BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 04/05 pp131–142
Volume 3, Number 2, June 2020
DOI: 10.26599/BDMA.2019.9020022

Online Real-Time Trajectory Analysis Based on Adaptive Time Interval Clustering Algorithm

Jianjiang Li, Huihui Jiao, Jie Wang*, Zhiguo Liu, and Jie Wu

Abstract: With the development of Chinese international trade, real-time processing systems based on ship trajectory have been used to cluster trajectory in real-time, so that the hot zone information of a sea ship can be discovered in real-time. This technology has great research value for the future planning of maritime traffic. However, ship navigation characteristics cannot be found in real-time with a ship Automatic Identification System (AIS) positioning system, and the clustering effect based on the density grid fixed-time-interval algorithm cannot resolve the shortcomings of real-time clustering. This study proposes an adaptive time interval clustering algorithm based on density grid (called DAC-Stream). This algorithm can perform adaptive time-interval clustering according to the size of the real-time ship trajectory data stream, so that a ship's hot zone information can be found efficiently and in real-time. Experimental results show that the DAC-Stream algorithm improves the clustering effect and accelerates data processing compared with the fixed-time-interval clustering algorithm based on density grid (called DC-Stream).

Key words: storm; trajectory clustering; adaptive; data mining; density grid

1 Introduction

With the development of science and technology, we have entered the era of big data[1,2], and data is of increasing significance to the development of the entire society. With the explosive growth of data and the increasing scale of data, it is important to extract useful information in real time. Different application scenarios have different algorithms to meet the high requirements of real-time[3–5]. Among the various types of data, the mining of massive trajectory data has become a research hotspot. With the development of the real-time big data processing framework represented by Storm, and the ability to take advantage of the combination of Storm and urban transportation not only the Storm distributed processing framework can discover the ship's trajectory, position, and speed in real-time, but the trajectory data can also be processed according to the individual needs of the maritime regulatory authorities. Further, the Storm can be combined with maritime traffic to improve data processing throughput and reduce trajectory processing run time. Storm's high fault tolerance, scalability, and high availability provide good system performance. Therefore, the combination of the big data real-time processing framework Storm and the maritime regulatory field is of great significance for planning maritime traffic[6–9].

With the advent of maritime information and the era of big data along with the establishment of shore-based Automatic Identification System (AIS) networks and trajectory databases, a large number of AIS ship historical trajectories can provide data support for exploring intelligent supervision algorithms, thus enabling the identification of ship motion patterns, including high-level situational awareness, such as ship

• Jianjiang Li, Huihui Jiao, and Zhiguo Liu are with the Department of Computer Science and Technology, University of Science and Technology Beijing, Beijing 100083, China. E-mail: lijianjiang@ustb.edu.cn; jiaohuihui0503@163.com; 415132399@qq.com.
• Jie Wang is with TravelSkey Technology Ltd., Beijing 101318, China. E-mail: wangjieblingbling@126.com.
• Jie Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA. E-mail: jiewu@temple.edu.
* To whom correspondence should be addressed.
Manuscript received: 2019-10-08; accepted: 2019-12-05

behavior prediction and ship anomaly detection, and improving the regulatory efficiency of the maritime sector[10–12]. Reference [13] proposed an automated method for the derivation and modeling of navigation trajectory based on AIS data of fishing vessels, and based on the mapping results of these trajectory models on the nautical chart, the identification and discovery of marine fishing areas were completed. Through the establishment of an AIS real-time management system, the high-flow and high-risk areas of ship navigation were mined. Based on this, the importance identification of the arrival area of the ship was realized, and then the behavior of the ship was analyzed[14]. Reference [15] proposed that based on model clustering, the whole trajectory is used as the unit of clustering, and the cluster is obtained by EM algorithm.

At present, there are numerous studies on ship trajectory analysis methods. In a cloud computing distributed environment, a parallel Density-Based Spatial Clustering of Application with Noise (DBSCAN) clustering algorithm has been designed and implemented. The spatial data is divided by a K-Dimensional (KD) tree space partitioning algorithm. The technical solution and implementation framework of a massive AIS data mining system based on Hadoop has been designed[16]. Reference [17] used an improved DBSCAN algorithm to mine historical AIS track topology. Another study proposed an adaptive kernel density-based method for the motion pattern mining of AIS trajectory data. Based on this, a particle filter is used to complete the anomaly detection of a ship[13].

A ship AIS positioning system cannot find the navigation characteristics of a ship in real-time, the problem of using a fixed-time-interval clustering effect is critical to trajectory data flow velocity fluctuation based on the Storm[18, 19]. This paper mainly completes the following work:

(1) An adaptive time interval clustering algorithm based on density grid and a distributed clustering framework flow are proposed on the basis of the real-time processing framework of Storm big data to examine the characteristics of streaming trajectory data.

(2) A real-time processing system for ship navigation is constructed by using the big data real-time processing framework Storm, the distributed log collection system Flume[20], and the distributed message queue Kafka[21]. Flume is used to collect the trajectory data of the ship simulation. The data access is completed by Kafka, the problem of inconsistent processing speed between the data collection module and the data processing module is alleviated, and data congestion is avoided. The pre-processing, clustering, and storage of ship trajectory data are completed by Storm. Finally, the hot zone information is visualized using the service management platform built by SpringMVC, Spring, and MyBatis framework.

The actual test results show that the proposed adaptive time interval clustering algorithm based on density grid (called DAC-Stream) algorithm not only improves the clustering effect, but also accelerates data processing.

2 Adaptive Time Interval Clustering Algorithm Based on Density Grid

At present, STREAM algorithm[22], Clustream algorithm[23], HPStream algorithm[24], and D-Stream algorithm[25] are the classical real-time data stream clustering algorithms. Among them, D-Stream is based on a density grid, does not need to retain a large amount of historical data, and can recognize clusters of arbitrary shapes. These characteristics are in line with the real-time trajectory clustering analysis in this paper. Therefore, some concepts from the D-Stream algorithm have been used to optimize the trajectory clustering analysis in this work to achieve better clustering results.

2.1 Total flow of the distributed clustering framework based on Storm

Some achievements have been made in a vehicle hot zone, traffic congestion prediction, and vehicle trajectory prediction because of the good combination of Storm and land transportation. With the increasing complexity of maritime traffic and the individual requirements of maritime regulatory authorities, certain requirements are imposed on the real-time processing of ship trajectories. To this end, this work chooses Storm as the processing framework to develop an online real-time trajectory system. The distributed processing framework of Storm is mainly used to complete the real-time clustering of ship trajectory data, and the hot zone in the navigation of the ship is efficiently found while improving the clustering speed.

The general flow of the distributed clustering framework based on Storm is shown in Fig. 1. Data pre-processing (PreBolt), local clustering (LocalBolt), global clustering (GlobalBolt), and write database (WriteBolt) are required in the Storm-based distributed clustering framework. Its topology is shown in Fig. 2.

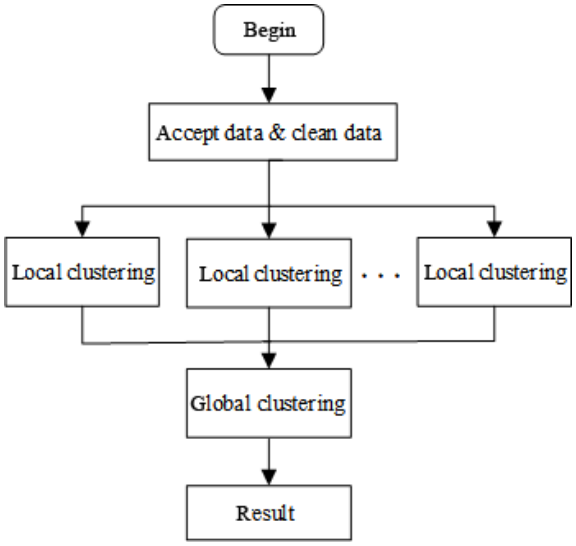


Fig. 1 Total flow of the distributed clustering framework based on Storm.

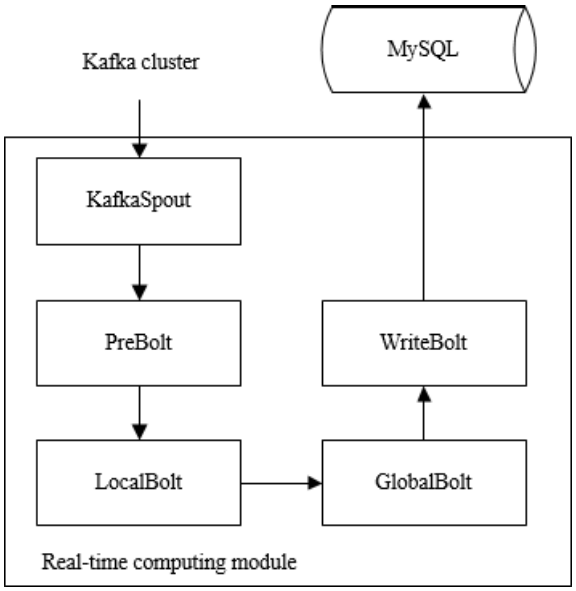


Fig. 2 DAC-Stream algorithm topology.

PreBolt completes the cleaning, de-duplication and de-noising work of trajectory data. Complex pre-processing work cannot be performed in PreBolt because the system is based on Storm's real-time processing system. Therefore, the data cleaning work mainly focuses on (1) removing duplicate data and deleting two identical ones; (2) deleting the missing points of important information, such as latitude and longitude or ship ID; and (3) reducing the trajectory data of the ship, retaining only the shipID (ship ID), mmsi (ship Maritime Mobile Service Idfentify (MMSI)), name, ship type (shiptype), lon (longitude), and lat (latitude), that can meet the business needs.

LocalBolt performs local clustering on multiple local nodes for pre-processed trajectory data and updates microcluster information in real time. The algorithm is described in detail in Section 2.3.

GlobalBolt globally clusters the results of local clustering to generate the final clustering results. A detailed description of the algorithm is provided in Section 2.4.

The hierarchical structure of local and global nodes in Storm is shown in Fig. 3. Among them, $S_1(T_s), S_2(T_s), \ldots, S_m(T_s)$ are data streams that are transmitted to different local nodes at time T_s.

2.2 Meshing, data point mapping, and grid information statistic

Data are divided into data dimensions, and then mapped to the corresponding mesh by a density grid-based clustering algorithm. Density statistics is performed on the basis of well-defined grid statistics (e.g., frequency, effective time density, etc.) and clustered according to mesh density. The density grid-based algorithm converts the grid statistics into density but does not preserve historical data. Consequently, the amount of stored data can be greatly reduced.

The following definitions regarding data space and meshing are provided.

Definition 1. For the data space S, if the data dimension of the cluster analysis is set as d, then

$$S = S_1 \times S_2 \times \cdots \times S_d \tag{1}$$

Definition 2. For any dimension in $S_i, i = 1, 2, \ldots, d$, which can be divided into n_i shares, then

$$S_i = S_{i,1} \cup S_{i,2} \cup \cdots \cup S_{i,j} \cup \cdots \cup S_{i,n_i} \tag{2}$$

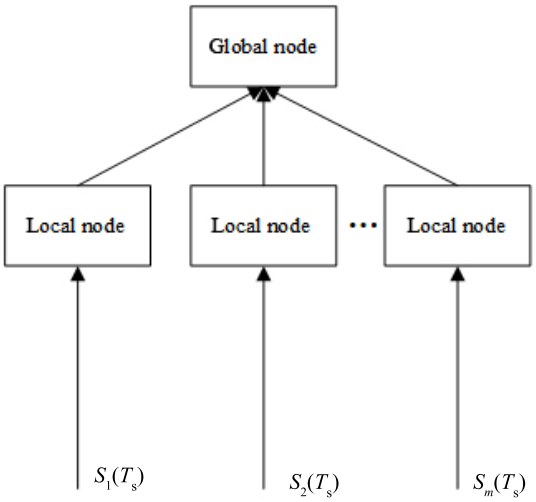


Fig. 3 Hierarchical structure of local and global nodes.

where $S_{i,j}$ is the j-th partition of the data space S divided by the dimension S_i. Then, the data space S is divided into $N = \prod_{i=1}^{d} n_i$ grids, where N is the total number of grids in the data space. In actual data space partitioning, the number of grids needs to be divided according to specific scenarios,

$$g = (k_1, k_2, \ldots, k_d) \tag{3}$$

Assume that grid $g = (k_1, k_2, \ldots, k_d)$ and data $x = (x_1, x_2, \ldots, x_d)$. When x is mapped to the space where g is located, it is recorded as

$$g(x) = (k_1, k_2, \ldots, k_d) \tag{4}$$

This work clusters ship trajectories. As such, the more important information in the data flow includes ship coordinates (including the longitude and latitude), time, and so on. To facilitate understanding, this work divides the data space into a two-dimensional space containing coordinate information and divides the mesh in the two-dimensional space. Time is represented as an "attenuation factor" (introduced in Section 2.3.1), not as a single dimension.

The latitude and longitude coordinates of the lower left corner of the grid are used to represent the grid to describe the spatial extent of grid representation. For convenience in calculation, the latitude and longitude coordinates are converted, and finally the Cartesian system coordinates are obtained.

For each point mapped to the grid, its weight in the grid can be defined. The more commonly used statistic is frequency, at which the data points are mapped to the grid. That is, for the trajectory data points in the current time stream, the initial weight of each point is set to 1, and the grid density of the map is increased by 1.

The data point mapping frequency taken as its weight is easily implementable and better reflects the trajectory of a target object in a real environment. After receiving the data points, Storm assigns them to the corresponding remote nodes according to various dimensional data of the data points. Storm then converts the latitude and longitude coordinates into rectangular coordinates and maps them to the corresponding grid.

2.3 Local clustering

The meshing and mapping of data points to grids are the basis for local clustering. Their corresponding principles are introduced in the preceding section. This section focuses on the method of performing local clustering. Figure 4 shows the basic flow of local clustering.

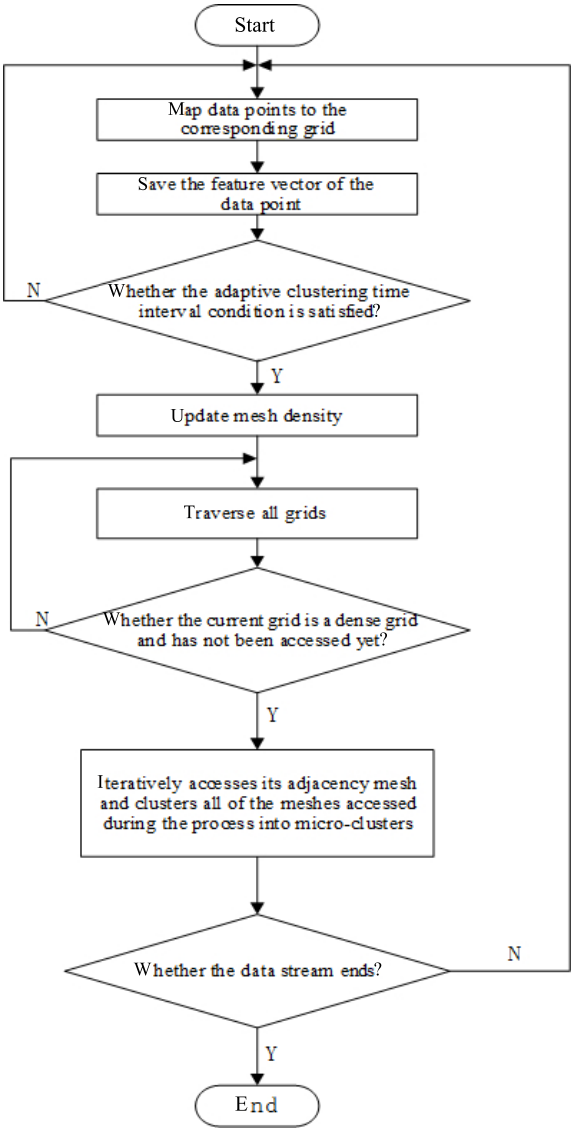


Fig. 4 Local clustering flow chart.

2.3.1 Basic concepts of data flow clustering

The following definitions are first made to clearly describe the algorithm in the local clustering process.

Definition 4. Dense mesh refers to a mesh with a density greater than the threshold D_h.

Definition 5. Sparse mesh refers to a mesh with a density less than the threshold D_l.

Definition 6. Transition grid refers to a grid with a density between D_h and D_l.

Definition 7. Microclusters constitute a grid set, which is a dense mesh with another adjacent dense mesh.

Definition 8. Connectivity: If the dense mesh is adjacent to any of the microclusters, the dense mesh is defined to be in communication with the microclusters.

Definition 9. Feature vector is a vector that holds information, such as data point coordinates, weights, and

initial mapping time.

According to Definitions 4, 5, and 6, this paper further divides the partitioned mesh into a dense mesh, transition mesh, and sparse mesh. The state of these grids is not static because they change state over time, and new data point mappings may change state. That is, a dense mesh may degenerate into a transition mesh or a sparse mesh, and a sparse mesh may evolve into a transition mesh or a dense mesh. Thresholds D_h and D_l will be manually entered on the basis of the relevant professional domain knowledge.

According to Definitions 7 and 8, grid clustering can be performed; that is, a dense grid connected to the microcluster in the current node is continuously merged into the microcluster. A cluster of any shape that can theoretically be formed is created.

Unlike static clustering, data stream clustering should consider the influence of time factors on clustering effects because real-time clustering generally involves data from a most recent period of time, but disregards data older than that from the current time. This article refers to these disregarded data as "expired data" or "historical data". Therefore, the concept of attenuation factor can be introduced to describe the change in data weight over time, i.e., the data weight decays over time.

For each data point x_i, a commonly used exponential decay factor is used to describe its decay process. The frequency is used as the statistical method of grid information, so the initial weight of each data point is 1. Then, the weight of the data point x_i after changing with time is

$$w(x_i, t_n) = \beta^{-\alpha\times(t_n - t_s)} \tag{5}$$

where $\beta > 1$ and $0 < \alpha < 1$, they are the parameters of the attenuation factor and are constant; $\beta^{-\alpha(t_n - t_s)}$ is the attenuation factor; t_n is the current time; t_s is the time at which the data point is initially mapped to the grid; and $w(x_i, t_n)$ is the weight of the data point at time t_n.

Without loss of generality, the weight of the known data point x_i at time t_i is w_{t_i}; then its weight at time t_n is

$$w(t_n) = \beta^{-\alpha(t_n - t_i)} \times w_{t_i} \tag{6}$$

Equations (5) and (6) show that the weight of the data gradually decreases with time, which is consistent with the characteristics of data stream clustering.

For grid g, the density $D(g, t_n)$, which is the sum of the weights of all the data points mapped to it, is easily derived,

$$D(g, t_n) = \sum_{i=0}^{N} w(t_n) \tag{7}$$

where N is the number of all data points in g at this time.

The density of g at the current time can be calculated not only from Eq. (7), but also from the density of the last update plus the weight of the new data mapped to the grid at current time t_n. The old data points in the grid are attenuated at the same rate after the most recent update; so the following equation can be obtained:

$$D(g, t_n) = D(g, t_i) \times \beta^{-\alpha\times(t_n - t_i)} + \sum_{i=0}^{k} w(x_i, t_n) \tag{8}$$

where k is the number of new data mapped to the grid at time t_n.

When no new data are mapped to the grid, the current grid density is expressed as follows:

$$D(g, t_n) = D(g, t_i) \times \beta^{-\alpha(t_n - t_i)} \tag{9}$$

2.3.2 Adaptive clustering interval

During the clustering interval, the system maps the data in the data stream to the grid because this stream is real-time and uninterrupted.

If the clustering interval is too short (i.e., clustering whenever new data flows in), then the change in the state of each grid may not be very large. Therefore, this clustering is more similar to the previous clustering, thereby wasting calculation. The resource and the total clustering time are expensive. If the clustering interval is too long, some moments that need to be clustered may be missed, and the meaning of real-time clustering is lost. Therefore, a suitable clustering interval is particularly important.

In another study, Ref. [26] considers that the effect of local clustering may change when the grid degenerates from a dense grid to a sparse grid or vice versa. Therefore, the smaller value of the time between the dense grid degenerating to the sparse grid or the sparse grid evolving to the dense grid can be taken as the clustering interval. However, Ref. [27] does not consider the change in the speed of the data stream in the two clustering time intervals, that is, the data speed transmitted to the trajectory analysis system in the data stream is assumed to be uniform.

However, in the actual trajectory analysis, the speed of the real-time data stream may fluctuate considerably because of the need for sensors, network communications, and a series of intermediate links in data acquisition, data transmission, and data processing.

The average speed of the data stream in the two clustering intervals is set to v. When v is faster, the amount of data to be processed in the clustering interval is larger, and the time required for the grid to evolve from a sparse grid to a dense grid is shorter. Therefore, the clustering time interval should be reduced; otherwise, if v is slow, the amount of data to be processed in the clustering interval is low, and the data mapped to the grid are reduced. Therefore, the time interval of clustering should be appropriately increased.

This work proposes an adaptive clustering time interval strategy based on the average speed of the data stream. That is, considering the change in the speed of the data stream, the grid state conversion time fluctuates, and the clustering time interval is adaptively selected according to its speed.

When a dense mesh decays to a sparse mesh over time, the following formula is obtained:

$$D_h \times \beta^{-\alpha(t_d - t_p)} \leqslant D_l \tag{10}$$

where t_d is the current clustering time and t_p is the previous clustering time.

In Eq. (10), the process of mapping new data points to the current grid is not considered in grid attenuation. When new data points are mapped to the current mesh, mesh attenuation is greatly affected, that is, the time when the dense mesh degenerates into a sparse mesh is extended, which may miss some moments that need to be clustered. Therefore, the case with newly added data points is not considered.

From Eq. (11), at least time T_1 is needed to solve the degradation of a dense grid into a sparse grid,

$$T_1 = t_d - t_p \geqslant \frac{1}{\alpha}\log_\beta\left(\frac{D_h}{D_l}\right) \tag{11}$$

For the evolution from a sparse grid to a dense grid, this paper assumes a certain law in mapping new data points to the grid; that is, new data points most likely fall into the edge of the dense mesh obtained from the previous clustering. The area enclosed by the sparse grid of "corner-connected" is shown in Fig. 5, where the dark grid represents a dense grid, and the light grid corresponds to a sparse grid. When the time interval of real-time clustering is short (usually tens of seconds), the data points of the previous cluster falling into the dense grid likely move to the grid around it in this cluster, i.e., the sparse net within the area of the grid. In this paper, a grid composed of a dense mesh and a sparse mesh is called a data mapping mesh.

When a sparse mesh evolves into a dense mesh, it is necessary to consider not only the change of the old

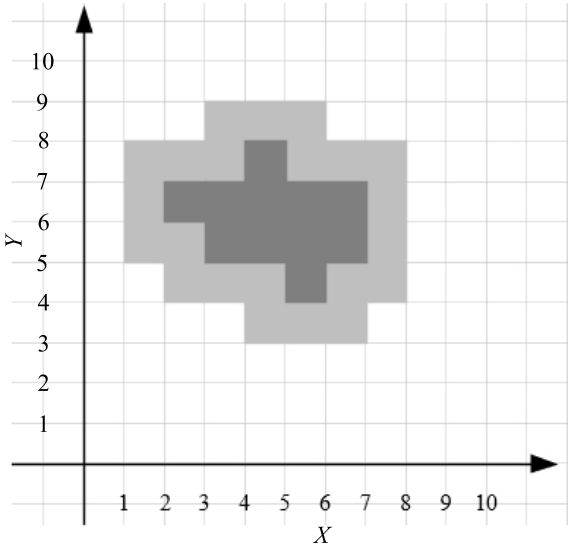


Fig. 5 Schematic of the data mapping grid.

data points in the grid with time but also the weight of the new data points in the data stream. To this end, we obtain formula (12),

$$D_l \times \beta^{-\alpha\times(t_d - t_p)} + \frac{v \times (t_d - t_p)}{n} \geqslant D_h \tag{12}$$

where v is the average speed of the data stream during the clustering time interval and n is the number of data mapping grids.

$$D_l \times \beta^{-\alpha\times(t_d - t_p)} + \frac{p_d}{n} \geqslant D_h \tag{13}$$

Then $D_l \times \beta^{-\alpha\times(t_d - t_p)}$ represents the sum of the weights of the old data points assigned to the sparse grid before the t_p time. The latter p_d/n represents the sum of the weights added after the new data points are mapped to the grid after time t_p. The attenuation of the new data points is ignored here because according to the previous clustering time interval, the new data points have only a small weight attenuation between the two clustering moments.

The time required for a sparse grid to evolve into a dense grid is T_2. Because the adaptive clustering time interval takes the time when the dense mesh degenerates into a sparse mesh or the sparse mesh evolves into a dense mesh. Then the adaptive clustering time interval G_{clu} is obtained as

$$G_{\text{clu}} = \min\{T_1, T_2\} \tag{14}$$

As p_d in Eq. (13) is actually a function that changes with time, directly solving T_2 is impossible.

At this point, if we assume that $t_d - t_p$ in Eq. (13) takes the minimum value of T_1 in Eq. (11) and substitute it into Eq. (13), it can be solved as follows:

$$p_d \geqslant \frac{\left(D_h^2 - D_l^2\right) \times n}{D_h} \tag{15}$$

Let p_d have a minimum value of $p_{\min}$, which can be obtained using Eq. (15). If the sparse grid is to be evolved into a dense grid at T_1, the amount of data to be inputted in this clustering interval is at least $p_{\min}$. If $p_d = p_{\min}$ at this time, Eq. (13) can be established by decreasing T_1. When p_d is greater than $p_{\min}$, T_1 can be further reduced.

Let the time when the data stream input data reaches $p_{\min}$ be t_p, then the solution of Eq. (14) is

$$G_{\text{clu}} = \begin{cases} T_1, & T_1 \leqslant T_p; \\ T_p, & T_1 > T_p \end{cases} \tag{16}$$

The meaning of Eq. (16) is that the clustering operation is performed immediately if the amount of data in the data stream reaches $p_{\min}$ in time interval T_1; otherwise, the clustering operation is performed in time interval T_1.

2.3.3 Local clustering algorithm

Algorithm 1 is a local clustering algorithm, and its process is described as follows.

Algorithm 1 Local clustering algorithm

Input:

grid collection grid_list, current time curtime, last clustering time preclutime, adaptive clustering time interval clu_interval, whether the grid has been accessed by a certain flag cluster is_visit.

Output:

micro cluster collection microclu_list.

```
1: Initialize the is_visit collection element to false;
2: while (Data still flows into Storm in the data stream)
3:  Read data point x_i information;
4:  Map x_i to the corresponding grid;
5:  Save the feature vector of x_i;
6:  if(curtime – preclutime == clu_interval)
7:   for (each grid_i in grid_list)
8:     Updating mesh density using Eqs. (8) and (9);
9:     if((grid_i is a dense grid && is_visit[grid_i]== false))
10:      is_visit[grid_i]←true;
11:      Access its neighboring grid grid_adj_k;
12:      if(Grid_adj_k is a dense grid && is_visit[grid_adj_k] ==
    false)
13:        is_visit[grid_adj_k]←true;
14:        Iteratively accessing the adjacency mesh of
    grid_adj_k;
15:      end if
16:    end if
17:    Store all the meshes accessed by the above process as a
    micro cluster into the microclu_list;
18:   end for
19:  end if
20:  preclutime ← curtime;
21:  Update the is_visit collection element to false;
22: end while
```

Line 1: Initialize the flag is_visit that has been accessed by the grid to a false value, i.e., all meshes are accessed.

Lines 2–5: If the data stream is not finished, the data point information is continuously read. Each dimension's information is sent to the corresponding local node based on the spatial cell where the data point read at this time belongs. On the local node, this data point is mapped to the already divided grid, and the feature vector of the data point is saved.

Lines 6–8: If the clustering time interval condition is met, then each grid in the grid set grid_list is traversed, and its grid density is updated using Eqs. (8) and (9).

Lines 9–11: During the traversal process, if the current grid grid_i is a dense grid and has not yet been accessed, its access state is set to accessed, and its adjacent mesh is accessed in all directions.

Lines 12–15: If one of the adjacency grids grid_adj_k is a dense grid and has not yet been accessed, the access state is set to be accessed, and the adjacency grid of grid_adj_k is iterated through the steps in Lines 9–16 until all the adjacency grids have been accessed.

Lines 17–19: All grids accessed from grid_i are stored as microclusters in a microclu_list for the subsequent transmission to the global node for global clustering. This local clustering is over.

Lines 20–22: The last clustering time is updated, and the grid access flag is reset to a false value by using the following clustering approach.

2.4 Global clustering

In the previous section, the adaptive clustering time interval strategy was used for local clustering on local nodes. The data stream speeds of local nodes may vary, so the resulting adaptive time intervals differ. To obtain a better clustering effect, this work selects the minimum value of the adaptive clustering time interval of each remote node as the time interval of all clustering nodes, and sends the result to the global node for global clustering after local clustering is completed.

However, the microclusters obtained by local clustering may contain more information about the grid, resulting in a larger amount of data transmission between the local node and the global node. For a global node, its bandwidth is constant. When many local nodes simultaneously transmit data to it, a large amount of bandwidth is consumed.

For the data transmitted from the local node to the

global node, the following conditions should be met:

(1) The information transmitted to the global node needs to be much smaller than the original data stream size processed by the local node.

(2) The result of the global cluster can be the result of the combination of the divided local clusters.

The data transmitted by the local node are referred to as "summary information". The above two conditions not only utilize the generality of the clustering data and reduce the amount of transmission but also ensure the reducibility of the clustering data in the global node.

Through the microcluster formed by many meshes, the boundary mesh can be determined so that the microcluster can be represented as a space surrounded by the boundary mesh, which saves the mesh information transmission inside the microcluster. At the same time, the global cluster can restore the microcluster through the boundary grid. The amount of summary information can be further reduced under the condition of ensuring reducibility.

In this work, the feature mesh of the microcluster is defined as follows: starting from the coordinates at any inflection point of the microcluster, a direction (clockwise or counterclockwise) is selected, and the inflection point along the mesh boundary is continuously searched until it returns to the original position. In the search process, the grid containing the inflection point is called the "feature grid". The inflection point can only belong to the feature mesh where it initially occurs in the selected direction.

The feature mesh is a subset of the boundary mesh that meets the first condition described above. The boundary information of the microcluster can be preserved because it contains all the inflection points of the microcluster boundary, thereby further restoring the entire microcluster information, which also meets the second condition described above.

For convenience of explanation, the coordinates in the figure represent the grid with the coordinates in the lower left corner. In Fig. 6, local clustering produces two microclusters. The microclusters on the left in Fig. 6 consists of 8 grids, and its characteristic grid is shown by the red box in Fig. 6, comprising five grids: (2,7), (2,5), (3,4), (4,5), and (3,7). The right microclusters comprise 12 grids, and their feature grids are also marked with red boxes, which are four grids: (7,8), (7,5), (9,8), and (9,5). The feature mesh reduces the amount of information transmitted by the microcluster.

Therefore, this work selects the feature mesh as the

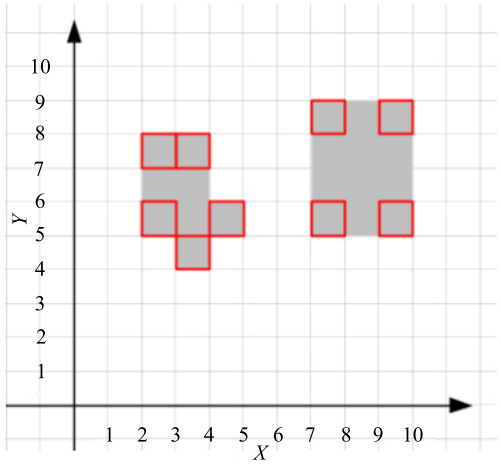


Fig. 6 Schematic of the feature cluster of the microcluster.

summary information of the microcluster and passes it to the global node.

After receiving the feature mesh sent by the local node, the global node restores the summary information represented by the feature mesh to the microcluster by using the restore function getReduction(). For the union of all local node clustering results, the formation of a larger cluster depends on the connection of the clusters.

Algorithm 2 is a global clustering algorithm.

Algorithm 2 Global clustering algorithm

Input:

summary information profileinfo_list of each local node, microclu_list before the merge, microclu_temp for temporarily storing the micro cluster, whether the two micro clusters can be merged is_merge.

Output:

The merged microcluster collection microclu_list.

```
for (each profileinfo in profileinfo_list)
    microclu ← getReduction(profile_info);
    Save microclu to microclu_list;
end for
    for (microclu_i in microclu_list)
        for (microclu_k in microclu_list)
        is_mergecanMerge(microclu_i ,microclu_k ) // i < k;
          if (is_merge)
            Store microclu_i and microclu_k in microclu_temp;
          end if
        end for
        if (is_merge)
            Remove the micro clusters that appear in microclu_temp from the microclu_list;
            Merge microclusters that appear in microclu_temp;
            The merged micro-cluster is stored in the last position of the microclu_list;
        end if
end for
```

Lines 1–4: Using the restore function getReduction(), the summary information of each local node transmitted to the global node is restored to the microcluster, and stored in the microcluster set before the merge.

Lines 5–11: According to the order of the element position in the microcluster set, starting from the first element representing the microcluster, the cycle determines whether it can merge with the latter one. After the cycle ends, it continues to determine whether the second element representing the microcluster can merge with the latter one until the last element representing the microcluster can merge with the latter one. For microclu_i, if the microcluster microclu_k that can be merged with it can be found, the return value of the canMerge() function (in this case is true) is assigned to the merge flag is_merge, and the two microclusters are stored in the set of temporarily saved microclusters.

Lines 12–17: For microclu_i in the microcluster set, if the merge flag is_merge is true, all microcluster (including microclu_i) that can be merged with microclu_i in the microclu_list are deleted, and the deleted elements are merged into one large cluster. It is placed in the last position in the collection.

3 System Implementation, Testing, and Algorithm Evaluation

3.1 Experimental environment

This experiment uses three servers in terms of hardware configuration. A distributed node is deployed on the virtual machine VMware by using the operating system CentOS6.5. The three servers include a Nimbus node and two Supervisor nodes. The server hardware configuration is given in Table 1. The required main software and version number are given in Table 2.

3.2 Test data

In this work, four different data volumes of AIS data[25] are selected as test data sets (Table 3). The data format and its meaning are given in Table 4. After preprocessing is completed, the test only needs information of five dimensions: ShipID, mmsi, name, lon, and lat. During the test, the relevant parameters in the specified cluster are $D_h = 12$, $D_l = 6$, $\beta = 2$, and $\alpha = 0.5$.

3.3 Actual test and result analysis

The adaptive time interval clustering algorithm based on density grid (called DAC-Stream) proposed and implemented in this work is compared with the fixed-time-interval clustering algorithm based on the density grid (DC-Stream). Figures 7–10 show the clustering results of the first, second, third, and fourth clusters at different times by using two clustering algorithms.

Here, Target 4 trajectory data in Table 3 are selected, and the fixed clustering time interval is estimated to be about 4 s. The comparison of Subgraphs (a) and (b) in Figs. 7–10 reveals that more clusters are produced by Subgraph (b) than by Subgraph (a) in the second and fourth clusters. Clustering with the DAC-Stream algorithm can more accurately discover the hot zone in ship navigation and improve the accuracy of the real-

Table 1 Hardware configuration.

CPU	Memory (GB)	Hard disk (GB)
Single core	4	50

Table 2 Software environment.

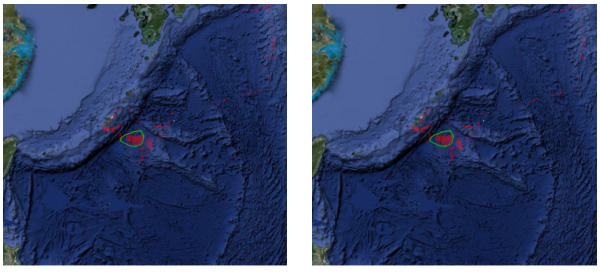
Name of software	Version number
VMware	12.5.2 build-4 638 234
CentOS	6.5
Python	2.7
Java	1.8.0_144
Zookeeper	3.4.10
Storm	1.1.1
Flume	1.8.0
Kafka	2.12–1.0.0
MySQL	5.6.16
Eclipse	4.7.0

Table 3 Test data set.

	Amount of data
Target 1	3000
Target 2	5000
Target 3	10 000
Target 4	15 000

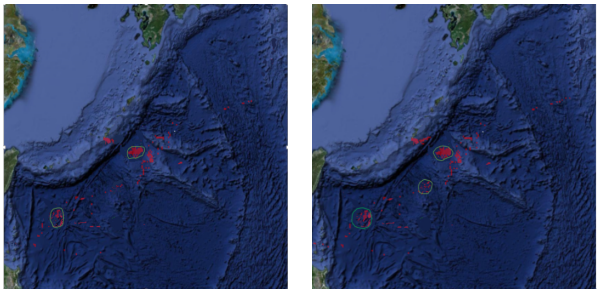
Table 4 AIS partial data format and its meaning.

Field	Type	Length	Remark
shipID	uint64	8	-
mmsi	uint32	4	-
shiptype	uint16	2	-
name	string	-	-
lon	int32	4	Longitude $[-1.8 \times 10^8, 1.8 \times 10^8]$
lat	int32	4	Latitude $[-0.9 \times 10^8, 0.9 \times 10^8]$



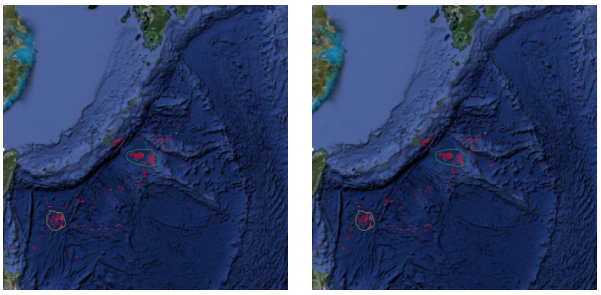
(a) DC-Stream (clustering moment t = 4 s) (b) DAC-Stream (clustering moment t = 4 s)

Fig. 7 First clustering results.



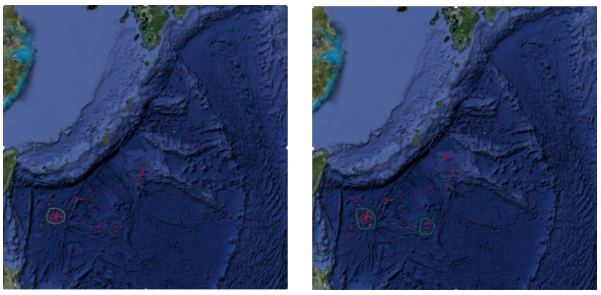
(a) DC-Stream (clustering moment t = 8 s) (b) DAC-Stream (clustering moment t = 6 s)

Fig. 8 Second clustering results.



(a) DC-Stream (clustering moment t = 8 s) (b) DAC-Stream (clustering moment t = 6 s)

Fig. 9 Third clustering results.



(a) DC-Stream (clustering moment t = 8 s) (b) DAC-Stream (clustering moment t = 6 s)

Fig. 10 Fourth clustering results.

time clustering algorithm over that of the DC-Stream algorithm. Therefore, the DAC-Stream algorithm for trajectory clustering can more timely detect the hot zone, and avoid traffic congestion and dangerous situations as much as possible.

Figures 7–10 present the visual comparison of the clustering effects of the DC-Stream algorithm and DAC-Stream algorithm. However, the merits and demerits of the clustering results can be determined by using special clustering evaluation indicators. In this work, the contour coefficients in the internal evaluation method are used as indicators.

The value of the contour coefficient is within $[-1, 1]$. The closer the value of the total contour coefficient to 1 is, the better the effect of clustering will be. K is the number of clusters and g_j is the point in the hot zone of $L_i = (g_1, g_2, \ldots, g_j, \ldots, g_n)$. The point in the hot zone is calculated using Eqs. (17) and (18). The contour coefficients of each point in the hot zone are calculated using Eq. (19). The average contour coefficient is used in this work. The clustering effects of the DC-Stream algorithm and the DAC-Stream algorithm proposed and implemented in this work are compared.

$$h(g_i) = \frac{\sum_{k=1,k\neq j}^{n} \operatorname{dist}(g_k, g_j)}{n} \tag{17}$$

$$p(j) = \min\left\{\frac{\sum_{1\leqslant k\leqslant n} \operatorname{dist}(g_j^i, g_k^l)}{n}\right\}, 1\leqslant l\leqslant m,\ l\neq i \tag{18}$$

$$h(g_i) = \frac{p(j) - h(j)}{\max\left(p(j), h(j)\right)} \tag{19}$$

Figure 11 shows a comparison of the contour coefficients of the DAC-Stream clustering algorithm and DC-Stream clustering algorithm. The contour coefficients of the DAC-Stream algorithm are much larger than those of the DC-Stream; that is, the clustering effect of the DAC-Stream clustering algorithm is better than that of the DAC-Stream algorithm.

The DAC-Stream algorithm can adaptively adjust the interval time of clustering. Therefore, when the speed of the data stream is faster, the amount of data that needs to be processed in a certain period is larger, and the grid changes from a sparse grid to a dense grid. When the time is shorter, the time interval of clustering also shortens. Conversely, if the speed of the data stream is slow, the amount of data to be processed is reduced in a certain period, and the data mapped to the grid are also reduced. The time interval of clustering should be increased. System overhead and latency can be reduced by adaptively adjusting the interval of clustering. The introduction of the feature grid in the global clustering

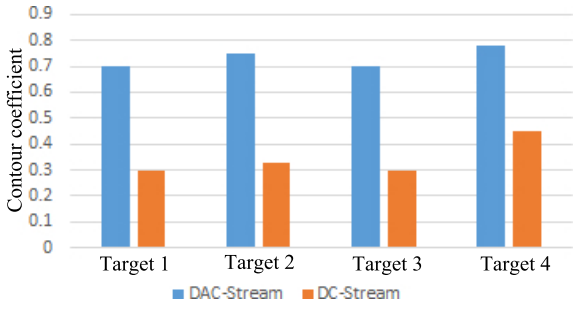


Fig. 11 Contour coefficient comparison between DAC-Stream and DC-Stream.

process reduces the amount of data transmission, thereby reducing the use of bandwidth, I/O, and the time of trajectory clustering. As the amount of data increases, the advantages of the DAC-Stream clustering algorithm become more apparent. Figure 12 presents a running time comparison diagram of clustering the test data in Table 3 by using the DAC-Stream clustering algorithm and DC-Stream clustering algorithm.

This section compares DAC-Stream with DC-Stream in terms of two aspects: the accuracy of the clustering algorithm and clustering time. The experimental results reveal that the precision of the proposed and implemented DAC-Stream clustering algorithm is higher than that of the DC-Stream, and the total clustering time of the former is shorter than that of the latter. In general, the DAC-Stream algorithm has a good effect on the online real-time clustering of streaming trajectory data.

4 Conclusion

With the development of the big data real-time processing framework represented by Storm and in comparison with urban transportation, the maritime traffic business has become more complicated, and the real-time requirements of the system are higher. A ship's navigation characteristics cannot be found in real-time with its AIS positioning system, and the clustering effect based on the density grid fixed-time-interval algorithm cannot meet the shortcomings of real-time clustering. The clustering effect based on the density grid fixed-time-interval algorithm cannot meet the real-time requirements. Clustering has some shortcomings, such as inability to find the hot zone in time. This work proposes DAC-Stream, which can adapt the time according to the size of the real-time ship trajectory data stream. Interval clustering can be achieved according to the size of the real-time ship trajectory data stream for efficient and timely acquisition of the hot zone information of the ship. Experimental results show that DAC-Stream improves the clustering effect and accelerates data processing compared with DC-Stream.

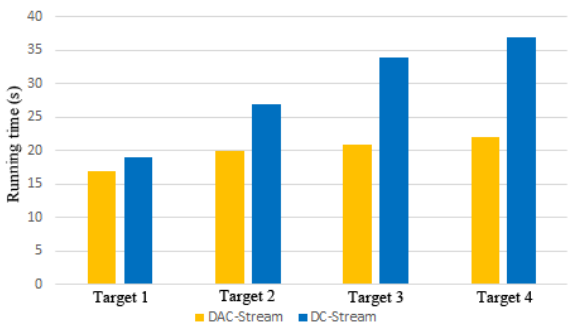


Fig. 12 Comparison of the clustering time of DAC-Stream and DC-Stream.

Acknowledgment

This work was supported by the National Key R&D Program of China (No. 2017YFB0202104).

References

[1] L. Shen and P. R. Stopher, Review of GPS travel survey and GPS data-processing methods, *Transport Reviews*, vol. 34, no. 3, pp. 316–334, 2014.

[2] G. Cugola and A. Margara, Processing flows of information: From data stream to complex event processing, *ACM Computing Surveys*, vol. 44, no. 3, pp. 1–62, 2012.

[3] F. Zhu, P. Chen, D. Yang, W. Zhang, H. Chen, and B. Zang, A GPU-based high-throughput image retrieval algorithm, in *Proceedings of the 5th Annual Workshop on General Purpose Processing with Graphics Processing Units*, New York, NY, USA, 2012, pp. 30–37.

[4] Z. Fang, D. Yang, W. Zhang, H. Chen, and B. Zang, A comprehensive analysis and parallelization of an image retrieval algorithm, in *IEEE International Symposium on Performance Analysis of Systems and Software*, Austin, TX, USA, 2011, pp. 154–164.

[5] W. Zhang, T. Bao, B. Zang, and C. Zhu, Optimizing bandwidth constraint through register interconnection for stream processors, in *Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques*, Brasov, Romania, 2007, pp. 199–208.

[6] R. Evans, Apache storm, a hands on tutorial, in *Proc. of IEEE International Conference on Cloud Engineering*, Tempe, AZ, USA, 2015, p. 2.

[7] P. Nesi, G. Pantaleo, and G. Sanesi, A hadoop-based platform for natural language processing of web pages and documents, *Journal of Visual Languages & Computing*, vol. 31, pp. 130–138, 2015.

[8] S. S. Situ, Design and implementation of real-time traffic information processing system based on storm, (in Chinese), master degree thesis, Zhongshan University, Guangzhou, China, 2015.

[9] S. S. Li, Design and implementation of real-time traffic information management system based on storm, (in Chinese), master degree thesis, Yangzhou University, Yangzhou, China, 2017.

[10] F. Mazzarella, M. Vespe, D. Damalas, and G. Osio, Discovering vessel activities at sea using AIS data: Mapping of fishing footprints, in *Proc. of International Conference on Information Fusion*, Salamanca, Spain, 2014, pp. 1–7.

[11] F. Mazzarella, V. F. Arguedas, and M. Vespe, Knowledge-based vessel position prediction using historical AIS data, in *Proc. of Sensor Data Fusion: Trends, Solutions, Applications*, Bonn, Germany, 2015, pp. 1–6.

[12] S. Kim, H. Kim, and Y. Park, Early detection of vessel delays using combined historical and real-time information, *Journal of the Operational Research Society*, vol. 68, no. 2, pp. 1–10, 2016.

[13] B. Ristic, B. L. Scala, M. Morelande, and N. Gordon, Statistical analysis of motion patterns in AIS data: Anomaly detection and motion prediction, in *Proc. of International Conference on Information Fusion*, Cologne, Germany, 2008, pp. 1–7.
[14] R. Laxhammar, G. Falkman, and E. Sviestins, Anomaly detection in sea traffic – A comparison of the Gaussian mixture model and the kernel density estimator, in *Proc. of International Conference on Information Fusion*, Seattle, WA, USA, 2009, pp. 756–763.
[15] S. Gaffney and P. Smyth, Trajectory clustering with mixtures of regression models, in *Proc. of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, San Diego, CA, USA, 1999, pp. 63–72.
[16] S. N. Shang, Design and implementation of massive AIS message data mining system based on cloud computing and distributed technology, (in Chinese), master degree thesis, Dalian Maritime University, Dalian, China, 2017.
[17] H. S. Qiu, Research on forecasting ship sailed track behavioral abnormalities algorithm based on Kalman filter, (in Chinese), master degree thesis, Hebei University of Technology, Tianjin, China, 2012.
[18] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, et al. Storm@twitter, in *Proc. of ACM SIGMOD International Conference on Management of Data*, Snowbird, UT, USA, 2014, pp. 147–156.
[19] D. Simonassi, G. Eisbruch, and J. Leibiusky, *Getting Started with Storm*. Sebastopol, CA, USA: O'Reilly Media Inc., 2012.
[20] D. Vohra, Apache flume, in *Proc. of Practical Hadoop Ecosystem*, Berkeley, CA, USA, 2016, pp. 287–300.
[21] K. Thein, Apache Kafka: Next generation distributed messaging system, *Journal of Scientific Engineering and Technology Research*, vol. 3, no. 47, pp. 9478–9483, 2014.
[22] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, Streaming-data algorithms for high-quality clustering, in *Proc. of International Conference on Data Engineering*, San Jose, CA, USA, 2002, p. 685.
[23] C. C. Aggarwal, J. Han, J. Wang, T. J. Watson, and P. S. Yu, A framework for clustering evolving data streams, in *Proceedings of the 29th International Conference on Very Large Data Bases, VLDB Endowment*, Berlin, Germany, 2003, pp. 81–92.
[24] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, A framework for projected clustering of high dimensional data streams, in *Proc. of Thirtieth International Conference on Very Large Data Bases, VLDB Endowment*, Toronto, Canada, 2004, pp. 852–863.
[25] W. R. Jia, Research on clustering analysis algorithm for real time data stream, (in Chinese), master degree thesis, North China Electric Power University, Beijing, China, 2017.
[26] S. W. Li, Research of data stream clustering methods based on density, (in Chinese), master degree thesis, Xidian University, Xi'an, China, 2017.
[27] X. Cai, Application and development of AIS, *Mechanical and Electrical Equipment*, vol. 28, no. 2, pp. 28–30, 2011.

Jianjiang Li is currently a professor at the University of Science and Technology Beijing, China. He received the PhD degree from Tsinghua University in 2005. He was a visiting scholar at Temple University from Jan. 2014 to Jan. 2015. His current research interests include parallel computing, cloud computing, parallel compilation, and big data.

Huihui Jiao is currently a master student in the University of Science and Technology Beijing, China. She received the bachelor degree from the North University of China in 2017. Her current research interests include parallel computing and cloud computing.

Jie Wang received the master degree from the University of Science and Technology Beijing, China, in 2019. She is now working in TravelSky Technology Ltd. Her research interests include parallel computing, cloud computing, and big data.

Zhiguo Liu is currently a master student in the University of Science and Technology Beijing, China. His research interests include parallel computing, cloud computing, and big data.

Jie Wu received the PhD degree from Florida Atlantic University in 1989. He is an IEEE fellow. He serves as the director of Center for Networked Computing and Laura H. Carnell professor at Temple University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications.