Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 15, No. 2, 2024

Illegal ship identification system based on trajectory similarity measurement

Enze Wu

School of Information Engineering, Jiangsu Maritime Institute, Nanjing, China

Abstract: The Automatic Identification System (AIS) relies on the identification code (MMSI) of a vessel to identify it and records its trajectory data, which is an important technical means for modern maritime management. However, there are some vessels that violate regulations by opening multiple AIS transponders, resulting in multiple location trajectory records for the same vessel. This behavior is difficult to distinguish through traditional means and has a negative impact on the accuracy and effectiveness of maritime management, thereby threatening the safety of navigation. This system uses big data processing tool Spark to efficiently process and analyze large amounts of AIS data and detects the similarity of vessel trajectories to automatically identify the violating vessels that open multiple AIS transponders. In this way, the system not only improves the detection efficiency of violations, but also provides more reliable evidence for maritime management departments to ensure the safety and order of navigation routes.

Keywords: Automatic ship identification system; Trajectory similarity detection; Channel safety.

1. Introduction

Automatic Identification System (AIS) is an indispensable tool in Marine traffic management [1]. It identifies ships through the Maritime Mobile Service Identity (MMSI), and records and transmits the position information, heading, speed and other data of ships in real time, thus forming complete track data. These data not only provide a reliable basis for maritime management, but also provide technical support for safe navigation of ships and sea area monitoring. However, with the increasing frequency of maritime activities, some ship violations begin to bring challenges to maritime management, especially the phenomenon that a single ship illegally opens multiple berth equipment. This behavior leads to the abnormal trajectory of the same ship, which brings great trouble to the management and monitoring of the ship.

Since multiple MMSI numbers will be generated after a ship opens multiple slipways, resulting in multiple similar tracks in AIS data, it is difficult for the existing system to accurately identify these anomalies [2], thus negatively affecting the effectiveness of the maritime management system. This not only increases the workload of the maritime management personnel, but also poses a potential threat to the navigation safety of the waterway [3].

To solve this problem, this paper combines big data technology and uses the method of trajectory similarity measurement to realize the identification of illegal ships.

2. System Architecture

The data acquisition layer is responsible for collecting the raw signals of AIS from the Jiangsu section of the Yangtze River, and the AIS signals contain important information such as the MMSI code of the ship, the time of the signal, the latitude and longitude, the heading, the speed, etc. Because of the complexity of the sources, Flume is used as a data acquisition tool to collect the AIS data from these different sources. As the signal sources are complex and come from many different data sources, Flume is used as a data acquisition tool to collect AIS data from these different signal sources respectively. Flume can flexibly adapt to multiple data sources and transfer the collected AIS signals to Kafka message queue in real time to ensure the reliability and efficiency of data collection.

The data processing layer uses the stream processing tool Flink and the batch processing tool Spark to process these AIS signals [4]. First, Flink subscribes to the AIS signals in Kafka and processes the signals in real time for basic data cleaning, including de-weighting, format conversion, and check summing of some fields. After the data cleaning is completed, Flink writes these processed signals to HDFS (Hadoop Distributed File System) according to the batch to ensure that the track data can be stored for a long time. After that, Spark regularly reads the trajectory data from HDFS and performs in-depth analyses to find similar patterns in the trajectory data. Spark's powerful computational capability makes it able to effectively process large-scale trajectory data and meet the demands of timed batch processing.

After the data processing is completed, the results processed by Spark are imported into MySQL database and Redis using Sqoop, which is convenient for the application layer to perform query and further analysis operations. This process stores the results of trajectory similarity analysis in the same database with other related data for quick access and application layer use.

The application layer provides rich functional interfaces and data presentation through collaboration between the front and back ends. The back end is developed using Java frameworks such as SpringBoot, MyBatis and implements RESTful style API interfaces [5], which are mainly used for similar trajectory querying and CRUD operations of data. The front-end part uses the Vue framework, and requests are made to the back-end interface through Axios to get the data of similar trajectories from MySQL. The acquired data is displayed through Echarts charts for statistical analysis [6], while the ship trajectory is visualized with the help of Gaode Map API, so that users can intuitively view the ship's movement trajectory and similar analysis results.

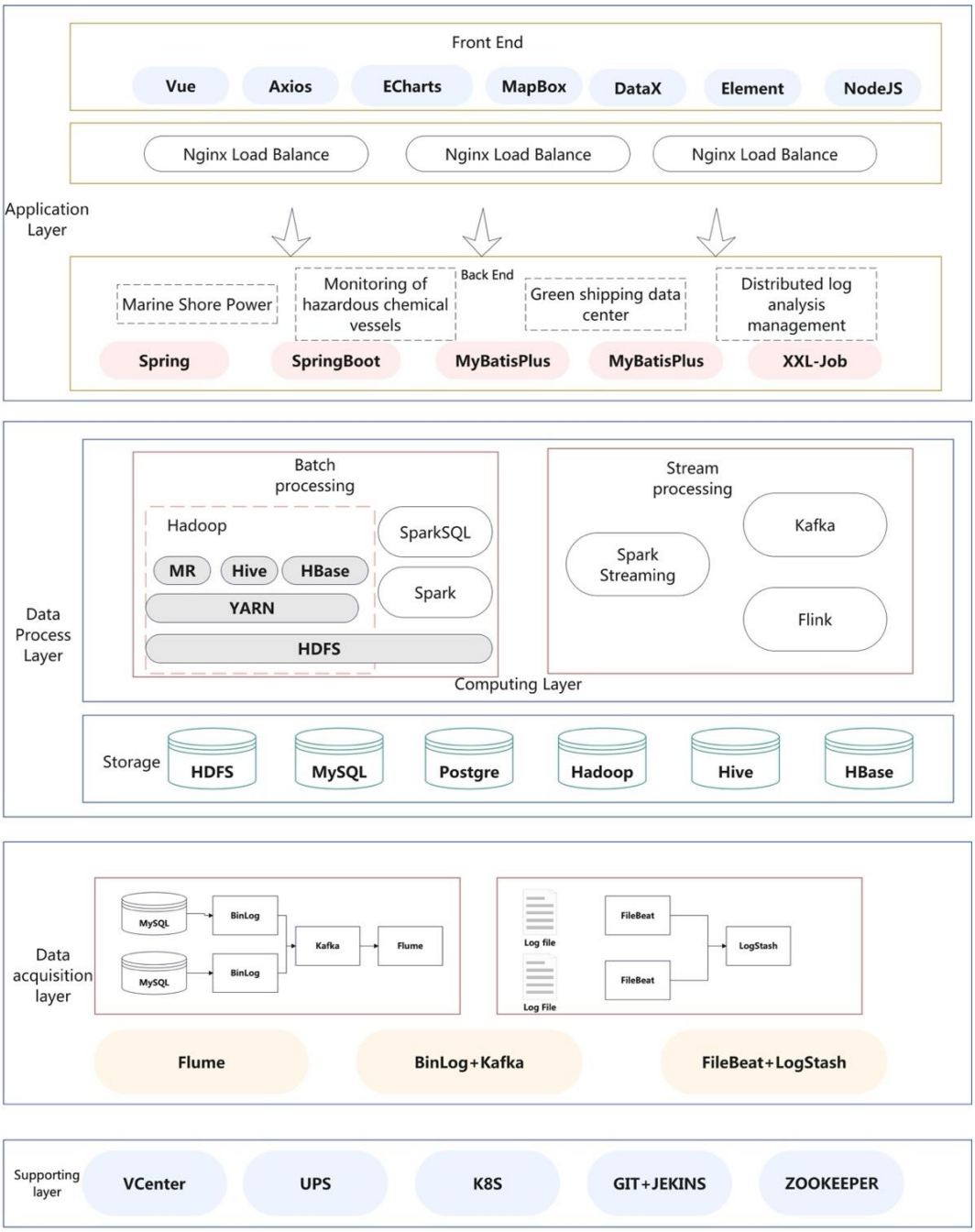

Fig.1 System architecture diagram

3. System Implementation

3.1. Data acquisition and storage

AIS data has high-frequency and massive characteristics, with the continuous movement of the ship, the device generates a large amount of positioning information in real time. In order to effectively process and transmit these data, it is first necessary to use the Flume tool to collect data from multiple data sources. Flume is able to flexibly adapt to the data streams from different sources, aggregating and transmitting them. After data collection is complete, all AIS data will be uniformly sent to the message queue Kafka. Kafka, with its high throughput and distributed architecture, can efficiently process and store these large-scale AIS data and provide a reliable streaming data pipeline for subsequent big data processing tasks. This architecture not only ensures real-time data availability, but also provides a solid foundation for downstream analyses, monitoring and warning systems.

Due to the differences in the quality of ship equipment and the influence of the strength of different river area signals [7], the raw AIS data often have various types of anomalies. For example, the data is beyond the reasonable range, the positioning point drift and so on. These anomalies will cause the trajectory data to be inconsistent with the actual situation, affecting the subsequent analysis and processing. Based on the window function provided by Flink, the system can detect and process the continuous AIS signals of the same ship over a period to ensure the continuity and reasonableness of the trajectory. Flink not only filters the signal interruptions or noisy data, but also eliminates the problem of point drift through real-time correction of the trajectory data, so as to generate a ship movement trajectory that is more in line with the actual situation.

The pre-processed trajectory data is stored in the HDFS distributed file system for subsequent analysis and processing. In order to optimize the resource utilization and reduce the pressure on the server, Flink does not write each piece of data to HDFS immediately after it is processed; instead, Flink adopts a batch writing strategy, storing the data temporarily in memory and writing the processed data to HDFS at the end of each 30-minute window, which not only improves the writing efficiency, but also reduces the impact of frequent writes. efficiency of the system and reduces the I/O overhead caused by frequent writes, but also ensures the stability of the system when processing large amounts of data. With this timed batch storage approach, the system can better handle massive AIS data, while ensuring the timeliness and effectiveness of data persistence.

This system has a high degree of flexibility and scalability through distributed data collection and storage technology. According to the change of AIS data processing volume, the system can dynamically adjust the server resources to ensure efficient and stable operation even when the data volume increases. With this technical architecture, when the data traffic is small, the system can reasonably allocate resources to save computation and storage costs; and when the data traffic surges, the system can quickly expand the computation and storage capacity to avoid performance bottlenecks. Such a flexible adjustment mechanism not only improves resource utilization, but also ensures the system's efficiency and sustainability in handling large-scale massive AIS data.

3.2. Similar trajectory recognition

The system employs Spark tools for similar trajectory identification and analysis, making full use of Spark's distributed computing capabilities in order to efficiently process large-scale AIS data. The system runs in Spark on YARN mode, and the cluster resources are dynamically scheduled and managed by YARN resource manager [8]. With this architecture, Spark can execute trajectory similarity computation tasks in a multipoint cluster in parallel, effectively improving the processing and computation efficiency. In addition, the highly flexible system of Spark on YARN mode can flexibly adjust the resource allocation according to the changes of processing demand, thus ensuring that the system can maintain efficient and stable performance when processing data of different sizes. The process of trajectory similarity detection using Spark is as follows.

Step 1: Trajectory aggregation. To perform similarity calculation on trajectories, consecutive position points of the same ship at a certain time need to be aggregated into trajectories. Each AIS data carries a unique identifier MMSI code of the ship, to generate the trajectory of each ship, the data needs to be grouped according to MMSI first.

Step 2: Trajectory similarity metric, which is the core link in similar trajectory detection. In this paper, the detection of similar trajectories is achieved by using the Longest Common Subsequence (LCSS), which is an algorithm commonly used for comparing the similarity of time series data, and its main advantage lies in its ability to flexibly cope with some inaccuracies in the trajectories, such as offsets, noises and missing points.

LCSS compares the points in two trajectories one by one and finds the longest overlap between them by means of dynamic programming. Unlike other strict similarity measures, LCSS allows a certain range of spatial and temporal deviations earlier, i.e., some points in the two trajectories do not need to match exactly but can be offset to a certain extent. Such an approach is well suited for processing AIS data. If a certain pair of positional points satisfy the distance and time tolerance conditions, they are matched and added to the common subsequence. Through such a matching process, LCSS can find the longest common path segment between two trajectories, which in turn yields their similarity metric. The more similar the two trajectories are, the longer their common subsequence is, and the higher the final output similarity score is. The pseudo-code for the trajectory similarity metric of this system is as follows.

Step 3: Parallelized trajectory similarity computation. Due to the large amount of AIS trajectory data, the detection of similar trajectories needs to be accelerated by distributed computation. All possible trajectory pairs are generated by cartesian operation, and then similar trajectories are filtered out by similarity threshold.

Step 4: Storage and Output. After similarity calculation, the data results will be output to Redis & MySQL.

Algorithm1 Similar trajectory discrimination

```
// Input:
// T1, T2: Two tracks, each a list of track points (latitude, longitude, timestamp)
// epsilon: Space tolerance distance threshold
// delta: Time tolerance threshold
function LCSS (T1, T2, epsilon, delta):
    // Gets the length of two tracks
    m = length(T1)
    n = length(T2)
    // Create a matrix L of size (m+1) x (n+1) to store dynamic programming results
                L = array[m+1][n+1]
    // Initializes the boundary of the matrix L
    for i = 0 to m:
        L[i][0] = 0
    for j = 0 to n:
        L[0][j] = 0
    for i = 1 to m:
        for j = 1 to n:
            // Take out the i-1 and j-1 points in trajectories T1 and T2
            p1 = T1[i-1]
            p2 = T2[j-1]
            // Calculate the spatial distance between two points
            spatial_dist = sqrt((p1.latitude - p2.latitude)^2 + (p1.longitude - p2.longitude)^2)
            // Calculation time difference
            time_diff = abs(p1.timestamp - p2.timestamp)
            // Determine whether the two points meet similar conditions
            if spatial_dist <= epsilon and time_diff <= delta:
                // Match, length plus 1
                L[i][j] = L[i-1][j-1] + 1
            else:
                // Otherwise, choose to skip the current point
                L[i][j] = max(L[i-1][j], L[i][j-1])
    return L[m][n]
```

3.3. Application layer

In the application layer, the user can open the webpage to

view the current position of each ship with the result of ship similar trajectory detection for disposal.

To display the ship trajectory information on the map, the system adopts the map API provided by Gaode for trajectory drawing. Firstly, the user can view the current positions of all the ships on the map. To realize the ship positioning function, Axios is used to request and interact with the back-end data, and the real-time position of the ship is displayed through the AutoNavi map API. The component-based structure of Vue.js makes the logic of map, ship marking and data interaction clear and controllable, and it is easy to modularize these functions, which is convenient for development and maintenance.

By requesting ship's position information from the back-end server through Axios, Axios can handle asynchronous requests, enabling the front-end to pull the latest ship's position information and update the display on a regular basis. At the same time, by setting the polling mechanism, real-time data is obtained regularly to ensure the dynamic display of the ship's position. After the map is initialized, the current position of the ship is marked on the map through the API and Marker function based on the acquired ship position data. The High-Definition Map API also provides rich functions to set the zoom level of the map, view center, etc. according to requirements. When the ship's position is updated, Vue's responsive data binding mechanism can automatically refresh the ship's position marker on the map to achieve a dynamic display of the ship's trajectory without users having to manually refresh the page. In this way, the system can display the current position and movement track of the ship on the map in real time.

The data request sent by the front-end through Axios will be processed by the back end built based on SpringBoot, which will first query whether the track information exists or not from Redis, and if it does not exist, then it will continue to query the track data from MySQL and update the track data into Redis. If the track data is updated, the system will update the data in MySQL and expire the old record in Redis. By handling the caching Loki in this way, the overhead caused by many repetitive queries for ship trajectories can be reduced.

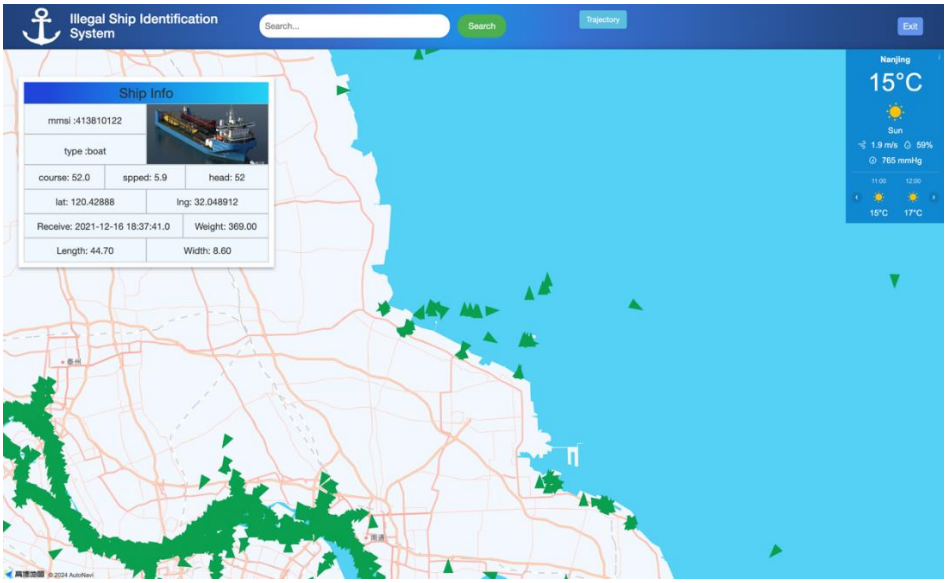

Fig.2 Ship positioning

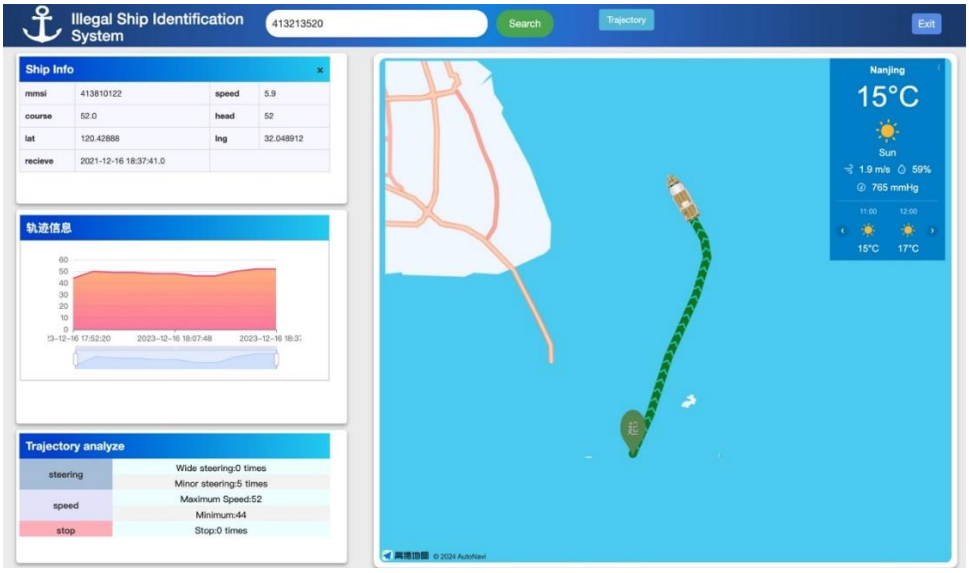

Fig.3 Ship trajectory analysis

4. Summary and Outlook

In this paper, we use various big data tools and front-end and back-end development tools to complete the ship similar trajectory detection system, to achieve the detection of similar trajectory ships under the background of massive data, so as

to identify the ships with illegal behaviors. The system can effectively detect ships with multiple slips open within a reasonable time, which ensures the safety of navigation. In the subsequent research, the accuracy and performance of the similar trajectory detection algorithm will be further improved, aiming to complete the identification of similar trajectories in near real-time.

Acknowledgements

This work was financially supported by the Research Program of Jiangsu Maritime Institute(2022ZKyb02), the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (22KJB580002), and the Excellent Teaching Team for QingLan Project of the Jiangsu Higher Education Institutions of China (Big Data Technology Teaching Team with Shipping Characteristic).

References

[1] Svanberg, Martin, et al. "AIS in maritime research." Marine Policy 106 (2019): 103520.

[2] Yang, Dong, et al. "How big data enriches maritime research–a critical review of Automatic Identification System (AIS) data applications." Transport Reviews 39.6 (2019): 755-773.

[3] Tijan, Edvard, et al. "Digital transformation in the maritime transport sector." Technological Forecasting and Social Change 170 (2021): 120879.

[4] Katsifodimos, Asterios, and Sebastian Schelter. "Apache flink: Stream analytics at scale." 2016 IEEE international conference on cloud engineering workshop (IC2EW). IEEE, 2016.

[5] Ehsan, Adeel, et al. "RESTful API testing methodologies: Rationale, challenges, and solution directions." Applied Sciences 12.9 (2022): 4369.

[6] Li, Deqing, et al. "ECharts: a declarative framework for rapid construction of web-based visualization." Visual Informatics 2.2 (2018): 136-146.

[7] Lee, Yi-Te, et al. "State-level HCC incidence and association with obesity and physical activity in the United States." Hepatology 74.3 (2021): 1384-1394.

[8] Cheng, Dazhao, et al. "Cross-platform resource scheduling for spark and mapreduce on yarn." IEEE Transactions on Computers 66.8 (2017): 1341-1353.