Design and Implementation of the Astronomical Observation Data Transmission Log System for Xinjiang Astronomical Observatory (Postprint)

Authors: Zhu Yan, Zhang Hailong, Ye Xinchen, Wang Jie, Wang Wanqiong, Tohtinur, Li Jia, Zhang Meng

Date: 2018-11-01T00:00:00+00:00

Abstract

Observation data generated by telescopes at the Xinjiang Astronomical Observatory is transmitted via dedicated data lines to the headquarters data center for long-term storage. The data transmission employs rsync; however, the process lacks detailed log records, making statistical analysis of transmission status impossible. This paper investigates the data transmission requirements of the Xinjiang Astronomical Observatory and designs and develops an observation data transmission logging system for the observatory. Based on the data storage servers at the Nanshan and Qitai stations of the Xinjiang Astronomical Observatory and the long-term storage at headquarters, the system designs the overall architecture of the logging system and implements log recording for the data transmission process. Utilizing shell multithreading technology and the Qt framework, the system develops backend service programs and a software interface for the logging system, enabling log analysis, statistical backup, and data transmission visualization pages.

Full Text

Design and Implementation of the Xinjiang Astronomical Observatory Observation Data Transmission Log System

Zhu Yan[12], Zhang Hailong[13]*, Ye Xinchen[1], Wang Jie[1], Wang Wanqiong[1], Tohtonur[1], Li Jia[1], Zhang Meng[12]

[1]Xinjiang Astronomical Observatory, Chinese Academy of Sciences, Urumqi 830011, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]Key Laboratory of Radio Astronomy, Chinese Academy of Sciences, Nanjing 210008, China

Abstract: Observation data generated by Xinjiang Astronomical Observatory telescopes is transmitted via dedicated data lines to the headquarters data center for long-term storage. While rsync is employed for data transfer, it lacks detailed logging capabilities, preventing statistical analysis of transmission performance. This paper investigates the data transmission requirements of Xinjiang Astronomical Observatory and presents the design and development of a comprehensive observation data transmission log system. Based on the data storage servers at Nanshan Station and Qitai Station, along with long-term storage infrastructure at the headquarters, the overall system architecture was designed to implement detailed logging throughout the data transmission process. Utilizing shell multi-threading technology and the Qt framework, we developed backend service programs and a graphical log system interface that enable log analysis, statistical backup, and data transmission visualization.

Keywords: Log; Data Transmission; Database; Visualization

0 Introduction

Due to the unique nature of astronomical observations, telescope observatories are often located far from astronomical data storage centers, requiring scientific data to be transmitted via dedicated lines to data centers for permanent storage. These exposed transmission lines frequently encounter various issues that interrupt data transfer. Effective control systems and detailed logging mechanisms are essential for promptly identifying problems and retransmitting data after fault recovery. An intuitive display and visual management interface would greatly facilitate data administrators in performing statistical analysis and monitoring. Log recording during transmission and subsequent log analysis constitute critical components of astronomical data transmission systems and represent the primary focus of this work. Based on practical challenges encountered during data transmission at Xinjiang Astronomical Observatory, we designed and developed a comprehensive data transmission log system that implements data query, statistics, and analysis functionalities.

Wicenec et al. developed the Next Generation Archive System (NGAS)[12], primarily designed to handle large data streams from telescopes. NGAS effectively meets the growing demands for data transmission and management from observational facilities, providing a URL-based command interface that supports dozens of commands including archiving and retrieval. Currently, NGAS has been deployed for data transmission across multiple data centers, with ALMA, MWA, and FAST already utilizing the system, and the Shanghai Astronomical Observatory planning its adoption. Rsync is a data mirroring and backup tool that synchronizes files between local and remote hosts by transferring only differential data, offering high transmission efficiency[3]. However, rsync logs are output to syslog by default, and while it supports specifying log file locations and customizing log fields, its logging information is too simplistic to meet the requirements for status monitoring and problem detection in massive astronomical data transfers. Real-world astronomical data transmission demands more

comprehensive logging content.

Logs provide audit trails of activities occurring on target systems and should remain simple and efficient while meeting requirements. Existing mature logging systems such as Glog[2] are overly complex and resource-intensive[4]. Neither NGAS nor rsync support multi-destination log output, and rsync lacks log rotation functionality. Requirements such as log file rotation configuration, multi-destination output, and writing daily or folder-based statistical results to both log files and database tables remain unmet by existing systems.

Funding: This work was supported by the National Basic Research Program of China (973 Program) (2015CB857100), the National Natural Science Foundation of China (11873082, U1531125, 11803080, 11503075), the Youth Innovation Promotion Association of the Chinese Academy of Sciences, and the CAS Astronomical Observatory Equipment Update and Major Instrument Operation Special Fund.

Received: 2018-09-04; **Revised:** 2018-09-08

Author Introductions: Zhu Yan, female, master' s student, research direction: astronomical information technology. Email: zhuyan@xao.ac.cn. Corresponding author: Zhang Hailong, male, senior engineer, research direction: data-intensive research. Email: zhanghailong@xao.ac.cn.

1 Astronomical Data Transmission Log System Design

Xinjiang Astronomical Observatory operates a 26m radio telescope at Nanshan Station, approximately 100km from the observatory headquarters. Raw observation data is transmitted via dedicated lines to the headquarters for offsite backup[5]. This log system is built upon the rsync software tool.

1.1 Overall System Architecture

[Figure 1: see original paper] Log System Structure Diagram

As shown in Figure 1, the log system comprises three main components: log collection, storage, and retrieval. The log collector periodically scans the local observation data storage directories and processes the results, with scan intervals configured in the configuration file. The storage service process receives output from the collector and stores it both as database table entries and log files. The log retrieval system displays log information based on administrator queries, controls backend service programs (start, stop, restart), and modifies configuration files.

1.2 Key Implementation Details

[Figure 2: see original paper] Implementation Details Diagram

As illustrated in Figure 2, the system implements a robust monitoring architecture consisting of monitoring scripts, control scripts, and monitoring objects. Two monitoring scripts execute monitoring tasks and mutually supervise each other. The control script manages input parameters for various scripts, such as loop intervals, based on configuration file settings. Monitoring objects comprise two components: update checking and data writing. The update checking script detects data differences between sender and receiver to determine whether synchronization is required, and exists only on the receiver-side log system. The data writing component consists of file information logging and statistical information logging. The file information script records details of newly added files from the latest storage directory scan, while the statistical information script calculates daily increments for the root storage directory and its subfolders.

1.2.1 Mutual Monitoring System Shell scripts on Linux systems may terminate abnormally or exit unexpectedly. The log system must continuously monitor to ensure stable operation. Monitoring Script 1, Monitoring Script 2, and the monitoring objects form a robust mutual monitoring system. The two monitoring scripts supervise each other while simultaneously monitoring the monitoring objects. The primary task of monitoring scripts is to check the running status of target processes; any script not present in the process list is immediately restarted. This architecture ensures that monitoring objects remain continuously operational, enabling the log system to run stably for extended periods without human intervention.

1.2.2 Linux Implementation The control script must run multiple scripts concurrently, requiring multiple background processes. The system implements multi-threading by appending the `&` operator to execution commands: `./script_{name}.sh &`. While the `&` operator makes the command immune to SIGINT signals and executes it in a subshell, the process terminates when its terminal closes. True background execution requires prefixing with `nohup` to immunize against SIGHUP signals: `nohup ./script_{name}.sh &`. The monitoring objects contain multiple scripts that are launched simultaneously using shell multi-threading technology. If any script within the monitoring object stops running, the control script automatically restarts it.

2 Implementation Details

2.1 Development Languages

The backend programs of the Xinjiang Astronomical Observatory data transmission log system are implemented in shell scripts, while the log retrieval system was developed using Qt in the Qt Creator integrated development environment. Shell serves as the bridge between users and the Linux system, accepting commands from users and executing them in the kernel. Shell scripts are programs composed of shell commands, featuring branching control structures and loop structures characteristic of programming languages, enabling convenient sta-

tistical computations on Linux through command combinations. Qt is a cross-platform C++ graphical user interface application development framework with high modularity and reusability that simplifies the development and maintenance of GUI applications[6], with internal components communicating through Qt' s unique signal-slot mechanism[7].

2.2 Log File Structure

The data transmission log system generates two types of logs. The first, called ordinary logs, records processed statistical information stored locally on machines, saved as text files in date-named log files that are created whenever data is stored. The second, called system logs, records activities generated by the log retrieval system itself, including login/logout records and interface operation logs that track user behavior trajectories. System log naming rules are determined by the configuration file, which provides both size-based and time-based rotation options. Certain information from ordinary logs must also be written to database tables.

(1) Ordinary Logs

Ordinary logs are written to text files while certain fields are simultaneously written to database tables. The output format is:

```
[<Timestamp>] [<log type>] <folder path> <file number> <data volume> <last modify time>
```

Example: `[2018-06-04 19:28:22][INFO]`

Field definitions: `<timestamp>` records the log date and time; `<log type>` represents the log level, which can be "INFO" , "WARNING" , "ERROR" , or "NOTICE" ; `<folder path>` indicates the absolute path of each folder under the data storage path; `<file number>` shows the file count in that folder; `<data volume>` represents the total file size in MB; `<last modify time>` indicates the folder' s last modification time.

(2) System Logs

System logs are written to text files with the following format:

```
<type>|<timestamp>|<user>|<operation>|<source file>|<log type>=<log message>,<status>
```

Example: `SYSTEM_{LOGIN}|2018-06-20 18:23:21|sam|logon|logindialog.cpp|INFO=log into system,SUCCEED`

Field definitions: `<type>` indicates log category (login or operation); `<timestamp>` records the time; `<user>` shows the administrator name; `<operation>` describes the action; `<source file>` indicates the source file; `<log type>` shows the log level; `<log message>` provides a descriptive string; `<status>` indicates operation success or failure.

2.3 Database Tables

The system implements six database tables:

1. **Files Table**: Records filename, file size, storage time, last modification time, and MD5 checksum for each file.
2. **Astronomical Data Table**: Stores telescope data storage information including total data volume (MB), total file count, storage time, and last modification time (users can configure recording intervals, e.g., 2 minutes).
3. **Folder Table**: Records data information for folders under the storage root directory, including auto-increment field, folder traversal count, folder name, total data volume, total file count, record time, last modification time, and total folder count under root.
4. **Daily Data Increment Table**: Tracks daily changes in astronomical data storage, including auto-increment field, date, daily added data volume, daily added file count, and reserved fields.
5. **Daily Folder Data Increment Table**: Records daily changes for each folder under the root directory, including auto-increment ID, date, folder name, folder data volume, folder file count, daily added data volume, daily added file count, and total folder count under root.
6. **Script Process Status Table**: Monitors backend script operation status for four scripts: data writing, control, daily statistics, and rsync data transmission, with update timestamps.

Database Table Structure

3 Log Retrieval System Implementation

The log retrieval system serves system administrators by providing a user-friendly interface for convenient log query and operation. To facilitate control of the entire astronomical data transmission log system, the interface enables administrators to start, stop, or restart backend programs and modify configuration parameters to control log output behavior.

3.1 Functional Architecture

[Figure 3: see original paper] Structure of Log Retrieval System

As shown in Figure 3, the log retrieval system comprises four functional modules: ordinary log retrieval, classified log query, log backup, and parameter management control.

3.2 Sub-module Descriptions

As shown in Figure 4, after successful login, the interface displays data storage status for both sender and receiver, including system status, total file volume, and total file count for each end. System status shows disk usage, utilization percentage, and available space.

[Figure 4: see original paper] Summary of Data Storage at Sender and Receiver

3.2.1 Ordinary Log Retrieval [Figure 5: see original paper] Ordinary Log File Retrieval

As shown in Figure 5, the ordinary log retrieval interface displays detailed file records including filename, file size, and file creation time. It supports querying records within specific time ranges down to the second, and keyword-based searches using filenames or partial filenames. Top buttons allow selection of sender or receiver data. The backup button at the bottom supports saving current search results to a local default directory configured in the Configure page.

3.2.2 Classified Log Query This interface categorically displays system log content generated by user interactions with the log retrieval system, including login logs and operation logs.

[Figure 6: see original paper] Log Overview
[Figure 7: see original paper] Different Types of Log Queries (Login Log)
[Figure 8: see original paper] Different Types of Log Queries (Operation Log)

Figure 6 shows the system log overview with categories and quantities. Figure 7 illustrates login logs recording each user' s login/logout time, status, action (login/logout), and descriptions. Figure 8 shows operation logs detailing user behavior trajectories, including operation time, content (e.g., query, backup), status (success/failure), and detailed descriptions.

3.2.3 Parameter Management Control This interface controls backend service startup, shutdown, restart, and parameter configuration.

[Figure 9: see original paper] System Configuration

Figure 9 shows the system configuration interface that automatically displays existing parameters:

- **OrdinaryLogBackupDir**: Default local path for remote backup of ordinary logs
- **SystemLogBackupDir**: Default path for system log backup
- **OrdinaryLogMaxSize**: Maximum size for individual ordinary log files (default 5MB); new files are created when this limit is exceeded
- **OrdinaryLogInterval**: Interval for writing log files, i.e., backend program scan interval for root directory (default 2 minutes)
- **MonitorInterval**: Backend program monitoring interval for checking shell script status (default 5 minutes)

Three checkboxes correspond to OrdinaryLogMaxSize, OrdinaryLogInterval, and MonitorInterval. Changed settings require clicking the restart button to reload backend services. Start and stop buttons control backend service operation.

3.2.4 Log Backup This interface provides both ordinary and system log backup functionality. Ordinary log backup supports archiving logs from sender or receiver servers, while system log backup supports archiving software interface logs.

[Figure 10: see original paper] Log Backup

As shown in Figure 10, system log backup supports archiving local logs from before today, six months ago, or one year ago to the default configured path. Ordinary log backup supports compressing and downloading log files for user-selected date ranges, including per-file, per-folder, and daily root directory change logs, using the backup path configured in the settings page.

4 Data Transmission Visualization

Building upon the log system, we developed a data transmission visualization page using HTML5, AJAX, PHP, and other web technologies, as shown in Figure 12. The visualization page displays transmission system status, enabling administrators and relevant personnel to monitor data transfer conditions. The page consists of six modules: (1) bar charts showing daily data storage and transmission; (2) bar charts displaying weekly data statistics; (3) server heartbeat monitoring showing sender and receiver status with automatic alerts upon anomalies; (4) pie charts illustrating server storage status to help managers determine when to add storage devices; (5) real-time transmission status from log records; and (6) line graphs showing data transmission rates over the past hour.

[Figure 11: see original paper] Visualization Page

The visualization system intuitively displays log content, providing administrators with convenient tools to assess system operation and identify problems promptly. Developed using modular techniques, the system can be integrated into larger observatory control systems in the future.

Conclusion

Based on the actual data transmission requirements of Xinjiang Astronomical Observatory, we constructed a log collection-storage-retrieval architecture and deployed backend service programs on both sender and receiver servers. We innovatively proposed a mutual monitoring system for script programs, implementing ordinary logs that record file information and MD5 checksums, along with statistical logs organized by day and storage folder. The receiver-side system additionally features automatic data update detection. The log retrieval system, installed as client software on third-party devices, reads configuration files to automatically connect to databases on both servers, enabling time-based and keyword-based remote log queries, backup of search results, and viewing of local login and operation logs with compression capabilities. The system supports configuration parameter modification and provides start/stop/restart functionality for both server systems. The data transmission visualization page

displays log content via web interface, assisting administrators in monitoring system status. The modular development approach facilitates future migration and reuse.

References

[1] Wicenec A, Knudstrup J, Johnston S. ESO' s Next Generation Archive System[J]. Messenger, 2002, 129:27-31
[2] Wicenec, A.; Knudstrup, J. ESO' s Next Generation Archive System in Full Operation[J]. Messenger, 2007, 129:27-31
[3] Zuo Zhende. Research and Tool Development of Real-time High-speed Backup for Structured Database Based on Rsync[D]. South China University of Technology, 2017.
[4] Yang Shanning. The Design and Implementation of a Lightweight and High-efficient Cross-platform Logging System[J]. Computer Knowledge and Technology, 2017, 13(28):77-78+115.
[5] Zhang Hailong, Zhu Yan, Nie Jun, Yuan Jianping, Wu Gang, Liu Jun, Wang Jie, Wang Wanqiong, Ye Xinchen, Tohtonur, Zhang Meng. Xinjiang Astronomical Observatory NSRT Data Storage System[J]. Astronomical Research & Technology, 2018, 15(02):181-187.
[6] Rischpater R. Application Development with Qt Creator[J]. 2014.
[7] Blanchette J, Summerfield M. C++ GUI Programming with Qt 4[J]. 2006, 45(6):747-749.

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