Time Series Reconstruction Methods for Massive Star Catalog Data Based on the Spark Distributed Framework: A Postprint

Authors: Qing Zhao, Quan Wenli, Chen Yarui, Cui Chenzhou, Fan Dongwei

Date: 2024-03-26T00:00:00+00:00

Abstract

Time series reconstruction constitutes a crucial data processing step in time-domain astronomy and serves as the foundation for light curve fitting and time-domain analysis research. In MapReduce distributed models such as Hadoop and Spark, tasks among nodes in distributed clusters are relatively independent during execution, necessitating minimal cross-node data transmission. This work proposes a non-blocking asynchronous execution pipeline wherein each distributed process continuously handles data from independent sky regions, while identification tasks for newly added celestial objects at block boundaries that affect other nodes are appended in delayed batches. The appending methodology is dynamically determined based on the varying progress across processes to ensure complete identification computation, thereby enhancing concurrent efficiency while preserving algorithmic precision. Additionally, diverse Join strategies between two tables are investigated from both theoretical and experimental perspectives, culminating in the proposal of a Join-free strategy. Finally, the aforementioned research is validated through the design of an efficient time series reconstruction system built upon the Spark distributed framework. Experimental results demonstrate significant efficiency improvements over previous approaches, establishing a robust foundation for astronomical time series data analysis in time-domain astronomy.

Full Text

Research on Time Series Reconstruction Method of Massive Astronomical Catalogues Based on Spark Distributed Framework

ZHAO Qing[1], QUAN Wen-li[1], CHEN Ya-rui[1;2], CUI Chen-zhou[3], FAN Dong-wei[3]

[1]College of Artificial Intelligence, Tianjin University of Science and Technology, Tianjin 300457, China
[2]Engineering Research Center for Integration and Application of E-Learning Technology, Ministry of Education, Beijing 100039, China
[3]National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100101, China

Abstract

Time series reconstruction is a crucial data processing step in time-domain astronomy and serves as the foundation for fitting light curves and conducting time-domain analysis. For many large-field time-domain surveys, it is necessary to complete this computational process within a single exposure cycle. With the rapid increase in astronomical data, existing methods for astronomical data processing struggle to simultaneously meet the accuracy and efficiency requirements of time-series reconstruction. The memory-based computing general-purpose distributed framework, Spark, holds the potential to improve the efficiency of this process. However, applying Spark directly often encounters issues. MapReduce distributed models like Hadoop and Spark require relatively independent tasks among distributed cluster nodes and minimal data transfer across nodes during execution. Otherwise, frequent communication becomes an efficiency bottleneck for the application of the model. However, due to the presence of boundary problems in cross-matching, it is inevitable to transmit newly added data at the boundaries, severely restricting the concurrency of the model and reducing the acceleration ratio in practical parallel model applications. Therefore, we propose a non-blocking asynchronous execution flow, where each distributed process handles continuous processing exclusively for independent sky regions. The delayed batch appending of additional identification tasks from block-edge newly added celestial bodies in other nodes is determined based on the progress of each process. This ensures that identification calculations are not omitted, thereby improving concurrent efficiency while maintaining algorithm accuracy. Additionally, a research study was conducted on different join strategies between two tables, examining them from both theoretical and experimental perspectives. Furthermore, a join-free strategy was proposed. Finally, the design of an efficient time-series reconstruction system based on the Spark distributed framework validates the aforementioned research. Experimental results demonstrate a significant improvement in the efficiency of the proposed time-series reconstruction algorithm compared to previous research, laying a solid foundation for the analysis of astronomical time-series data in time-domain astronomy.

Keywords: time domain astronomy; cross-match calculation; time series reconstruction; distributed computation; Spark

1 Introduction

In time-domain astronomy, a fundamental task is to perform cross-identification on continuous multiple observations from the same telescope to determine the correspondence of each celestial object across different observation records, thereby generating time-series data. This process often involves a contradiction between the massive data volume and high-speed requirements. Taking the Ground-based Wide-Angle Cameras (GWAC) as an example, this data processing must be completed within a single exposure cycle. Therefore, designing and developing efficient time-series data generation methods for large-field time-domain survey observations is of great significance.

In recent years, advances in time-domain observation technology have provided extremely rich data resources for time-domain astronomy research. For instance, Gaia DR3 released astrophysical parameters for 1.59 billion sources and 470 million celestial objects collected over 34 months. GWAC generates 1.1 GB of data every 15 seconds, with star catalog data reaching up to 2 TB per observation night. The Large Synoptic Survey Telescope (LSST) produces 15 TB of data per night, with a 10-year operation expected to collect 60 PB of data. Faced with such vast accumulated data and newly archived data generated at high speed daily, traditional cross-identification algorithms struggle to meet both accuracy and efficiency requirements simultaneously.

First, the boundary source leakage problem after data partitioning affects both accuracy and efficiency. Current main solutions include: Zhao et al. added additional edge data to each star catalog to solve this problem, but the implementation is complex and inefficient. Du et al. and Yu et al. adopted a hybrid identification calculation method based on HTM and HEALPix indexing, which reduces cross-boundary data transmission but significantly increases storage and computation overhead, while still having a certain proportion of source leakage and accuracy loss. Therefore, the edge source leakage problem needs further optimization.

Second, time-series identification requires cross-identification calculations among multiple continuous observation star catalogs, resulting in higher computational complexity than traditional multi-band cross-identification, necessitating further research on parallel computing frameworks. Du et al. proposed a multi-threaded identification algorithm to accelerate the computation process. Zhao et al. proposed a multi-core parallel cross-identification algorithm capable of cross-identifying star catalogs with over 100 million records and 470 million records. However, since these methods were not executed in large-scale distributed environments, they still have certain performance limitations for time-series reconstruction needs among multiple star catalogs. In studies of time-series for homologous star catalogs, Yu et al. implemented time-series reconstruction for homologous star catalogs in an MPI parallel environment, achieving parallel execution with up to 6 processes. Although identification efficiency was improved, this parallel environment is not easily scalable and

struggles to cope with further expanding data volumes. Xu et al. proposed a fast time-series reconstruction algorithm based on MongoDB, which has certain reference value for non-relational data selection in time-series reconstruction problems but lacks sufficient integration with large-scale distributed frameworks.

Therefore, for time-series reconstruction of large-scale homologous star catalogs, it is necessary to introduce new parallel frameworks to further improve efficiency while ensuring accuracy. Cloud computing represented by the MapReduce ecosystem such as Hadoop and Spark has gradually become an important solution for big data processing challenges. Among them, Spark, as a general-purpose distributed framework based on in-memory computing, has greater efficiency advantages. In recent years, it has been applied in astronomical information processing. For example, Brahem et al. utilized Spark' s efficient data processing characteristics to propose an astronomical data query processing method. Zečević et al. proposed a scalable astronomical analysis framework based on Spark. Liu et al. combined MapReduce/Spark to propose an unsupervised clustering-based pulsar candidate screening scheme. It is evident that adopting distributed computing technology has become a development trend for astronomical research under massive data.

However, direct application of Spark is not suitable for this problem. MapReduce distributed models like Hadoop and Spark require relatively independent tasks among multiple processes during execution; otherwise, frequent inter-node communication becomes an efficiency bottleneck for model application. However, due to the classic boundary source leakage problem in cross-identification, data transmission for newly added edge data cannot be avoided, which severely reduces the actual acceleration ratio of this model in practical applications. Therefore, we designed a non-blocking asynchronous execution flow where each distributed process completely focuses on continuous processing of data from independent sky regions. Additional identification tasks caused by newly added celestial bodies at block edges are delayed and appended in batches based on the different progress among processes, ensuring no omission in identification calculations while significantly improving concurrent efficiency.

Furthermore, when transforming cross-identification calculations into Join operations between two tables, the choice of optimization strategy is also critical to efficiency. Considering these issues, this paper designs a distributed time-series reconstruction algorithm suitable for massive star catalogs, focusing on three key optimizations: (1) For the boundary source leakage problem, a non-blocking asynchronous execution flow suitable for Spark architecture is designed, reducing data transmission while ensuring accuracy and effectively improving algorithm efficiency; (2) Time-series identification is transformed into Join operations between large and small tables, comparing two different Join implementation strategies and a Join-free operation strategy, providing applicable scenarios for different algorithms; (3) The research content is verified through the design and implementation of a time-series identification system based on the Spark

distributed framework.

2 Design of Asynchronous Non-Blocking Multi-Process Execution Mode and Distributed Update Synchronization

After distributed data partitioning of star catalogs, a boundary source leakage problem occurs: due to positional errors, some celestial objects at partition boundaries may cross boundaries in another observation and appear in another block. Therefore, edge data requires additional processing to improve matching accuracy. In time-series reconstruction, cross-identification occurs between continuous observation star catalogs, and newly added celestial objects at block edges require cross-node synchronous updates. If multi-nodes in the distributed architecture operate in blocking mode, global updates for newly added celestial objects can be guaranteed, but this means that calculations for all celestial objects in each observation record must be completed before the next observation data can be processed, which constrains the efficiency advantages of the distributed framework and leads to waste of computing resources.

To address these problems, the asynchronous non-blocking multi-process execution mode proposed in this paper can significantly improve efficiency and concurrent processing capability. The so-called non-blocking mode means that when newly added celestial objects at edges are updated to adjacent blocks, the adjacent blocks will not stop their current batch processing to handle the new data. Instead, they continue with batch identification calculations for star catalogs, with each process being independent in processing progress, allowing each process to efficiently process subsequent data within the block one by one. Identification omissions caused by asynchronous progress among processes will be centrally appended after batch identification calculations are completed. This non-blocking multi-process mode can maximize the utilization of computing resources at each node and achieve high-efficiency parallel processing.

This paper uses the earliest star catalog on the identification timeline as the reference catalog. Each celestial object in subsequent observation records will undergo pairwise spherical distance calculation with celestial objects in the same partition of the reference catalog. The calculation formula is:

$$d = \arccos[\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos(\lambda_1 - \lambda_2)]$$

where (λ_1, ϕ_1) are the coordinates of celestial object A and (λ_2, ϕ_2) are the coordinates of celestial object B. When the angular distance between two celestial objects is very small, the angular distance formula can be approximated as:

$$d^2 \approx [(\lambda_1 - \lambda_2)\cos\phi]^2 + (\phi_1 - \phi_2)^2$$

Due to certain error radii, assuming the error radii of the two star catalogs are r_1 and r_2, two celestial objects can be identified as the same if they satisfy the

following relationship:

$$d \leq |r_1| + |r_2|$$

Since different observation star catalogs are all identified with the reference catalog, only the reference catalog requires redundant storage of edge replicas, reducing the amount of replica storage data. A small number of newly added celestial objects during the identification process will be appended to the reference catalog, and if they are at block edges, they need to be simultaneously updated in the reference catalogs of multiple computing nodes.

Due to the designed non-blocking multi-process computing mode, each distributed process completely focuses on continuous processing of data from independent sky regions. When newly added celestial objects at block edges are synchronized across nodes, if identification calculations are missed due to progress being ahead, they will be compensated in batches later. This makes the system more concurrent, with higher computing resource utilization, making it more suitable for time-series reconstruction calculations of massive star catalogs.

The specific update flow diagram is shown in Figure 1 [Figure 1: see original paper]. Since identification computing tasks are executed asynchronously, the identification progress of the current block and adjacent blocks is not synchronized. The solution to the specific update synchronization problem for newly added celestial objects is:

Figure 1 Reference catalog edge celestial object cross-node update flow diagram

(1) **Adjacent block identification progress is ahead.** In this case, one possibility needs to be considered: if edge celestial object X is a supernova, its first appearance might be in this adjacent block. Before X is updated to the adjacent block' s reference catalog, it may already exist. As shown in Figure 2 [Figure 2: see original paper], a new celestial object X appears in block A on the 15th. Because X is at the edge between blocks A and B, it is added to block A' s reference catalog while being sent to the node where block B resides. If X first appears in block B on the 10th and has already been sent to A, then block A will not have new celestial object X on the 15th. Due to transmission delay, X was not updated to block A' s reference catalog in time, so it is discovered again, while block B already had celestial object X before the 15th. Therefore, when the node where block B resides receives celestial object X, it only needs to identify it with newly added celestial object data from the 11th to the 25th in block B' s reference catalog: if identification succeeds, discard X because it was previously discovered in B; if identification fails, add X to block B' s reference catalog and identify it with all celestial objects in block B from the 11th to the 25th.

Figure 2 Schematic diagram of edge update when adjacent block identification progress is ahead

(2) **Adjacent block identification progress is behind.** Add edge celestial objects to the adjacent block' s reference catalog and identify them with subsequent celestial objects based on observation time. If the observation time of subsequent star catalog objects is earlier than that of the celestial object, identification is not performed.

3 Comparison of Different Join Strategies Between Two Tables

To achieve distance calculation and threshold judgment for star pairs under the same block number, this paper considers two methods: one uses Join operations in the database, while the other directly employs HDFS file data and operator operations in Spark without using Join. This chapter details the Join-based method, while the non-Join method is described in Section 4.2.

To implement distance calculation and threshold filtering, non-equivalent Join is adopted, where the Join condition is that the spherical distance between a record in the reference catalog and a record in the catalog to be identified is less than the threshold, i.e., formulas (1)-(5). Successfully joined results are identified record pairs. To improve processing performance, Hash Join is used, mapping Join field values to buckets in a hash table to reduce reshuffle or reordering, which has good adaptability for large-scale data.

Considering that reference catalog data needs to be identified one-to-one with other star catalogs on the continuous time axis, the reference catalog data is treated as the small table, while other star catalogs on the continuous time axis are integrated as the large table. To further improve performance, this paper experimentally compares two hash join strategies suitable for small and large tables: Shuffle Hash Join (SHJ) and Broadcast Hash Join (BHJ).

SHJ implementation repartitions both the large and small tables according to the join key using the HashPartitioner method, i.e., the Shuffle operation. The purpose is to place data with the same key from both tables in the same partition, then perform join operations on data within the same partition.

BHJ implementation broadcasts the small table data to other nodes in Spark, then independently performs join operations on each node, maximizing CPU resources dedicated to join operations. This Join strategy sacrifices node space to some extent to avoid time-consuming Shuffle operations, but broadcasting a large table may put significant pressure on the driver node because broadcasting large tables is expensive.

This paper implements identification calculations between two tables based on these two Join strategies and conducts comparative studies. Experimental results are presented in Section 5.3, showing that when computing resources are

sufficient, using Join operations for identification calculations, BHJ is more advantageous; when computing resources are insufficient, SHJ should be chosen.

Figure 3 [Figure 3: see original paper] shows the Join flow diagram between two tables. First, the textFile operator reads HDFS metadata to create RDD (Resilient Distributed Datasets) [13], then the map operator and toDF function convert RDD to DataFrame. The two tables are joined based on Healpix index to return a new DataFrame, then the withColumn function adds a new column (angular distance between two celestial objects) and the filter function filters out data with angular distance less than the threshold. Finally, the qualified data is output to HDFS (Hadoop Distributed File System) [14], completing the time-series reconstruction process of star catalog data.

Figure 3 Join flow diagram between two tables

4.1 Design Architecture of Massive Star Catalog Data Time-Series Reconstruction Method Based on Spark

This paper proposes a massive star catalog time-series reconstruction method based on Spark in asynchronous non-blocking mode. The overall framework is shown in Figure 4 [Figure 4: see original paper]. In the preprocessing stage, raw star catalog data is effectively filtered to reduce storage and identification costs and generate metadata files, then HEALPix index is calculated and data is formatted. Notably, the adjacent block indices where celestial objects reside must also be recorded to allow redundant storage of adjacent block edge data in the reference catalog. Finally, star catalog data is partitioned into different HEALPix block files through Spark' s custom partition function, awaiting the next identification calculation. Subsequently, the preprocessed star catalogs undergo distributed identification calculation, and in the final stage, time-series data for celestial objects is generated.

Figure 4 Overall framework diagram of massive star catalog time-series reconstruction method based on Spark

4.2 Join-Free Identification Calculation Method

Chapter 3 introduced the method of integrating distance calculation and threshold filtering into database join operations. This method is simple to program and implement, but frequent Join operations still affect efficiency. To further optimize efficiency, a Join-free method is considered here, which directly applies distance calculation formulas and threshold filtering to HDFS data files. Data from different HEALPix partitions execute identification calculation tasks in parallel, with the reference catalog that needs frequent read/write operations stored in memory. Star catalogs on the continuous time axis are sequentially identified with celestial objects in the reference catalog. The identification calculation flow is shown in Figure 5 [Figure 5: see original paper]. This method will be compared with the Join-based method and previous excellent methods.

(1) After reading star catalog data from HDFS, first create the reference catalog for the current partition.

(2) If there are celestial objects to be identified in the current partition and edge celestial objects for the current partition exist in the Redis database, then update the edge celestial objects to the reference catalog according to the solution for reference catalog update synchronization problems.

(3) If there are no edge celestial objects for the current partition in the Redis database or the update is completed, then perform identification calculation between the next time period star catalog and the reference catalog: if matching fails, the celestial object is considered new and added to the reference catalog; if matching succeeds, the data of the two celestial objects is recorded.

(4) If the new celestial object is an edge object, it indicates that edge source leakage may have occurred in adjacent partitions. First, add it to Redis data. Through step (2), this edge celestial object will be added to the reference catalog of adjacent partitions. By redundantly storing edge celestial objects in the reference catalog, edge source leakage is avoided.

(5) Repeat step (2) until identification of all time period star catalogs for the current partition is completed.

Figure 5 Time-series reconstruction calculation flow diagram based on Spark

5.2 Star Catalog Preprocessing Performance Evaluation

To analyze the preprocessing performance of star catalog files, Figure 6 [Figure 6: see original paper] shows the file sizes before and after preprocessing and preprocessing time for different datasets. The preprocessed star catalog files for AST3-I datasets are about 28% of the original files, while for AST3-II datasets they are about 83% of the original files. The preprocessing time is within an acceptable range. According to the processed file sizes, preprocessing raw star catalog data can greatly reduce data storage volume, save storage time, and also save unnecessary cross-node data transmission time, thereby improving time-series reconstruction efficiency.

5.3 Comparison of Different Join Algorithms Between Two Tables

To verify the performance of BHJ and SHJ algorithms for identification calculation between two tables, experiments were conducted using the Tess4 dataset under different nodes and data volumes.

As shown in Figure 7a [Figure 7: see original paper], when the star catalog

data volume is fixed, the running time of BHJ algorithm for identification calculation between two tables under different nodes is less than SHJ. When the number of computing nodes is 16, the time using BHJ is 56.6% less than using SHJ. As shown in Figure 7b, when the number of computing nodes is 32, under different star catalog data volumes, the time using BHJ is also less than using SHJ. Because the Shuffle operation in SHJ algorithm is very time-consuming, causing star catalog data to be transmitted between different nodes. Notably, when the computing node memory is configured as 1 GB and the two table data sizes reach 150 MB and 600 MB respectively, the BHJ algorithm experiences insufficient memory, leading to identification calculation failure. Therefore, when computing resources are sufficient, using Join operations for identification calculation, BHJ is more advantageous; when computing resources are insufficient, SHJ should be chosen.

Note: a) Comparison of Join between two tables under different computing nodes; b) Comparison of Join between two tables under different data volumes.
Figure 7 Comparison of Join between two tables

5.4.1 Performance of Join-Free Identification Method at Different HEALPix Levels

Different HEALPix levels result in different star catalog partitioning granularities. Deeper levels mean finer granularity, i.e., more blocks, and consequently more edge data, requiring additional time to process edge data and involving cross-node updates. Smaller level numbers increase the number of celestial objects per partition, thereby increasing computational load. For homologous star catalogs, the distribution of celestial objects in most target sky regions is uneven. In the star catalog partitioning of this paper' s experiments, some partitions contain only a few celestial objects. Therefore, when the level is too small, nodes with few celestial objects remain idle most of the time, wasting computing resources and failing to fully exploit distributed computing performance. Thus, HEALPix level selection has a significant impact on time-series reconstruction performance, requiring optimal HEALPix level selection in advance to reduce computational complexity.

To select an appropriate HEALPix level, experiments were conducted using 64 computing nodes to perform time-series reconstruction on multiple datasets with similar data volumes at different HEALPix levels. As shown in Figure 8 [Figure 8: see original paper], for all datasets in the experiments, the computation time is shortest at HEALPix level 10. Decreasing to levels 8 and 9 or increasing to levels 11 and 12 increases computation time. Since HEALPix is a quadtree structure, each level increase multiplies the number of star catalog blocks by about 4, increasing the number of identification calculation tasks. If there are too many tasks, task startup and switching overhead increases significantly, degrading performance. Each level decrease reduces the number of blocks to 1/4, but increases the data volume per partition by about 4 times, resulting in too few tasks that cannot fully utilize cluster parallel computing

capability, slow task execution, and potential out-of-memory exceptions. For the experimental environment in this paper, HEALPix level 10 is the optimal choice, used in subsequent experiments. Of course, this value may differ in different test environments.

Figure 8 Join-free identification computation time at different HEALPix levels

5.4.2 Performance Evaluation of Join-Free Method with Different Nodes

To meet the efficiency and accuracy requirements in time-series reconstruction, this paper proposes a Join-free identification method based on Spark' s asynchronous non-blocking mode, which breaks through traditional Join algorithms and can further improve time-series reconstruction efficiency.

As shown in Figure 9 [Figure 9: see original paper], the identification computation time for each dataset decreases as the number of computing nodes increases. When the number of computing nodes is fixed, except for the Tess5-13 dataset, identification computation time increases with dataset file size. The total file size of the Tess5-13 dataset is similar to that of the HD117688 dataset, but the identification computation time for Tess5-13 is about 56% more than HD117688 because Tess5-13 has more edge data requiring time to process, resulting in longer computation time.

Figure 9 Join-free identification computation time at different HEALPix levels

5.4.3 Comprehensive Performance Evaluation of Join-Free Identification Method

To verify the overall performance of the Join-free method, experiments were conducted using the AST3-I dataset under different computing nodes, with a dataset size of 100 GB. As shown in Figure 10 [Figure 10: see original paper], identification computation time decreases as the number of nodes increases, taking only 6.18 minutes with 64 computing nodes.

Figure 10 Comprehensive experimental performance with different nodes

Figure 11 [Figure 11: see original paper] shows the speedup ratio under different computing nodes. As the number of computing nodes increases, acceleration effects become apparent, partly due to the excellent performance of the reference catalog replica strategy and reduced cross-node data transmission. As computing nodes increase, identification computation time gradually decreases, showing a basically linear growth trend.

Figure 11 Speedup ratio variation with number of computing nodes

5.4.4 Comparison with Others' Work

Currently, time-series reconstruction among continuous multiple star catalogs is still relatively rare. The research work of Yu et al. is most similar to this paper' s work. They implemented parallel computing through MPI programming, with an experimental environment of Ubuntu operating system, Intel i7-4790 CPU (8 cores @ 3.6 GHz), and 16 GB memory. This paper compares experimental results using the same dataset with their results under 6 processes. The results show that although Yu et al.' s method provides a visual interface with advantages in usability, the identification computation performance of this paper' s method using 5 computing nodes can surpass theirs. As shown in Figure 12 [Figure 12: see original paper], when the number of computing nodes continues to increase, this paper' s method can achieve continuous performance improvement. With 64 nodes, the identification computation time for HD88500, HD117688, and HD136488 datasets is 30%, 18%, and 19% of Yu et al.' s method time, respectively. This shows that the algorithm in this paper has achieved significant performance improvements. Moreover, as the number of nodes in the distributed environment increases, this paper' s method can achieve near-linear speedup because the method only redundantly stores the reference catalog, while other star catalogs are partitioned according to specific HEALPix levels. The main parallelization method is data parallelism, with computing tasks on different nodes being independent of each other, and multi-node synchronous updates and appended identification for newly added celestial objects being asynchronous. Therefore, the identification computation process rarely involves cross-node data transmission, and as computing nodes increase, the parallel execution efficiency can basically achieve linear speedup. In summary, the large-scale distributed architecture based on Spark and good speedup ratio make this method highly scalable and well-adapted for time-domain astronomical observation equipment with continuously increasing data volumes. Additionally, this paper' s time-series reconstruction method requires only one indexing method, effectively saving data storage space. In terms of computational accuracy, the theoretical source leakage rate of this paper' s method is 0, while Yu et al.' s hybrid HTM and HEALPix indexing method has a source leakage rate of about 4.5%. It is evident that introducing new parallel frameworks and algorithm design plays a decisive role in program performance improvement.

Figure 12 Comparison with others' methods

5.4.5 Comparison of Different Identification Methods

To compare the performance of the Join-free identification method with the two Join algorithms, experiments were conducted using the HD88500 dataset on 64 computing nodes. As shown in Table 3 , the Join-free identification method is obviously more efficient than traditional Join-based identification methods. Join-based identification calculation methods have problems with cross-node data transmission and data skew, while the Join-free identification calculation method avoids massive cross-node data transmission through prepro-

cessing, greatly reducing identification computation time and improving identification efficiency, enabling faster identification results.

Table 3 Comparison of different identification methods

6 Conclusion

This paper addresses the challenges of time-series reconstruction for massive astronomical star catalog data, combining the latest theoretical methods for star catalog indexing, algorithm design, and data update to solve the efficiency and accuracy problems of massive astronomical star catalog time-series reconstruction, providing data support for subsequent research. This paper first proposes a time-series reconstruction method under asynchronous non-blocking mode, introducing a reference catalog replica strategy that reduces data transmission while ensuring accuracy. For cross-node data update and synchronization problems in distributed environments, this paper provides corresponding solutions for different situations. Additionally, this paper studies and compares Broadcast Hash Join and Shuffle Hash Join algorithms between two tables from both theoretical and experimental perspectives, providing applicable scenarios for each. Finally, a Join-free time-series reconstruction method based on the Spark distributed framework is proposed, achieving more efficient time-series reconstruction of massive astronomical star catalogs. Experiments demonstrate that the time-series reconstruction algorithm designed in this paper is efficient, feasible, and practically valuable, providing a good foundation for the development of time-domain astronomy. The large-scale distributed computing architecture based on Spark and near-linear speedup ratio make this method valuable for time-series reconstruction tasks for larger-scale time-domain telescopes such as GWAC.

References

[1] Cordier B, Wei J, Atteia J L, et al. Proceddings of Science, DOI: https://doi.org/10.22323/1.233.0005
[2] Vallenari A, Brown A G A, Prusti T, et al. A&A, 2023, 674: A1(2023)
[3] 万萌, 吴潮, Ying Zhang, 等. 天文研究与技术, 2016, 13(3): 373
[4] Ivezic Z, Tyson J A, Acosta E, et al. AJ, 2019, 873(2): 111
[5] Zhao Q, Sun J, Yu C, et al. Transactions of Tianjin University, 2011, 17(1): 62
[6] Du P, Ren J J, Pan J C, et al. Science China: Physics, Mechanics and Astronomy, 2014, 57(3): 577
[7] Yu C, Li K, Tang S, et al. MNRAS, 2020, 496(1): 629
[8] 徐丹滢, 赵青, 权文利, 等. 天文学进展, 2022, 40(2): 298
[9] Brahem M, Yeh L, Zeitouni K. Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA: ACM, 2018: 229
[10] Zečević P, Slater C T, Jurić M, et al. AJ, 2019, 158(1): 37

[11] 刘莹, 马智, 游子毅, 等. 天文学报. 2022, 63(3): 10
[12] Armbrust M, Xin R S, Lian C, et al. Proceedings of the 2015 ACM SIGMOD international conference on management of data, New York: ACM, 2015: 1383
[13] Zaharia M, Chowdhury M, Das T, et al. Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, Berkeley: USENIX Association, 2012: 15
[14] Shvachko K, Kuang H, Radia S, et al. Proceedings of the IEEE 26th symposium on mass storage systems and technologies (MSST), USA: IEEE, 2010: 10

Note: Figure translations are in progress. See original paper for figures.