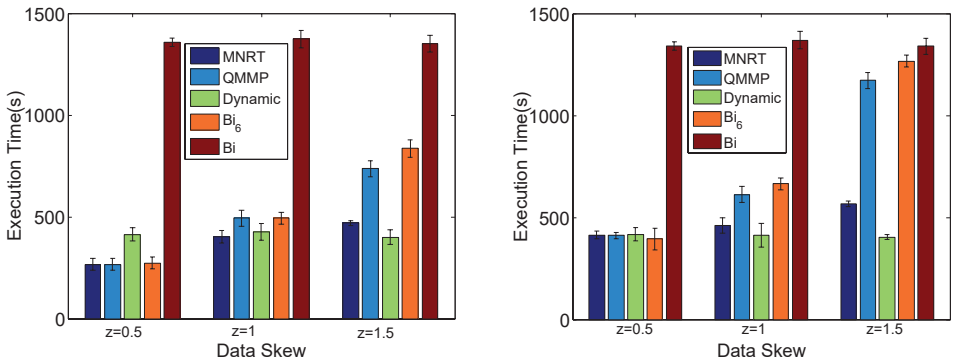
着交换来寻求均衡的暴力方法.这使得在海量 key 及均衡要求较高的情况下,其均衡调整的迁移计划制定时间大大增加;

- PKG[20]:将一个 key 拆分为更小的粒度后,有选择性地分发负载至不同并行节点上来避免负载倾斜问题.具体来说,PKG 将 key 拆分为两个部分,之后将拆分的两个部分置于不同的处理节点上,进而根据两个负载节点的负载,有选择性地分发后续到来的元组.本文使用了 PKG 提供的源码(https://github.com/gdfm/partial-key-grouping);
- Dynamic[22]:即矩阵模型的具体实现方法.Dynamic 将节点分布在矩阵的长和宽两个维度上进行数据流的连接操作,做连接的两条流通过这两个方向流入.然而,由于 Dynamic 设定矩阵内节点的个数为 2 的幂次方个,这使得矩阵形状的变化存在一定的局限性,从而浪费计算资源;
- Bi 和 Bi_6[19]是二分图模型处理数据流上的连接操作实现.Bi_6 表示在二分图的每组有 6 个子组,Bi 表示在二分图的各边没有子组划分.Bi 方法需要广播一侧的流数据到另一侧的所有处理节点上,这增大了网络代价;Bi_6 将处理节点分为子组,在一定程度上限定了该架构应对数据倾斜的特性.

5.2 实验结果

5.2.1 可扩展性-全历史连接操作

图 3 显示了将 12GB 的 TPC-H 数据处理完成的系统耗时情况.实验将元组装载的速度设置为最大,以使各个负载节点的计算能力充分发挥.图 3(a)展示了等值连接 EQ_5 在不同数据倾斜情况下的执行时间:当数据倾斜较小时,系统负载也较为均衡,因此 MNRT 和 QMMP 方法根据 key 定位使得数据的广播量减少,进而使得每个负载节点的负载较小;随着数据倾斜变严重,QMMP 对 key 的拆分动作增多,这使得后续连接操作中的元组广播频率增加,从而增加了节点的负载均衡计算,这一状况在图 3(b)中体现得尤为明显.对于二分图方法,当数据倾斜度严重时,Bi_6 方法中的分组受到了挑战,例如,当 z=1.5 时,大粒度的 key 占比增加,这造成了子组内广播的数据量增大,从而增加了负载节点的计算量,使系统的执行时间延长.Bi 方法由于不分子组,即全广播连接运算,这使得其架构内的负载节点的计算负载最大,从而导致其延时较高.基于矩阵模型的 Dynamic 方法随机分发元组,因此执行时间不受数据倾斜的影响;同时,由于该方法保证了两条数据流中的任意元组可碰面,等值或非等值的连接操作谓词不影响其性能,如图 3(a)和图 3(b)所示.



(a) EQ_5 查询语句 (b) B_{NCI} 查询语句

Fig.3 Execution time

图 3 执行时间测试

图 4 显示了各种方法在装载数据过程中的节点使用情况,Dynamic 方法限定节点个数是 2 的幂次方,每次扩展仅仅是简单的将过载节点由 1 个扩展为 4 个,势必造成资源浪费.本文的方法 MNRT 和 QMMP 能够根据装载量按需分配负载节点,因此,相对于 Dynamic 能够使用更少的节点资源.然而,从图上不难看出,二分图方法以优化内存为出发点也使用了较少的节点个数.具体来说:Bi_6 初始化时二分图各边(R/S)有 6 个子组,每个子组有一个

负载节点,随着装载量的变大,其子组内的节点个数按照装载的量扩容;Bi 初始化时二分图的每个边仅有一个节点,然而,由于其仅仅考虑以内存为优化目标,当 CUP 的计算缺乏时,其吞吐量会受到影响.这也体现在图 3 展示的执行时间上.

图 5 展示了各种方法在不同数据倾斜度情况下扩容的调整效率.如图 4 所示:Dynamic 使用了更多的并行节点个数,当系统扩容时,其需要迁移的数据量也是最多的,进而增大了调整的时间.QMMP 能够从元组粒度级别进行调整,该方法能够更快地计算出迁移计划,进而其调整的速度稍微领先于 MNRT.二分图方法中,Bi 在单条流对元组是随机存放的,所以它也更便利于系统中并行节点的均衡扩容.

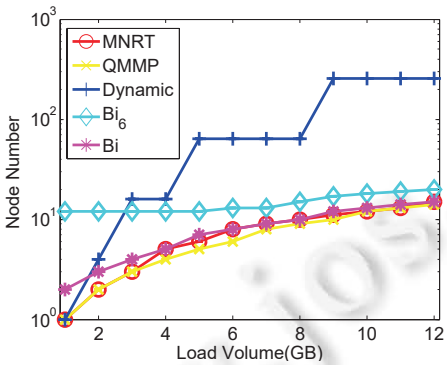


Fig.4 Parallel processing node number

图 4 并行处理节点使用个数

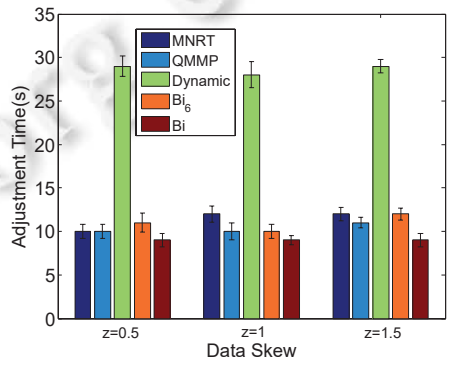


Fig.5 System balance adjustment time

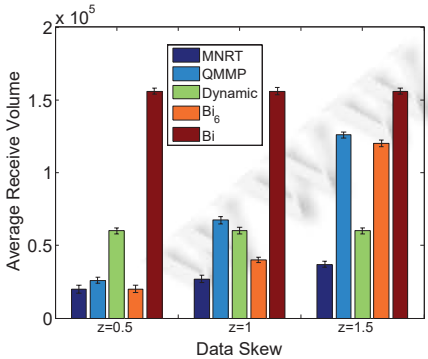
图 5 系统均衡的调整时间

由图 3~图 5 可以看出:本文提出的方法能够在使用相对较少节点的情况下,快速地处理完 12GB 的数据量.

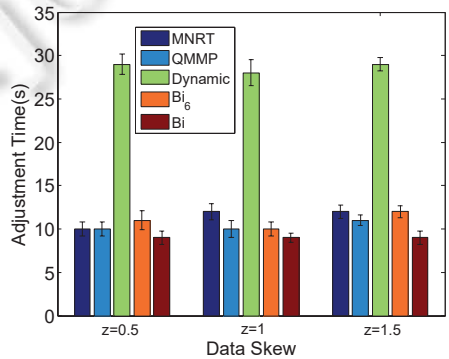
5.2.2 动态性-基于窗口模型的连接操作

为了进一步验证数据的倾斜及查询操作对性能的影响,本节使用窗口模型展示了不同的查询任务在不同倾斜数据分布情况下,每个负载节点平均接收到的元组个数.该组实验使用了 64GB 的 TCP-H 数据量,将窗口大小设置为 180s,元组发送速率约为每秒 3×10^5 条,该速度确保了每个负载节点的 CPU 可以满负荷.

图 6(a)和图 6(b)分别展示了 EQ_5 和 B_{NCI} 的查询任务在不同数据倾斜分布情况下及不同计算架构中每个节点平均接收到的元组数量.图 6 中,Dynamic 方法不受数据倾斜的影响,原因是该方法在矩阵的两个维度方向都是采用了随机分发方式.所以只要窗口内的数据量固定,无论数据是何种分布以及操作是何种连接,各个并行节点接收到的数据量是相同的.因此,该方法能够将计算负载均衡地分配到每各个负载节点上,但是该方法使用了较多的节点资源.



(a) EQ_5 查询语句



(b) B_{NCI} 查询语句

Fig.6 Subsequent network load

图 6 后续网络负载

本组实验结果从各个方法中节点的数据流入量测试各个节点的计算负载情况.MNRT 方法尽量将属于同一 key 的元组存储至同一个节点,其对不同粒度的 key 区别对待,这使得流数据中元组的路由方向性更好.即便在范围连接 B_{NCI} 操作中,MNRT 依然能够很好地控制元组数据的流向.然而,QMMP 方法频繁地拆分 key,虽然能够使系统快速制定出迁移计划,但是拆分 key 增加了连接流的广播量.在倾斜较小的情况下,对于等值连接 Eq5, Bi_6 方法根据路由能够更好地将连接请求分发至选中的子组,从而使节点接收到较少的处理连接请求.随着数据倾斜度加大,子组内的广播数据量加大.例如当 z=1 时,最大的 key 占据了一半的数据量,这一现象在范围连接操作 B_{NCI} 中尤为明显.Bi 由于是不分组的全广播,各个负载节点接收的连接请求在等值连接或范围连接中是同样的.

5.2.3 真实数据性能

本节使用 10GB 微博数据进行 TOPK 操作,以验证方法的有效性,该操作基于窗口模型进行.本组实验用于验证适用于聚合操作的均衡调整方法性能,由于 Dynamic 和 Bi 方法更倾向于做数据流的连接操作,故此组实验不涉及这两种方法.本组实验中,系统要求将不均衡容忍度调整至 0.05 以内.

图 7 展示了各种方法在不同均衡容忍度的要求下,制定迁移计划的效率.由于 PKG 方法中没有迁移动作,因此也没有迁移计划的制定动作.在图 7 中,QMMP 以元组为粒度拆分 key,无论要求的均衡容忍度为多大,它均能够较快地制定出迁移计划.MNRT 兼顾了整体系统代价,所以其迁移计划的制定比 QMMP 稍慢.Readj 方法由于要配对各个节点及节点内的 key 来尝试着交换不同节点中的 key 使系统均衡,这种方法计算复杂度较高,因此制定迁移计划时间更长.在图 8 中,各个方法都使用可靠性保证消息机制,同时,对于操作中的计数使用了并行的方式,这些并行计数的中间结果被定时(Storm 上的 tick 机制)汇总到一个聚合节点上.中间结果的定时汇总操作在拆分 key 更频繁的方法上体现出了更大的处理延时,从而降低了系统的整体吞吐量而使系统的处理速度变慢.Readj 由于对严格的均衡容忍度敏感,增长了迁移计划的制定时间,最终影响了整体系统的性能表现.

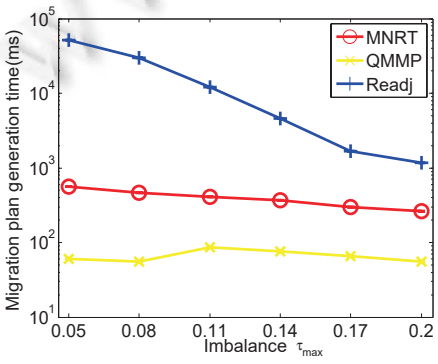


Fig.7 Migration plan generation time

图 7 迁移计划制定时间

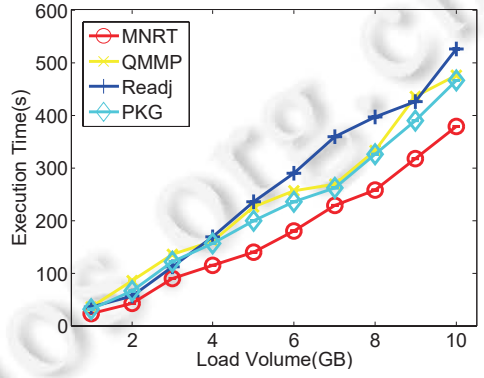


Fig.8 System performance

图 8 系统整体性能

6 总 结

在分布式流处理系统中,并行处理节点间不均衡负载会导致个别高负载节点处理效率降低而其他节点出现大量的空闲运算资源,从而降低了系统的整体利用率.本文通过对并行流处理系统在均衡调整过程中的代价分析,提出了一种对 key 按需拆分,尽量合并的均衡调整方法.该方法既方便于系统的均衡调整,又能减少系统在后续操作中对数据的广播动作,进而有效地减少了系统的额外负载,最终提升了系统的处理性能.然而,对于文中提到的多目标代价优化的实现,本文只给出了简单的利用分步贪心方法来获取各个代价标准的相对较优解,因此在未来的工作中,我们会进一步研究更优的方法来解决对目标代价的优化问题,将各个性能指标归一化,使系统达到更便捷、更优的均衡特性.

References:

[1] Gong XQ, Jin CQ, Wang XL, Zhang R, Zhou AY. Data-Intensive science and engineering: Requirements and challenges. Chinese Journal of Computers, 2012,35(8):1−16 (in Chinese with English abstract).

[2] Li JZ, Liu XM. An important aspect of big data: Data usability. Journal of Computer Research and Development, 2013,50(6): 1147−1162 (in Chinese with English abstract).

[3] Qian Z, He Y, Su C, Wu ZJ, Zhu HY. TimeStream: Reliable stream computation in the cloud. In: Proc. of the ACM European Conf. on Computer Systems. 2013. 1−14. [doi: 10.1145/2465351.2465353]

[4] Zhou Y, Ooi BC, Tan KL. Dynamic load management for distributed continuous query systems. In: Proc. of the IEEE Computer Society. 2014. 322−323. [doi: 10.1109/ICDE.2005.54]

[5] Zhu Y, Rundensteiner EA, Heineman GT. Dynamic plan migration for continuous queries over data streams. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Paris, 2015. 431−442. [doi: 10.1145/1007568.1007617]

[6] Zhou YL, Ooi BC, Tan KL, Wu J. Efficient dynamic operator placement in a locally distributed continuous query system. LNCS 4275, 2006. 54−71. [doi: 10.1007/11914853_5]

[7] 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. 2013. 725−736. [doi: 10.1145/2463676.2465282]

[8] Jin CQ, Qian WN, Zhou AY. Analysis and management of streaming data: A survey. Ruan Jian Xue Bao/Journal of Software, 2004, 15(08):1172−1181 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1172.htm

[9] Xu Y, Kostamaa P, Zhou X, Chen L. Handling data skew in parallel joins in shared-nothing systems. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. 2008. 1043−1052. [doi: 10.1145/1376616.1376720]

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

[11] Kwon YC, Balazinska M, Howe B, Rolia J. SkewTune: Mitigating skew in mapreduce applications. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. 2012. 25−36. [doi: 10.1145/2213836.2213840]

[12] Gufler B, Augsten N, Reiser A, kemper A. Load balancing in MapReduce based on scalable cardinality estimates. In: Proc. of the IEEE Int'l Conf. on Data Engineering. 2012. 522−533. [doi: 10.1109/ICDE.2012.58]

[13] Xing Y, Zdonik S, Hwang JH. Dynamic load distribution in the Borealis stream processor. In: Proc. of the Int'l Conf. on Data Engineering. IEEE, 2005. 791−802. [doi: 10.1109/ICDE.2005.53]

[14] Xing Y, Hwang JH, Cetintemel U, Zdonik S. Providing resiliency to load variations in distributed stream processing. In: Proc. of the Int'l Conf. on Very Large Data Bases. ACM Press, 2006. 775−786.

[15] Abadi DJ, Ahmad Y, Balazinska M, *et al*. The design of the Borealis stream processing engine. In: Proc. of the 2005 CIDR Conf., 2005. 277−289.

[16] Fang J, Zhang R, Fu TZJ, Zhang ZJ, Zhou AY, Zhu JH. Parallel stream processing against workload skewness and variance. arXiv preprint arXiv:1610. 05121, 2016.

[17] Shah MA, Hellerstein JM, Chandrasekaran S, Franklin MJ. Flux: An adaptive partitioning operator for continuous query systems. In: Proc. of the Int'l Conf. on Data Engineering. 2003. 25−36. [doi: 10.1109/ICDE.2003.1260779]

[18] Gedik B. Partitioning functions for stateful data parallelism in stream processing. Int'l Journal on Very Large Data Bases, 2014, 23(4):517−539. [doi: 10.1007/s00778-013-0335-9]

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

[20] Nasir MAU, Morales GDF, Garcia-Soriano D, Kourtellis N, Serafini M. The power of both choices: Practical load balancing for distributed stream processing engines. In: Proc. of the IEEE Int'l Conf. on Data Engineering. IEEE, 2015. 137−148. [doi: 10.1109/ICDE.2015.7113279]

[21] Apache storm. http://storm.apache.org/

[22] 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]

[23] Nasir MAU, Morales GDF, Kourtellis N, Serafini M. When two choices are not enough: Balancing at scale in distributed stream processing. In: Proc. of 2016 IEEE 32nd Int'l Conf. on Data Engineering, 2016. 589−600.

[24] 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 on Conf. on Information and Knowledge Management. ACM Press, 2016. 1773−1782. [doi: 10.1145/2983323.2983773]

[25] Okcan A, Riedewald M. Processing theta-joins using MapReduce. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. 2011. 949−960. [doi: 10.1145/1989323.1989423]

[26] Fang JH, Wang XT, Zhang R, Zhou AY. Flexible and adaptive stream join algorithm. In: Proc. of the Asia-Pacific Web Conf. Springer Int'l Publishing, 2016. 3−16. [doi: 10.1007/978-3-319-45817-5_1]

[27] Bruno N, Kwon YC, Wu MC. Advanced join strategies for large-scale distributed computation. Proc. of the VLDBEndowment, 2014,7(13):1484−1495. [doi: 10.14778/2733004.2733020]

[28] The TPC-H benchmark. http://www.tpc.org/tpch

附中文参考文献:

[1] 宫学庆,金澈清,王晓玲,张蓉,周傲英.数据密集型科学与工程:需求和挑战.计算机学报,2012,35(8):1−16.

[2] 李建中,刘显敏.大数据的一个重要方面:数据可用性.计算机研究与发展,2013,50(6):1147−1162.

[8] 金澈清,钱卫宁,周傲英.流数据分析与管理综述.软件学报,2004,15(8):1172−1181. http://www.jos.org.cn/1000-9825/15/1172.htm

房俊华(1985－),男,河南沈丘人,博士生,主要研究领域为分布式流处理.

王晓桐(1994－),女,硕士,主要研究领域为分布式流处理.

张蓉(1978－),女,博士,副教授,主要研究领域为分布式计算.

周傲英(1965－),男,博士,教授,博士生导师,CCF杰出会员,主要研究领域为Web数据管理,数据密集型计算,内存集群计算,分布事务处理,大数据基准测试和性能优化.