算法是针对单机环境设计的多数据流频繁伴随模式发现算法,该算法为了节省内存(压缩 segment)和减少索引维护代价,设计和采用 Seg-tree 树型索引结构.本文设计的 FCP-DM 算法是针对分布式计算环境,拥有较充裕的内存,因此采用了分布式哈希索引.虽然 CooMine 算法对基于 Seg-tree 的数据查找操作进行了大量优化,但是 FCP-DM 算法中基于哈希索引的查询操作(如查找元素、伴随模式等)要比 CooMine 算法中基于 Seg-tree 的查询操作具备更高的查询效率,而哈希索引的内存使用效率要低于 Seg-tree 索引结构.因此,在 8 台机器上运行 FCP-DM 算法的计算效率接近于在 8 个计算节点上单独运行的 CooMine 算法的计算效率之和.

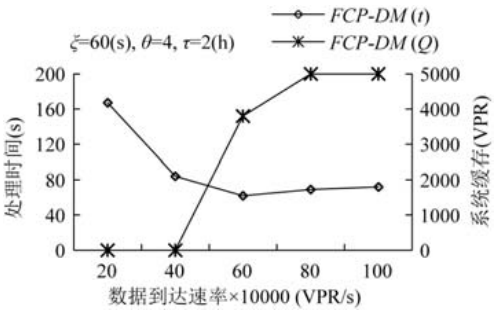

Fig.7 Processing capability of FCP-DM
图 7 FCP-DM 方法处理能力

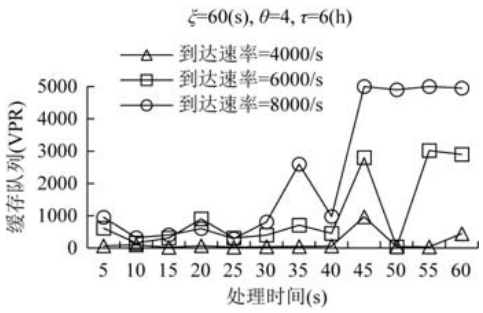

Fig.8 Processing capability of CooMine
图 8 CooMine 算法处理能力

为更加直观地比较 FCP-DM 算法和 CooMine 算法的性能,分别令这两种算法处理相同规模的过车记录数据集,并调整过车记录数据集的规模来比较两者的处理时间.该组实验是将整个数据集一次性地加载到系统中,算法处理时间是指从处理第 1 条数据开始至最后一条数据处理完成的时间跨度.图 9 所示的实验结果表明,FCP-DM 算法(采用 8 个物理节点)的处理时间要明显小于 CooMine 算法的处理时间,并且随着数据规模的扩大,FCP-DM 算法处理时间的增长速率明显小于 CooMine 算法的时间增长速率.

图 10 测试数据到达速率为 40 000 条/s 时 FCP-DM 算法处理不同规模数据时的可扩展性.该组实验中,分别采用 2、4、6、8 个计算节点处理不同规模的数据集.实验结果表明,随着计算节点个数的增加,FCP-DM 的处理时间明显下降.VPR 数量越多,处理时间的下降趋势越明显,表明 FCP-DM 方法在处理大规模数据流时具备良好的可扩展性.

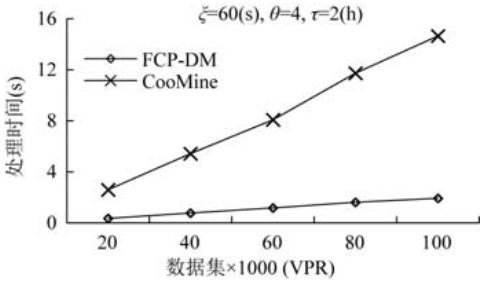

Fig.9 Comparison of FCP-DM and CooMine w.r.t. processing time
图 9 FCP-DM 与 CooMine 处理时间比较

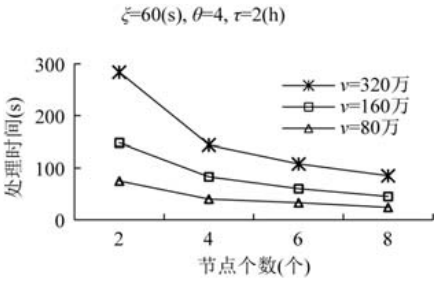

Fig.10 The scalability of FCP-DM
图 10 FCP-DM 的可扩展性

5.2.2 FCP-DM 算法相关参数对算法性能影响的评价

图 11 和图 12 主要对本文所采取的负载迁移策略的性能进行验证,主要测试单个 PartitionPE 所能承载的 segment 的最大数量β、数据到达速率以及数据集规模对于 PartitionPE 数量的影响.该组实验共采用 4 个物理计算节点.实验开始之前,我们对测试数据集中的所有元素的标识进行遍历,得到一个能够包含测试数据集中所有元素的标识范围$[w_s,w_e]$.实验开始时,初始化一个 PartitionPE,令其 key 值范围为$[w_s,w_e]$,也就是说,所有初始的 PartitionPE 能够接收所有的 segment.当初始 PartitionPE 接收但来不及处理的 segment 数量达到阈值β时,该

PartitionPE 就会分裂,这样就会产生多个 PartitionPE.这里需要注意的是,PartitionPE 一旦创建就不会被删除,本组实验将记录处理过程中所有的 PartitionPE 的数量.

图 11 的实验结果表明,当β值一定时,随着数据到达速率的增加,PartitionPE 的个数明显增多,这是因为数据到达越快,PartitionPE 所积压的未处理的 segment 越多,这样发生负载迁移的次数越多,导致更多的 PartitionPE 产生.当数据到达速率一定时(例如 30 000VPR/s 时),随着β值的增大,PartitionPE 的数量逐渐减少.这是因为随着β值的增大,PartitionPE 能够缓存的 segment 增多,使得 PartitionPE 的数量减少.图 12 中,令 FCP-DM 处理不同规模的数据集并观察 PartitionPE 的数量.当数据到达速率和β值固定时,我们发现数据集的规模对 PartitionPE 的数量没有显著影响,这表明 PartitionPE 的数量只与数据到达速率和β值有关,与待处理的数据集规模无关.

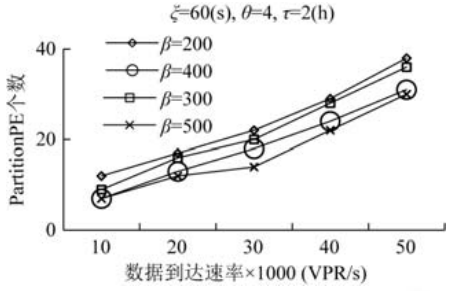

Fig.11 Number of PartitionPE w.r.t. β

图 11 参数β对 PartitionPE 个数的影响

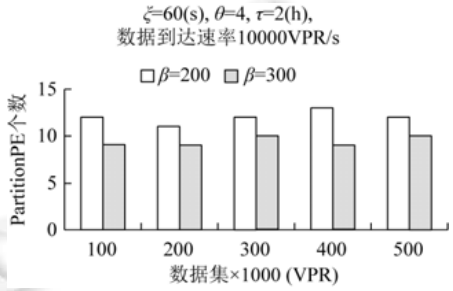

Fig.12 Number of PartitionPE w.r.t. scale of data

图 12 数据集规模对 PartitionPE 个数的影响

在延迟删除策略中,调整参数 Δt 的值可以改变对过期数据的删除周期.图 13 测试了参数 Δt 对于算法运行时内存使用状况的影响.实验策略是随机抽取 100 万条 VPR 作为测试数据集,令数据发送速率为 10 000VPR/s,然后观察不同时刻的所有计算节点的内存使用状态.图 13 的纵坐标表示所有计算节点运行 FCP-DM 算法时消耗的内存之和,横坐标表示算法的运行时间.实验结果表明,算法运行初期 Δt 取不同值时,内存使用量均增大,这是因为此时内存中没有过期数据,致使内存累积的数据增加.当算法运行 50s 之后,内存使用量出现明显波动,这是因为算法每隔 Δt 时间都会对过期数据进行删除.总的来说,Δt 取值越小,平均的内存使用量也越少,但是删除次数也相对增加.图 14 测试了参数ξ对 FCP-DM 处理时间的影响.图 14 所示实验结果表明,算法处理时间随着ξ值的增大而增加.这是因为,ξ越大,segment 长度越大,此时所产生的 CP 数量也越大.由于 FCP-DM 需要对所有的 CP 进行处理,因此,FCP-DM 算法的处理时间随着 CP 数量的增大而明显增加.

FCP-DM 算法的目标是准确发现给定数据集中所有的 FCP,即查全率和查准率应为 100%.由于 CooMine 算法的目标也是准确发现给定数据流中所有的 FCP[1],其查全率和查准率均为 100%.表 1 比较了 FCP-DM 算法和 CooMine 算法在同一数据集上所发现的 FCP 数量.当参数θ和 k 取值一定时,两种算法在该实验数据集上发现的 FCP 数量相同,从而验证了 FCP-DM 算法的查全率和查准率均为 100%.

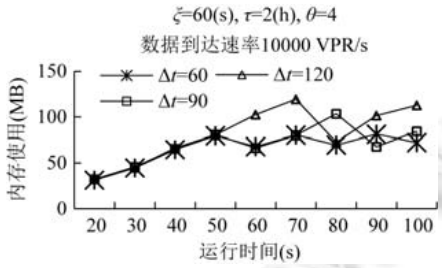

Fig.13 Memory consumption w.r.t. Δt

图 13 参数 Δt 对于内存使用的影响

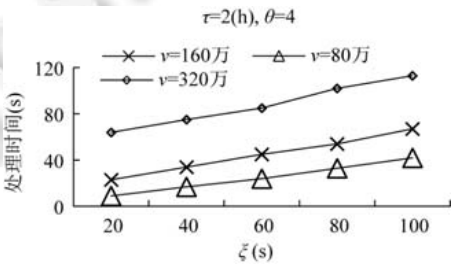

Fig.14 Processing time w.r.t. ξ

图 14 ξ对处理时间的影响

Table 1 Comparison of FCP-DM and CooMine w.r.t. the number of discovered FCP

表 1 FCP-DM 和 CooMine 挖掘结果数量比较

	FCP-DM				CooMine		
	k=2	k=3	k=4		k=2	k=3	k=4
θ=3	953	54	7	θ=3	953	54	7
θ=4	182	7	0	θ=4	182	7	0
θ=5	32	2	0	θ=5	32	2	0

5.2.3 FCP-DM 挖掘结果评价

图 15 和图 16 主要测试数据规模和参数θ对 FCP-DM 算法挖掘结果的影响,其中,k 表示 FCP 的长度,ξ、τ和θ的含义见定义 2 和定义 3.该组实验中,一个 FCP 代表一组频繁伴随车辆.图 15 所示实验结果表明,当 FCP 定义中相关参数(ξ、τ、θ)的值固定时,随着数据规模的扩大,FCP-DM 发现的 FCP 数量逐渐增多;当数据规模一定时,FCP 的长度(k 值)越大,FCP 的数量越少,也就是说,用户指定一组频繁伴随车辆中的车辆数目越大,现实中这样的车辆组合越少,越符合我们的直观认识.FCP 定义中参数θ将对 FCP 的数量产生较大影响,图 16 所示实验结果表明,随着θ值的增大,FCP 数量变小,也就是说,如果用户要求一组 FCP 伴随出现的数据流越多,那么这样的 FCP 数量越少,实验结果同样符合预期.目前,FCP-DM 算法已被应用至山东省某地市智能交通综合管控平台,用于快速发现频繁伴随车辆.图 17 和图 18 给出了 FCP-DM 算法在该交通管控平台发现频繁伴随车辆的部分结果展示.其中,地图中的红色箭头是车辆伴随行驶方向,图片右侧是发现的频繁伴随车辆组合.

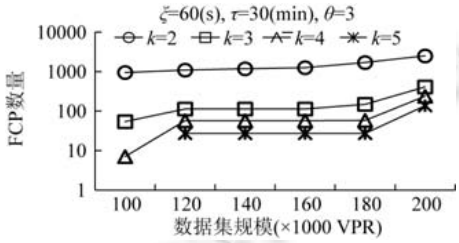

Fig.15 Number of FCP w.r.t. scale of data

图 15 数据规模对 FCP 数量的影响

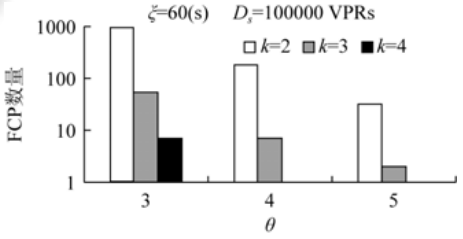

Fig.16 Number of FCP w.r.t. θ

图 16 参数θ对 FCP 数量的影响

Fig.17 Demonstration of applying FCP-DM (1)

图 17 FCP-DM 实际应用展示(1)

Fig.18 Demonstration of applying FCP-DM (2)

图 18 FCP-DM 实际应用展示(2)

6 总 结

本文研究了海量多数据流 FCP 挖掘问题,提出了 FCP-DM——一个可部署在分布式流数据处理平台的分布式挖掘方法.在该方法中,本文重点研究分布式环境下多计算节点协同挖掘大规模数据流所形成的 FCP 时面临的负载均衡问题、数据分布式存储背景下的 FCP 生成问题以及连续数据流的 FCP 增量挖掘问题,最后通过大量实验对 FCP-DM 方法的各项性能和挖掘结果进行充分验证,并给出了算法在实际应用中的效果展示.后续工作将继续研究如何对发现的大量 FCP 进行排序优化,从而将更符合用户偏好的 FCP 排在挖掘结果的前面.

References:

[1] Yu ZQ, Yu XH, Liu Y, Li WZ, Pei J. Mining frequent co-occurrence patterns across multiple data streams. In: Proc. of the 19th Int'l Conf. on Extending Database Technology. Brussels, 2015. 73–84.

[2] Chang JH, Lee WS. Finding recent frequent itemsets adaptively over online data streams. In: Proc. of the 9th SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Washington, 2003. 487–492.

[3] Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Proc. of the 20th Int'l Conf. on Very Large Data Bases. Santiago de Chile, 1994,1215:487–499.

[4] Manku GS, Motwani R. Approximate frequency counts over data streams. In: Proc. of the 28th Int'l Conf. on Very Large Data Bases. Hong Kong, 2002. 346–357.

[5] Yu JX, Chong Z, Lu H, Zhou A. False positive or false negative: Mining frequent itemsets from high speed transactional data streams. In: Proc. of the 30th Int'l Conf. on Very Large Data Bases. Toronto, 2004. 204–215.

[6] Leung CS, Khan QI. Dstree: A tree structure for the mining of frequent sets from data streams. In: Proc. of the 2006 IEEE Int'l Conf. on Data Mining. Hong Kong, 2006. 928–932.

[7] Li J, Maier D, Tufte K, *et al*. No pane, no gain: Efficient evaluation of sliding-window aggregates over data streams. In: Proc. of the 11th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Baltimore, 2005. 39–44.

[8] Mozafari B, Thakkar H, Zaniolo C. Verifying and mining frequent patterns from large windows over data streams. In: Proc. of the 29th Int'l Conf. on Data Engineering. Cancun, 2008. 179–188.

[9] Karp RM, Papadimitriou CH, Shenker S. A simple algorithm for finding frequent elements in streams and bags. ACM Trans. on Database Systems, 2003,28(1):51–55.

[10] Chi Y, Wang H, Yu PS, Muntz R. Moment: Maintaining closed frequent itemsets over a stream sliding window. In: Proc. of the Int'l Conf. on Data Mining. Brighton, 2004. 59–66.

[11] Silva A, Antunes C. Multi-relational pattern mining over data streams. Data Mining and Knowledge Discovery, 2015,29(6): 1783–1814.

[12] Li HF, Zhang N, Zhu JM, Cao HH. Frequent itemset mining over time-sensitive streams. Chinese Journal of Computers, 2012,35(11):2283–2293 (in Chinese with English abstract).

[13] Guo J, Zhang P, Tan J. Mining frequent patterns across multiple data streams. In: Proc. of the 20th ACM Conf. on Information and Knowledge Management. Glasgow, 2011. 2325–2328.

[14] Mao YX, Chen TB, Shi BL. Efficient method for mining multiple-level and generalized association rules. Ruan Jian Xue Bao/ Journal of Software, 2011,22(12):2965−2980 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3907.htm [doi: 10.3724/SP.J.1001.2011.03907]

[15] Ao X, Luo P, Li C, *et al*. Online frequent episode mining. In: Proc. of the 31st Int'l Conf. on Data Engineering. Seoul, 2015. 891–902.

[16] Patnaik D, Laxman S, Chandramouli B, Ramakrishnan N. Efficient episode mining of dynamic event streams. In: Proc. of the 12th IEEE Int'l Conf. on Data Mining. Brussels, 2012. 605–614.

[17] Vanchinathan HP, Marfurt A, Robelin CA. Discovering valuable items from massive data. In: Proc. of the 21st ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Melbourne, 2015. 1195–1204.

[18] Saber S, Reza A, Florent M. Fast parallel mining of maximally informative *k*-itemsets in big data. In: Proc. of the 15th IEEE Int'l Conf. on Data Mining. 2015. 350–368.

[19] Wang L, Yang B, Chen YH, Zhang XQ, Orchard JF. Improving neural-network classifiers using nearest neighbor partitioning. IEEE Trans. on Neural Networks and Learning Systems, 2017,28(10):2255–2267.

[20] Han SY, Chen YH, Tang GY. Fault diagnosis and fault-tolerant tracking control for discrete-time systems with faults and delays in actuator and measurement. Journal of the Franklin Institute, 2017,354(12):4719–4738.

[21] Neumeyer L, Robbins B, Nair A, Kesari A. S4: Distributed stream computing platform. In: Proc. of the 2010 IEEE Int'l Conf. on Data Mining Workshops. Washington, 2010. 170–177.

附中文参考文献:

[12] 李海峰,章宁,朱建明,曹怀虎.时间敏感数据流上的频繁项集挖掘算法.计算机学报,2012,35(11):2283–2293.

[14] 毛宇星,陈彤兵,施伯乐.一种高效的多层和概化关联规则挖掘方法.软件学报,2011,22(12):2965−2980. http://www.jos.org.cn/1000-9825/3907.htm [doi: 10.3724/SP.J.1001.2011.03907]

于自强(1984－),男,山东青岛人,博士,讲师,CCF 专业会员,主要研究领域为时空数据分布式查询,流数据分布式计算.

禹晓辉(1977－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为海量数据管理与分析.

董吉文(1984－),男,博士,教授,CCF 高级会员,主要研究领域为数字图像处理.

王琳(1984－),男,博士,副教授,CCF 专业会员,主要研究领域为机器学习,数据反向建模.