YOURFS: A READ-ONLY FILE SYSTEM FOR NON-LANDING DATA SCENARIOS ON ASTRONOMICAL SCIENCE PLATFORMS

By Jun Han

National Astronomical Observatories, Chinese Academy of Sciences, and National Astronomical Data Center, 20A Datun Road, Chaoyang District, Beijing, China

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 that, this paper presents a POSIX-compliant read-only file system designed for non-landing data scenarios. The system integrates the FUSE (File system in USEr space) 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. This software aims to provide a curated and user-customized virtual view of massive datasets. It is suitable for data-filtering and small-file reading scenarios and can fully meet user requirements in those scenarios. The results demonstrate that YourFS effectively balances the accessibility and cost-efficiency of massive datasets, offering a viable technical paradigm for server-side data management in cloud-based science platforms.

Introduction

Astronomy has entered an era of big data due to the development of numerous ground-based and space-based photometric, spectroscopic, and digital sky surveys. For the projects constructed in the past few years, the data volume of sky surveys such as the Sloan Digital Sky Survey, *Large Sky Area Multi-Object Fiber Spectroscopic Telescope* (*LAMOST*), *Gaia*, and the *Transiting Exoplanet Survey Satellite* are still in the range of hundreds of terabytes [1,2]. In recent years, several next-generation large telescopes and surveys around the world, *e.g.*, the *Square Kilometre Array*, the *Five-Hundred-Meter Aperture Spherical Radio Telescope*, and the *Vera C. Rubin Telescope*, have been constructed, resulting in a sharp increase in data volume, and the data volume has already begun to reach the scale of petabytes (PB) or even exabytes (EB) [3–5].

Making data open is a tradition in astronomy, and the problems of data-storage and computing have been initially solved through large data centres. Cyber-infrastructure is becoming a crucial part of daily scientific research in astronomy, as the data sets are too large for astronomers to download and analyse using their own computers. Several pioneers also have designed many on-line services and tools to make data easily discoverable, such as SkyServer and CasJobs to provide browser and sql interfaces [6], and *Simbad* and *VizieR* to allow astronomers to query, explore, and analyse data on-line [7,8]. Modern science's reliance on enormous data sets, particularly evident in astronomy, has precipitated a critical demand for server-side analytical infrastructures, and usually a cloud scientific platform is provided for astronomers to work [9,10].

Science platforms offer a range of working environments yet exhibit shortcomings in data management. In terms of data management, different science platforms have proposed

distinct solutions, which are technically divided into two categories. One category involves creating datasets based on scientific requirements and mounting them *via* links to provide data support for users' working environments[11–14]. The other category offers users access to interfaces through an application-programming interface (API) approach[15,16]. Given that the vast majority of astronomical software performs data processing by directly reading files through the POSIX protocol, the first method is more universally applicable. While the second method offers greater flexibility, it is not supported by most astronomical software and is thus more suitable for some newly developed astronomical software applications. The volume of astronomy data is immense, with individual datasets containing millions or even tens of millions of files. For instance, each sky-survey campaign of *LAMOST* yields an extensive set of spectral files[*], and different scientists have varying research focusses and require distinct subsets of data[17]. To facilitate data access for scientific users, platforms often mount entire datasets or copy filtered results to users. However, researchers typically need only 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 common storage engines such as Ceph, BeeGFS, and GlusterFS, access permissions are typically controlled through ACLs (Access Control Lists). However, those implementations are often tied to hardware configurations, resulting in significant usability limitations and practical inconveniences.

For scientific users, data requirements are not static, and accessible scopes dynamically expand as data-protection periods expire or data-retrieval scopes change. Considering that the vast majority of current users prefer POSIX-compliant file systems and the widely adopted NFS (Network File System), this paper proposes a read-only file system with file-isolation capabilities, which implements essential POSIX interfaces. It enables users to mount remotely NFS shares transparently on local systems through an interface while defining file access scopes *via* a database, thereby achieving seamless cross-platform data sharing. In the technical-architecture section, this paper details the technical architecture and design of the file system, followed by a section which introduces the software-usage methods and application scenarios in astronomical science platforms. The last section presents system-test results and performance evaluation across usage scenarios.

The technical architecture

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 software is developed in the C programming language, with its release version (v1.0) published and open-sourced on GitHub[†]. To achieve POSIX compliance, server-side data access, and data-isolation capabilities, the system employs FUSE (Filesystem in Userspace)[‡] to implement the 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. That architecture ensures both protocol compatibility and granular security control in multi-user environments. The technical architecture is shown in Fig. 1.

Fuse and NFS

FUSE establishes a dual-layer architecture for user-space file-system development, con-

[*] https://www.lamost.org/lmusers/

[†] https://github.com/techwander/yourfs

[‡] https://github.com/libfuse/libfuse

[§] https://github.com/sahlberg/libnfs

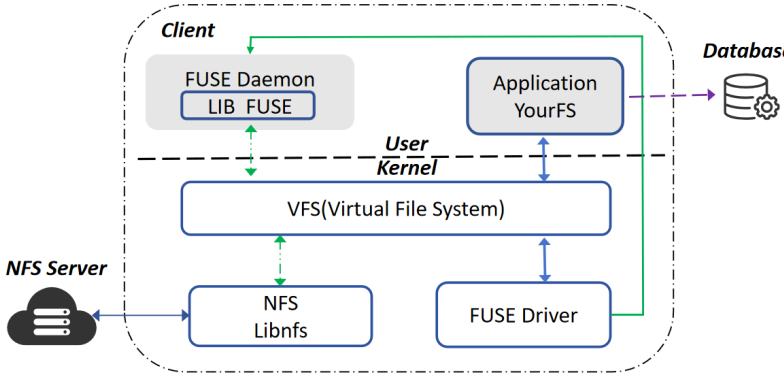

Fig. 1

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.

sisting of a kernel-resident module and the libfuse user-space library. The kernel module mediates VFS [virtual file system] interactions *via* the `/dev/fuse` interface, while libfuse provides standardized programming abstractions for implementing file-system logic in user space, effectively bridging application-level operations with kernel-level file-system services. It demonstrates cross-version compatibility across Linux distributions, and that compatibility framework has made FUSE the *de facto* solution for implementing POSIX interfaces in modern distributed file systems.

NFS is designed to achieve cross-network file sharing and transparent access. Its core design philosophy enables client systems to access remote-server storage resources with the same operational semantics as local file manipulation, while maintaining cross-platform compatibility. As one of the most prevalent distributed file systems, NFS employs libnfs as its critical protocol-implementation library, which a supports cross-platform development interface, the NFSv3/v4 protocol, Kerberos authentication, and Access Control List (ACL) permission models.

Database

The database subsystem implements a metadata-driven authorization mechanism, where NFS-accessible paths are explicitly defined through sql-based policy configurations. That design ensures end-to-end access control validation before any file operation reaches the libnfs protocol layer. The database table implements a hierarchical storage model through three essential columns: parent, name, and realpath. Parent is the parent path of the file, name is the name of the file, and realpath is the absolute file path of the file in the NFS server mount point. This system implements conditional access control through database-driven authorization. When granting file access to a specific user, the corresponding lists are inserted into the designated database table. Conversely, those records remain unpopulated if access permission is not granted. The database tables can be configured for multi-user shared access with identical data-accessibility scopes, or alternatively, user-specific schemas may be independently defined to establish distinct access boundaries through metadata-based isolation mechanisms.

Runtime

The following describes the runtime logic when a user lists files in a directory in the YourFS file system. When a user process (*e.g.*, the `ls` command) is executed, the system calls `readdir()` to request directory contents. It is intercepted by the VFS layer and forwards it to the 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 the directory in the database and queries metadata from a backend source *via* libnfs. The metadata is returned to libfuse and written to `/dev/fuse` *via* libfuse. The driver extracts the entries, passes them to the VFS, and the VFS populates the user process's buffer. The ls command receives the entries and displays them on the terminal.

Usage and application

In this section, we will detail the software's specific usage guidelines, including directory-organization methodology and database schema, along with its application to astronomical science platforms.

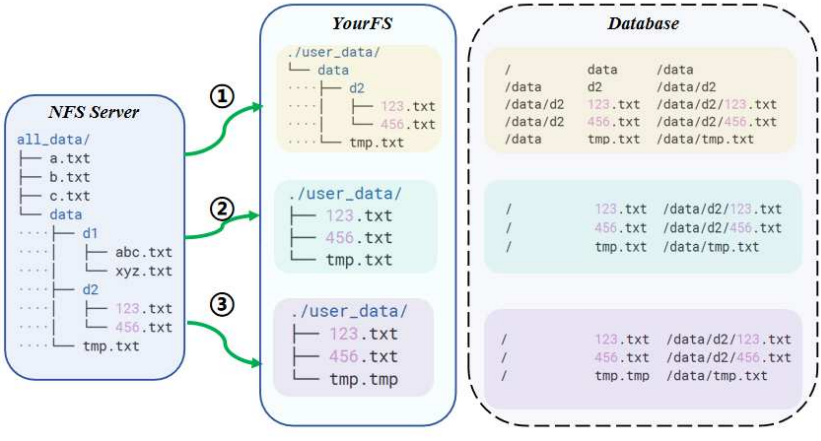

Fig. 2

This figure illustrates three examples of directory-listing customization. The left panel shows the file list in the NFS Server, the middle panel displays the mounted YourFS view (Partial File Display, Restructured Directory Hierarchy, Alias Creation for Specific Files), and the right panel presents the raw content stored in the database table.

Usage

The implementation of this file system primarily involves two phases: database construction and mounting operations. The database stores metadata defining accessible files, and the system dynamically generates a virtual directory structure based on the schema of the database tables, exposing only the authorized data to users. The NFS server hosts an enormous volume of directories and files, often reaching tens of millions of files in astronomical fields. In practical usage, however, users may require only a small subset of the data or even

just a handful of files out of that vast repository.

To enhance clarity, three use-case scenarios have been designed to demonstrate the workflow and benefits of the file-system architecture. Fig. 2 illustrates three examples of directory-listing customization. The first example is used to display partial files, and it is usually used in partial-data-release scenarios. One needs only to write the file information involved to the data table. In astronomical applications, after cross-matching datasets, users often need to access only a handful of files, which are typically scattered across deeply nested directory hierarchies. That fragmented storage structure creates significant usability challenges. Example 2 demonstrates a directory restructuring example where modifying the parent-directory hierarchy consolidates those 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. That allows files to appear under modified names or paths in the virtual file system without altering their physical storage locations on the NFS server.

The YourFS file system supports both SQLite3* and PostgreSQL† databases currently. Users can maintain backups of filtered data, enabling direct loading during subsequent usage without requiring redundant cross-referencing or retrieval operations.

Application to astronomical science platforms

In our science platform, there exists a vast amount of observational data; for instance, the *LAMOST* telescope has accumulated tens of millions of spectral-data records. In our previous solution, to facilitate user access to those data, we developed a data-retrieval system. That system allows users to obtain lists of desired data. Based on those lists, the system packages the corresponding files and generates download links. Users then utilize those links to download the data onto their data-processing virtual machines. To minimize data-download requirements and storage usage, we have integrated the platform with the YourFS system to design an automated data-retrieval and -loading system. The system framework is illustrated in Fig. 3.

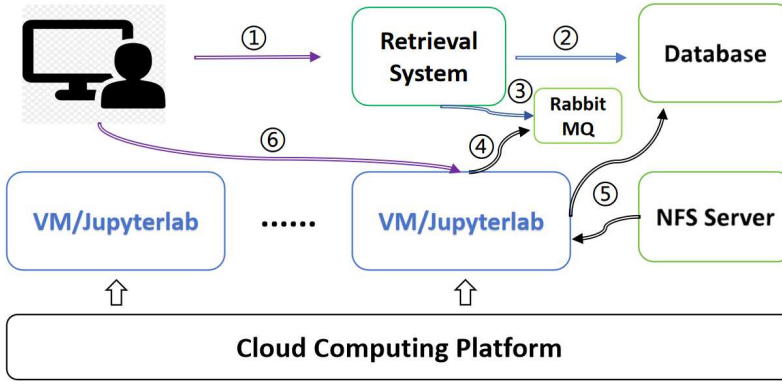

Fig. 3

This diagram depicts a typical application scenario of the YourFS file system. Integration and testing have been completed on the NADC science platform. The YourFS software is deployed on VM/JupyterLab environments to provide users with data access services.

*https://www.sqlite.org

†https://www.postgresql.org

The user first accesses the data retrieval system through a web page. Upon retrieving the required samples, the backend of the retrieval system automatically creates a data table in the metadata-management database and writes the access information to the RabbitMQ message queue, including account, mount point, database, and so on. When the user's virtual machine detects information belonging to it in the queue, the VM automatically executes the relevant scripts. Those scripts simultaneously connect to the metadata database and mount the data directory from the NFS server. Consequently, when the user accesses the virtual machine, they can immediately view the required data with minimal latency. Since only partial metadata operations are involved, that process completes exceptionally quickly — typically within 3 seconds (<10 000 files) — enabling users to gain near-real-time access to the required data.

Performance test and comparison

Performance test

User experience is a critical metric for assessing storage-system functionalities. To enable objective evaluation, this test scope covers common user operations, including listing files and reading files. We designed a testing environment where the NFS and client reside within the same local-area network (LAN). The SQLite3 and PostgreSQL databases are deployed on the client and another server within the LAN, respectively. We tested the time required to list files in a single directory as the number of files increases from 1000 to 10 000. The test result is listed in Fig. 4.

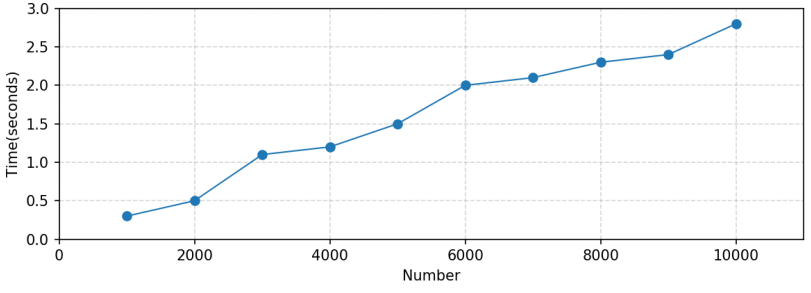

Fig. 4

This figure lists the time required (in seconds) as the number of files increases from 1000 to 10 000. Postgres is used as the database.

The tested values represent the average of 100 consecutive test iterations. The results indicate favourable response times. During testing, due to the caching mechanism, response times tended to decrease in subsequent test iterations. Notably, during the 5000-files test, the maximum response time reached over 2 seconds. In such scenarios, the user experience becomes noticeably degraded. Therefore, to ensure better performance, it is recommended to keep the number of files per directory below 5000. In the test using PostgreSQL as the database, the response times were similar to those of SQLite3.

In a gigabit local-area-network (LAN) environment, the random-file read speed can reach approximately 50 megabytes per second (MB/s) for a 1 GB file, which is around 70% of the performance of native NFS. Although significant performance gaps exist compared to the native NFS service, the solution delivers adequate user experience in most scenarios since retrieved datasets typically involve minimal data volumes. That is primarily due to the FUSE-

based file system, where frequent context switches between user space and kernel space become the dominant source of latency. The software was originally designed to provide a curated, user-customized virtual view of massive datasets for numerous fragmented small files, and its read performance fully meets the demand for instant access to small files.

Comparison with other storage engines

In the field of computer science, a diverse array of storage engines has been developed to cater to different requirements, including mainly file storage, object storage, and block storage. Those storage engines serve as the underpinning of data-storage systems, with high performance and security as their core attributes, yet they lack flexibility in terms of file-level access-permission control. Although file-permission control is achievable through the ACL mechanism, it is inherently bound to individual users. Furthermore, it does not offer the directory-structure customization approach referred to in the usage section.

The file system proposed in this paper is a secondary encapsulation architecture based on file storage. That file system can build a virtual file system on any NFS-protocol-supported file system. It is non-exclusive of other file systems, which are adopted as its underlying storage substrate. By incorporating a database layer, it introduces file-level access-control capabilities. Although its performance is slightly inferior to that of traditional storage engines, it excels in flexibility and is thus more suitable for rapidly building customized small and even medium-sized datasets, while also delivering user-acceptable performance in practical applications.

Conclusions

YourFS presents an innovative solution to critical challenges in astronomical big-data management. By integrating FUSE's POSIX compatibility with NFS's distributed-access capabilities, coupled with a database-driven dynamic permission mechanism, the system provides a method for data accessibility and storage efficiency. Performance evaluations demonstrate the system's robust capabilities in operational scenarios. To ensure optimal responsiveness, maintaining fewer than 5000 files per directory is strongly recommended.

YourFS is designed for trusted multi-user environments (*e.g.*, internal scientific platforms), where users are granted read-only access to the entire shared file system by default. Its core value lies in providing a curated, user-customized virtual view of massive datasets, rather than enforcing strict isolation for proprietary or confidential data. That design choice aligns with the open-data tradition in astronomy and the requirements of non-landing data scenarios in trusted cloud platforms. For the performance, compared with native distributed file systems, the user-space FUSE layer introduces context-switch overhead, and the NFS backend cannot fully exploit the throughput of distributed storage architectures. YourFS is optimized for lightweight, on-demand data access in cloud scientific platforms, focussing on reducing data migration and storage costs instead of pursuing peak I/O throughput for ultra-large datasets. That positioning matches the typical usage patterns of astronomers who only require small subsets of massive survey data. The software has been successfully deployed on the cloud-computing platform and public dataset-sharing applications of the Chinese National Astronomical Data Center. It delivers outstanding practical performance, especially in data filtering and small-file reading, and can fully meet user requirements.

The upcoming release version 2.0 will implement advanced caching optimizations to boost software performance and significantly increase data-query efficiency, and the source code will also be open-sourced on GitHub.

Acknowledgements

Special thanks to my son for his support. The name 'YourFS' originates from his nickname 'Youyou'. This work was supported in part by the National Key RD Program of

China (2022YFF0711500,2020SKA0120201), and in part by the Strategic Priority Research Program of the Chinese Academy of Science (XDB0550101).

References

(1) D. G. York *et al*., *AJ*, **120**, 1579, 2000.
(2) G. R. Ricker, in D. Apai & P. Gabor (eds.), *Search for Life Beyond the Solar System. Exoplanets, Biosignatures & Instruments* (https://astrobiology.nasa.gov/nai/seminars/featured-seminar-channels/conferences-and-workshops/2014/3/16/search-for-life-beyond-the-solar-system-conference/index.html), 2014.
(3) P. E. Dewdney *et al*., *Proceedings of the IEEE*, **97**, 1482, 2009.
(4) R. Nan *et al*., *Int. J. Mod. Phys. D*, **20**, 989, 2011.
(5) Ž. Ivezić *et al*., *ApJ*, **873**, 111, 2019.
(6) W. O'Mullane *et al*., *IEEE International Conference on Web Services (ICWS'05)*, **1**, 2005, p. 33.
(7) M. Wenger *et al*., *A&AS*, **143**, 9, 2000.
(8) F. Ochsenbein, P. Bauer & J. Marcout, J., *A&AS*, **143**, 23, 2000.
(9) M. Taghizadeh-Popp *et al*., *Astronomy and Computing*, **33**, 100412, 2020.
(10) C. Cui *et al*., in J. Wang *et al*.(eds.), *Proceedings of the 3rd International Conference on Wireless Communcation and Sensor Networks (WSCN 2016)* (Atlantis Press), **44**, 2017.
(11) V. Navarro, *ESA Data Exploitation Platforms: Accelerating Space and Navigation Science. SciOps 2022: Artificial Intelligence for Science and Operations in Astronomy (SCIOPS)* (Zenodo), 2022.
(12) M. Diaz *et al*., *Astronomical Data Analysis Software and Systems*, **XXVII**, 522 (Astronomical Society of the Pacific), 2020, p. 323.
(13) S. A. Russo *et al*., *Astronomy and Computing*, **41**, 100648, 2022.
(14) S. Stetzler *et al*., *AJ*, **164**, 68, 2022.
(15) M. Taghizadeh-Popp *et al*., *Astronomy and Computing*, **33**, 100412, 2020.
(16) R. Nikutta *et al*., *Astronomy and Computing*, **33**, 100411 2020.
(17) H. Yan *et al*., *The Innovation*, **3**, 100224, 2022.

REVIEWS

Conjuring the Void: The Art of Black Holes, by Lynn Gamwell (MIT Press), 2025. Pp. 208, 22·9 × 30·5 cm. Price $44·95 (about £33) (hardbound; ISBN 978 0 262 04996 2).

Gamwell is a non-fiction author and art curator, her books concentrating on the relationship between science, its history, and art. When encountering 'black holes' and 'art' in the same phrase, I usually think of Peter Galison (director of the Black Hole Initiative, philosopher, professor of the history of science and physics at Harvard, author, and film-maker), and indeed he is the first person thanked in the extensive acknowledgements. After a foreword by Neil deGrasse Tyson and an Introduction, there are three chapters of vastly different lengths: 'Black Hole Basics' (24 pages), 'Historical and Philosophical Background to Imaging Black Holes' (18), and 'Artistic and Scientific Images of Invisible Objects' (126), followed by a one-page 'Conclusion'. Each chapter has several sections (three, four, and ten, respectively) which more or less alternate between science and art (or, in either case, its history). Thus, we have the sequence (at the end of the third chapter) 'Supermassive Black Holes', 'Black Holes as a Metaphor for Nothing, Silence, and Blackness', 'Gravitational Waves Caused by Merging Black Holes', 'Immersive Art About Black Holes', and 'Direct Observation of Black Holes by the Event Horizon Telescope'.