National Astronomical Observatories
Chinese Academy of Sciences
National Astronomical Data Center
国家天文科学数据中心

YourFS:A read only file system for data non-landing scenarios in the astronomical cloud

Jun Han (hanjun@nao.cas.cn)

National Astronomical Observatories, Chinese Academy of Science, Beijing, China
National Astronomical Data Center, Beijing, China

Abstract: With astronomy entering an era of petabyte to exabyte scale data from next-generation telescopes and surveys, existing cloud platforms face critical data management challenges. Traditional approaches of mounting entire datasets or copying filtered subsets create trade-offs between access efficiency and storage costs, while conventional storage engines lack flexible permission control. To address this, this paper presents a POSIX-compliant read-only file system designed for non-landing data scenarios. The system integrates FUSE for POSIX interface implementation, libnfs for server-side data retrieval, and a database-driven metadata layer that enables file-level access isolation through SQL-defined policies. Performance testing indicates that maintaining under 5,000 files per directory ensures optimal user experience, while read operations achieve speeds of more than 50MB/s under a gigabit local area network – sufficient for most application scenarios in astronomical data workflows. Our results indicate that YourFS effectively balances accessibility and cost-efficiency for massive datasets, providing a viable paradigm for server-side data management in cloud platforms.

Backgroud: Making data open is a tradition in astronomy, and data storage and computing have been initially solved through large data center. Cyber-infrastructure is becoming a crucial part for scientific research in astronomy, as the data sets are too large for astronomers to download and analyze using their own computers. Modern science's reliance on enormous datasets have jointly precipitated a critical demand for server-side analytical infrastructures, and usually a cloud scientific platform is provided for astronomers to work.

While cloud computing scientific platforms provide certain working environments, they still exhibit shortcomings in data management. The volume of astronomical data is immense, with individual datasets containing millions or even tens of millions of files. Different scientists have varying research focuses and require distinct subsets of data. To facilitate data access for scientific users, platforms often mount entire datasets or copy filtered results to users. However, researchers typically only need to examine a small fraction of the data. While the mounting approach reduces data migration costs, it diminishes users' efficiency in locating specific data. Conversely, the copying method improves usability for researchers but dramatically increases data storage costs. For scientific users, data requirements are not static, and accessible scopes dynamically expand as data protection periods expire. Considering that the vast majority of current users prefer POSIX compliance file systems and the widely adopted NFS, this paper proposes a read-only file system with file isolation capabilities, which implements essential POSIX interfaces.

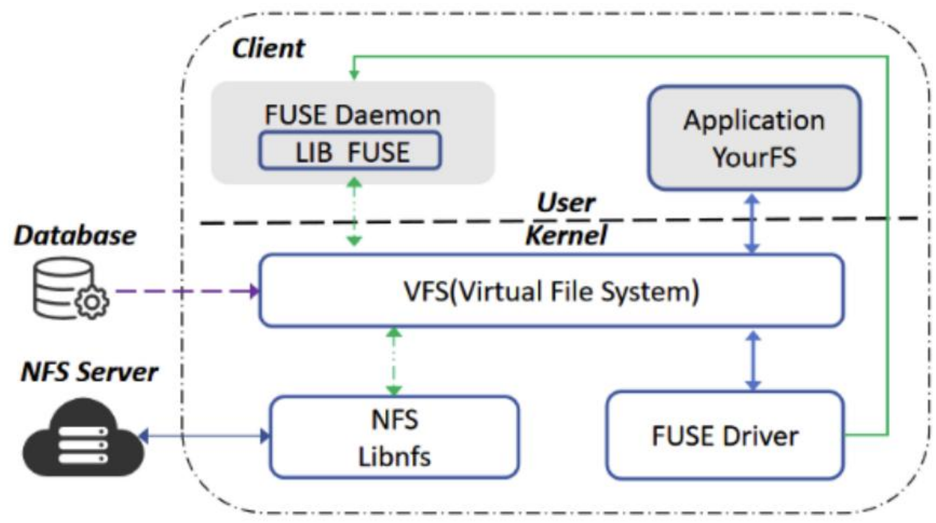

Figure 1. The technical architecture of YourFS. The proposed system ensures POSIX compliance through FUSE implementation, facilitates server-side data retrieval via libnfs integration, and implements access scope definitions through database-driven metadata constraints, achieving both protocol standardization and security enforcement.It enables users to transparently mount remote NFS shares on local systems through an interface while defining file access scopes via a database, thereby achieving seamless cross-platform data sharing.

The Architecture and Runtime: This file system is designed to be user-centric and fully customized, delivering a "what you envision is what you obtain" experience, thus designated as YourFS. The architecture is shown in figure 1. The software is developed in C programming language, and the release version 1.0 has been published in github. To achieve POSIX compliance, server-side data access, and data isolation capabilities, the system employs FUSE to implement POSIX interface design. It dynamically invokes libnfs for data loading operations and utilizes a database to define file accessibility scopes, thereby establishing file-level access isolation mechanisms through metadata-based permission constraints.

When a user process(e.g., ls command) is executed, the system call readdir() to request directory contents. It is intercepted by VFS layer and forwards it to FUSE Driver. The driver encapsulates the request into a FUSE protocol message and writes it to the /dev/fuse character device. The FUSE Daemon monitors the device via libfuse, which decodes the protocol message and triggers the readdir() callback function registered in YourFS. Then YourFS executes its custom logic to retrieve directory in the database and queries metadata from a backend source via libnfs. The metadata is returned to libfuse and writed to /dev/fuse via libfuse. The driver extracts the entries, passes them to VFS, and VFS populates the user process's buffer. The ls command receives the entries and displays them on the terminal.

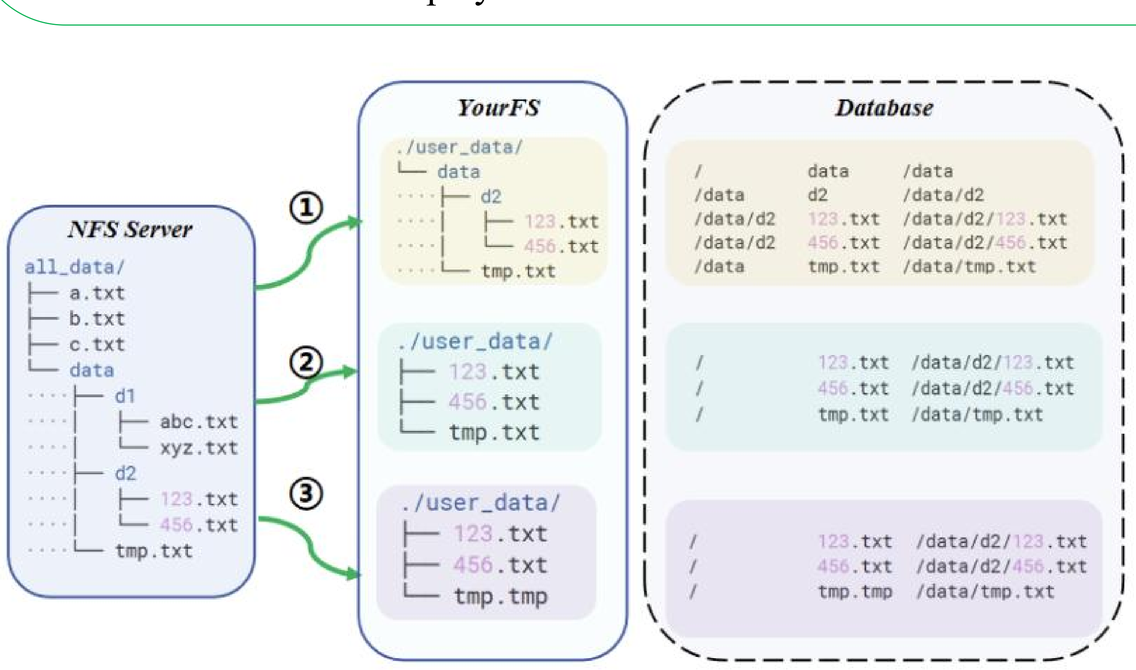

Figure 2. Three examples of directory listing customization. The left shows the files in NFS Server, the middle displays the mounted YourFS view (Partial File Display, Restructured Directory Hierarchy, Alias Creation for Specific Files), and the right presents the raw content stored in the database.

Usage and Test: After cross-matching datasets, users often need to access only a handful of files, which are typically scattered across deeply nested directory hierarchies. The first example is used to display partial files. You only need to write the file information involved to the data table. Example 2 demonstrates a directory restructuring example where modifying the parent directory hierarchy consolidates these files into a unified view, achieving a flat, user-friendly structure for data access. Example 3 demonstrates how users can tag special files by creating aliases. This allows files to appear under modified names or paths in the virtual filesystem without altering their physical storage locations on the NFS server.Performance testing indicates that maintaining under 5,000 files per directory ensures optimal user experience, while read operations achieve speeds of more than 50MB/s under a gigabit local area network – sufficient for most application scenarios in astronomical data workflows. The upcoming release v2.0 will implement advanced caching optimizations to boost software performance and significantly increase data query efficiency.