Research on Parallelized Stream Data Micro Clustering Algorithm

Ke Ma[1], Lingjuan Li[1], Yimu Ji[1], Shengmei Luo[1], Tao Wen[2]

[1]College of Computer,Nanjing University of Posts and Telecommunications, Nanjing, China, 210003

[2]ZTE Corporation, Nanjing, China, 210012

lilj@njupt.edu.cn，wen.tao1@zte.com.cn

Keywords: clustering; stream data; distriubuted algorithm; MapReduce; Micro-clustering

Abstract. Analysis and mining of stream data is a hot research topic in recent years. In order to improve the clustering efficiency, based on MapReduce, this paper proposes a Parallelized Stream Data Micro Clustering Algorithm PSDMC for the micro-clustering phase of CluStream algorithm. PSDMC algorithm uses a series of containers to store real-time stream data according to the arrival time. Each map node produces real-time local micro-clusters per unit time (such as 1 second). The reduce node puts together these real-time local micro-clusters to produce real-time global micro-clusters by using DBSCAN and the micro clustering method of CluStream. The global micro-clusters will be used to renew local micro-clusters in every map node and be used to create snapshots to store into Pyramidal Time Frame. Analysis shows that the efficiency of PSDMC algorithm can increase nearly linearly with the increase of map nodes while the clustering accuracy can be guaranteed.

Introduction

Data stream is a sequence composed by the arriving data according to the arrival time, which is boundless and changes constantly [1]. With the development of data acquisition technology, lots of data is produced in the form of data stream in recent years, such as the traffic flow data stream. The analysis and mining of this data is becoming a hot topic. Stream data is dynamic data which is acquired and mined at the same time. Compared with classical data, stream data has the features of potentially unbounded, high speed and real-time. These features restrict us that the data mining algorithms for stream data should be single-scan algorithms.

The mining of stream data has lots of tasks, such as classification, clustering and association analysis. This paper concentrates on the researches of clustering algorithm for real-time stream data. The classical stream data clustering algorithm CluStream achieves a two-level clustering frame, which divides stream data clustering into two parts: online micro-clustering and offline macro-clustering [2]. The algorithm proposed in this paper is a parallelized clustering algorithm aims to the online macro-clustering part. When the speed of data stream is too fast while one single computer cannot process, single-node mining algorithm could not meet the mining demand. In this situation, we need parallelized algorithm to deal with it. MapReduce is a programming model for parallelized computing of big data[3]. This paper uses MapReduce model to produce a parallelized clustering algorithm PSDMC (Parallelized Stream Data Micro Clustering), which adds a container to store stream data temporarily. In PSDMC, map nodes produce local micro-clusters according to unit time and the reduce node produces global micro-clusters. This algorithm can have a nearly linear increase with the increase of map nodes while the accuracy can be guaranteed.

Classical Stream Data Clustering Algorithm CluStream

C.C.Aggarwal, Jiawei Han and other researchers produced the famous stream data clustering algorithm CluStream [4], which has two important concepts: micro-cluster and pyramidal time frame. They also divided the process of stream data clustering into two parts: online micro-clustering and

offline macro-clustering. The online part processes the new arriving data in time while the offline part uses these statistical results and user's inputs to find the final clustering result.

The CluStream algorithm uses micro-clusters to record the statistical information which represents the locations of data. These micro-clusters are defined as an expanding (to record the attributes of time) of cluster feature vectors. These additional time attributes can solve the stream data problems naturally. Assuming that n d-dimension data points are get together to a cluster, which is defined as a micro-cluster, whose feature vectors CFT is a tuple which has (2d+3)-dimensions:

$$(\overrightarrow{CF2^x}, \overrightarrow{CF1^x}, CF2^t, CF1^t, n) \quad (1)$$

In the tuple, the first part represents the quadratic sum of each data point's feature vector in the cluster; the second part represents the algebraic sum of each data point's feature vector in the cluster; the third part represents the quadratic sum of each data point's arrival time in the cluster; the forth part represents the algebraic sum of each data point's arrival time in the cluster and the last part represents the number of data points in the cluster.

The CluStream algorithm uses k-means algorithm to initialize q micro-clusters; cased on these initial micro-clusters, it uses online part to process each subsequent arriving point, which can be absorbed to one old micro-cluster or create a new micro-cluster to maintain the total of micro-clusters. The specific practices are: After initializing micro-clusters, for each data point Xik, we calculate the distances of Xik and q cluster centers and put Xik into the closest micro-cluster Mp. If Mp is the closest micro-cluster to Xik, but the distance is longer than the radius of Mp, we should create a new micro-cluster (with a new ID) for Xik. At the same time, we should delete one oldest micro-cluster or merge two closest micro-clusters so that we can maintain the total of micro-clusters.

In practical applications, we often pay the closest attention to the clustering results in the nearest time and pay farther attention to earlier results. In order to do this, the CluStream algorithm uses pyramidal time frame to divide time axis into time of different granularity while the nearer time has the finer granularity. The time granularity and the maximum number of each time granularity are defined by the parameters α and l (which are set by users). The maximum number of each time granularity is αl+1. The time granularity of i-th row in pyramidal time frame is αi. Table 1 is an example of snapshots stored in the form of pyramidal time frame (α=2, l=2 and present time is 29).

TABLE I. AN EXAMPLE OF PYRAMIDAL TIME FRAME

Order of Snapshots	Clock Times(Last 5 Snapshots)
0	29 ~~28~~ 27 ~~26~~ 25
1	~~28~~ 26 ~~24~~ 22 ~~20~~
2	28 ~~24~~ 20 ~~16~~ 12
3	24 ~~16~~ 8
4	16

In Table 1, each number represents the snapshot at the clock time of the number, and snapshots marked by a crossbar are stored repeatedly. This storing method can meet the demand of selecting clustering results during different time.

Parallelized Stream Data Micro Clustering Algorithm PSDMC

MapReduce is an emerging parallel programming model invented by Google. It puts parallelization, fault tolerance, data distribution, load balancing into a library, and the system sums up all operations on the data to map stage and reduce stage. To submit handling procedures of all operations to the MapReduce programmer only need to define Map function and reduce function.

Based on the size of the input data and the job configuration information, MapReduce system can automatically initialize the job to multiple same map tasks and reduce tasks, and they respectively read different input data blocks and order the map function and reduce function to process [5].

The target of parallelized stream data micro Clustering algorithm PSDMC is: use the mechanism of MapReduce to parallelize the micro-clustering phase of CluStream. The basic idea of PSDMC is as follows:

- The word "data" is plural, not singular. It uses a series of containers to store real-time stream data according to the arrival time.
- It assigns the online micro-clustering tasks in CluStream to each map nodes and every map node produces real-time local micro-clusters according to the arriving data.
- The reduce node merges these real-time local micro-clusters into real-time global micro-clusters and uses these global micro-clusters to produce real-time snapshot, then stores the snapshot into pyramidal time frame.
- The real-time global micro-clusters will be also sent to each map nodes to renew their local micro-clusters. After renewing the local micro-clusters, each map node will start the task of the next unit time.

Fig. 1 is the internal module sketch map of PSDMC algorithm.

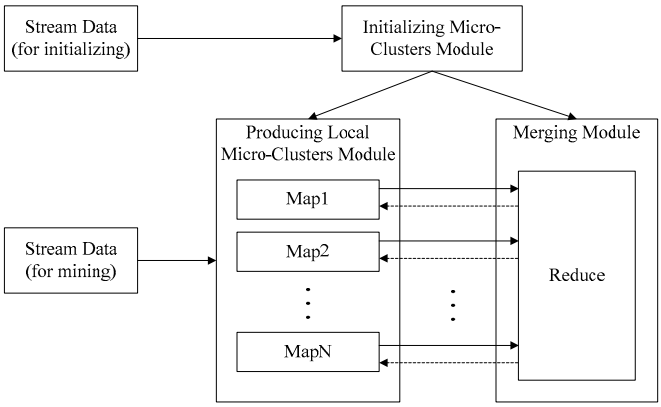

Figure 1. Internal modules of PSDMC algorithm

The minimum time granularity of pyramidal time frame in PSDMC is set by user's demand, and the size of unit time in the following parts equals the minimum time granularity.

A. Design of Local Micro Clustering in Map Nodes

The arriving stream data (except which arrives in the first unit time) is equally distributed to each map nodes. Each map node has a container to receive the arriving data.

The main tasks of each map node are: Map node receives the initial micro-clusters produced by initializing micro-clusters module and uses them to produce initial local micro-clusters. Map node stores the arriving stream data according to its arrival time. Map node receives the newest real-time global micro-clusters produced by reduce node and renews its local micro-clusters; After renewing the local micro-clusters, each map node takes out the stream data of the earliest first unit time in the container, and deal with this data according to the method of the online part in CluStream. After processing the stream data of one unit time, the map node produces its newest local micro-clusters and sends them to the reduce node. Fig. 2 is the flow chart of producing newest micro-clusters in one map node:

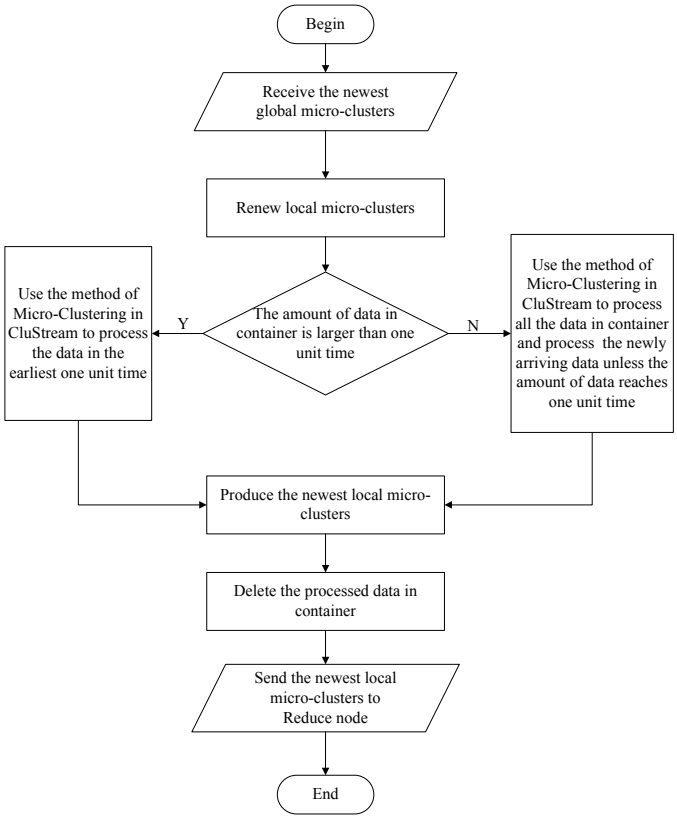

Figure 2. The flow of producing newest micro-clusters in one map node

B. Design of Generating Global Clusters in Reduce Node

The main tasks of reduce node are: storing global micro-clusters; receiving local micro-clusters from each map nodes to produce newest global micro-clusters and sending them to the map nodes; creating real-time snapshots and storing them according to the rules of pyramidal time frame.

The initializing module computes the stream date of the first unit time to produce initial micro-clusters and rsends these micro-clusters to the reduce node, which are also the global micro-clusters of the first unit time. The reduce node stores the micro-clusters and uses them to create snapshot and then stores it according to the rules of pyramidal time frame.

The reduce node receives local micro-clusters from each map nodes and combines them with the old global micro-clusters stored in reduce node to produce real-time global micro-clusters, which uses the offline phase of CluStream and modified DBSCAN [6] algorithm. The pseudo-code is as follow:

- The word “data” is plural, not singular.
- Input: set of old global micro-clusters stored in reduce node C_pre (the size of C_pre is k); set of the newly receiving local micro-clusters from each map nodes C (the size of C is n*k)
- Output: set of the newest global micro-clusters C_new;

Output: set of the newest global micro-clusters C_new;

Begin:

- For each micro-cluster in C: if the micro-cluster is changed from one micro-cluster in C_pre, then put the micro-cluster into set C1;Else put the micro-cluster into set C2
- For each micro-cluster in C1: subtract the cluster feature vector (CFT) of the corresponding micro-cluster in C_pre which it changed from; If the number of points in the micro-cluster is 0 (after subtacting), then delete the micro-cluster from C1;
- Create a k-dimension array count[], whose initial values are 0, the value of count represents The times of revision (every item in the array corresponds to one micro-cluster in C_pre);
- For each micro-cluster in C1: add the micro-cluster to the corresponding micro-cluster in C_pre and the corresponding value in count[] +1;

- Take the micro-clusters in C2 as points whose coordinate equals the core of the corresponding micro-cluster and scanning radius (Eps in DBSCAN algorithm) equals the radius of the corresponding micro-cluster;
- Use DBSCAN algorithm on these data points and produce micro-clusters according to the clustering results, put these micro-clusters into set C3;
- For each micro-clusters in C3: if the data point (which represents the core of micro-cluster) can be absorbed into one micro-cluster in C_pre, then add the micro-cluster to the corresponding micro-cluster in C_pre and the corresponding value in count[] +1;Else put the micro-cluster into set C4;
- Sort the micro-clusters in C_pre according to the values in count[] (descend order).
- Sort the micro-clusters in C4 according to the number of points in each micro-cluster (descend order).
- If m (the size of C4) < k*30%, then delete the last m micro-clusters and add all the micro-clusters of C4 into C_pre; Else delete the last (k*30%) micro-clusters and add the first (k*30%) micro-clusters of C4 into C_pre;
- Modify the IDs of micro-clusters in C_pre and then produce the newest global micro-clusters C_new;
- Output C_new;

End

30% in the bottom third step is default threshold, whose function is to avoid the situation that if elements in C4 are too many, causing us to delete too many old micro-clusters. The value 30% can be set depending on circumstances.

After producing newest global micro-clusters, the reduce node sends them back to each map nodes and uses them to create real-time snapshot for storage.

Performance Analysis

As a result of that the unit time equals the minimum time granularity of pyramidal time frame; every time, each map node computes stream data of one unit time; and the reduce node merges the newest local clustering results to produce snapshot every unit time. The stored results in this paper's parallelized algorithm are corresponding to which in CluStream.

The processing mode in each map nodes is similar with which in the online phase of CluStream, but every map node adds a container to store stream data temporarily, so the memory consumption of one single node increases.

However, in time complexity respect, the time complexity of each map node in one unit time is o(q*n*d2), in which q is the amount of local micro-clusters, n is the amount of stream data in one unit time and d is the dimension of stream data. In the worst situation, the time complexity of reduce node in one unit is O(k*q*i2+m2) , in which q is the amount of global micro-clusters (which equals q in map node), i is the amount of global micro-clusters' IDs, k is the amount of map nodes and m is the amount of micro-clusters which need to use DBSCAN algorithm. In commonly situation, n is much bigger than k and m is very small, so the processing time of reduce node is much shorter than the time of each map node. Therefore, the efficiency of PSDMC algorithm can increase nearly linearly with the increase of map nodes while the clustering accuracy can be guaranteed.

Conclusions

This paper designs a parallelized stream data micro clustering algorithm for the micro-clustering phase of CluStream algorithm. Performance analysis shows that the parallelization thought is feasible and the research results are effective for parallelized online clustering mining on stream data.

Acknowledgment

Results of this paper are supported by ZTE Corporation fund of university-industry cooperation.

References

[1] L. Huang. Mining Stream Data: A Survey. Journal of Software.2004 ,Vol. 15, No. 1. pp. 1-7

[2] J. W. Han, M. Kamber. Data Mining: Concepts and Techniques. Second Edition, Elsevier Inc. 2006, pp. 487-489.

[3] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in OSDI'04: Proceedings of the 6th Symposium on Operating System Design and Implementation. San Francisco, California, USA. USENIX Association, pp. 137-150. December 6-8,2004.

[4] **C.**C Aggarwal, Jiawei Han, Jianyong Wang, and P.S. Yu. A Framework for Clustering Evolving Data Streams. Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003. pp.81-92

[5] L. Li and M. Zhang The Strategy of Mining Association Rule Based on Cloud Computing. Proceedings of the 2011 International Conference on Business Computing and Global Informatization (BCGIN), Shanghai, China, 2011, pp. 475 - 478

[6] H.B.Gao, J. Hou, J.F.Liu, Research on Distributed Stream Data Clustering Algorithm of Density and Central Points. Computer Application and Software, 2013. Vol. 30, No. 10. pp.181-184