Postprint: Fast Time-Series Reconstruction for Massive Astronomical Catalog Data Using MongoDB

Authors: Xu Danying, Zhao Qing, Quan Wenli, Song Hongzhuang

Date: 2023-06-07T00:00:00+00:00

Abstract

The explosive growth of astronomical data has led to low efficiency in generating astronomical time-series data using traditional scientific computing methods, directly impacting the scientific output of time-domain astronomy. To address this issue, this article proposes a fast cross-matching method for homogeneous source catalogs that reduces distance calculations, along with a MongoDB-based application scheme. The approach focuses on optimizing raw data access and storage, as well as improving cross-matching computational speed, to solve the efficiency problems in batch time-series reconstruction of large-scale astronomical catalogs. Experimental results demonstrate that, compared with methods based on traditional multi-band cross-matching algorithms and relational databases, this method can more effectively improve the generation efficiency of time-series data, providing new insights for time-series reconstruction and light curve generation from large-scale catalog data of frequently-sampled telescopes in the era of time-domain astronomy.

Full Text

Preamble

ChinaXiv Partner Journal, Vol. 40, No. 2

June 2022

PROGRESS IN ASTRONOMY Vol. 40, No. 2, June 2022
doi: 10.3969/j.issn.1000-8349.2022.02.10

Research on Fast Time Series Reconstruction of Massive Astronomical Catalog Data Based on MongoDB

XU Dan-ying, ZHAO Qing, QUAN Wen-li, SONG Hong-zhuang
(College of Artificial Intelligence, Tianjin University of Science & Technology, Tianjin 300457, China)

Abstract

The explosive growth of astronomical data has led to low efficiency in generating astronomical time series data using traditional scientific computing methods, directly impacting the scientific output of time-domain astronomy. To address this issue, this paper proposes a fast cross-identification method for homogeneous catalogs that reduces distance computations, along with a MongoDB-based implementation scheme. The research focuses on novel improvements in raw data access optimization and identification computation speed to solve the efficiency problem of batch time series reconstruction for large-scale astronomical catalogs. Experimental results demonstrate that compared with traditional multi-band cross-identification algorithms and relational database approaches, this method can more effectively improve the generation efficiency of time series data, providing new insights for time series reconstruction and light curve generation from large-scale catalog data obtained by frequently-sampled telescopes in the time-domain astronomy era.

Keywords: MongoDB; geospatial index; memory access optimization; identification optimization
Classification Code: P129
Document Code: A

1 Introduction

In recent years, "time-domain astronomy" has attracted widespread attention in the astronomical community, and developing high-performance software for processing massive astronomical data suitable for time-domain astronomy has become a focus of astronomical informatics research worldwide. Time series reconstruction of catalog data refers to the process of determining the data for each celestial object at different time points through cross-identification of stars on calibrated catalogs, and then sorting these data chronologically to generate time series for each object. This is an important data processing step in time-domain astronomy research and forms the foundation for fitting light curves and conducting time-domain analysis studies.

Efficient time series reconstruction methods for large-scale catalog data can significantly improve the generation efficiency of astronomical time series data products and promote the rapid development of time-domain astronomy research. However, current time series reconstruction for large-scale catalogs still faces the computational challenge of performing batch cross-identifications across multiple catalogs over long time intervals. Traditional cross-identification methods cannot meet the efficiency requirements of time series reconstruction problems and require further algorithmic optimization. Conventional

multi-band cross-identification methods for heterogeneous catalogs suffer from low computational efficiency due to complex distance calculations and additional processing required for edge data. For homogeneous catalogs, time series reconstruction faces an even greater challenge: it requires repeated cross-identifications over long time intervals, and traditional methods are insufficient to meet the efficiency demands, necessitating further optimization.

In recent years, many scholars have conducted in-depth research on the retrieval, storage, and fusion of massive astronomical catalog data. Gao et al. [1, 2] proposed cross-identification methods based on HTM indexing and KD-Trees, but these adopted a relational database model that could only support medium-scale catalogs with hundreds of thousands of entries and could not avoid the problem of missing sources at partition boundaries. Zhao et al. [3, 4] proposed a multi-core parallel cross-identification algorithm based on MPI that efficiently obtained all data requiring identification within the error radius through a fast adjacent block encoding derivation algorithm based on bit operations, solving the edge missing source problem. However, they used the traditional relational database MySQL, which still could not meet the identification demands for batch multi-catalog processing. Xu et al. [5, 6] proposed a method based on two-dimensional spatial grids with equal RA/Dec partitioning, but due to its special partitioning approach, it was only applicable to data covering medium-sized sky areas of 100–200 square degrees. Du et al. [7, 8] adopted a hybrid identification algorithm based on both HTM and HEALPix partitioning to solve the missing source problem in catalog data. Although this significantly reduced missing sources, it had long preprocessing times and introduced redundant computations during cross-identification. Moreover, it used SQL Server relational database, which can only run on Windows platforms, and its parallel implementation and coexistence model struggle to handle increasingly large data demands.

For massive astronomical catalog data, non-relational databases with their flexible models, rich functionality, unstructured storage, and high scalability are more suitable for data processing and show greater promise for improving performance in massive astronomical data reconstruction calculations. Currently, more researchers are recognizing the advantages of non-relational databases. Wan [9] proposed a MonetDB-based processing solution for GWAC data management, using MonetDB's built-in SQL to implement cross-identification calculations on simulated data generated by a simulator. MonetDB, optimized for scientific data computing characteristics, achieved good results. However, this method still has scalability issues, as experiments have shown [10] that MonetDB provides insufficient support for large-scale accumulated data, with unstable data ingestion times that can easily cause data congestion. Feng et al. [11] used the non-relational database MongoDB to store aeromagnetic big data, proving MongoDB's excellent performance in data write performance and spatial query efficiency. Cheng [12] designed a flexible database structure based on MongoDB and used sharding technology to deploy a MongoDB cluster for horizontal scaling to store data. Zhou et al. [13] proposed a geographic spatial data analysis scheme based on Spark and MongoDB, utilizing MongoDB's

geospatial indexing and query mechanisms to process geographic spatial data. These experiments demonstrate that MongoDB not only provides fast query speeds and supports distributed clusters, but is also highly suitable for processing high-concurrency, large-batch geographic spatial data.

Additionally, there has been research on cross-identification for homogeneous catalogs. Li et al. [14] proposed an optimization algorithm for homogeneous catalog identification in parallel environments, using dynamic programming to partition data to ensure load balancing in parallel environments. Their research results were applied to generating light curves from catalogs to describe how celestial brightness changes over time. However, they still used the hybrid identification approach based on HTM and HEALPix partitioning to solve boundary missing source problems, and the redundant computations made the computational load large. Their parallel environment was also not suitable for scaling and struggled to cope with further data volume increases.

Although traditional cross-identification algorithms have been studied for many years and the methods are relatively mature, considering the high sampling frequency of telescopes in time-domain astronomy, time series identification requires repeated identifications between continuous catalogs over long time axes, demanding higher efficiency. Further improvements in storage access and identification computation simplification are essential. This paper studies homogeneous catalogs that have consistent telescope systematic errors, with minimal positional variation of stars and only small common errors in certain local regions. Therefore, not all stars require strict distance identification, allowing us to consider reducing distance computation to some extent, optimizing algorithm complexity, and improving efficiency to make large-scale batch identification between homogeneous catalogs possible.

Considering these issues, this paper proposes a high-performance time series data reconstruction algorithm for massive homogeneous catalog data in a MongoDB environment, tailored to the characteristics of astronomical catalog data. The efficiency problem in large-scale catalog identification manifests in two main aspects: (1) Data storage and access efficiency. To address this, we utilize MongoDB's document-based sharded cluster storage and geospatial indexing to optimize data access. (2) Computation optimization. We propose a fast cross-identification algorithm that reduces distance dependency to optimize algorithm complexity, comparing it with traditional distance-based methods and improving accuracy through range filtering in specific regions and local identification calculations. For boundary missing source processing, we only use the first day's data as the reference catalog with added redundant data, eliminating the need for redundant data thereafter, thereby improving efficiency and facilitating distributed data partitioning while reducing inter-node data communication. The overall algorithm design framework is shown in Figure 1 [Figure 1: see original paper].

2 Fast Cross-Identification Algorithm for Catalog Data

Cross-identification algorithms are a crucial component of time series reconstruction. Current cross-identification algorithms adopt a strict distance computation approach for every star in a catalog. While these traditional methods offer high precision, they involve complex trigonometric operations and are extremely time-consuming due to massive computational loads. Since time series reconstruction requires batch cross-identification computations across multiple catalogs, these time-consuming traditional methods clearly cannot meet real-time query requirements, necessitating further algorithm optimization.

To address this problem, this paper proposes a fast identification algorithm that reduces reliance on distance computations within local regions to improve efficiency. The homogeneous catalog data studied in this paper has consistent systematic errors after positional calibration. Inspired by image alignment algorithms, we utilize image position and brightness information to find optimal matches between target and reference catalogs. Based on this principle, we aim to quickly locate matching celestial objects within local regions through position comparison, range filtering, and magnitude information comparison, eliminating most distance computations. Strict distance calculations are performed only in regions with abnormal star counts or magnitude anomalies. Simultaneously, we process edge data to reduce errors caused by missing sources, optimizing algorithm complexity while ensuring accuracy and improving the precision of the fast identification algorithm.

2.1 HEALPix Data Range Filtering Strategy

Catalog data volumes are enormous. We first process raw catalog files to extract key columns such as serial number, right ascension, declination, and brightness, then select an appropriate sky region level and calculate the HEALPix index to create an index table stored in MongoDB, as shown in Table 1 .

Cross-identification determines whether two stars are the same based on their angular distance, with identification occurring only when the angular distance is below a threshold. Stars on the celestial sphere are generally partitioned by positional information, and stars with large positional separations naturally cannot be the same object. Based on this principle, we can use range filtering to reduce the computational load and frequency of identification calculations.

The principle of range filtering is illustrated in Figure 2 [Figure 2: see original paper]. Identification calculations are first limited to the same HEALPix partition. Within the selected HEALPix partition, stars are further limited by region based on the two star index tables. According to the error radius requirements, an appropriate range is formed around the queried star to filter out stars with large distance differences, avoiding invalid calculations for stars that are too far apart and thereby improving the efficiency and accuracy of cross-identification distance calculations.

2.2 Distance-Free Identification Computation Method

Considering that the identification computations involved in time series reconstruction primarily concern homogeneous catalogs with consistent telescope systematic errors, and inspired by image alignment algorithms, we can perform distance-free identification of stars through comparison of star positions and brightness in local sky regions. That is, by selecting an appropriate HEALPix level based on the error radius, stars falling within the same block under this partitioning are directly considered matched, thereby reducing computationally intensive trigonometric operations. However, several situations may arise during the comparison of position and brightness information between two catalogs that could lead to matching errors: (1) A star in the reference catalog near the edge of a partition may fall into an adjacent partition in the catalog to be identified, causing boundary missing source phenomena; (2) Multiple stars within the same partition may have relatively close positions, leading to confusion through direct RA matching; (3) New stars may appear in a sky region at certain times in the catalog. To ensure identification accuracy, we perform special processing for these situations, as shown in Figure 3 [Figure 3: see original paper].

When situations (1) and (3) occur, they typically manifest as inconsistent star counts within the same partition. We mark partitions with inconsistent counts as special regions and, to avoid boundary missing sources, also mark surrounding partitions as special regions. When situation (2) occurs, we compare the magnitude brightness information of the pre-matched star pair, and if it exceeds the brightness threshold, we consider these two stars require high attention and mark the block as a special region. Stars in partitions marked as special regions will undergo strict distance-based identification with stars in the corresponding special regions of the other catalog to minimize errors from distance-free computation and ensure the accuracy of the fast identification algorithm.

Figure 4 [Figure 4: see original paper] shows the overall flowchart of the fast identification algorithm. The basic principles are as follows: (1) First, select a fine-grained sky region partitioning level based on the error radius to ensure the number of stars in each partition is minimized. (2) If the number of stars in the same partition of both catalogs is identical and equals 1, indicating no other stars exist in this partition besides the target star, brightness comparison can be performed directly. If the brightness difference is below the threshold, matching is performed directly through HEALPixID. (3) When the star count exceeds 1, to find corresponding stars between the two partitions, we first perform range filtering on stars in the same partition. Stars within the range then undergo brightness comparison: if the brightness difference is below the threshold, the stars are matched directly; stars with brightness differences exceeding the threshold cause their partition and surrounding partitions to be marked as special regions, which will then undergo strict distance calculations with corresponding partitions in the reference catalog. (4) If the star counts differ within the same partition, indicating possible boundary missing sources where stars in this partition have fallen into other partitions, we address this by performing

strict distance-based identification on this partition and surrounding partitions to improve identification accuracy.

3 Research on Storage and Access Optimization for Catalog Data

With the explosive growth of catalog data, data access speed has become critically important in astronomical applications. For time series reconstruction, due to the enormous identification volume between multiple catalogs over long time axes, fast spatiotemporal data access design is key to overall performance improvement, with data storage structures and index design being particularly challenging aspects for improving access speed.

To address this issue, we selected MongoDB, a distributed non-relational document database, for data storage, as shown in Figure 5 [Figure 5: see original paper]. As the NoSQL database closest to traditional relational databases, MongoDB combines improved read/write performance with convenient index structure establishment. Its storage design, geospatial indexing and query mechanisms based on 2D and 2Dsphere, and horizontal scalability give it excellent performance for storing astronomical data. Additionally, its data-oriented model and automatic failover capability provide high-concurrency read/write capabilities and good system stability. MongoDB stores data in BSON format, which can contain various types of documents and embed other data within them. This free storage model allows it to add or reduce fields for each record as needed when facing massive scientific data storage [15].

3.1 Research on Catalog Data Indexing

For catalog data, indexing is key to fast queries and plays an important role in various astronomical data processing and analysis tasks [16]. Common multidimensional indexing techniques include spatial indexes such as R-Tree, G-Tree, and KD-Tree. However, these indexes are not ideal in practical applications because it is difficult to ensure that spatially close points belong to the same branch or leaf node. Moreover, as data volume increases, the depth of these indexes grows, query efficiency decreases, and they fail to meet requirements for fast indexing and access. Among MongoDB's many extended features, its geospatial indexing provides excellent support for location-based data processing and computation. Geospatial indexing is a pseudo-two-dimensional index that can map two-dimensional space to one-dimensional space while ensuring adjacent regions remain close in one-dimensional encoding.

MongoDB's spatial indexes are divided into 2D, 2Dsphere, and geoHaystack [17]. geoHaystack is a special index that optimizes return results within small areas, improving efficiency for planar geometric queries. The key-value pair-based geospatial indexes are 2D and 2Dsphere. 2Dsphere can be understood as a spherical latitude-longitude index supporting queries of spherical geometric entities, but it only supports latitude-longitude data. The 2D index can be

understood as a planar 2D index supporting indexing of planar maps and temporally continuous point data. MongoDB constructs 2D indexes using Geo-Hash technology, not the internationally common GeoHash description method with 32 grids per level, but rather a planar quadtree approach. This partitioning 思想 is very similar to the HEALPix indexing 思想 in astronomy. This hierarchical iterative quadtree partitioning and encoding ensures that the vast majority of spatially close regions have similarly close encodings. The 2D index is shown in Figure 6 [Figure 6: see original paper].

We can fully utilize MongoDB's geospatial indexing to achieve rapid data location and improve query efficiency. According to experimental results in Section 4.3, 2D indexes are significantly more efficient than 2Dsphere indexes for large-volume catalog data, so this paper chooses to build 2D indexes on catalog data, with specific experiments detailed in Section 4.3.

3.2 Research on Catalog Data Storage

To process large-scale catalog data storage and load without requiring large computers, a database's horizontal scalability is crucial. MongoDB sharding is horizontal database scaling that divides entire datasets into many chunks based on a given shard key and stores these chunks on different nodes, with each node maintaining a data subset. This paper uses MongoDB's sharding functionality for distributed data storage, distributing data evenly across shards to prevent a single node from storing excessive data, which would cause overload and slow data retrieval.

To ensure data security in catalog data, we use MongoDB replica sets to save data copies on multiple nodes, enabling "near real-time synchronization" during replication for high availability. The replica set structure uses a typical "one primary, two secondary" architecture, as shown in Figure 7 [Figure 7: see original paper], where both primary and secondary nodes are data nodes that store complete data [18].

This paper uses the "one primary, two secondary" replica set structure as a single shard, creating shard one "replcopy1" and shard two "replcopy2" on separate machines, with two shards and one config node forming a cluster. The cluster structure is shown in Figure 8 [Figure 8: see original paper].

3.3 Storage Cluster Data Balancing Mechanism

Load balancing of cluster data is crucial for read/write efficiency, as uneven data distribution can cause certain nodes to become overloaded. MongoDB partitions catalog data through shard keys. Before sharding, a data collection is a single chunk; sharding splits the collection into multiple chunks based on the shard key, with these chunks distributed across different shards. During data read/write operations, shard key selection and data balancing mechanisms are critical for balanced data distribution.

For convenient data migration and balanced distribution, we adopt hash-based sharding. As shown in Figure 9 [Figure 9: see original paper], hash partitioning has natural static load balancing characteristics that generally pursue uniform data distribution across partitions. Users need not consider which partition a column value or set of values should reside in, as the database automatically handles this work. Because data must be located through a unified hash function, hash partitioning can effectively distribute data evenly across physical storage when data duplication rates on the partitioned columns are low.

This paper creates a hash shard key on the HEALPix partition ID column (named hid) of catalog data. HEALPix level partitioning also plays a critical role in whether overall catalog data storage is balanced. The size of HEALPix block partitioning affects the number of stars stored per HEALPixID and the accuracy of the fast identification algorithm. As shown by experiments in Section 4.2.2, when the HEALPix level is 13, it ensures that the catalog data stored per HEALPixID is relatively balanced while achieving overall optimization of fast identification algorithm efficiency and accuracy. Creating a hash shard key on HEALPixID randomly and evenly distributes data across nodes, ensuring both data balance and read concurrency.

For real-time catalog data queries, besides considering storage balance among nodes, we must also consider balancing data access frequency. Nodes with hot data will be accessed intensively. In such cases, we can consider disabling MongoDB's balancer for manual sharding during shard design, or modifying MongoDB's balancing mechanism to incorporate average data access time intervals as a factor in addition to chunk quantity load, thereby addressing data access frequency imbalance to some extent.

4 Experiments

4.1 Hardware and Software Environment

The experimental data used in this study is part of the HD8500 dataset publicly available on the China Virtual Observatory website, captured by AST3. For comparative analysis between the fast identification algorithm and traditional identification algorithms, we used an AMD Ryzen 5 3500U x8 computer with 8 GB memory, deepin15.5 operating system, and C++ programming language. MongoDB read/write performance testing and comprehensive testing were conducted on an Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz computer with 8 GB memory, Windows 10 operating system, and C++ and Python programming languages.

4.2.1 Accuracy Comparison Test with Integrated Brightness Information

Traditional cross-identification uses a single position-based method. While highly precise, this approach involves enormous computational load. Considering that this paper studies time series reconstruction of homogeneous

catalogs, for long-period variable sources with slow brightness variations, we can integrate star brightness information to further improve efficiency and identification accuracy during distance-free computation. The experiment uses whether the brightness difference between two stars is below a brightness threshold to determine if the brightness difference is too large, thereby judging whether they are the same star. To verify the necessity of incorporating brightness information comparison, we conducted an accuracy comparison experiment using AST3-captured HD8500 data a0507.2 and a0507.3, with 37,818 and 34,940 catalog entries respectively, and a brightness threshold of 0.02.

Figure 10 [Figure 10: see original paper] shows the accuracy of two algorithms: one with and one without brightness information comparison in the fast identification algorithm. The results clearly demonstrate that incorporating star brightness information comparison significantly improves identification accuracy, stabilizing it above 99%.

4.2.2 Evaluation Metrics Comparison Under Different HEALPix Levels

During cross-identification, HEALPix level partitioning affects both data distribution balance and identification result accuracy. To rigorously determine the 优劣 of level partitioning impacts, we use the F1 score as the evaluation metric for HEALPix level partitioning. The F1 score is calculated from precision (P) and recall (R), briefly introduced as follows:

(1) **Precision**: Represents the proportion of predicted positive samples that are truly positive. The formula includes true positives (TP) and false positives (FP):

$$P = \frac{TP}{TP + FP}$$

(2) **Recall**: Represents the proportion of actual positive samples correctly predicted. The formula includes false negatives (FN):

$$R = \frac{TP}{TP + FN}$$

(3) **F-Measure (F-Score)**: Since P and R sometimes contradict each other, they must be considered comprehensively. The most common method is F-Measure, the weighted harmonic mean of Precision and Recall:

$$F_1 = \frac{(a^2 + 1)PR}{a^2(P + R)}$$

When parameter $a = 1$, this becomes the most common F1 score used in this experiment:

$$F_1 = \frac{2PR}{P + R}$$

As the formula shows, F1 comprehensively combines P and R results, with higher F1 indicating more effective experimental methods.

As shown in Figure 11 [Figure 11: see original paper], lower HEALPix levels result in larger sky partitions with more stars falling into the same region, causing confusion and matching errors. Higher HEALPix levels produce smaller partitions, where overly small regions can cause stars near boundaries to fall into adjacent partitions, preventing matching between stars that should be identified and resulting in data loss. Therefore, while precision can reach 100% at HEALPix level 17, recall is extremely low, indicating massive identification result loss under overly fine sky partitioning, which would severely impact light curve generation for time series reconstruction. At level 13, the F1 score is highest, so we select HEALPix level 13 to optimize algorithm performance.

4.2.3 Evaluation Metrics Comparison of Four Distance-Reducing Identification Methods

To verify the effectiveness of our proposed fast identification method, we compare the F1 scores of four identification methods that reduce distance computations:

(1) If star counts are equal in identical HEALPix blocks between catalogs, directly match after range filtering, considering stars within range as successfully matched. When counts differ, directly match stars within range and consider extra stars as identification failures.

(2) If star counts are equal in identical HEALPix blocks, directly match after range filtering. When counts differ, mark both HEALPix blocks as special regions and perform strict distance identification on stars within both blocks, without marking or identifying surrounding blocks.

(3) If star counts are equal in identical HEALPix blocks, directly match after range filtering. When counts differ, mark both HEALPix blocks as special regions, perform strict distance identification on stars within both blocks, and also mark and identify surrounding blocks.

(4) If star counts are equal and equal to 1 in identical HEALPix blocks, compare brightness and match if below threshold. If counts are equal and greater than 1, match stars that remain after range filtering and brightness comparison. When counts differ or corresponding star brightness exceeds the threshold, mark both HEALPix blocks as special regions, perform strict distance identification on stars within both blocks, and also mark and identify surrounding blocks.

As shown in Figure 12 [Figure 12: see original paper], method (1) performs worst because it doesn't consider boundary missing source problems or one-to-many/many-to-one identification issues, resulting in low recall and precision from direct block-based matching. Method (4), which adopts different identification approaches based on whether star counts are equal, integrates mag-

nitude information, incorporates range filtering, and considers boundary missing sources, significantly improving algorithm performance. Therefore, our fast identification algorithm adopts method (4).

4.2.4 Speed and Accuracy Comparison Between Fast and Traditional Identification Algorithms

To demonstrate the performance of the fast identification algorithm, we selected partial data from AST3-captured HD8500 files as test data and compared it with traditional identification algorithms on a single node. Figure 13 [Figure 13: see original paper] shows the required computation time, revealing that the fast identification algorithm requires significantly less time than traditional methods for the same data volume.

Figure 14 [Figure 14: see original paper] compares the data volumes of identification results from both algorithms, showing minimal difference. Using traditional strict distance-based identification results as the standard, we compared them with fast identification algorithm results. As shown in Figure 15 [Figure 15: see original paper], the blue curve represents the accuracy of the fast identification algorithm, which remains extremely high at over 99.5%.

These comparisons demonstrate that compared with traditional identification algorithms, the fast identification algorithm incorporates range filtering, brightness comparison, and distance-free identification, avoiding massive distance calculations while maintaining high accuracy, significantly reducing identification time and improving efficiency.

4.3.1 MongoDB Index Performance Comparison Test

MongoDB's geospatial indexes provide higher efficiency for geographic data queries. We compared the efficiency of MongoDB's 2D planar index, 2Dsphere index, and B-tree index built on the HEALPixID column for querying catalog data.

Table 2 MongoDB Index Comparison

Index Type	Catalog Data Query Volume
2D Index	$5.0\$\times10^{\{4\},1.5\times10}\{5\},3.0\times10^{\{5\},6.0\times10}\{5\},7.5\times10\hat{}\{5\}\$$
B-tree Index	(same volumes)
2Dsphere Index	(same volumes)

Table 2 shows that for large data volumes, 2Dsphere spherical index efficiency is significantly lower than 2D planar index efficiency. Therefore, we build 2D indexes on catalog RA/Dec coordinates.

4.3.2 Hash Partitioning Data Balance Test

Hash functions enable random data sharding. We can utilize this static load balancing characteristic of hash partitioning to evenly distribute catalog data across different shard nodes within a certain range. We conducted data distribution comparison tests using both hash indexes and B-tree indexes on the HEALPixID column as shard keys.

As shown in Figure 16 [Figure 16: see original paper], using ordinary B-tree indexes as shard keys leads to uneven data distribution with huge disparities in data storage between nodes. In contrast, hash partitioning demonstrates obvious load balancing effects, achieving basically uniform data distribution across all nodes and dispersing read/write pressure.

4.4 Comprehensive Test

Traditional time series data generation methods import catalog files into relational databases, use cross-identification results to assign MatchID markers to stars across different time periods, then match stars with identical ID markers in the database [19], export results to files, and finally generate time series data from these files. This experiment selected partial catalog files from AST3-captured HD88500 data as test data, with each catalog file containing approximately 29,000–39,000 entries. We compared our method with traditional time series generation methods.

As shown in Table 3 , the MongoDB-based fast identification algorithm is clearly more efficient than traditional methods, with the advantage becoming more pronounced as data volume increases.

Table 3 Comparison Between Traditional and Fast Identification Methods

HD88500 Catalog Files	Traditional Method (min)	Fast Method (min)

5 Conclusion

This paper designs a fast time series reconstruction method for massive astronomical catalog data based on MongoDB. On one hand, it utilizes MongoDB's document-based storage and geospatial indexing to improve data storage and query efficiency. On the other hand, it reduces distance computations in traditional cross-identification methods through range filtering in specific regions, magnitude information comparison, and local identification calculations instead of global computation. Experimental results demonstrate that this research achieves certain improvements in data access optimization and identification algorithm efficiency, particularly for continuous temporal identification of massive homogeneous catalogs. The proposed fast time series reconstruction method has

good application value and can effectively solve access and computational efficiency problems in batch time series reconstruction of large-scale astronomical catalogs, thereby advancing research on stellar light curves in the time-domain astronomy era.

References

[1] Gao D, Zhang Y X, Zhao Y H. *Astronomical Research & Technology*, 2005, 2(3): 186
[2] Gao D. PhD Thesis. Beijing: National Astronomical Observatories, Chinese Academy of Sciences, 2008: 1
[3] Zhao Q, Sun J, Yu C, et al. ICA3PP, 2009: 604
[4] Zhao Q, Sun J, Yu C, et al. *Transactions of Tianjin University*, 2011, 17(1): 62
[5] Xu Y, Wu C, Wan M, et al. *Astronomical Research & Technology*, 2013, 10(3): 273
[6] Xu Y. Master's Thesis. Hubei: China Three Gorges University, 2013: 1
[7] Du P. Master's Thesis. Shandong: Shandong University, 2013: 1
[8] Du P, Ren J J, Pan J C, et al. *Science China: Physics Mechanics & Astronomy*, 2014, 57(3): 577
[9] Wan M. PhD Thesis. Beijing: National Astronomical Observatories, Chinese Academy of Sciences, 2016: 1
[10] Yang C, Weng Z J, Meng X F, et al. *Journal of Computer Research and Development*, 2017, 54(2): 248
[11] Feng L, Yang Z Y, Li W J, et al. *Journal of Geology*, 2019, 43(03): 421
[12] Cheng J L. Master's Thesis. Anhui: Anhui University, 2020: 1
[13] Zhou Y, Liu C, Xu S N, et al. *Geomatics & Spatial Information Technology*, 2018, 41(09): 71
[14] Li K, Yu C, Tang S, et al. 15th IEEE International Symposium on Parallel and Distributed Processing with Applications. Guangzhou, China, 2017: 1074
[15] Ren M F, Li X J, Cui M M, et al. *Computer Knowledge and Technology*, 2019, 15(34): 1
[16] Xu S L. Master's Thesis. Liaoning: Dalian Maritime University, 2018: 1
[17] Kristina Chodorow. *MongoDB: The Definitive Guide* (2nd Edition). Translated by Deng Q, Wang M H. Beijing: Posts & Telecom Press, 2014: 121
[18] Zhang W J, Cai J L. *MongoDB from Beginner to Business Practice*. Beijing: Beijing Electronics Press, 2019: 47
[19] Xiong C C, Fu L Y, Zhao Q. *Computer Applications and Software*, 2021, 38(04): 17

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