Journal of Research in Science and Engineering (JRSE)
ISSN: 1656-1996 Volume-7, Issue-10, October 2025
DOI: 10.53469/jrse.2025.07(10).03

Multi-level Ship Trajectory Query System Based on Flink

Jin Tao, Enze Wu, Taizhi Lv

College of Information Engineering, Jiangsu Maritime Institute, Nanjing 211170, Jiangsu, China

Abstract: ***Against the backdrop of increasingly frequent global maritime transportation and sustained growth in ship traffic, how to efficiently process and display vast amounts of ship trajectory data has become an important issue in smart maritime management. Traditional trajectory query systems mostly rely on batch processing, which is difficult to meet the multiple demands for real-time performance, visualization levels, and query efficiency. To address this challenge, this paper designs and implements a multi-level ship trajectory query system based on Flink, combining a stream computing framework with a front-end and back-end separation architecture to achieve real-time processing, hierarchical management, and dynamic visualization of ship trajectories. The system possesses strong practicality and promotional value. The back-end of the system uses Spring Boot to build core services, while the front-end uses HTML and JavaScript frameworks to implement interactive interfaces for users and administrators respectively. Apache Flink is introduced as a real-time data processing engine to perform high-throughput aggregation and thinning processing on AIS ship data. Trajectory data with different granularities are displayed according to different map zoom levels, thus ensuring trajectory integrity while optimizing query efficiency and front-end rendering performance.***

Keywords: Flink, Ship trajectory, Stream computing, Multi-level visualization, Real-time processings.

1. Introduction

With the continuous development of global trade, maritime transportation, as an important mode of international logistics, has become increasingly prominent, and the number of ships and navigation frequency continue to increase [1][2]. The Automatic Identification System (AIS) serves as a core means of monitoring ship movements, capable of collecting a large amount of real-time location information, speed, direction, and other data, providing important data support for maritime supervision, shipping scheduling, and traffic safety. However, AIS data is characterized by high frequency, high dimensionality, and continuity. Faced with the increasingly large scale of trajectory data, traditional storage and processing methods have become difficult to meet the dual requirements of real-time performance and visual display [3][4].

Currently, common ship trajectory systems mostly adopt offline batch processing methods, which present significant bottlenecks in trajectory rendering and query efficiency, making it difficult to cope with complex business scenarios and changing user interaction demands. Especially in map trajectory display, as the zoom level changes, how to dynamically adjust the granularity of trajectory data and achieve the loading and presentation of trajectory information at different levels becomes a key issue affecting system performance and user experience [5]. Therefore, exploring a new system architecture that supports real-time processing, hierarchical visualization, and efficient querying of massive trajectory data holds significant research and application value [6].

Apache Flink, as a new-generation distributed stream processing framework, boasts advantages such as low latency, high throughput, and robust state management, making it highly suitable for handling real-time aggregation and trajectory thinning of AIS data. By integrating a front-end and back-end separation technical architecture, utilizing HTML and JavaScript frameworks to construct interactive interfaces, employing Spring Boot to implement business logic and data interfaces, and complementing this with MySQL for managing trajectory data storage across different time granularities, it is possible to effectively achieve high-performance response and multi-level display in a trajectory query system. Therefore, the design of a multi-level ship trajectory query system based on Flink not only aligns with the development trend of smart maritime affairs but also provides a practical and feasible solution for the efficient management and visualization of massive trajectory data.

This paper aims to design and implement a multi-tiered ship trajectory query system based on Flink, addressing issues such as poor real-time performance, limited display options, and weak scalability in current ship trajectory data processing. By introducing a stream computing framework, real-time aggregation and hierarchical trajectory thinning of high-frequency AIS data are achieved [7][8]. Combined with a front-end and back-end separation architecture, interactive map display is realized. The system can dynamically adjust trajectory data granularity according to map zoom levels, balancing trajectory completeness and system response efficiency, thereby enhancing the ship trajectory query and visualization experience. The construction of this system not only has strong practical engineering value but also serves as a demonstration for the intelligent upgrade of maritime information systems. This project focuses on two core issues: "high-concurrency real-time trajectory processing" and "multi-tiered data visualization". It integrates the stateful computing capabilities of Flink with the visualization strengths of modern web development frameworks, forming a trajectory query solution with real-time response capabilities, flexible data abstraction abilities, and excellent user experience. This not only provides technical support for areas such as ship supervision, shipping scheduling, and maritime early warning but also offers valuable references for the architectural design of traffic trajectory systems. With the advancement of concepts like smart ports and digital shipping, the expansion potential of this system in practical applications will become more prominent, continuously supporting the digital transformation of related industries. Building a system architecture that integrates stream computing, hierarchical trajectory display, and real-time interaction functions has become a hot and challenging topic in current research on ship trajectory systems both domestically and internationally. Leveraging the technical advantages of Apache Flink in the field of stream computing, combined with the efficient rendering capabilities of HTML and JavaScript frameworks on the front end and the modular service framework of Spring Boot on the back end, exploring a scalable, visual, and real-time responsive ship trajectory query system holds significant practical and forward-looking value.

2. System Design

2.1 Overall Architecture

To achieve efficient collection, processing, storage, and display of ship trajectory data, the overall functional structure of this system is divided into multiple relatively independent yet cooperating modules. Interaction between modules is facilitated through a unified data interface, ensuring that the system possesses good scalability, maintainability, and high concurrency processing capabilities. The system primarily includes the following functional modules:

1) Data acquisition module

This module is responsible for collecting ship trajectory data from external AIS devices or data sources and pushing it into the Kafka message queue. The collected data typically includes fields such as MMSI, timestamp, latitude and longitude, heading, and speed. Leveraging Kafka's high throughput characteristics, reliable transmission of massive real-time trajectory data is achieved, providing data support for subsequent processing.

2) Flink stream processing module

As the core processing unit of the system, the Flink module undertakes the real-time computation task of trajectory data. The system performs windowed processing on ship data based on Event Time, and aggregates it according to the set time granularity (such as 5 minutes, 10 minutes, 30 minutes, 50 minutes) to generate multi-level trajectory data. This module is also responsible for handling out-of-order data, data cleaning, trajectory slicing, and other logics, ensuring the temporal accuracy and completeness of the subsequent displayed data.

3) Data storage module

The processed trajectory data will be stored in the MySQL database according to its time granularity. The system has established independent data table structures for different time levels to achieve hierarchical management of trajectory data. This module design follows database optimization strategies such as efficient indexing and partitioned storage to improve trajectory query efficiency.

4) Backend service module

The backend service, built on the Spring Boot framework, provides a unified API interface responsible for data interaction with the frontend system. This module encapsulates logic such as trajectory querying, condition filtering, and paging loading. It supports rapid retrieval of required trajectory information based on conditions such as ship number, time range, and trajectory level, and returns the results in JSON format to the frontend for invocation.

5) Front-end display module

The system frontend is implemented based on an HTML and JavaScript framework, responsible for the visual display of ship trajectories. This module integrates map components, dynamically requests trajectory data of different granularities according to the user's current zoom level, and renders them as continuous and smooth track lines. It also provides interactive query, timeline playback, and other functions to enhance the user experience.

6) System management module

This module is primarily designed for system administrators, offering functions such as basic configuration management, data table monitoring, and operational status viewing. Administrators can utilize this module to adjust database structures, configure Kafka connection parameters, and monitor Flink jobs, ensuring the continuous and stable operation of the system.

Through the collaborative work of the aforementioned modules, the system achieves an integrated ship trajectory management capability, encompassing data collection, stream processing, hierarchical storage, multidimensional query, and visual display, effectively enhancing the utilization efficiency and application value of massive trajectory data. The functional module diagram of the system is shown in Figure 1.

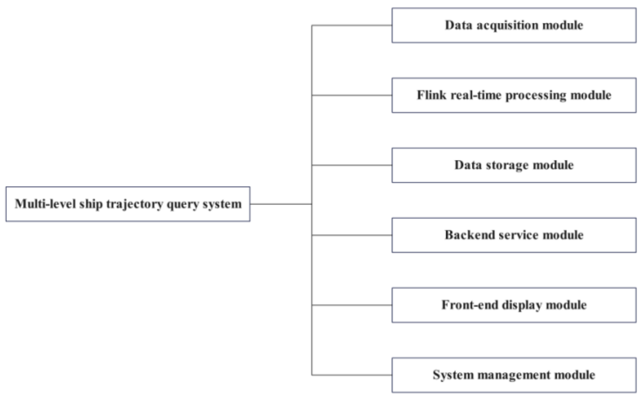

Figure 1: Functional Module Diagram

2.2 Functional Module Design

The core function of the trajectory data processing and storage module is to utilize Apache Flink to read the ship AIS trajectory data pushed by Kafka in real-time, perform aggregation processing using a fixed time window (Tumbling Window) based on event time, and ultimately store the results hierarchically in a MySQL database for subsequent querying, analysis, and visualization. This module possesses characteristics such as high throughput, low latency, and fault tolerance, making it suitable for real-time processing scenarios involving massive trajectory data. The overall system process is as follows:

1) Kafka data access: Real-time consumption of trajectory data published by topics (such as "ais") in Kafka through FlinkKafkaConsumer. The data format is a string, containing fields such as timestamp, MMSI (unique identifier for ships), longitude and latitude, and speed.

2) Event time extraction and watermark generation: Utilize the Flink API, utilizing the time field (receive) from the trajectory data as the event time, and establish a maximum allowable delay time (such as 2 minutes) to generate watermarks, ensuring accurate aggregation even in the event of data disorder.

3) Fixed time window aggregation processing: Group by MMSI using keyBy, and use time windows (such as 5 minutes, 10 minutes, or 30 minutes) to perform aggregation calculations on the trajectory points of each ship within that time period, such as trajectory point sets, average speed, maximum speed, path range, etc. Compared to session windows, fixed time windows help maintain the stability of time period division and are more suitable for periodic visualization display and analysis requirements.

4) Result mapping and hierarchical storage:

First, the original trajectory data is directly written into the MySQL table based on the original fields. Then, the aggregated results obtained through time window calculations (such as average speed, trajectory segments) are written into the corresponding time-separated data table. Finally, using a custom Sink or JDBC Sink, the structured results are written into MySQL, enabling hierarchical data management and efficient query support.

The data flow diagram of this module is shown in Figure 2 below:

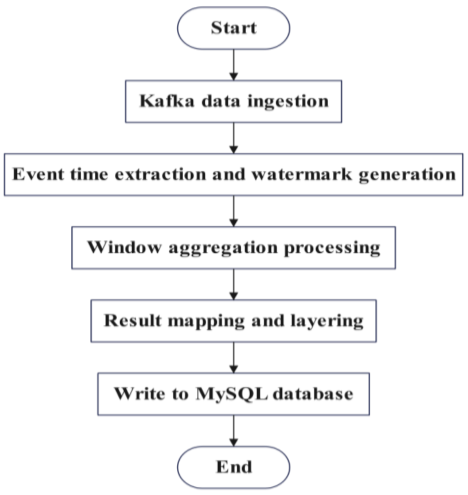
Start
Kafka data ingestion
Event time extraction and watermark generation
Window aggregation processing
Result mapping and layering
Write to MySQL database
End

Figure 2: Flowchart of trajectory data processing

3. System Implementation

In this system, to meet users' demands for high-precision analysis of trajectory data, a function for displaying ship trajectory data based on 5-minute intervals is provided. The front end is developed using an HTML and JavaScript framework, which sends request parameters (such as MMSI number, time range, and time interval) to the back end via fetch. The back end, based on Spring Boot, receives the request and queries the 5-minute interval data stored in the hierarchical trajectory data table in MySQL.

The interface employs a combination of tables and maps, allowing users to dynamically view the ship's trajectory points on the map, while simultaneously viewing the corresponding longitude, latitude, speed, and time information in the table. The map component utilizes an open-source interactive map library (Amap JS API). Trajectory points are connected by lines to form a complete trajectory, and users can interact with the map through dragging and zooming operations.

The key to implementing this feature lies in the layered processing of backend data and the visual rendering of trajectory points on the frontend, ensuring data loading efficiency and user interaction experience. The interface implementation effect is shown in Figure below.

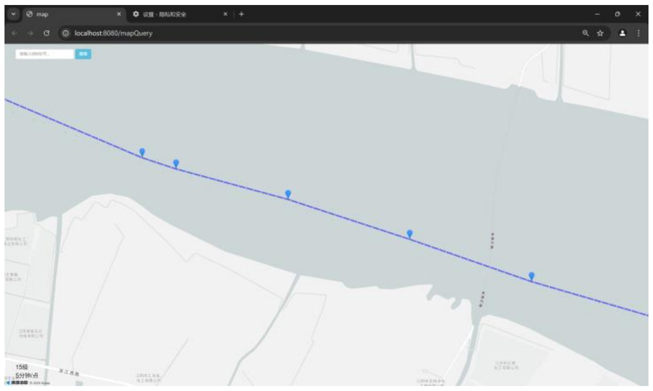

Figure 3: Implementation of the ship trajectory 5-minute/point interface

To support trajectory analysis with varying precision requirements, the system provides a trajectory display function based on a 10-minute time interval. Similar to the 5-minute interface, after the user selects a trajectory interval of 10 minutes on the interface, the system sends a request to the backend through a parameterized interface.

The backend accesses the 10-minute stratified data table stored in MySQL based on the received time interval parameter, and quickly returns the matching trajectory data. The frontend loads the returned data into the map and connects it with polylines to form a trajectory path. It also supports a timeline playback function, facilitating user analysis of ship movement trends.

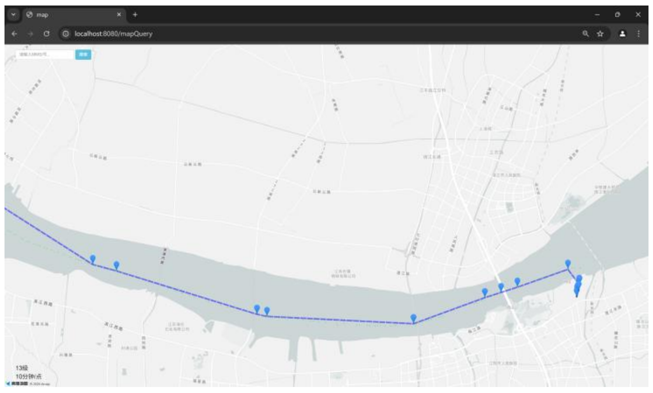

Figure 4: Implementation of the ship trajectory 10-

minute/point interface

The system also implements a trajectory display function based on 30-minute intervals. This function is primarily used to observe the overall trend and directional changes of ships during long-distance voyages, and is suitable for large-scale path evaluation and operational scheduling analysis.

This module is designed to be consistent with the 5-minute and 10-minute interfaces, featuring a unified interaction process and map display logic. Users simply need to switch the time interval within the interface. The system will automatically call the corresponding backend interface and query the distinct_ais_new_30 layered table, returning data that is sparser but covers a wider range.

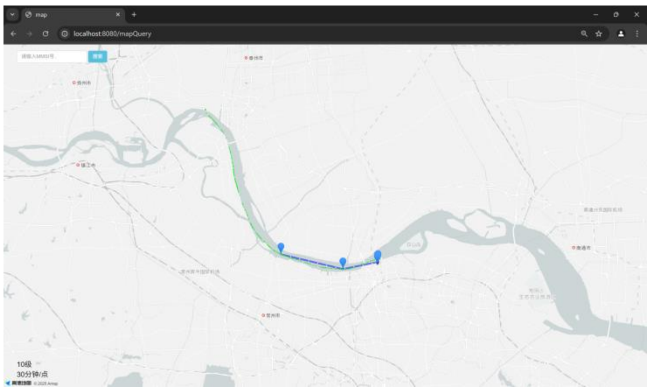

Figure 5: Implementation of the ship trajectory 30-minute/point interface

To meet users' needs for comprehensively grasping the ship's operational trajectory, the system provides a "All Trajectories" display function. Users can enter the ship's identification code (MMSI) and the start and end time to query and view all trajectory points of a specific ship within a designated time period, and the results will be returned in chronological order.

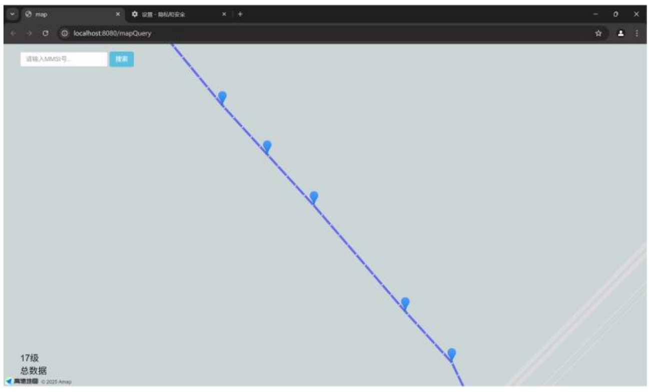

Figure 6: Implementation of the entire interface for ship trajectory

4. Conclusion

This paper systematically explores the design and implementation of a multi-level ship trajectory query system based on Flink, covering aspects such as requirement analysis, technology selection, architecture design, database design, function implementation, and system testing. By utilizing Apache Flink to implement real-time stream processing of large-scale AIS data, combining Spring Boot to build efficient and stable backend services, and employing HTML and JavaScript frameworks to achieve a dynamic interactive interface with front-end and back-end separation, the system successfully demonstrates the ability to dynamically display trajectories at different time intervals based on map zoom levels, meeting the needs of multi-level and multi-granularity trajectory queries.

Although this system has achieved relatively comprehensive functions, there is still room for further improvement. In the future, efforts can be made to enhance algorithm optimization, such as introducing machine learning technology to improve trajectory prediction and anomaly detection capabilities. In terms of system architecture, consideration can be given to splitting into microservices to enhance module independence and elastic expansion. At the same time, the front end can enhance the data interaction experience, supporting more dimensions of trajectory analysis and visualization effects.

Acknowledgements

This work was financially supported by the funding of the Science and technology project of the Jiangsu Maritime Institute (2023ZKyb06), and the Fundamental Computer Education and Teaching Research Project of the Association of Fundamental Computing Education of Chinese Universities (2025-AFCEC-488).

References

[1] Tengesdal, Trym, T. A. Pedersen, and T. A. Johansen. "A Comparative Study of Rapidly-exploring Random Tree Algorithms Applied to Ship Trajectory Planning and Behavior Generation." Journal of Intelligent & Robotic Systems 111.1(2025).
[2] Tengesdal, Trym, T. A. Pedersen, and T. A. Johansen. "A Comparative Study of Rapidly-exploring Random Tree Algorithms Applied to Ship Trajectory Planning and Behavior Generation." (2024).
[3] Chen, Xinqiang, M. Wang, and W. C. Li. "Ship imaging trajectory extraction via an aggregated you only look once (YOLO) model." Engineering Applications of Artificial Intelligence: The International Journal of Intelligent Real-Time Automation 130. Apr. (2024): 107742.1-107742.9.
[4] Xiong, Yong, et al. "Data-driven ship trajectory tracking control method." Chinese Journal of Ship Research 20.1(2025):232-246.
[5] Liu, Zhao, et al. "An online method for ship trajectory compression using AIS data." Navigation News Nov./Dec. (2024).
[6] Gil, Mateusz, et al. "A multiparameter simulation-driven analysis of ship turning trajectory concerning a required number of irregular wave realizations." Ocean Engineering 299(2024).
[7] Pan, Jiale, et al. "A Graph Representation Learning Approach for Imbalanced Ship Type Recognition Using AIS Trajectory Data." Intelligent Transportation Systems, IEEE Transactions on 26.8(2025): 12049-12067.
[8] Jiang, Fan, and H. Yan. "Research on an improved ship trajectory clustering method." Proceedings of SPIE 13018.000(2024):8.