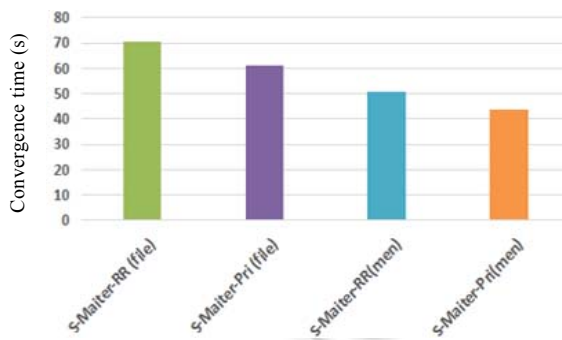

Fig.10 Memory mode vs.external memory mode on convergence time (Pokec)

图 10 内存模式与外存模式的收敛时间比较(Pokec)

然而,当运行比较大的数据集(如 Synthetic-13m)时,由于数据集过大,4GB 的内存已经容纳不下了,只能在外存模式或分布式环境下进行计算.为此,我们用 S-Maiter-RR(file)、S-Maiter-Pri(file)与 distributed Maiter-RR、distributed Maiter-Pri 在 Synthetic-13m 数据集上进行对比实验.

图 11 显示了 PageRank 算法(运行 Synthetic-13m 数据集)在分布式 Maiter[4]与单机外存 S-Maiter 下的收敛时间对比.当分布式 Maiter[4]在一个计算节点上计算时,虽然在内存运算,但是相对于 S-Maiter 的内存模式,分布式 Maiter[4]在单个计算节点上并没有使用多线程进行更新计算,S-Maiter 在内存模式下的计算速度快于外存模式(如图 10 所示),S-Maiter 在外存模式下的计算速度快于分布式 Maiter[4]在单个计算节点上的计算速度.集群数量大于 1 时,分布式 Maiter[4]存在大量的网络开销,但随着集群大小从 2 增长到 16 时,Maiter[4]的性能也会逐渐提升.当集群大小为 16 时,distributed Maiter-RR 需要 953s,distributed Maiter-Pri 需要 780.83s,S-Maiter-RR(file)需要

1 013.42s,S-Maiter-Pri(file)需要 848.167s.我们可以看到,单机运行 S-Maiter 的收敛时间与 16 台分布式集群上 Maiter[4]的收敛时间相差不多.S-Maiter 在合理的时间内,只用了 1 台计算机解决了分布式 Maiter[4]用 16 台计算机集群解决的问题,减少了获取和管理大规模集群需要的昂贵代价,也不用考虑复杂的图分区工作.

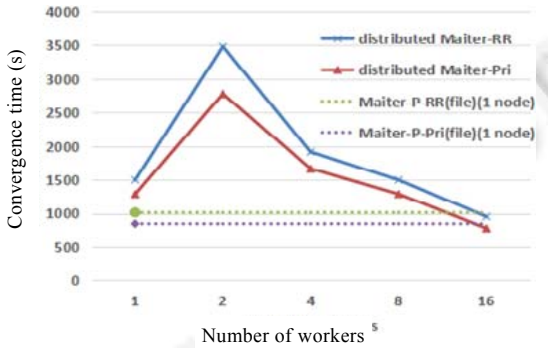

Fig.11 S-Maiter (1 computing node) vs. distributed Maiter (1~16 computing nodes) on convergence time

图 11 单机 S-Maiter 与分布式 Maiter (1~16 个节点)的收敛时间对比

4.6 update线程与I/O线程数量比

在使用 SSD 时,我们可以开启多个 I/O 线程提高读取数据的存储量,提高计算效率.为了探究 update 线程与 I/O 线程的数量比对 S-Maiter 性能的影响,本文设计了在不同比例下的 S-Maiter 执行迭代算法的收敛速度对比实验.图 12 显示了在不同的 update 线程与 I/O 线程比例下,WCC(运行LiveJournal数据集,内存预算大小 0.5GB)在 S-Maiter-RR(file)下收敛的平均时间.从图中可以看出:随着 update 线程与 I/O 线程数量比例不断增大,收敛速度出现先提高后降低的现象,并在 update 线程与 I/O 线程数量比为 8:2 时收敛速度最快.update 线程与 I/O 线程的数量比对 S-Maiter-Pri(file)的影响类似,在此不再详述.

经分析发现:如果 I/O 线程相对于 update 线程过多,就会出现缓存结构数据过多,update 线程执行不过来,出现“撑着了”的情况;如果 I/O 线程相对于 update 线程过少,就会出现 update 线程执行太快,但是 I/O 线程太少,数据跟不上,出现“吃不饱”的情况.这两种情况导致了图 12 所示的现象,所以应该避免这两种情况的发生.

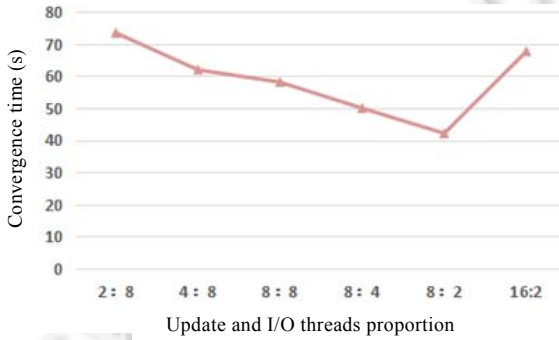

Fig.12 update and I/O threads proportion (LiveJournal)

图 12 update 线程与 I/O 线程比例(LiveJournal)

5 相关工作

随着大规模数据集的产生以及学术界和工业界对图数据分析的深入研究,出现了一系列大规模图数据处理系统,来完成各种复杂的大规模图数据分析任务.

- 基于同步方式的分布式图处理系统

PEGASUS[8]和 GBase[23]基于 MapReduce,并且支持矩阵向量乘法.Pregel[5]并不基于 MapReduce,Pregel[5]在概念模型上遵循 BSP 模型[24],整个计算过程由若干顺序执行的超级步(super step)组成,系统从一个超级步迈向下一个超级步,直达到算法的终止条件.Blogel[25]基于 BSP 模型,扩展了以顶点为中心的编程模型,提出了以块为中心(block-centric)的编程模型.它以块为单位进行图处理,开发人员可以综合考虑块内其他顶点的状态来指定图顶点执行不同的计算.GraphX[26]是一个基于 Spark[16]构建的图处理系统,支持弹性分布式图(resilient distributed graph,简称 RDG)的抽象数据结构.它将图存储为表格结构,并将图计算操作实现成了对几个表格的分布式连接(join)操作,以利用 Spark[16]提供的底层计算引擎实现高效的内存图处理.Pregelix[27]是一个基于 Hyracks[28]构建的 BSP 模型的分布式外存实现版本,它将图数据和消息数据存储为数据多元组(data tuples),通过分布式连接操作来实现数据分发.PrIter[17]是基于 Hadoop Online[29]构建的 BSP 模型的图处理系统,它支持带优先级的图处理,可以确保加速图计算的收敛,尤其适合在线 top-*k* 查询.Trinity[11]利用分布式内存键值对存储来支持在线图查询操作和离线的 BSP 图处理.

- 基于异步方式的分布式图处理系统

分布式版本的 GraphLab[6]同时支持同步图处理和异步图处理,采用拉取(pull)模式从邻居顶点获取状态信息,并基于 Gather-Apply-Scatter(GAS)模型,通过分布式锁来保证图计算过程中需要满足的多种一致性需求. GraphLab 的同步计算表现出了较好的性能,性能优于大部分同步图处理系统.然而对于异步图计算,仅在支持部分异步图算法(置信传播算法 Belief Propagation 等)上有着不错的性能,对于大多数图算法(PageRank 等),异步图处理的性能却相当糟糕[6],甚至比同步计算性能差数十倍.究其原因,是由于为了实现异步数据一致性而实现的分布式锁,它造成了极大的性能开销,导致得不偿失.近年来,除了 GraphLab[6],也涌现除了其他几个典型的支持异步图处理的系统.GRACE[30]支持用户可定制的图顶点调度和消息选择处理策略,然而它的运行环境是单机共享内存环境下,而不是分布式环境,计算调度等在共享内存环境下容易处理的问题一旦转移到分布式环境下将变得非常难以解决,这导致了 GRACE 系统的可用性受限;Giraph++[31]提出以子图为中心(graph-centric)的处理模型,通过图数据分割后,允许子图内图顶点多次迭代,而全局同步被尽量推迟执行,避免频繁全局同步带来的大量系统等待开销; GiraphUC[32]相对应于 BSP 模型提出 BAP(barrierless asynchronous parallel)模型,并基于开源的同步图处理系统 Giraph[33]实现,它的核心思路与 Giraph++[31]类似,都是通过区分本地同步和全局同步来提高性能.

- 基于累加计算的分布式图处理系统 Maiter

Maiter[4]采用基于推送(push)模式的异步累加式的图处理方法,各图顶点之间交互的信息并不是顶点状态而是顶点状态的变化,并且累积这些变化信息即可得到最终的收敛结果.它不要求包括本地同步或者全局同步等的任何同步操作,各计算节点在交互信息时完全独立,这样即达到了理想的完全异步状态.Maiter 目前已经支持一大类算法,包括 PageRank,SSSP,SimRank 等.本文提出的 S-Maiter 同样基于异步累加计算模型,利用单机大容量磁盘和并行计算技术进行图处理优化,对于没有必要在分布式环境执行的图处理问题或者缺少分布式计算条件的情况下提供一种更好的解决方案.分布式 Maiter 利用分布式集群结合分布式计算的特点加速图计算,而 S-Maiter 利用单机大容量磁盘结合图处理的 I/O 特点和并行计算特点来加速图计算.

- 单机图处理系统

X-Stream 遵循以边为中心的计算模型.X-Stream 会将部分中间结果写回磁盘以便后续的处理,这将产生双倍的 I/O 代价,导致额外的计算成本和数据加载开销.此外,X-Stream 也不支持顶点的选择调度策略.GraphChi 将图数据在磁盘中组织成若干个分片,每个分片包含一系列顶点信息和以及与这些顶点相关的入边和出边信息,GraphChi 要求每个分片都能够装进内存.GraphChi 在计算之前必须将整个分片全部载入内存,这种约束阻碍了计算和 IO 的并行性.在相同内存限制下,GraphChi 相对 S-Maiter 会产生更多的分片,随之也产生了更多的数据传输和随机 I/O.S-Maiter 可以不间断地流式计算分片里的数据,并且将可变的顶点值数据缓存到内存中,减少了随机 I/O,最小化 I/O 代价.TurboGraph[12]虽然可以无延迟地处理图数据,但是 TurboGraph 是针对 SSD 设计的,S-Maiter 不仅可以应用在 SSD,还可应用在廉价的机械硬盘.

6 结 论

本文提出了流式处理的异步计算方案 ASP,一种用基于单机大容量磁盘的方法解决大图计算的方案,包括适用于异步累加迭代的图存储模型和计算模型.并基于该模型实现了基于外存的图计算框架 S-Maiter,有效解决了频繁更新和读取磁盘数据导致大量 I/O 的问题.实现了对磁盘数据的完全顺序访问,有效利用了内存和 CPU 资源,能并行化图数据的流式载入和更新函数的异步执行.通过一系列的实验,其结果表明:S-Maiter 比 Graphchi 和 X-Stream 更快,比 Maiter 性价比更高.

References:

[1] Yu G, Gu Y, Bao YB, Wang ZG. Large scale graph data processing on cloud computing environments. Chinese Journal of Computers, 2011,34(10):1753−1767 (in Chinese with English abstract).

[2] Li XT, Li JZ, Gao H. An efficient frequent subgraph mining algorithm. Ruan Jian Xue Bao/Journal of Software, 2007,18(10): 2469−2480 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/2469.htm [doi: 10.1360/jos182469]

[3] Pan W, Li ZH, Wu S, Chen Q. Evaluating large graph processing in MapReduce based on message passing. Chinese Journal of Computers, 2011,34(10):1768−1784 (in Chinese with English abstract).

[4] Zhang Y, Gao Q, Gao L, Wang C. Maiter: An asynchronous graph processing framework for delta-based accumulative iterative computation. IEEE Trans. on Parallel & Distributed Systems, 2014,25(8):2091−2100. [doi: 10.1109/TPDS.2013.235]

[5] Malewicz G, Austern MH, Bik AJC, Dehnert JC, Horn I, Leiser N, Cazjkowski G. Pregel: A system for large-scale graph processing. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2010. 135−146. [doi: 10.1145/1582716.1582723]

[6] Low Y, Gonzalez J, Kyrola A, Bickson D, Guestrin C, Hellerstein JM. Distributed GraphLab: A framework for machine learning and data mining in the cloud. Proc. of the VLDB Endowment, 2012,5(8):716−727. [doi: 10.14778/2212351.2212354]

[7] Kyrola A, Blelloch GE, Guestrin C. GraphChi: Large-Scale graph computation on just a PC. In: Proc. of the 10th USENIX Symp. on OSDI. 2012. 31−46.

[8] Kang U, Tsourakakis CE, Faloutsos C. PEGASUS: A peta-scale graph mining system implementation and observations. In: Proc. of the 9th IEEE Int'l Conf. on Data Mining. IEEE Computer Society, 2009. 229−238. [doi: 10.1109/ICDM.2009.14]

[9] Chen R, Weng X, He B, Yang M. Large graph processing in the cloud. In: Proc. of the 2010 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2010. 1123−1126. [doi: 10.1145/1807167.1807297]

[10] Krepska E, Kielmann T, Fokkink W, Bal H. HIPG: Parallel processing of large-scale graphs. ACM SIGOPS Operating Systems Review, 2011,45(2):3−13. [doi: 10.1145/2007183.2007185]

[11] Shao B, Wang H, Li Y. Trinity: A distributed graph engine on a memory cloud. In: Proc. of the 2013 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2013. 505−516. [doi: 10.1145/2463676.2467799]

[12] Han WS, Lee S, Park K, Lee JH, Kim MS, Kim J, Yu H. TurboGraph: A fast parallel graph engine handling billion-scale graphs in a single PC. In: Proc. of the 19th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2013. 77−85. [doi: 10.1145/2487575.2487581]

[13] Roy A, Mihailovic I, Zwaenepoel W. X-Stream: Edge-Centric graph processing using streaming partitions. In: Proc. of the 24th ACM Symp. on Operating Systems Principles. ACM Press, 2013. 472−488. [doi: 10.1145/2517349.2522740]

[14] Cheng J, Liu Q, Li Z, Fan W, Lui JCS, He C. VENUS: Vertex-Centric streamlined graph computation on a single PC. In: Proc. of the 2015 IEEE 31st Int'l Conf. on Data Engineering (ICDE). IEEE, 2015. 1131−1142. [doi: 10.1109/ICDE.2015.7113362]

[15] Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I. Spark: Cluster computing with working sets. HotCloud, 2010,10(10-10):95.

[16] Zhang Y, Gao Q, Gao L, Wang C. Imapreduce: A distributed computing framework for iterative computation. In: Proc. of the 2011 IEEE Int'l Symp. on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW). IEEE, 2011. 1112−1121. [doi: 10.1109/IPDPS.2011.260]

[17] Zhang Y, Gao Q, Gao L, Wang C. PrIter: A distributed framework for prioritized iterative computations. In: Proc. of the 2nd ACM Symp. on Cloud Computing. ACM Press, 2011. 13. [doi: 10.1145/2038916.2038929]

[18] Engle C, Lupher A, Xin R, Zaharia M, Franklin MJ, Shenker S, Stoica I. Shark: Fast data analysis using coarse-grained distributed memory. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2012. 689−692. [doi: 10.1145/2213836.2213934]

[19] Kang U, Tong H, Sun J, Lin CY, Falontsos C. GBASE: A scalable and general graph management system. In: Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. San Diego: DBLP, 2011. 1091−1099. [doi: 10.1145/2020408.2020580]

[20] Bertsekas DP. Distributed asynchronous computation of fixed points. Mathematical Programming, 1983,27(1):107−120. [doi: 10.1007/BF02591967]

[21] Stanford dataset collection. http://snap.stanford.edu/data/

[22] Snap. http://github.com/snap-stanford/snap

[23] Gerbessiotis AV, Valiant LG. Direct bulk-synchronous parallel algorithms. Journal of Parallel Distributed Computing, 1994,22(2):251−267. [doi: 10.1006/jpdc.1994.1085]

[24] Yan D, Cheng J, Lu Y, Ng W. Blogel: A block-centric framework for distributed computation on real-world graphs. In: Proc. of the VLDB Endowment. 2014. 1981−1992. [doi: 10.14778/2733085.2733103]

[25] Xin RS, Crankshaw D, Dave A, Gonzalez JE, Franklin MJ, Stoica I. GraphX: Unifying data-parallel and graph-parallel analytics. In: Proc. of the Computer Science. 2014.

[26] Borkar V, Borkar V, Jia J, Carey MJ, Condie T. Pregelix: Big(ger) graph analytics on a dataflow engine. Proc. of the VLDB Endowment, 2014,8(2): 161−172. [doi: 10.14778/2735471.2735477]

[27] Borkar V, Carey M, Grover R, Onose N, Vernica R. Hyracks: A flexible and extensible foundation for data-intensive computing. In: Proc. of the IEEE Int'l Conf. on Data Engineering. IEEE Computer Society, 2011. 1151−1162. [doi: 10.1109/ICDE.2011.5767921]

[28] Hadoop. http://hadoop.apache.org/

[29] Wang G, Xie W, Demers A, Gehrke J. Asynchronous large-scale graph processing made easy. In: Proc. of the CIDR. 2013.

[30] Balmin A, Balmin A, Corsten SA, Tatikonda S, Mcpherson J. From "think like a vertex" to "think like a graph". Proc. of the VLDB Endowment, 2013, 7(3):193−204. [doi: 10.14778/2732232.2732238]

[31] Han M, Daudjee K. Giraph unchained: Barrierless asynchronous parallel execution in pregel-like graph processing systems. In: Proc. of the VLDB Endowment. 2015.

[32] Apache Giraph. http://giraph.apache.org

附中文参考文献:

[1] 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] 李先通,李建中,高宏.一种高效频繁子图挖掘算法.软件学报,2007,18(10):2469−2480. http://www.jos.org.cn/1000-9825/18/2469.htm [doi: 10.1360/jos182469]

[3] 潘巍,李战怀,伍赛,陈群.基于消息传递机制的 MapReduce 图算法研究.计算机学报,2011,34(10):1768−1784.

李金吉(1993－),男,吉林松原人,硕士生,主要研究领域为大数据处理,分布式系统.

张岩峰(1982－),男,博士,教授,CCF 专业会员主要研究领域为大数据处理,分布式系统,云计算.

巩树凤(1991－),男,博士生,主要研究领域为大数据处理,分布式系统.

于戈(1962－),男,博士,教授,博士生导师,CCF 会士,主要研究领域为数据库,分布式系统,嵌入式系统.

高立新(1968－),女,博士,教授,博士生导师,主要研究领域为社交网络,路由策略,网络虚拟化,云计算.