Design and Implementation of LAMOST Heterogeneous Environment Information Retrieval System (Postprint)

Authors: Wang Zheng, Wang Feng, Tian Yuan, Li Jian, Zhao Yongheng

Date: 2020-06-09T00:00:00+00:00

Abstract

Environmental information constitutes part of the operational status of the Large Sky Area Multi-Object Fiber Spectroscopic Telescope (LAMOST, also known as the Guo Shoujing Telescope), serving to assist in the telescope' s daily maintenance, observation, and post-processing of data. This environmental information originates from multiple subsystems, with data storage designed by different organizations, resulting in a complex data storage environment that is not conducive to unified retrieval. Based on asynchronous coroutines, we propose a method for heterogeneous environmental information retrieval. This approach utilizes a database proxy object factory to integrate multiple databases and retrieval field information, provides remote services through clients and Web browsers, and achieves retrieval of heterogeneous databases via custom retrieval commands. The method fully considers future upgrade requirements, retains multiple database and application interfaces, and facilitates the integration of additional databases and applications. Currently, the retrieval system developed based on this method has been deployed in the LAMOST operational environment, where it simplifies operational environment retrieval tasks in practice, improves work efficiency and data acquisition accuracy, and enhances maintenance and observation efficiency.

Full Text

Design and Implementation of a Heterogeneous Environment Information Retrieval System for LAMOST

Wang Zheng[13], Wang Feng[23], Tian Yuan[1], Li Jian[1], Zhao Yongheng[13]

[1]Key Laboratory of Optical Astronomy, National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100101, China

[2]Center for Astrophysics, Guangzhou University, Guangzhou 510006, China
[3]University of Chinese Academy of Sciences, Beijing 100049, China

Abstract

Environmental information constitutes a critical component of the operational status of the Large Sky Area Multi-Object Fiber Spectroscopic Telescope (LAMOST, also known as the Guo Shoujing Telescope), serving to support daily maintenance, observation activities, and subsequent data processing. This information originates from multiple subsystems whose data storage architectures were designed by different institutions, resulting in a complex storage environment that impedes unified retrieval. We propose a heterogeneous environment information retrieval method based on asynchronous coroutines. This approach utilizes a database proxy object factory to integrate multiple databases and retrieval field information, providing remote services through both client applications and web browsers to enable retrieval across heterogeneous databases via custom search commands. The method fully accommodates future upgrade requirements by preserving multiple database and application interfaces to facilitate the integration of additional databases and applications. The retrieval system developed based on this method has been deployed in the LAMOST operational environment, where practical operation has demonstrated simplified retrieval procedures, improved work efficiency, enhanced data acquisition precision, and increased maintenance and observation effectiveness.

Keywords: Information retrieval; Asynchronous coroutines; LAMOST

Introduction

LAMOST is a large-aperture reflective Schmidt optical telescope with a wide field of view, independently developed by China and installed at the Xinglong Observatory of the National Astronomical Observatories. The telescope can simultaneously observe 4,000 targets, making it the world' s most productive telescope in terms of spectra obtained per observation [1]. To date, LAMOST has observed, produced, and released over 10 million astronomical spectra.

In addition to producing astronomical spectra, LAMOST' s various subsystems generate millions of real-time status records daily, which are stored in different types of databases [2]. As shown in [Figure 1: see original paper], the LAMOST operational environment comprises a collection of status information from multiple subsystems, providing diverse environmental data including meteorological and DIMM information from the Weather subsystem, active optics and mount wind speed data from the TCS subsystem, guide star information from the GUID system, camera temperature data from the ICS subsystem, observation target information from the OSS system, and observation data from the OCS system. This constitutes a heterogeneous storage system composed of multiple database types including MySQL, SQLite, and PostgreSQL [3-4].

Environmental information provides crucial reference data for telescope observation, maintenance, and subsequent data processing. Based on application and importance, this information can be categorized as follows:

a. **Critical Alerts:** Emergency events affecting telescope safety. During observations, sudden conditions such as excessive humidity or high winds trigger immediate alerts through audible alarms and conspicuous flashing colors, prompting observers to cease operations and close the telescope dome to protect the instrument.

b. **Observation Assistance:** Visual displays showing the relationship between observation fields and targets, assisting resident astronomers in selecting optimal observation objects.

c. **Fault Alarms:** Subsystem malfunction notifications that alert observers to fault locations during observations, enabling timely repairs. During routine maintenance, on-site personnel must quickly retrieve status information from various subsystems to immediately identify and address any anomalies, ensuring the telescope remains in optimal operating condition. Off-site maintenance personnel can remotely access subsystem status information to assist on-site staff. By analyzing long-term environmental data in conjunction with maintenance logs and observation issues, maintenance teams can develop effective maintenance plans that reduce costs and downtime while improving observation efficiency.

d. **Problem Traceability:** During data processing, personnel can retrieve environmental information from the time of observation to identify root causes of problematic spectra.

e. **Environment Display:** Centralized real-time display of current observation and operational status on large screens.

Currently, various status information sources in the LAMOST operational environment remain isolated, requiring complex retrieval statements for each database type within the local network. Off-site personnel must traverse multiple proxy jumps to access the internal network, creating operational complexity and security vulnerabilities. Retrieval syntax varies across databases, resulting in high error rates and low efficiency. Cross-database retrieval across heterogeneous systems is impossible, and terminal-based retrieval alone offers limited functionality while posing risks of accidental data deletion or modification. An urgent need exists for a heterogeneous information retrieval solution.

Several methods exist for heterogeneous information access, including open-source retrieval tools such as Solr (based on Lucene), ELK, and Kata, which analyze and organize information to build unified index databases. However, these tools require extensive specialized training, feature complex usage, and involve long development cycles [5,6], making them unsuitable for LAMOST' s urgent data retrieval requirements. Addressing these practical constraints,

this paper proposes a rapid heterogeneous environment information retrieval method.

This paper first briefly introduces the composition of LAMOST environmental information, then presents the system architecture and overall design in Section 1. Detailed descriptions of module implementation and system deployment and testing follow in Sections 2 and 3.

1 System Functionality and Design

The heterogeneous environment information retrieval system organizes and consolidates environmental data from heterogeneous databases, shielding users from internal complexity while unifying and simplifying retrieval syntax to reduce data access errors. Developed to ensure original data security, the system provides multiple retrieval pathways for quick, precise data acquisition, thereby improving work efficiency.

The LAMOST heterogeneous environment information retrieval system comprises three layers based on different retrieval requirements: the information resource layer, service layer, and application layer, as illustrated in [Figure 2: see original paper].

The **information resource layer** includes the database proxy object factory module and database asynchronous proxy module. The database proxy object factory module maintains existing databases and retrievable information, while the database asynchronous proxy module handles proxies for multiple database types including MySQL and PostgreSQL.

The **application layer** consists of environment information retrieval requesters, available through both terminal clients and web clients. Terminal clients perform retrieval via command line, while web clients access the system through browsers for retrieval and result visualization.

The **service layer** functions as the environment information retrieval server, receiving client requests, converting them into database-specific retrieval syntax based on request content, and returning consolidated results to requesters.

2 System Implementation

During development, the system was divided into functionally decoupled modules including the database proxy factory module, reception module, command verification module, command parsing module, result aggregation module, and application module, simplifying logical control.

2.1 Development and Runtime Environment

The LAMOST heterogeneous environment retrieval system was developed under Python 3.7 and operates on various Linux distributions. Based on comprehen-

sive research and practical engineering requirements, we primarily adopted the following Python third-party resources:

- **Asyncio:** Python 3.7' s standard asynchronous coroutine library, providing asynchronous support for numerous third-party libraries. Coroutines are user-scheduled code segments executing in user space with extremely high efficiency, making them ideal for high-concurrency network communications [7-8].
- **Aiomysql, aiopg, aiosqlite:** Three Python 3.7 libraries supporting asynchronous database operations for MySQL, PostgreSQL, and SQLite databases, respectively.
- **Pyzmq:** The Python development library for ZeroMQ. As a high-performance open-source messaging middleware, ZeroMQ offers rich communication libraries and multiple communication modes to satisfy diverse business requirements [9].

2.2 Reception Module

The reception module is an asynchronous coroutine server responsible for receiving client retrieval requests and returning results. Based on request origin, retrieval accesses are categorized into three types: telescope internal network, National Astronomical Observatories internal network, and wide-area internet, with the telescope internal network being the primary operational mode. To ensure stable environmental information service, the system limits the number of external network retrieval requests, rejecting new connections once the limit is reached.

2.3 Data Module

The data module comprises the database proxy object factory module and asynchronous data proxy module.

LAMOST' s observation and maintenance requirements for environmental data retrieval are relatively fixed. Based on common retrieval needs, the system implements a database proxy object factory module. As shown in [Figure 3: see original paper], this module functions as a custom virtual database that assigns numbers to all current databases and maps custom retrieval data names to corresponding databases and field names.

The database asynchronous proxy module serves as the system's database proxy, handling specific data retrieval and result reception. Based on database type, it includes MySQL asynchronous proxy, PostgreSQL asynchronous proxy, and SQLite asynchronous proxy. To improve database utilization, each proxy module maintains a connection pool with multiple database connections to ensure real-time retrieval for multiple clients.

2.4 Command Verification Module

The command verification module validates command format, content, and permissions. Given diverse retrieval requirements and varying user authorization levels, retrieval portability and security are paramount. shows the custom simplified retrieval command format; only commands conforming to this format proceed to subsequent operations. To enhance data security, the system performs security checks on user-submitted requests, permitting only retrieval operations on databases.

TABLE:1 Custom Search Command Format

Command format	Function
`select <name>`	Data within 24 hours
`select <name> <new>`	The newest data
`select <name> <start time>`	Data from start time to current time
`select <name> <start time> <end time>`	Data from start time to end time

2.5 Command Parsing Module

The command parsing module analyzes, converts retrieval commands, and selects appropriate asynchronous database proxies. The database proxy object factory module contains no specific telescope operational environment information. Based on retrieval content, the parsing module obtains database location and corresponding field names from the factory module, reconstructs retrieval statements, selects the appropriate asynchronous database proxy based on database information, and submits the statements for execution.

2.7 Application Module

The application module handles retrieval command submission and result display, available through command-line clients and web clients.

LAMOST' s internal network contains hundreds of computers with diverse operating systems, making individual client installation impractical. Since maintenance staff only need to retrieve environmental data without further analysis or recording, we utilize the built-in NetCat tool package in Linux and MacOS for environmental data retrieval. Windows systems can download and configure NetCat with minimal setup. [Figure 4: see original paper] demonstrates NetCat-based command-line retrieval of operational environment information.

The HTTP proxy module provides web services for external access, using Apache httpd as the web server. As shown in [Figure 5: see original paper], the HTTP proxy converts browser-sent HTTP requests into custom retrieval commands, forwards them to the service layer, then transforms retrieval results back into HTTP responses for browser display.

3 System Testing

To verify whether the system meets LAMOST operational environment retrieval requirements, we deployed it on a multi-segment server within LAMOST to provide retrieval services across different network domains. [Figure 6: see original paper] illustrates the system deployment network topology, showing simultaneous service provision to the LAMOST internal network, National Astronomical Observatories internal network, and external internet.

The multi-segment server is configured with an Intel Xeon processor (4 CPUs, 14 cores each) and 128 GB memory, co-located with databases inside the LAMOST telescope. The LAMOST internal network test computer uses an Intel Core i3-2100 (dual-core) with 4 GB memory, while the National Astronomical Observatories internal network and external network test computers use an Intel Core i7-4790 (dual-core) with 16 GB memory.

As the server handles other network transmission and data processing tasks, we limited testing to 100 concurrent clients per network segment, with each client executing 3,000 retrieval requests to avoid impacting normal operations.

As shown in [Figure 7: see original paper], across 300,000 retrievals, the 172 network segment completed in 9.936 seconds (average 0.033 ms per retrieval), the 10 network segment in 14.448 seconds (average 0.048 ms), and the external network in 21.155 seconds (average 0.071 ms). During this period, server CPU utilization was 43.5% and memory consumption was negligible. Although external network retrieval required the most time, the 0.071 ms retrieval speed fully satisfies operational requirements with low server resource consumption that does not affect other applications.

4 Summary and Outlook

This system leverages Python 3.7' s latest asynchronous coroutine features to reduce information acquisition difficulty and improve system throughput. Through the database proxy object factory pattern, it integrates multiple databases and enhances system extensibility. The custom retrieval command approach simplifies the user interface, improves usability, and enables staff to promptly access telescope operational environment information. The command screening mechanism protects sensitive operational details of the large telescope, improves service precision, and maintains data security.

The system provides a foundational framework for future unified management of all LAMOST databases and establishes multiple database interfaces to facilitate subsequent information resource expansion.

References

[1] Zhao Yongheng. Technology of automatic observation of astronomical telescope [J]. E-Science Technology & Application, 2012, 3(4): 11-16.

[2] Luo Ali, Tian Yuan, Song Jing, et al. Design and implementation of LAMOST observatory control system [J]. E-Science Technology & Application, 2012, 3(4): 76-85.

[3] Li Weimin, Xing Xiaozheng, Hu Hongzhuan, et al. Research on the parallel control system of the multioptical fiber on the spherical focal plate for LAMOST [J]. Chinese journal of mechanical engineering, 2002, 38(7): 116-120.

[4] Li Zhangqi. Construction of historical data center based on heterogeneous database [J]. Electronic Technology & Software Engineering, 2019, (18): 154-158.

[5] Yin Qi, Li Qing, Huang Peng. Design of aircraft fault heterogeneous information retrieval system based on Solr [J]. Aeronautical Science & Technology, 2017, 28(04): 30-36.

[6] Pan Chunhua. Research and practice of multi-level monitoring platform for university data center operation and maintenance based on ELK [J]. China Education Information, 2020, (07), 93-96.

[7] Liu Jian, Huang Caisheng. Research on high concurrency architecture based on CO process [J]. Digital technology & application, 2018, 36(04): 85-86.

[8] Tian Yuan, Wang Feng, Li Jian, et al. LAMOST resource monitoring system based on python coroutines technique [J]. Astronomical research technology, 2018, 15(4): 456-464.

[9] Deng Hui, Zhong Wenjie, Fu Yingxue, etl. The Design of Communication Framework for a New Generation of Telescope Automous Control System Based on ZeroMQ [J], Astronomical research and technology, 2018, 15(03), 308-314.

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