Distributed MySQL Cone Search for Massive Astronomical Data: Postprint

Authors: Yang Chao, Liang Bo, Dai Wei, Wei Shoulin, Hui Deng, Wang Feng

Date: 2020-11-12T00:00:00+00:00

Abstract

With the progression of sky survey observation programs, traditional database technologies can no longer meet the storage and retrieval performance requirements for massive astronomical datasets. This paper addresses the high concurrency and high-performance challenges associated with massive astronomical data storage and cone search by employing database middleware technology. When data volume reaches the storage threshold of traditional databases, middleware technology enables storage across database clusters through database and table sharding, thereby fully integrating the advantages of relational databases and distributed technologies. In this work, by integrating the DIF plugin with MySQL databases, a pseudo-spherical index is established within distributed databases to satisfy the cone indexing requirements of massive astronomical data.

Full Text

Research on Cone Search of Distributed MySQL for Massive Astronomical Data

Yang Chao[1], Liang Bo[1,2], Dai Wei[1,2], Wei Shoulin[1,2], Deng Hui[3], Wang Feng[3]

[1]School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650000, China
[2]Yunnan Key Laboratory of Computer Technology Application, Kunming University of Science and Technology, Kunming 650000, China
[3]Center for Astrophysics, Guangzhou University, Guangzhou 510006, China

Abstract: With the advancement of sky survey programs, traditional database technologies can no longer meet the storage and retrieval performance requirements of massive astronomical data. This paper addresses the high concurrency

and performance challenges in storing massive astronomical data and performing cone searches by employing database middleware technology. When massive data reaches the storage threshold of traditional databases, middleware technology enables storage in database clusters through sharding, fully integrating the advantages of relational databases and distributed technology. In this work, we integrate the DIF plugin with MySQL databases to establish a pseudo-spherical index in distributed databases, which satisfies the cone search requirements for massive astronomical data.

Keywords: massive data; distributed middleware; cone search; DIF plugin; MySQL

Funding: This work is supported by the National Key R&D Program of China (2018YFA0404603), the NSFC-CAS Joint Fund for Astronomy (U1831204, U1931141), the National Natural Science Foundation of China (11961141001, 11903009), the Guangzhou University "Innovation Strengthening School Project" (2017KZDXM062), the Yunnan Provincial Key R&D Program (2018IA054), and the Yunnan Provincial Applied Basic Research Projects (2017FB001, 2018FB103). We also acknowledge the support from the National Astronomical Observatories-China Cloud Astronomy Big Data Joint Research Center.

Received date: ; **Revised date:**

Author biographies: Yang Chao, male, master' s student, research interests: massive astronomical data storage and retrieval processing techniques. Email: chao_{yang}@stu.kust.edu.cn. Dai Wei, male, associate professor, research interests: astronomical technology and methods. Email: daiwei@astrolab.cn. Wei Shoulin, male, associate professor, research interests: astronomical technology and methods. Email: weishoulin@astrolab.cn. Deng Hui, female, professor, research interests: astronomical technology and methods. Email: denghui@gzhu.edu.cn. Wang Feng, professor, doctoral supervisor, research interests: distributed data storage and computing, IoT technology, cloud computing.

Corresponding author: Liang Bo, male, senior engineer, research interests: computer database technology. Email: liangbo@cnlab.net

0 Introduction

In recent years, with the continuous improvement of large astronomical observation facilities and unprecedented enhancement in data acquisition capabilities, numerous astronomical survey projects have emerged both domestically and internationally. For instance, China' s independently designed LAMOST (Large Sky Area Multi-Object Fiber Spectroscopic Telescope) collects 20 GB of spectral data per night [2]; the Vera C. Rubin Observatory' s LSST (Large Synoptic Survey Telescope) acquires up to 15 TB of raw observation data nightly; and the Square Kilometre Array (SKA), a large-scale scientific facility with international collaboration and the world' s largest radio observatory, is expected

to produce approximately 300 PB annually. Against this backdrop of big data, exploring efficient and scalable retrieval methods for massive astronomical data has become an urgent challenge in the field of astronomical observation.

Traditional scientific data retrieval relies on relational databases, which offer satisfactory efficiency for non-massive structured data through indexing mechanisms. However, as astronomical observations have expanded from optical to full-band coverage, traditional databases cannot withstand the impact of exponential data growth. NoSQL systems typically employ key-value storage formats where all data values under the same key are stored together for fast access. Researchers have proposed systems like AstroSpark [4] and FASTDB [5] based on distributed array databases, which offer advantages such as low latency, scalability, and high cost-effectiveness. Nevertheless, due to their underlying storage format, these systems lack advanced features of the relational model and compromise ACID properties [6].

In recent years, NoSQL has evolved into NewSQL. NewSQL systems are essentially relational DBMSs that support SQL, indexing, and schema, maintaining not only ACID transaction properties but also achieving the same scalability as NoSQL. Their basic architecture is a parallel DBMS that outperforms MapReduce frameworks in query load [7]. Distributed relational databases combine traditional relational databases, distributed clusters, and distributed transactions, offering high compatibility with traditional databases while better supporting SQL and transaction processing [8].

However, even in powerful distributed relational databases, the direct retrieval approach cannot meet current astronomical data retrieval requirements due to the increasingly large data volumes published by major sky survey projects. Effective spherical indexing is essential for accessing large-scale catalog datasets. Currently, widely used pseudo-spherical indexing methods include HTM (Hierarchical Triangular Mesh), HEALPix (Hierarchical Equal Area isoLatitude Pixelisation), and Q3C (Quad Tree Cube) [9]. After establishing appropriate spherical indexes, cone search can be implemented for large-scale catalog datasets. Cone search, a specialized retrieval method in astronomy [14], defines position information in celestial coordinates using right ascension (R.A.), declination (Dec), and angular distance (SR). It typically queries a conical region centered at (Tra, Tdec) with radius SR to retrieve information about relevant celestial objects, i.e., target objects (Sra, Sdec). This paper integrates the DIF indexing tool into distributed MySQL database clusters, enabling distributed relational databases to establish pseudo-spherical indexes for efficient cone search.

Section 1 investigates the basic framework of database middleware technology, proxy modes, and high-availability architecture, using sharding and read-write separation to horizontally scale MySQL databases while supporting transactions and ACID properties. Section 2 introduces the DIF indexing tool, which establishes pseudo-spherical indexes suitable for sharding rules and implements cone search through built-in functions. Section 3 presents the experimental framework design and cone search services under this architecture. Section 4

compares DIF tool functions with the great-circle formula for cone search, analyzes database node configurations, HEALPix index levels, and performance differences between standalone and distributed retrieval.

1 Database Middleware

Database middleware encapsulates underlying data to enable database operations similar to single-database usage. Two typical middleware patterns exist: server-side proxy (database proxy) and client-side proxy (data source proxy). The database proxy approach uses a proxy server to manage multiple database instances. Clients establish connections to the proxy server through data sources, and all SQL operations are distributed by the proxy to underlying databases, with results integrated and returned to clients through the proxy.

The data source proxy approach internally manages multiple ordinary data sources. Client SQL operations undergo parsing, rewriting, and other processing by the data source proxy before being distributed to ordinary data sources for execution, with results merged and returned by the proxy [10].

Analyzing the SQL processing workflow, database middleware can parse client transaction requests, perform SQL parsing, optimization, and routing analysis, split them into executable thread tasks for databases, and distribute them to multiple database servers according to predefined sharding rules, effectively alleviating load pressure on single databases and achieving peak shaving. To support high availability, a configuration center and monitoring service can be added to form a simple high-availability architecture. The monitoring service detects cluster status and pushes change information to the configuration center upon changes. The database proxy then pulls configuration updates from the configuration center to refresh database configurations, as shown in [Figure 1: see original paper].

To reduce the load and overhead of cone search services, master-slave read-write separation distribution and sharding can be employed, enabling horizontal database scaling and improving cone search efficiency and concurrency.

1.1 Sharding

Sharding is the core function of database middleware. Common partitioning methods include vertical and horizontal partitioning. In distributed systems, horizontal sharding is typically used to address single-database bottlenecks and relieve access pressure. Using MySQL relational databases for horizontal partitioning, the global relationship N table is divided into several disjoint subsets according to horizontal sharding rules to satisfy completeness, reconstructability, and disjointness properties, corresponding to formulas (1), (2), and (3) [11].

1.2 Read-Write Separation

Since read operations far outnumber write operations in databases, a master-slave distribution is commonly used, where the master node handles writes and slave nodes handle reads. Read-write separation distributes load across multiple nodes. Maintaining data consistency requires synchronization mechanisms such as master-slave replication, Paxos, Raft, Term, ZAB, and other protocol algorithms [12]. MySQL commonly uses MySQL Proxy as the read-write separation intermediate layer through an embedded Lua parser to define query processing [13].

2 DIF Plugin

The Dynamic Index Facility (DIF) [16] is an open-source MySQL/MariaDB database plugin compiled from C++ libraries, Perl scripts, and SQL stored procedures. Its approach "discretizes" 2D space and maps it to 1D space using pixelization methods. Each pixel is assigned a unique index ID. Exhaustive retrieval has O(n) time complexity. For cone search, pseudo-spherical indexes are needed to reduce time complexity, with most indexes based on tree structures. B-Tree [17] indexing reduces time complexity to $\log_2 N$, but its depth increases with data volume, making it difficult to apply to massive data indexing. The B+ tree-based index method is the most commonly used dynamic index structure in database systems, which is also the method DIF uses to establish pseudo-spherical indexes through MySQL.

The DIF indexing tool employs HTM (Hierarchical Triangular Mesh) and HEALPix (Hierarchical Equal Area isoLatitude Pixelisation), two of the most widely used pseudo-spherical indexing methods. HTM, a classic sky partitioning method, was first applied to the Sloan Digital Sky Survey (SDSS) [9]. HEALPix replaces HTM' s triangular partitioning with equal-area quadrilateral blocks, though both share quadtree-like hierarchical recursion patterns [15]. Currently, HEALPix-related pseudo-spherical indexing and efficient cone search have been widely applied in massive astronomical data [9].

This paper uses the DIF indexing tool to pre-establish HEALPix Nest indexes and implements cone search through the DIF_{Circle}(RA, DEC, SR) function, where RA is right ascension, DEC is declination, and SR is angular distance. DIF uses the healpix_{base} toolkit to calculate HEALPix Nest pixels intersecting with the cone, then queries corresponding pixels through MySQL and returns the result set.

3 Design and Implementation

To achieve efficient cone search for massive astronomical data, this paper proposes a solution based on database middleware, MySQL databases, and the DIF indexing tool. [Figure 2: see original paper] illustrates the main framework design.

In this framework, the cone search service does not directly access the underlying database. Instead, it first forwards cone search commands to the database middleware service through the proxy server according to predefined strategies. The middleware performs SQL parsing, routing, rewriting, execution, and result set merging. The underlying communication with MySQL databases occurs through DBI/DBD-MySQL modules, enabling function calls through MySQL database functions.

4 Experiments and Discussion

To verify the advantages of the proposed distributed cone search framework, this paper conducted comparative experiments between standalone MySQL and the distributed architecture for cone search.

4.1 Test Environment

The distributed test platform consists of a cluster of Sugon servers with Intel(R) Xeon(R) CPU E7-4807 processors, 16 GB memory, 2$×$256 GB SSD storage, and gigabit network connections between servers, running Ubuntu 18.04.4 LTS. One server hosts the middleware system, while three others run MySQL 5.7.31 and DIF 0.5.5. The standalone test environment uses servers with identical configuration.

4.2 Test Data

This test uses the source data table from Gaia Data Release 2, which contains basic source parameters. As shown in , irrelevant columns for cone search are filtered out, extracting source_{id}, ra, and dec columns as test data for establishing HEALPix Nest indexes using the DIF tool.

The comparison between DIF and great-circle formula cone search is presented in . The experiment was conducted in the distributed test environment with search center (0, 50) and radius of 1 degree. The first method uses the DIF tool SQL template for cone search centered at (ra, dec) with SR as angular distance, where (RA, DEC) are right ascension and declination in degrees (deg) and SR is in arcminutes. The second method establishes a DEC index, filters the range (DEC-1, DEC+1), and uses the great-circle formula to calculate spherical angular distances less than SR, where SR is in degrees. The great-circle formula calculates the spherical angular distance d between two points p1(RA, DEC) and p2(ra, dec) as shown in equation (4) [18]:

$$d = \arccos[\sin(DEC)\sin(dec) + \cos(DEC)\cos(dec)\cos(RA - ra)] \quad (4)$$

As shown in , both methods yield identical result counts and intersection numbers, confirming that DIF implementation produces consistent results with the optimized great-circle formula, while DIF queries demonstrate higher efficiency.

4.3 Test Results and Analysis

To evaluate the impact of database node count and HEALPix Nest index order on retrieval efficiency, comparative experiments were conducted to determine optimal configurations. [Figure 3: see original paper] shows cone search performance with database node counts of 1, 3, 15, 30, 60, and 100, using search center (40, 0) degrees, with SR ranging from 30 to 240 arcminutes on the horizontal axis and retrieval time in milliseconds on the vertical axis. To eliminate cache interference, query cache was temporarily disabled using `set global query_{cache}_{size}=0` or `set global query_{cache}_{type}=0`.

The comparison reveals that retrieval time decreases as nodes increase from 1 to 30, but increases when nodes further increase to 60 and 100. Therefore, this paper selects 30 nodes as the optimal configuration.

To compare the impact of different HEALPix Nest index orders on cone search efficiency, experiments were conducted at 30 nodes with center (40, 0) degrees and radius 60 arcminutes, with results shown in [Figure 4: see original paper].

Fan D et al. [19] conducted comparative experiments between index order and retrieval efficiency, concluding that order=12 was optimal for their dataset, noting that different data densities may require different choices. Berriman et al. [20] compared HTM and HEALPix performance on Solaris, Windows, and Windows Server databases, suggesting that higher index levels improve performance, but excessive levels cause performance degradation due to overly fine granularity and hardware I/O throughput limitations. Combining these findings with [Figure 4: see original paper], this paper selects index order 12 as the optimal solution for the test dataset, balancing retrieval accuracy and efficiency.

Based on the comparison in [Figure 3: see original paper], stress tests were performed on 30 nodes versus standalone systems using Jmeter with 500 concurrent users, querying center (40, 0) degrees, radius 60 arcminutes, and HEALPix order 12. Results are presented in and [Figure 5: see original paper].

As shown in , cone search is data-intensive and CPU-consuming. The distributed middleware solution not only accelerates retrieval efficiency but also provides greater throughput. [Figure 5: see original paper] shows that standalone MySQL cone search queries consume over 90% CPU, while the distributed middleware framework maintains approximately 50% CPU usage. The proposed distributed database middleware solution for massive astronomical data effectively improves cone search efficiency and provides valuable references for future sky survey projects.

5 Conclusion

The ever-growing volume of massive astronomical data continuously presents challenges in data storage and efficient retrieval. This paper proposes a solution to extend MySQL database systems through database middleware and the DIF tool. The middleware implements database sharding to alleviate storage

and retrieval pressure from massive astronomical data. To further accelerate astronomical data retrieval, the DIF tool is integrated with MySQL to enable efficient cone search for massive astronomical data. Testing demonstrates that the distributed retrieval solution with index order 12 can efficiently perform cone search on massive astronomical data, reducing load pressure while ensuring data security and database high availability. Future work will compare whether different indexing algorithms (HTM, HEALPix, Q3C, etc.) suit this architecture, investigate the impact of columnar databases or vector engines on cone search, compare retrieval performance between MariaDB and MySQL (both supported by DIF), and explore GPU acceleration for DIF algorithms and improved cone search algorithms.

References

[1] Caife A. M. M. Big telescope, big data: towards exascale with the Square Kilometre Array. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 2020, 378: 2166.

[2] Wang Yang, Li Peng, Ji Yimu, Fan Weibei, Zhang Yujie, Wang Ruchuan, Chen Guoliang. A survey of high-performance computing and astronomical big data research. *Computer Science*, 2020, 47(01): 1-6.

[3] Abell, Allison P A, Anderson J, et al. *LSST Science Book, Version 2.0* [C]. LSST Corporation. 2009.

[4] Brahem M, Lopes S, Yeh L, et al. AstroSpark: towards a distributed data server for big data in astronomy [C]// ACM SIGSPATIAL PhD Symposium. ACM, 2016.

[5] Qiu Nengjun, Chen Mei, Li Hui, Li Hongyuan, Huang Menglin, Zhu Ming. Research and implementation of array database system FASTDB. *Computer Engineering and Design*, 2016, 37(04): 1107-1112.

[6] Grolinger K., Higashino W.A., Tiwari A., Capretz M.A. Data management in cloud environments: NoSQL and NewSQL data stores. *J. Cloud Comput.*, 2013, 2(1): 1-24.

[7] Stonebraker M, Abadi D J, Dewitt D J, et al. MapReduce and parallel DBMSs: friends or foes? [J]. *Communications of the ACM*, 2010, 53(1): 64-71.

[8] Wang S, Zhong Y, Wang E. An integrated GIS platform architecture for spatiotemporal big data [J]. *Future Generation Computer Systems*, 2019, 94(MAY): 160-172.

[9] Zhang Hailong, Ye Xinchen, Li Huijuan, Wang Jie, Wang Wanqiong, Tuohutinuer, Nie Jun, Cui Chenzhou, Liu Liang, Chen Junyi, Chen Xiao, Xue Yansong, He Boliang, Li Changhua, Zhao Qing, Xiao Jian, Fan Dongwei, Cao Zihuang, Li Shanshan, Mi Linying, Yang Zherui. A survey of astronomical data retrieval and publication. *Astronomical Research & Technology*, 2017, 14(02): 212-228.

[10] Wang Zheng, Wang Feng, Tian Yuan, Li Jian, Zhao Yongheng. Design and implementation of LAMOST heterogeneous environment information retrieval system [J/OL]. *Astronomical Research & Technology*: 1-12 [2020-10-09].

[11] Qi Shaoyang. Design and implementation of SQL processing module in MySQL-based distributed access middleware [D]. Nanjing University, 2016.

[12] Wu Hao, Chen Kang, Wu Yongwei, Zheng Weimin. Research on consistency protocols for big data systems based on RDMA and NVM. *Big Data*, 2019, 5(04): 89-99.

[13] Xiang Kai. Research and application of database middleware for massive high concurrency [D]. Shanghai Jiao Tong University, 2015.

[14] Plante, Raymond, Williams, Roy, Hanisch, Robert, etc. Simple Cone Search Version 1.03 [J]. 2008.

[15] Gorski K M, Hivon E, Banday A J, et al. HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere [J]. *Astrophysical Journal*, 2004, 622(2): 759-771.

[16] Luciano N, Giorgio C. Multiple Depth DB Tables Indexing on the Sphere [J]. *Advances in Astronomy*, 2010, 2010: 11.

[17] Yu L, Fu G, Jin Y, et al. MPDBS: A multi-level parallel database system based on B-Tree. IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. IEEE 2015; 1-7.

[18] Fan Dongwei, He Boliang, Li Changhua, Han Jun, Xu Yunfei, Cui Chenzhou. Comparison of spherical distance calculation methods and accuracy. *Astronomical Research & Technology*, 2019, 16(01): 69-76.

[19] Fan D, Xu Y, Han J, et al. A Simple Survey of Cross-Matching Methods [J]. *ASPC*, 2019, 523: 551.

[20] Berriman G B, Good J C, Shiao B, et al. A Study of the Efficiency of Spatial Indexing Methods Applied to Large Astronomical Databases [J]. arXiv preprint arXiv:1806.08866, 2018.

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