Vol. 16 (2026): Proceedings of the 2026 International Conference on AI and Technological Innovation Applications (AITIA 2026)
E-ISSN: 3105-0522 P-ISSN: 3105-0514
DOI: https://doi.org/10.70267/aitia.20266473

A Review of Distributed Computing Technologies for Financial Data Preprocessing

Zhenze Yang*

Southern University of Science and Technology, Shenzhen 518055, China

**Corresponding author: Zhenze Yang.*

Abstract

Financial data preprocessing is a key link in financial analysis and modeling. With the exponential growth of data scale, the single-machine architecture is facing severe bottlenecks. Distributed computing provides a feasible path to break through performance limitations through multi-node collaborative processing. This article systematically sorts out the research status and technical progress of distributed computing in the field of financial data preprocessing. First of all, analyze the common characteristics and pre-processing task genealogy of financial data, and combine Tang Yao's stock linkage effect research to show the typical process of single-machine preprocessing pipelines and its scale bottlenecks; Then review the mainstream frameworks such as Hadoop, Spark and Flink from the perspective of technological evolution, and summarize the three core empowerment mechanisms of data parallelism, computing parallelism and stream processing; On this basis, classify and summarize the application research of distributed computing in data cleaning, feature engineering, real-time processing and other links, and take Ma Chiyu's financial news sentiment analysis based on SparkR as a case to verify the performance advantages of distributed preprocessing in actual tasks; Finally, we will comment on the limitations of existing research and look forward to the future direction of adaptive preprocessing, explainable attribution, privacy protection calculation, etc. This article aims to provide a systematic reference for distributed computing applications in the field of financial data preprocessing.

Keywords

data preprocessing, financial data, distributed computing, Apache Flink

1. Introduction

Data is the basis of financial analysis and decision-making, but raw data in the financial field can hardly be directly used for modeling - this is not caused by poor data quality, but by the "natural unavailability" state determined by its inherent characteristics. This unavailability is first manifested as the prevalence of noise and abnormal values: In the sales flow of fresh supermarkets and the high-frequency market of the stock market, input errors, sensor failures and human operation errors will introduce anomalous values. If not processed, it will seriously distort the statistical inference and even invalidate the model. Secondly, there is a formal misalignment between the data storage structure and the modeling needs. Financial data is mostly saved in the form of transaction records, and prediction tasks usually require timing samples aggregated by "date-single product" or "securities-time". If the form is not converted, the model cannot be trained. In addition, it is

difficult to directly integrate multi-source heterogeneous data. Fresh pricing needs to be linked to sales details, wholesale prices, loss rate and single product files. Stock forecasting needs to integrate market sequences, fundamental indicators and industry relationship atlas. These data lack a unified structure and granularity, and cannot form a complete analysis view without integration. The non-availability of the above three dimensions makes data preprocessing an insurmountable front-end link in financial modeling.

In terms of distributed computing, Hadoop MapReduce lays the foundation for massive data processing, Spark improves iteration efficiency with memory computing, and Flink compresses the streaming processing delay to milliseconds. For financial scenarios, research has verified the acceleration effect of Spark in retail data cleaning, the performance advantages of distributed frameworks in financial indicator calculation, and the applicability of stream processing in timing preprocessing. Tang Yao built a typical single-machine preprocessing pipeline in the study of stock linkage effects, while Ma Chiyu's financial news sentiment analysis based on SparkR showed a significant performance improvement of distributed schemes. However, there are still three shortcomings in the existing research: There is a lack of systematic review for financial data preprocessing scenarios, and most of the results only focus on isolated technical points; The adaptation mechanism of distributed framework and specific preprocessing methods is still unclear; The landing cases of stream processing in financial real-time pre-processing are still relatively limited. This article tries to make up for the above gaps. On the one hand, it systematically sorts out the application status and empowerment mechanism of distributed computing in financial data preprocessing; On the other hand, the two typical cases of single-machine and distributed are introduced for comparative analysis to verify the synergy between preprocessing method optimization and distributed acceleration. The marginal contribution of this article is to integrate “method improvement” and “distributed acceleration” into the unified analysis framework, which not only provides reprodible preprocessing optimization schemes, but also quantifies the actual performance gains brought by distributed computing.

2. Fundamentals of Distributed Computing Technologies

2.1 Evolution of Distributed Computing Technologies

Distributed computing technology has undergone three major stages of development.

The MapReduce Era (2004–2010). The MapReduce programming model introduced by Google in 2004 abstracted complex parallel computing into two stages—Map and Reduce—thereby significantly reducing the complexity of distributed programming. As an open-source implementation of MapReduce, Hadoop became the de facto standard for early big data processing [1]. The advantages of MapReduce lie in its strong fault tolerance and excellent scalability, yet its drawbacks are equally evident: Intermediate results are frequently dropped to disk, resulting in low efficiency of iterative calculations; the programming model is single and difficult to express complex data flows. The Spark Era (2010–2015). Developed by the AMPLab at UC Berkeley, Spark introduced Resilient Distributed Datasets (RDDs) and in-memory computing mechanisms, boosting the speed of iterative computations by orders of magnitude. RDDs achieve fault tolerance through lineage, avoiding the redundant replication found in Hadoop. The Spark ecosystem (Spark SQL, MLlib, GraphX, and Structured Streaming) has enabled its widespread application in areas such as batch processing, machine learning, graph computing, and stream processing, establishing it as the de facto standard for big data processing [2]. The Era of Stream Processing (2015–Present). Driven by the growing demand for real-time capabilities, true stream processing frameworks such as Apache Flink and Kafka Streams have gradually emerged. Flink employs an operator-based stateful stream architecture that supports event-time processing, exactly-once semantics, and millisecond-level latency, thereby addressing the limitations of Spark Streaming's micro-batch processing [3]. Stream processing technology enables use cases such as real-time risk control, dynamic pricing, and intraday anomaly detection.

The three major frameworks that emerged across these three eras provide a solid theoretical foundation for the practical application of distributed computing; the advantages and limitations of these frameworks are summarized in Table 1.

Table 1: Comparison of Mainstream Distributed Computing Frameworks

Framework	Core Abstraction	Processing Model	Advantages
Hadoop MapReduce	Key–Value (K, V) Pairs	Batch Processing	High stability, low cost, and a mature ecosystem
Spark	RDD/DataFrame	Micro-Batch Processing / Batch Processing	In-memory computing, rich operators, and a comprehensive ecosystem
Flink	DataStream	Native Stream Processing	Millisecond-level latency, state management, and event-time processing

2.2 Core Enabling Mechanisms

2.2.1 Data Parallelism

Data parallelization is to segment large-scale data sets according to specific dimensions and process them in parallel by multiple nodes, which greatly improves the efficiency of cleaning, coding and other tasks. Take the mutual information calculation of large-scale classified data as an example, traditional serial computing takes a very long time. Spark divides the data set into multiple subsets through column transformation, distributes it to each node of the cluster for parallel calculation, and uses two variable-length arrays to cache intermediate results. Finally, it achieves high scalability and operating efficiency on the cluster of 24 computing nodes. This effectively reduces the cost of data communication between nodes compared with traditional methods.

2.2.2 Computational Parallelism

Computational parallel is to decompose computing tasks into multiple sub-tasks and assign them to different nodes for parallel execution. In financial preprocessing, computational parallelity also plays a great role. Take FP-Growth [8] as an example, when the amount of data exceeds the memory capacity in a single-machine environment, the FP tree cannot be fully constructed. The core idea of distributed implementation is to partition the transaction data set by time or user, and each node independently builds a local FP tree and mines the local frequent item set; By summarizing the local frequent item sets of each node, the global frequent item set is obtained. The key to this scheme is to design a reasonable partition strategy to balance the local tree scale of each node and avoid a single node becoming a bottleneck. In practical applications, random disruption or range partitioning can be introduced to achieve load balancing.

2.2.3 Stream Processing

Stream processing regards data as a continuously arriving boundless flow and achieves millisecond-level response, which is suitable for real-time monitoring, online detection and other scenarios. As a mainstream stream processing engine, Flink realizes fault tolerance through a distributed snapshot mechanism and guarantees exactly-once processing semantics. Take the real-time reminder of taxi fatigue driving as an example, based on the distributed real-time data stream processing architecture built by Flink, it can perform millisecond-level analysis and status monitoring of vehicle sensors and GPS data flow. Once an abnormal driving mode is detected, it will immediately trigger an early warning to ensure driving safety.

3. Core Tasks and Challenges in Financial Data Preprocessing

3.1 Common Characteristics of Financial Data

Financial raw data can rarely be directly used for modeling. This stems from three inherent attributes. First, noise and abnormal values are common. Input errors, sensor failures, and human operation errors will introduce anomalous values, which will seriously distort statistical inference without processing. Second, the storage form and modeling requirements are misaligned. The data is mostly saved as a transaction record, and the prediction task requires the timing sample aggregated by “date-target”. Third, it is difficult to integrate multi-source heterogenes. Stock analysis needs to integrate market sequences, fundamental indicators and news

public opinion at the same time. These data lack a unified structure and granularity. The above characteristics determine that preprocessing is an indispensable pre-link in financial modeling.

3.2 Typical Preprocessing Tasks

Financial data preprocessing follows the progressive vein of "integration → cleaning → feature engineering". In the multi-source integration stage, with the securities code and the transaction date as the key, the market, finance, public opinion and other heterogeneous tables are connected into an analytical wide table. During this period, it is necessary to unify the time granularity and perform categorical encoding. Processing missing values and abnormal values in the cleaning stage: choose forward filling or mean interpolation for missing values according to the notch span; The distribution characteristics of abnormal values are identified by quartile distance truncation or Isolation Forest algorithm [4], supplemented by robust standardization to mitigate the interference of the outliers. Time-series aggregation compresses stroke-by-stroke or minute-level data into daily frequency indicators, and the window division needs to be matched with business logic. Feature engineering calculates the moving average, momentum index and lag term, and through the filtering method [5], the frequency domain component [6] is extracted through wavelet transformation if necessary, and redundant variables are eliminated. Unstructured text is aligned with the subject on the timeline through word division and vectorization, forming event-driven characteristics.

The above process is directly applied in financial empirical research. In the study of stock linkage effect, Tang Yao [7] carried out single-machine serial preprocessing of the text (plate classification, shareholder information) and time-series data (14-dimensional market field) of nearly 2,000 stocks in Shenzhen. After being parsed by the crawler, the text is stored in key-value pairs. The timing sequence is aligned with the deep proof index as the benchmark and filled in the missing value, and finally spliced into a multi-dimensional sample set. The method runs normally at a scale of 300 trading days, providing a clean data basis for subsequent linkage discrimination.

3.3 Nature of Large-Scale Processing Challenges

When the data volume expands from research samples to the whole history of the whole market, the single-machine preprocessing architecture faces a triple bottleneck. In terms of storage, TB-level data far exceeds the memory capacity of a single machine; In terms of calculation, the time complexity of time-series aggregation and sliding window operation increases linearly with the amount of data; On I/O, the disk reading and writing speed lags behind the CPU, and the resource utilization rate is low. These bottlenecks mean that the migration of preprocessing pipelines from single-machine serial to distributed parallel is a necessary path to deal with the scaling of financial data. Chapter 4 will elaborate on how the distributed computing framework can break through the above limitations.

4. Applications of Distributed Computing in Financial Data Preprocessing

4.1 Distributed Acceleration of Data Cleaning and Integration

Data cleaning is the most time-consuming part of preprocessing, including format conversion, field extraction, abnormal value filtering, etc. The distributed framework significantly improves efficiency through data parallel: large-scale data is stored in HDFS by time dimension, and each node processes local data in parallel. When the task fails, only a single partition needs to be recalculated. Spark SQL provides a declarative interface where users express data cleaning logic through operations such as filter, withColumn, and dropDuplicates, while the system automatically optimizes the execution plan. Multi-source integration involves large table connection. Spark optimizes performance through two strategies: partition connection and broadcast connection - small table broadcasts to all nodes to avoid redistribution, and large tables are executed in parallel after partitioning by pressing the connection key.

4.2 Distributed Acceleration of Feature Engineering

Feature engineering is the link with the highest calculation density in the pretreatment pipeline. In terms of time-series aggregation, the Spark window function declares the combination of groupBy and agg as computational logic. If the data is partitioned according to the grouping key, each node completes local

aggregation independently without cross-node mixing. The sliding window feature defines the window boundary through rowsBetween, and the internal increment of each partition calculates the local results. In the feature selection link, the filter method calculates the information gain or variance threshold in parallel according to the column, and each node calculates the local statistics and generates the global screening standard.

4.3 Stream Processing for Real-Time Data Preprocessing

In scenarios such as real-time risk control and high-frequency trading, preprocessing needs to be completed at the millisecond level. Flink maintains an independent state for each account through KeyedProcessFunction, and abnormal rules are instantly evaluated in the stream and triggers alarms. Dynamic indicator calculation relies on event time processing and sliding window incremental update - the aggregate value is updated when new data arrives, and the old data is returned when it leaves. The RocksDB state backend can support the large-scale status storage of thousands of stock indicators in the whole market. Online standardization is by maintaining the mean and standard deviation estimates of the sliding window, and the estimated value is standardized and updated as soon as each data arrives.

4.4 Practical Cases of Distributed Data Preprocessing

Ma Chiyu [1] constructed a distributed preprocessing pipeline for financial news sentiment analysis based on the SparkR platform, covering four stages: web scraping, text tokenization, feature selection, and sentiment classification. In the crawler stage, the URL to be crawled is initialized into an RDD for parallel downloading, and the title, time and text are extracted regularly; In the word segmentation stage, a custom function is used to match terms against a sentiment lexicon, and reduceByKey is employed to calculate word frequencies; in the classification stage, a Random Forest algorithm is used for distributed training on a five-node cluster.

The performance data for this scheme are shown in Table 2. When processing 100,000 financial news items, a single machine took 432 seconds, whereas a cluster required only 20 seconds—a 21-fold speedup. When the data volume increased to 200,000 records, a single machine could no longer handle the task, whereas a cluster took 39 seconds. In the scalability test, increasing the number of nodes from 3 to 5 reduced the time required for the same task from 47 seconds to 39 seconds, representing an efficiency gain of approximately 18%.

Table 2: Performance Evaluation of SparkR-Based Distributed Data Preprocessing

Test Item	Data Volume	Single-Machine Execution Time	Cluster Execution Time	Speedup
Random Forest Classification	100,000 Records	432 s	20 s (5 nodes)	21 x
Random Forest Classification	200,000 Records	Unable to Complete	39 s (5 nodes)	—
Scalability Test	200,000 Records	—	47 s → 39 s (3 → 5 nodes)	18%

4.5 Empirical Evidence of Distributed Acceleration Performance

The preceding analysis theoretically explains the principles by which distributed computing accelerates financial data preprocessing. In practical applications, existing studies have experimentally verified its acceleration effect. As reported in references [8-10], distributed computing has yielded significant performance improvements in stages such as data cleaning, feature engineering, and stream processing; specific results are presented in Table 3.

Table 3: Summary of Empirical Evidence on the Acceleration Effects of Distributed Computing

Result	Source	Experimental Scenario	Method	Performance Improvement
Accelerated Data Cleaning	Huang Liming [8]	Massive Data Analysis Tasks	DNN	Average processing latency reduced from 320 ms to 180 ms
Accelerated Feature Engineering	Shi Sheng [9]	Cluster-Based Parallel Computing Experiments	Spark	Achieved a 5.327-fold increase in computing capacity and a 2.098-fold speedup under workloads exceeding the performance limits of a single machine
Reduced Stream Processing Latency	Li Ya [10]	Smart Grid Big Data Analytics	Storm	Single-node throughput reached 23,532 records/s. Cluster throughput increased nearly linearly with the number of nodes, while security-level

				classification accuracy remained approximately 93.4%

4.6 Recommendations for Framework Selection

Financial data preprocessing tasks are diverse; framework selection must take into account task characteristics and latency requirements. Spark is recommended for batch processing historical data; its in-memory computing capabilities enable efficient handling of terabyte-scale data, while the DataFrame API reduces development costs. Flink is recommended for real-time stream processing; its true stream processing architecture ensures millisecond-level latency, while event-time processing accommodates the out-of-order nature of financial data. For massive data storage, HDFS or cloud object storage is recommended due to low costs and high scalability; for hybrid scenarios, Spark Structured Streaming or Flink's batch processing mode can be considered to achieve a unified framework for handling both batch and stream processing.

5. Case Study: Distributed Data Preprocessing for Fresh Food Retail Data

5.1 Scenario Description and Data Characteristics

The data of this case comes from Question C of the 2023 National College Student Mathematical Modeling Competition "Automatic Pricing and Replenishment Decisions for Vegetable Products" [11]. The data contains 251 single products, which are divided into 6 categories of flowers and leaves, cauliflower, aquatic rhizomes, eggplants, peppers and edible fungi, with a total sales record of 1095 days. There are the following quality problems in the original data: some records have an abnormal value with a markup rates as high as 200%; some single product sales records are intermittently missing; some single product sales days are very few and the sales volume is extremely low; the sales volume is significantly affected by seasons and holidays.

5.2 Improved Preprocessing Methods

In the practice of data preprocessing for fresh food supermarkets, and based on the aforementioned analysis of preprocessing methods, this paper proposes two improvements that effectively enhance data quality.

Classification of individual items based on dual indicators. The original classification relied solely on the single metric of "days of supply"; this paper proposes a dual-metric classification method based on supply continuity and seasonal concentration. The supply continuity indicator is defined as $S_c = \frac{\text{Total supply days over three years}}{1095}$, and the quarterly concentration indicator is defined as $S_s = \frac{\max_k N_k}{\sum_k N_k}$ ($where\ N_k$ represents the number of supply days in each quarter). Classification rules: Year-round vegetables ($S_c > 0.8$), seasonal vegetables ($S_c < 0.05$), and seasonal vegetables ($0.05 \leq S_c \leq 0.8$), with seasonal characteristics verified using S_s. The classification results are highly consistent with real-world observations. For example, "Sichuan Chinese toon" was correctly classified as a seasonal spring vegetable.

Two-stage anomaly detection using IQR and SVR. The original 3σ criterion is applicable only to univariate normal distributions and cannot detect anomalies arising from non-linear relationships between sales volume and pricing. Inspired by Lv Daohuan et al. [11], the IQR method was employed for the first-stage screening of the markup rates—which exhibited a skewed distribution—to exclude univariate extreme outliers (markup rates >120%). The second stage employs Support Vector Regression (SVR) to capture outliers within nonlinear relationships. An SVR model (RBF kernel, $C = 1.0, \varepsilon = 0.1$) was constructed using historical sales, pricing, and holiday characteristics as inputs and actual sales as the output; prediction residuals were calculated, and samples with residuals exceeding the 95th percentile were identified as outliers.

Sales and pricing data for foliage plants were selected for the comparative experiment, and the results are presented in Table 4. The IQR+SVR scheme reduced the misclassification rate from 0.8% to 0.3% and increased the model's R^2 from 0.83 to 0.89 after data removal, thereby validating the effectiveness of the improved method.

Table 4: Comparison of Different Outlier Detection Strategies

Method	Original R^2	False Positive Rate	R^2 *After Outlier Removal*
Linear Regression $+3\sigma$	0.68	0.8%	0.83
IQR+SVR	0.79	0.3%	0.89

5.3 Implementation and Performance Analysis of Distributed Acceleration

5.3.1 Distributed Architecture Design

For TB-level data processing needs, Spark clusters can be used to realize distributed preprocessing. For example, partition the data according to the store ID and date. Each node processes local data in parallel to localize data, avoid cross-node data transmission, and realize data parallel. Next, through the groupBy+agg operation of Spark, the time-series aggregation is realized, and each group is executed in parallel. Then divide the transaction data by date, explore the correlation of various data, and then combine the global results to realize the calculation parallel. Finally, in the real-time sales monitoring scenario, the Flink stream processing architecture can be used to access the sales flow in real time, dynamically update the single product sales statistics and abnormal detection model, and upgrade from "T+1 batch processing" to "second-level response".

5.3.2 Performance Analysis

According to Amdahl's Law, the theoretical acceleration ratio of parallel systems is limited to the parts of the program that must be executed serially. Let p be the proportion of the program's code that can be parallelized, and $1-p$ be the proportion of the serial part; then the theoretical speedup S using N processing nodes is:

$$S = \frac{1}{(1-p) + \frac{p}{N}}.$$

In this case study of fresh food supermarket data preprocessing, different types of operations have different parallel characteristics. First, there are data cleaning operations, such as field parsing, missing value imputation, and univariate anomaly detection. These operations are processed independently on a per-row basis and do not rely on data across rows; theoretically, complete parallelism ($p \to 1$) is achievable. The speedup of such operations is primarily limited by the balance of task distribution among nodes. Secondly, there's the time-series aggregation operation, which involves grouping and statistically analyzing data by store, individual product, and date. By employing appropriate partitioning (such as partitioning by store ID), each node can independently perform local data aggregation, requiring only minimal serial operations during the final result merging stage. Operations of this type exhibit a very high degree of parallelism, yet the global merging of final results creates a serial bottleneck. Association Rule Mining (FP-Growth): The distributed implementation partitions data by date; each node independently constructs a local FP-tree and mines local frequent itemsets, after which the global results are merged. The local mining phase can be fully parallelized, whereas the global merging phase requires aggregating frequent itemsets from all nodes, entailing a serial dependency.

Based on the above analysis, the parallelization ratio p for each preprocessing task in this case is high; however, serial stages—such as data merging and global synchronization—still exist. As the scale of data grows (e.g., expanding from a single store to a thousand stores), distributed computing distributes the parallel computational load by increasing the number of nodes (N); however, the presence of a serial component imposes a theoretical upper limit on the speedup ratio. Therefore, in the design of distributed preprocessing systems, identifying and optimizing serial bottlenecks is key to improving overall efficiency.

5.3.3 Summary of the Case Study

This case shows the collaborative application of distributed computing and improved preprocessing methods in the fresh supermarket scenario. The improved method (dual-indicator classification, IQR+SVR two-stage abnormal detection) has effectively improved the data quality; Distributed computing can theoretically compress the processing time of TB-level data from tens of hours to several hours through data parallel and computing parallel; The improvement of preprocessing methods and distributed acceleration can be applied together to support large-scale financial data analysis.

6. Challenges and Future Prospects

6.1 Limitations of Existing Studies

Although distributed computing has made significant progress in the field of financial data preprocessing, the existing research is still insufficient. It is reflected that the existing research focuses on the performance

test of a single framework or a single preprocessing link. The evaluation indicators are not uniform, the experimental designs are different, and it is difficult to compare different methods horizontally, thus lacking a systematic evaluation system. Moreover, the lack of standardized evaluation data sets and benchmark tasks for the characteristics of financial data restricts technical selection and academic progress.

The adaptation of distributed and preprocessing methods is insufficient, and the parallel mechanism of the distributed framework and the adaptation of specific preprocessing methods have not been deeply discussed. For example, the load balancing strategy of FP-Growth algorithm in a distributed environment, the optimization of the partition strategy of time-series aggregation tasks, and the distributed implementation of anomaly detection algorithms lack systematic theoretical analysis and experimental verification.

6.2 Future Research Directions

Adaptive preprocessing. The selection and parameter setting of the current preprocessing method are highly dependent on manual experience. In the future, the system can introduce a meta-learning mechanism to automatically identify data distribution characteristics (deflection, peak, missing mode, timing mode), recommend the best preprocessing combination from the method library, and dynamically adjust parameters through downstream task feedback. Adaptability will reduce labor costs and improve the quality of preprocessing.

Explainable attribution. The interpretability of preprocessing results will become a research hotspot. In the abnormal detection, not only the “abnormal” label should be output, but also the attribution analysis should be given: which characteristics lead to abnormal determination? Is it a data collection error or a real event? Combined with external knowledge atlases (such as financial report disclosure calendars and policy release timelines), abnormal points can be associated with external events, data errors can be distinguished from real market movements, and analysts can be provided with an actionable basis for decision-making.

Privacy protection calculation. In cross-agency joint analysis scenarios, data cannot be stored centrally. Federal learning allows participants to complete preprocessing locally, exchange only encrypted intermediate statistics, and realize “data immobility model movement”. Differential privacy technology can add noise to the preprocessing output to protect the privacy of individual data. The combination of these technologies and distributed computing will promote cross-institutional coordination in financial risk control, anti-fraud and other scenarios.

Cloud-native distributed computing. Cloud native architecture (containerization, service grid, Serverless) is reshaping the infrastructure of distributed computing. In financial data preprocessing, cloud native capabilities can be realized: flexible expansion of resources, dynamic adjustment of computing resources according to the amount of data; Multi-tenant isolation ensures the data security of different business lines; automation of operation and maintenance reduces cluster management costs. In the future, cloud native will promote the evolution of distributed computing from “moving to the cloud” to “born in the cloud”.

7. Conclusions

This article systematically reviews the research status and application progress of distributed computing in the field of financial data preprocessing. Financial data has common characteristics such as high dimension, strong timing, high noise, multi-source heterogeneity, etc., which determines that its preprocessing tasks cover data integration, missing processing, abnormal detection, time-series aggregation, feature engineering and other links. When the data scale expands from research samples to the whole history of the whole market, the triple bottlenecks of storage, computing and I/O are increasingly prominent, and the single-machine architecture is difficult to compete. Distributed computing technology has gone through three main stages of development: MapReduce, Spark and Flink. Data parallelism, computing parallelism and stream processing constitute its core empowerment mechanism. Spark is recommended for batch processing scenarios, Flink is recommended for real-time streaming processing scenarios, and mass storage relies on HDFS or cloud object storage. Huang Liming, Shi Sheng, Li Ya and other studies have verified the actual effect of distributed acceleration in data cleaning, cluster parallel and stream processing scenarios respectively. At the specific practical level, Tang Yao's stock linkage effect research shows a typical single-machine preprocessing pipeline: In terms of text, it crawls plate classification and shareholder information and analyzes and stores it. In terms

of timing, it is aligned and filled and spliced into a multi-dimensional sample set. The pipeline can be operated in fashion to process medium-scale data, but its expansion capacity is limited. In contrast, Ma Chiyu's financial news sentiment analysis based on SparkR shows the performance advantages of distributed preprocessing - the processing time of 100,000 news is compressed from 432 seconds on a single machine to 20 seconds in a cluster, accelerating by 21 times; When the amount of data increases to 200,000, the single machine can no longer be completed, and the cluster only takes 39 seconds. This comparison intuitively illustrates the necessity of migration from single-machine serial to distributed parallel migration. At the level of method improvement, strategies such as dual-index classification and IQR+SVR two-stage abnormal detection can effectively improve data quality and cooperate with distributed accelerated applications to provide strong support for large-scale financial data analysis.

At present, the field of financial data preprocessing is still facing a number of challenges. The evaluation system has not been unified. The adaptation mechanism of distributed frameworks and specific preprocessing methods needs to be analyzed in depth. The landing cases of stream processing in financial real-time preprocessing are still relatively limited.

Looking to the future, adaptive preprocessing, interpretable attribution, privacy protection computing and cloud-native architecture will become important development directions. With the continuous growth of financial data and the continuous improvement of real-time requirements, the role of distributed computing in financial data preprocessing will become more and more important. We look forward to the academic and industrial circles working together to jointly promote technological innovation and engineering implementation in this field.

References

[1] Ma, C. Y. (2016). *Research on sentiment analysis of online financial information and its relationship with stock market volatility: An implementation based on SparkR* (Master's thesis, Hefei University of Technology, Hefei, China).

[2] Li, Y. (2017). *Research on an online big data analysis and decision-making system for smart grids* (Master's thesis, North China Electric Power University, Beijing, China).

[3] Han, J., & Kamber, M. (2012). *Data mining: Concepts and techniques* (pp. 256–267). Beijing, China: China Machine Press.

[4] Chen, Z. Y., & Liu, Y. (2021). Anomaly detection and root cause analysis for financial data. *Software, 42*(7), 119–123.

[5] Li, Z. (2023). *Research on feature selection methods for continuous data and their applications* (Master's thesis, China University of Petroleum–Beijing, Beijing, China).

[6] Xiong, Z. B. (2005). *Wavelet denoising preprocessing of financial time series* (Master's thesis, Renmin University of China, Beijing, China).

[7] Tang, Y. (2017). *Research on stock linkage effects based on heterogeneous information processing* (Master's thesis, Harbin Institute of Technology, Harbin, China).

[8] Huang, L. M. (2024). Research on massive data analysis methods based on machine learning. *China Computer & Communication (Theory Edition), 36*(1), 84–86.

[9] Shi, S. (2021). *Research on Spark-based parallel algorithms for finite element clusters* (Master's thesis, Hunan University, Changsha, China). https://doi.org/10.27135/d.cnki.ghudu.2021.003799

[10] National Undergraduate Mathematical Contest in Modeling Organizing Committee. (2023). *2023 Higher Education Press Cup National Undergraduate Mathematical Contest in Modeling, Problem C: Automatic pricing and replenishment decision-making for vegetable commodities* [Competition problem].

[11] Lv, D. H., Tao, Y. Q., Yu, Y. L., et al. (2024). Research on preprocessing methods for satellite on-orbit experimental data. *Computer Simulation, 41*(5), 62–67.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

This paper is an output of the science project.