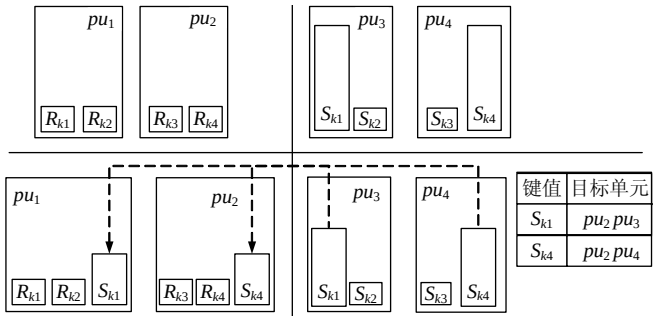


Fig.4 Example of S2SM

图 4 S2SM 算法举例

4.4 状态一致性迁移

基于二部图连接模型设计的 BiStream 系统[12]实现了动态扩展(或缩减)查询节点的资源管理策略,但其仅支持基于窗口的连接模型,无法对全历史数据的在线连接提供状态一致性的保证.本节引入内存文件系统 Tachyon[23]存储算子的状态信息,并使用算子状态管理器(operator states manager,简称 OSM)[24]动态迁移不同节点间的处理单元,以保证连接算子的状态一致性和可扩展性.我们利用该管理器实现了全历史数据在线连接操作的动态资源管理.状态管理器的架构图如图 5 所示.节点中的每个处理进程(storm 中称作 worker)管理自身的处理单元(pu_1~pu_n).状态管理器负责维护处理单元的数据不被当作垃圾回收,并将处理单元按照维持、迁出和迁入这 3 种状态进行管理.维持状态的处理单元仍运行在本进程中;迁出状态的处理单元将被迁移至其他节点,相关状态信息保存至文件系统(file server);迁入状态的处理单元由其他节点迁入该节点,并将其状态信息通过文件系统加载至恢复线程(retriever).该管理器确保了全历史数据在线连接操作的动态资源管理.

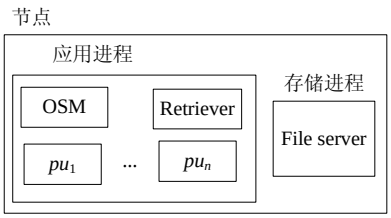


Fig.5 Architecture of OSM[24]

图 5 OSM 架构[24]

5 实验与结果分析

5.1 实验准备

1) 实验环境

本文实验平台用 1GB 网络连通 14 个物理节点,其中 5 个是使用 kafka 的数据发送节点,1 个是 Storm 的 nimbus 节点,其余 8 个是 Storm 的 supervisor 节点.数据发送与 nimbus 各节点配置是:CPU:Intel E5-2620 2.00GHz,Memory:4GB;supervisor 各节点配置是:CPU:两个 Intel E5-2620 2.00GHz,Memory:64GB;操作系统:Ubuntu-14.04.3;Storm 版本 0.9.5.

2) 数据集

我们使用合成数据和真实数据进行实验对比.首先,以 TPC-H[25]作为基准测试,并利用 *dbgen* 产生数据集.所有数据在发送至 Storm 之前,均存放于数据发送节点的 Kafka 中.为测试数据集的不同倾斜程度,在连接属性上使用不同倾斜度 z 的 Zipf 分布.默认情况下,我们令 z=1,并产生 10GB 数据.其次,第 2 个数据集是 Linear Road

Benchmark(LRB)[26],该基准测试模拟高速公路的收费系统,统计不同路段、不同方向的车速信息.我们产生了1 200 万条车辆数据信息,如图 6 所示,车速分布具有明显的倾斜性.

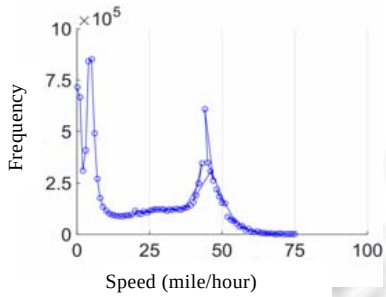


Fig.6 Speed distribution of LRB

图 6 LRB 车速分布图

3) 查询任务

本文共选取 4 个查询任务,其中两个是 TPC-H 提供的等值连接 Q3 和 Q5,一个是文献[11,12]中使用的范围查询(band).Band 查询描述为:

- SELECT *, FROM LINEITEM *L*1, LINEITEM *L*2
  WHERE *ABS*(*L*1.*orderkey*–*L*2.*orderkey*)≤1
  AND (*L*1.*shipmode*=‘TRUCK’ AND *L*2.*shipinstruct*=‘NONE’) AND *L*1.*Quantity*>48

另一个是 LRB 场景下,通过自连接操作统计同一车道下车速相同的车辆信息,其查询描述为:

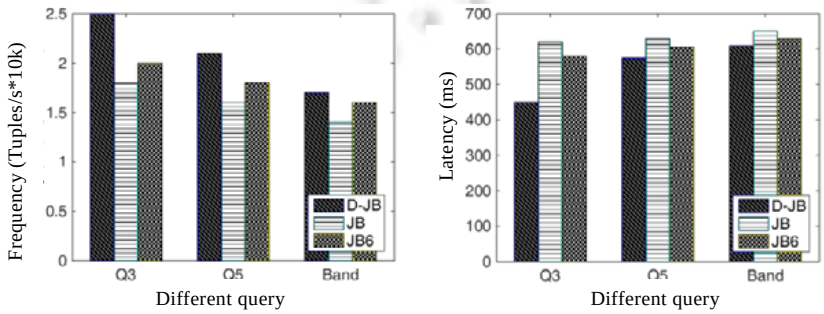
- SELECT *, FROM LRB *L*1, LRB *L*2
  WHERE *L*1.*Spd*=*L*2.*Spd* AND *L*1.*Lane*=*L*2.*Lane* AND *L*1.*Dir*=*L*2.*Dir* AND *L*1.*VID*!=*L*2.*VID*

4) 对比模型

本文使用 3 种算法对比分析查询性能:D-JB,JB 算子和 JB6.D-JB 是本文提出的算法,表示动态二部图连接模型.JB 算子表示平均分配集群节点至二部图的各侧.JB6 表示平均分配节点后,二部图各侧内部的处理单元分成 6 个子组做随机路由.

5.2 吞吐率和延迟

我们使用 TPC-H 中 *z*=1 的 10GB 数据,针对 3 个不同的查询任务,对比了 3 个模型的吞吐率和延迟.如图 7(a)所示,由于 D-JB 动态调整了处理单元的负载情况,其吞吐率最高.而 JB 算子需要做单侧的全网广播操作,通信量较大,故其吞吐率最低.针对不同查询任务的 topology 处理延迟而言,图 7(b)说明,D-JB 的延迟均低于 JB 算子和 JB6.



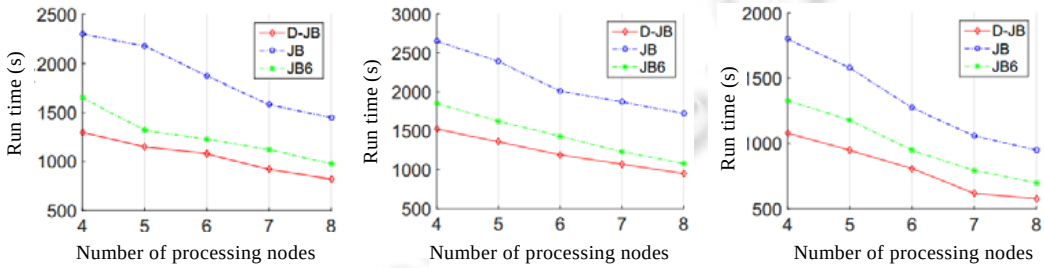
(a) Throughput (b) Latency

Fig.7 Throughput and latency

图 7 吞吐率和延迟

5.3 系统扩展性

当集群节点动态改变时,我们进一步分析 D-JB,JB 算子和 JB6 的可扩展性.由于 JB 算子和 JB6 不支持全历史数据连接的动态扩展,在增加处理节点时,我们将 TPC-H 数据暂存于 Kafka 的消息队列,待重启 Storm 的 topology 后继续发送数据(需减去重启 topology 的时间,以获得计算连接操作的实际运行时间).如图 8(a)~图 8(c)所示:D-JB 的运行时间是最短的,其扩展性最好;此外,由于 Q5 涉及到的连接操作最复杂,涉及的数据流最多,其运行时间高于其他两个查询任务.



(a) Q3 查询 (b) Q5 查询 (c) Band 查询

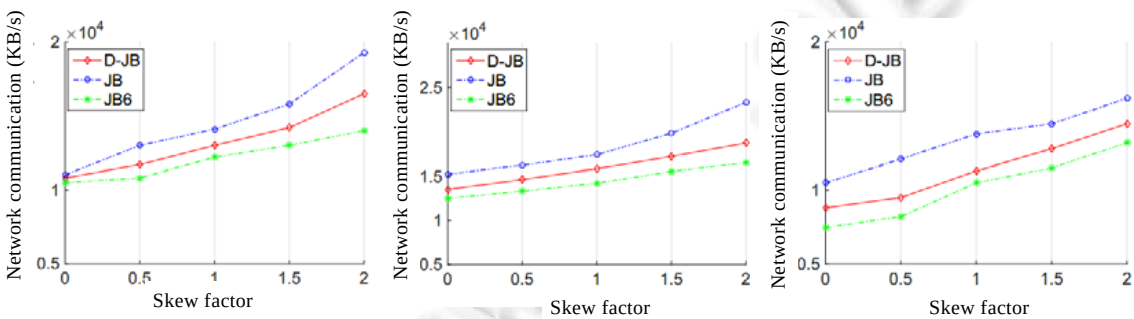
Fig.8 Run time

图 8 运行时间

5.4 通信代价

为验证 D-JB 模型应对不同倾斜度数据流的处理情况,我们通过改变 Zipf 的分布情况,测试 3 种模型的平均网络通信代价和数据迁移总量.对比图 9(a)~图 9(c)我们发现:

- JB 算子需要做全网广播操作,其平均网络通信代价最高;
- JB6 仅做子网广播操作,其通信代价最低;
- D-JB 由于做了数据迁移操作,其通信代价略高于 JB6;并且,由于 Q5 涉及到的数据流最多,其通信代价高于 Q3 和 Band.



(a) Q3 查询 (b) Q5 查询 (c) Band 查询

Fig.9 Network communication cost

图 9 网络通信代价

此外,根据不同倾斜度,图 10 给出了执行 D-JB 算法的数据迁移总量.可以发现:随数据倾斜度的增加,由数据迁移引起的网络通信总量也随之增加,但迁移总量相对于全部数据流量来讲相对较低.

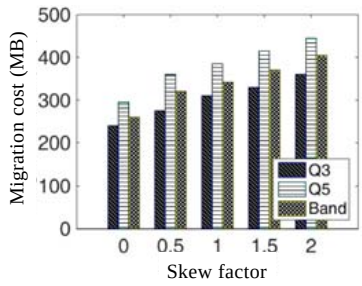


Fig.10 Total migration

图 10 迁移总量

5.5 归一化优化目标的选参设置

本文为设计归一化模型,为 3 个代价设置相关权重参数.首先,路由表仅在中心节点更新,维护代价最低;其次,在连接操作时,由于数据分裂导致的复制相同 K_{tup} 数据至多个节点,网络开销较高;最后,数据分裂导致多条数据在不同节点间迁移,网络开销最高.如图 11 所示,我们分别将α,β和γ设置为 0.2,0.3 和 0.5(conf_1);0.1,0.3,0.6(conf_2)和 0.1,0.4,0.5(conf_3).以 Q3 查询为例,我们可以发现,conf_1 的查询吞吐率最高.

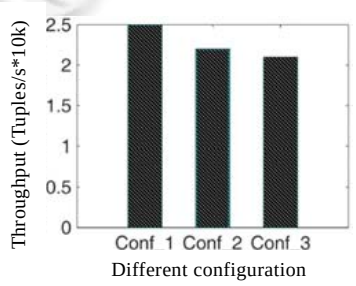


Fig.11 Throughput

图 11 吞吐率

5.6 真实数据流分析

除使用合成数据 TPC-H 外,我们还使用 LRB 的真实数据测试算法的有效性.执行第 5.1 节的查询任务,根据速度分布的倾斜性和数据量的不断增加,记录其执行时间.如图 12 所示,D-JB 算法的执行时间最短.

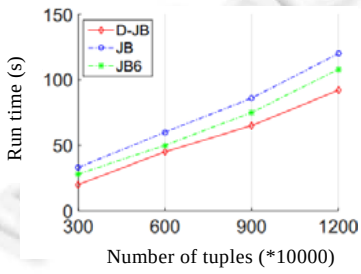


Fig.12 LRB run time

图 12 LRB 运行时间

6 总 结

本文提出了一种应对倾斜数据流的在线连接方法.基于二部图连接模型,在单侧节点内,我们设计了基于键值和元组的混合划分样式;在双侧节点间,我们制定了重新分布处理单元的策略;并通过引入状态管理器,实现

了可扩展的全历史数据连接操作.实验结果表明,本文提出的方法在系统性能、可扩展性和应对倾斜数据的能力等方面是可行且有效的.文中提到的数据迁移问题,文本通过制定归一化的优化目标并利用启发式规则以获取较优解.接下来,根据集群环境的不同,我们需要解决自动选取归一化优化目标的最优配置问题,并寻找更佳的优化方法使连接操作更加快速,系统更加稳定.

References:

[1] Sun DW, Zhang GY, Zheng WM. Big data stream computing: Technologies and instances. Ruan Jian Xue Bao/Journal of Software, 2014,25(4):839−862 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4558.htm [doi: 10.13328/j.cnki.jos.004558]

[2] Cui XC, Yu XH, Liu Y, Lü ZY. Distributed stream processing: A survey. Journal of Computer Research and Development, 2015, 52(2):318−332 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2015.20140268]

[3] Wang CK, Meng XF. Relational query techniques for distributed data stream: A survey. Computer Journal of Computers, 2016, 39(1):80−96 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2016.00080]

[4] Neumeyer L, Robbins B, Nair A, Kesari A. S4: Distributed stream computing platform. In: Proc. of the 10th EEE Int'l Conf. on Data Mining Workshops. Sydney: IEEE, 2010. 170−177. [doi: 10.1109/ICDMW.2010.172]

[5] Toshniwal A, Taneja S, Shukla A, Ramasamy K, Pater JM. Storm@Twitter. In: Proc. of the 2014 ACM Int'l Conf. on Management of Data. Snowbird: ACM Press, 2014. 147−156. [doi: 10.1145/2588555.2595641]

[6] Squall. https://github.com/epfldata/squall/

[7] Calcite. http://calcite.apache.org/

[8] Hwang JH, Balazinska M, Rasin A, Cetintemel U, Stonebraker M. High-Availability algorithms for distributed stream processing. In: Proc. of the 21st Int'l Conf. on Data Engineering. Tokyo: IEEE, 2005. 779−790. [doi: 10.1109/ICDE.2005.72]

[9] Fernandez RC, Migliavacca M, Kalyvianaki E, Pietzuch P. Integrating scale out and fault tolerance in stream processing using operator state management. In: Proc. of the 2013 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2013. 725−736. [doi: 10.1145/2463676.2465282]

[10] Walton CB, Dale AG, Jenevein RM. A taxonomy and performance model of data skew effects in parallel joins. In: Proc. of the 17th Int'l Conf. on Very Large Data Bases. Barcelona: Morgan Kaufmann Publishers, 1991. 537−548.

[11] Elseidy M, Elguindy A, Vitorovic A, Koch C. Scalable and adaptive online joins. Proc. of the VLDB Endowment, 2014,7(6): 441−452. [doi: 10.14778/2732279.2732281]

[12] Lin Q, Ooi BC, Wang Z, Yu C. Scalable distributed stream join processing. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data. Melbourne: ACM Press, 2015. 811−825. [doi: 10.1145/2723372.2746485]

[13] Ananthanarayanan R, Basker V, Das S, Gupta A, Jiang H. Photon: Fault-tolerant and scalable joining of continuous data streams. In: Proc. of the ACM 2013 Int'l Conf. on Management of Data. New York: ACM Press, 2013. 577−588. [doi: 10.1145/2463676.2465272]

[14] Zaharia M, Das T, Li H, Hunter T, Shenker S. Discretized streams: Fault-tolerant streaming computation at scale. In: Proc. of the ACM SIGOPS 24th Symp. on Operating Systems Principles. Farmington: ACM Press, 2013. 423−438. [doi: 10.1145/2517349.2522737]

[15] Spark streaming. https://spark.apache.org/streaming/

[16] Spark. http://spark.apache.org/

[17] Zaharia M, Chowdhury M, Das T, Dave A, Ma J. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proc. of the 9th USENIX Symp. on Networked Systems Design and Implementation. San Jose: USENIX, 2012. 15−28.

[18] Vitorovic A, Elseidy M, Koch C. Load balancing and skew resilience for parallel joins. In: Proc. of the 32nd IEEE Int'l Conf. on Data Engineering. Helsinki: IEEE, 2016. 313−324. [doi: 10.1109/ICDE.2016.7498250]

[19] Fang JH, Zhang R, Wang XT, Fu TZJ, Zhang ZJ, Zhou AY. Cost-Effective stream join algorithm on cloud system. In: Proc. of the 25th ACM Int'l Conf. on Information and Knowledge Management. Indianapolis: ACM Press, 2016. 1773−1782. [doi: 10.1145/2983323.2983773]

[20] Fang JH, Zhang R, Fu TZJ, Zhang ZJ, Zhou AY, Zhu JH. Parallel stream processing against workload skewness and variance. In: Proc. of the 26th Int'l Symp. on High-Performance Parallel and Distributed Computing. Washington: ACM Press, 2017. 15−26. [doi: 10.1145/3078597.3078613]

[21] Karmarkar N, Karp RM. An efficient approximation scheme for the one-dimensional bin-packing problem. In: Proc. of the 23rd Annual Symp. on Foundations of Computer Science. Chicago: IEEE, 1982. 312−320. [doi: 10.1109/SFCS.1982.61]

[22] Kafka. http://kafka.apache.org/

[23] Li HY, Ghodsi A, Zaharia M, Shenker S, Stoica I. Tachyon: Reliable, memory speed storage for cluster computing frameworks. In: Proc. of the 2014 ACM Symp. on Cloud Computing. Seattle: ACM Press, 2014. 1−15. [doi: 10.1145/2670979.2670985]

[24] Ding JB, Fu TZJ, Ma RTB, Winslett M, Yang Y, Zhang ZJ, Chao HY. Optimal operator state migration for elastic data stream processing. HAL-INRIA, 2015,22(3):1−8.

[25] TPCH. http://www.tpc.org/tpch

[26] Arasu A, Cheniack M, Maier D, Maskey AS, Ryvkina E, Stonebraker M, Tibbetts R. Linear road: A stream data management benchmark. In: Proc. of the 30th Int'l Conf. on Very Large Data Bases. Toronto: Morgan Kaufmann Publishers, 2004. 480−491. [doi: 10.1016/B978-012088469-8.50044-9]

附中文参考文献:

[1] 孙大为,张广艳,郑纬民.大数据流式计算:关键技术及系统实例.软件学报,2014,25(4):839−862. http://www.jos.org.cn/1000-9825/4558.htm [doi: 10.13328/j.cnki.jos.004558]

[2] 崔星灿,禹晓辉,刘洋,吕朝阳.分布式流处理技术综.计算机研究与发展,2015,52(2):318−332. [doi: 10.7544/issn1000-1239.2015.20140268]

[3] 王春凯,孟小峰.分布式数据流关系查询技术研究.计算机学报,2016,39(1):80−96. [doi: 10.11897/SP.J.1016.2016.00080]

王春凯(1981－),男,河南开封人,博士,主要研究领域为分布式数据流管理.

孟小峰(1964－),男,博士,教授,博士生导师,CCF会士,主要研究领域为Web数据管理,移动数据管理,XML 数据管理,云数据管理,隐私保护.