Container-Based Deployment and Testing of Pulsar Data Processing Software Postprint

Authors: Ye Xinchen, Zhang Hailong, Wang Jie, Zhang Meng, Zhang Yazhou, Wang Wanqiong, Li Jia, Du Xu

Date: 2023-01-18T00:00:00+00:00

Abstract

With the rapid development of astronomical observation technologies, astronomical data processing software has become increasingly complex, making the deployment and management of data processing environments progressively more cumbersome. By leveraging container technology to encapsulate pulsar data processing environments into container images and establishing a private image repository, researchers can select appropriate images based on data processing requirements, thereby enabling rapid deployment of processing environments. We performed coherent dedispersion processing on pulsar baseband data across physical machines, virtual machines, and containers, comparing resource utilization and data processing efficiency among different platforms. Experimental results indicate that container performance is comparable to that of physical machines; in multi-task concurrent data processing tests, containers demonstrate more rational resource allocation, enhancing computational resource utilization efficiency relative to virtual machines. We implemented a container-based data processing architecture deployment on the pulsar data processing servers at Xinjiang Astronomical Observatory, and designed and developed a graphical user interface for container management. By optimizing functionalities including multi-user login, authentication, and data volume mount management, we improved the efficiency of employing container technology for astronomical data processing.

Full Text

Deployment and Testing of Pulsar Data Processing Software Based on Container Technology

Xinchen Ye[1,2,4], Hailong Zhang[1,2,3,4*], Jie Wang[1,2,4], Meng Zhang[1,2], Yazhou Zhang[1,2], Wanqiong Wang[1], Jia Li[1], Xu Du[1,2]

[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
[4] National Astronomical Data Center, Beijing 100101, China

Abstract

With the rapid advancement of astronomical observation technology, astronomical data processing software has become increasingly complex, making the deployment and management of data processing environments progressively cumbersome. This study employs container technology to encapsulate the pulsar data processing environment into container images and establishes a private image repository, enabling researchers to rapidly deploy processing environments by selecting appropriate images based on their specific needs. We perform coherent dedispersion on pulsar baseband data using physical machines, virtual machines, and containers to compare resource utilization and processing efficiency across different platforms. Experimental results demonstrate that container performance is comparable to that of physical machines; in multi-task concurrent processing tests, containers exhibit more rational resource allocation, significantly improving computational resource utilization efficiency compared to virtual machines. We have implemented a container-based data processing architecture on the pulsar data processing server at Xinjiang Astronomical Observatory, designing and developing a graphical user interface for container management. By optimizing functionalities such as multi-user login, authentication, and data volume mount management, we enhance the efficiency of astronomical data processing using container technology.

Keywords: pulsar; data processing; container; virtualization

Pulsars are the remnants of massive stars that have undergone core collapse and supernova explosions at the end of their evolution, characterized by small volume, high density, rapid rotation, and strong magnetic fields. Researchers utilize pulsars to study the birth and evolution of the universe [1], and such research holds significant scientific importance for advancing astrophysics, particle physics, and satellite navigation. Since the discovery of the first radio pulsar, observational and theoretical studies have progressed rapidly, with the number of discovered pulsars continuously increasing. As of October 2022, the total number of confirmed pulsars has reached 3,341 (https://www.atnf.csiro.au/research/pulsar/psrcat/).

The Nanshan 25-meter Radio Telescope (25 m NSRT) in Xinjiang is operated by the Xinjiang Astronomical Observatory, Chinese Academy of Sciences. Researchers have used this telescope to conduct long-term timing observations of nearly 300 pulsars and have obtained large amounts of pulsar observation data

from radio telescopes such as Parkes and FAST. Commonly used pulsar data processing software and dependency libraries include FFTW, PGPLOT, PSRCAT, SIGPROC, PSRCHIVE, TEMPO/TEMPO2, DSPSR, and PINT, among others, which have complex interdependencies. With the development of astronomical observation techniques, the volume of observational data continues to increase, and desktop computers can no longer meet the data processing demands. Reference [5] suggests in its investigation of the core functional requirements of the China Virtual Observatory that online computing services can more efficiently process massive astronomical datasets. Deploying data processing environments on public data servers often consumes significant time and effort from researchers, and managing software versions and dependency relationships presents a considerable challenge.

Early research attempted to use Hypervisor-based virtual machine technology (hereinafter referred to as "virtual machines") for deploying scientific data processing environments and managing software versions. Reference [7] employed virtual machines to deploy a hybrid cloud, enabling A&A's data and applications to operate consistently regardless of the underlying infrastructure. However, virtual machines exhibit disadvantages in high input/output and high-concurrency scientific data processing: (1) the virtualization mechanism of virtual machines causes excessive consumption of computational resources; (2) virtual machines have weak cross-platform portability, making data processing environments difficult to share; and (3) virtual machines lack application-level monitoring capabilities and flexibility in resource scheduling.

Container technology [8] offers a promising solution to the problems in deploying and managing scientific data processing environments. Container technology does not fully virtualize the operating system; instead, it achieves lightweight OS-level virtualization by sharing the host kernel, utilizing Namespaces and Cgroups for isolation and management of virtual environments. Reference [9] evaluated the performance of virtual machines and container technology, demonstrating that containers outperform virtual machines in CPU performance, memory throughput, disk I/O, and load testing. Reference [10] proposed that container technology is an ideal choice for building scientific data processing platforms, as assigning tasks for different scientific objectives in isolated execution environments can significantly reduce the complexity of data processing and resolve dependency library version conflicts. Reference [11] adopted a Docker/VM-based architecture to implement server-side data batch processing using programming languages such as Python. Reference [12] used container technology to achieve rapid deployment of the visibility function calibration software SAGECaL, with studies showing that the automated deployment approach using containers greatly improved deployment efficiency for SAGECaL distributed clusters. Reference [13] tested the execution of astronomical software in container environments, demonstrating that container technology can help researchers reconfigure data processing environments and generate test results. Reference [14] created a Docker-based container framework (Kliko) for running one or more related computational jobs, implementing a web-based con-

tainer scheduler and specialized visualization tools for astronomical data output. Reference [citation error] encapsulated the software environment required for research into container images, facilitating replication and extension of results by other researchers.

Container management typically relies on Command-Line Interfaces (CLI), requiring researchers to invest time in learning how to use containers for data processing, which reduces the efficiency of scientific data processing. Existing tools have attempted to use Graphical User Interfaces (GUI) for container management, such as Portainer (https://www.portainer.io/), minikube GUI (https://minikube.sigs.k8s.io/docs/tutorials/setup_{minikube}_{gui}/), and DockStation (https://dockstation.io/). However, these tools are not optimized for scientific data processing, leaving room for improvement in scenarios involving the creation of data processing environments using container technology.

This paper implements the encapsulation and rapid deployment of the pulsar data processing environment at Xinjiang Astronomical Observatory based on the Docker open-source container engine, compares and tests the resource utilization and stability of pulsar data processing in containers versus virtual machines, and designs and implements a graphical user interface for controlling the container framework with functional optimizations tailored to astronomical data processing requirements. The container framework has been deployed on the pulsar data processing server at Xinjiang Astronomical Observatory and applied to actual scientific data processing tasks.

1.1 Image Packaging

To enhance the flexibility of data processing for researchers and improve resource utilization, this paper hierarchically encapsulates pulsar data processing software and dependencies into standard container images. The software components included in each image layer are shown in Table 1 .

Using Ubuntu as the base image, we systematically encapsulated dependency environments and software versions with different relationships across various layers. For instance, to process images generated by PGPLOT using the Pillow library, or when running older programs, a Python 2 container environment can be created to use the PIL library. Graduate students new to pulsar data processing can directly utilize the complete pulsar data processing environment to quickly initiate scientific research, and thanks to container isolation, they need not worry about configuration errors affecting other containers running on the same server. Experienced researchers can select specific software versions according to their requirements and combine multiple containers to form data processing pipelines.

1.2 GUI-Based Container Configuration

In response to researcher requirements, this paper attempts to test container encapsulation for data processing environments with graphical interfaces. The complex dependencies required for astronomical data processing lead to dependency conflicts when deploying pulsar data processing software using existing VNC images from DockerHub (https://hub.docker.com/), making implementation difficult. Therefore, when deploying astronomical data processing environments with graphical interfaces, this paper builds upon images with pre-installed pulsar processing software, implements the graphical interface using the Xfce desktop, and utilizes TightVNCServer for remote access. For physical machine configuration, x11-xserver-utils must be installed in the host system to allow Docker to access the X11 display interface. During container creation, local Unix ports need to be shared and environment variables modified to avoid issues when running GUI applications inside containers.

2.1 Architecture Design

We have deployed a container-based data processing architecture on the pulsar data processing server at Xinjiang Astronomical Observatory. The architecture design encompasses the hardware environment, software deployment, and graphical user interface, with specific details illustrated in Figure 1 [Figure 1: see original paper].

2.2 Hardware Environment

The core of the pulsar data processing environment is the data processing server, model Inspur NF5460M4, with configuration details shown in Table 2 .

The data processing server uses 10 Gigabit Ethernet to mount multiple NFS servers for expanded storage space and connects to the management server and public data server via Gigabit Ethernet. Deploying the container management module and private image registry on separate servers facilitates system maintenance and enables rapid migration of research group data processing environments to high-performance computing clusters.

2.3.1 Basic Software

The basic software components deployed in the data processing architecture are as follows:

- **Docker 19.03.8**: Docker is used as the container engine, interfacing with the graphical user interface developed in this paper through a middleware layer.
- **cAdvisor 0.43.0**: cAdvisor analyzes container resource usage and performance metrics, collecting real-time information from containers and determining host system resource utilization.

- **InfluxDB 1.5.3**: Stores data generated by cAdvisor, enabling persistent storage of monitoring data.
- **Grafana 8.3.2**: A visualization tool that presents monitoring data and user container information to administrators.
- **Tomcat 5.0.28**: The application server that provides services for the graphical user interface.
- **MySQL 5.0.24**: Used for storing user information.

2.3.2 Private Image Registry

Due to network bandwidth limitations and requirements for private image packaging, this paper utilizes Harbor 2.3.4 as a private registry tool for storing and managing images of the pulsar data processing environment. The private image registry is deployed on the public data server. Since researchers do not directly interact with Harbor for image operations, actual image usage permissions are controlled through the graphical user interface.

2.4 Graphical User Interface

Researchers can log into the graphical user interface to manage user information, images, containers, and data volumes. Through page operation guidance, researchers need not learn container operations or implementation principles. The graphical user interface homepage is shown in Figure 2 [Figure 2: see original paper]. The management page only records operation information, while the middleware layer is responsible for converting this information into specific commands to control the container framework and execute user instructions. The functional modules of the graphical user interface are illustrated in Figure 3 [Figure 3: see original paper].

2.4.1 Multi-User Management and Authentication

Container frameworks are typically managed centrally by administrators during usage. In scientific data processing, servers are often shared by multiple researchers, making flexible multi-user management more suitable for practical needs. This paper implements multi-user management in the graphical user interface, allowing researchers to directly manage their personal containers, images, and data volumes. User permissions are controlled by the graphical user interface based on information in the database, and upgrades to the underlying container framework or changes to the hardware platform do not affect researcher operations.

Unlike traditional management systems with independent authentication methods, this paper accomplishes authentication through verification of an "initial container." When the system creates a new user, an "initial container" is simultaneously generated, which contains a complete pulsar data processing environment that researchers can directly use for data processing. When researchers use the graphical user interface for container management, they only need to

input the port number of the "initial container" and the user password within it. Authentication is then performed directly inside the container through the middleware layer. The advantage of this authentication approach is that it eliminates the need to record dedicated management passwords, and passwords are not stored in the database, thereby enhancing both convenience and security for researchers.

2.4.2 Container Management

The container management module enables starting/stopping containers, deleting containers, creating containers, encapsulating containers, and exporting containers. The container creation function adopts a guided approach, where the system generates containers based on operation information and records the generation details. Researchers can log into containers through automatically assigned port numbers. The container export function can package containers into .tar files and store them on the public data server, allowing researchers to save exported containers using generated download addresses within a specified time period.

2.4.3 Image Management

The image management module enables deletion and sharing of images. The image sharing function allows encapsulated containers to be shared with other researchers within the system.

2.4.4 Data Volume Management

To enable flexible use of container data volumes in multi-user environments and meet the demands of astronomical data processing, this paper categorizes data volumes into three types: (1) read-only shared data volumes, (2) read-write shared data volumes, and (3) personal data volumes. Read-only shared data volumes store public data, and their read-only attribute effectively prevents accidental deletion, with only designated accounts possessing write permissions. Read-write shared data volumes are divided into two categories: (1) read-write shared data volumes on HDDs, facilitating data transfer between different users and storage of temporary large-capacity intermediate data; and (2) read-write shared data volumes on SSDs, designated for data processing requiring high I/O operations. By default, personal data volumes are accessible only to their owners, but the data volume sharing function allows personal data volumes to be shared with other researchers in read-only or read-write modes, facilitating collaboration and communication.

3.1.1 Pulsar Data Coherent Dedispersion

The interstellar medium is a low-temperature plasma. When pulsar signals travel through the interstellar medium to reach Earth, they experience dispersion, causing higher frequency components to arrive earlier and lower frequency

components to arrive later [15]. The observational time delay Δt between frequencies v_1 and v_2 caused by the interstellar medium is given by:

$$\Delta t = k_{\mathrm{DM}} \cdot \mathrm{DM} \cdot (v_1^{-2} - v_2^{-2})$$

where k_{DM} is the dispersion constant with a value of 4.148808 MHz^2 pc^{-1} cm^3 s, and DM is the dispersion measure, which represents the integral of electron density along the line of sight between the pulsar and Earth, expressed as:

$$\mathrm{DM} = \int n_e \, dl$$

where n_e is the electron number density and l is the line-of-sight distance between the pulsar and the observing telescope. The interstellar medium severely affects pulsar signals through dispersion, causing pulse profile broadening and distortion that can, in extreme cases, render the pulse profile unobservable. Therefore, dedispersion processing of pulsar observation data is essential. There are two methods for pulsar data dedispersion: incoherent dedispersion and coherent dedispersion.

Coherent dedispersion treats the dispersion effect as a filter. The process involves first performing a Fourier transform on the pulsar data, multiplying the frequency-domain data by the inverse of the equivalent filter transfer function (the chirp function), and then performing an inverse Fourier transform to obtain the dedispersed pulsar data. The discrete chirp function [17] is:

$$c_p = H\left[1 + \frac{2\pi k_{\mathrm{DM}} \mathrm{DM} f_0^2}{f(f - f_0)}\right] \tag{1}$$

where N is the number of frequency-domain points in the discrete signal; f_0 is the central frequency of observation; and B is the bandwidth.

Coherent dedispersion can theoretically completely eliminate the dispersion effects of the interstellar medium, but the process involves Fourier transforms and inverse transforms, requiring substantial CPU and memory resources.

3.1.2 Test Environment and Data

To compare pulsar data processing performance across physical machines, virtual machines, and containers, this paper performs coherent dedispersion on pulsar data in an experimental environment. The test environment information is shown in Table 3 , and the test data information is shown in Table 4 .

3.1.3 Single-Task Comparison Test

This paper performs coherent dedispersion on baseband data of pulsar J0437-4715 using TEMPO2 on physical machines, virtual machines, and containers, with identical thread counts and unnecessary programs disabled. The test results are shown in Tables 5 through 7 . Comparison of the test results reveals minimal differences across platforms, with containers exhibiting processing times closer to those of physical machines.

3.1.4 Multi-Task Comparison Test

On three physical machines with identical configurations (hardware information shown in Table 3), we run three and five threads/virtual machines/containers respectively. A test script is used to enable each thread/virtual machine/container to simultaneously perform coherent dedispersion on the data in each round of testing (simulating a multi-user processing scenario), with a total of five test runs.

In the comparative experiments, the time when the last thread/virtual machine/container completes data processing is taken as the duration for a single test. The data processing time consumption is shown in Figures 4(a) and 4(b) [Figure 4: see original paper]. To compare the resource allocation balance across different platforms, the overall standard deviation (σ) is calculated for the test results:

$$\sigma = \sqrt{\sum (x - \bar{x})^2} \tag{2}$$

The overall standard deviation of time consumption is shown in Figures 4(c) and 4(d) [Figure 4: see original paper].

Experimental results demonstrate that containers consume less time than virtual machines for processing equivalent computational workloads of pulsar data. Comparative analysis of time consumption fluctuations and overall standard deviation indicates that container technology provides more balanced resource allocation than virtual machines, with more stable data processing. Comparison with physical machine test results shows that balanced resource allocation during multi-task execution can deliver superior overall data processing performance.

3.2 Pulsar Data Processing Pipeline Test

Using the deployed data processing architecture (server configuration shown in Table 2), we create containers and implement a pulsar timing data processing pipeline. The test uses observational data published in 2004 from the Parkes telescope [15], with data information shown in Table 8 . Pulsar data processing typically involves the following steps: data download, data preprocessing, and specific data analysis. Based on the data processing requirements of each step,

multiple containers are generated using image files from the private image registry to achieve automatic data preprocessing, and analysis is performed using pulsar data processing software within the containers. The container information used is shown in Table 9 .

The data processing pipeline is as follows:

1) Due to public server access restrictions (specified container exposed ports), Container A is used to download data folders and transfer them to the "rf" subdirectory of the personal folder.
2) A looping script running on Container B monitors .rf files in the "rf" subdirectory of the personal folder. When new .rf files are detected, it automatically invokes the PSRCHIVE command for preprocessing: `pam -DFTp -e FT *.rf`.
3) Container B transfers the preprocessed data to the "ft" subdirectory of the personal folder.
4) Container C is used to view the pulse profiles of preprocessed data files: `pav -DFTp *.FT`, as shown in Figure 5 [Figure 5: see original paper].
5) Files with high signal-to-noise ratios are selected to create standard profiles (.std) using the `paas` command from PSRCHIVE.
6) PSRCHIVE and PSRCAT software are used to generate .tim and .par files. After data fitting, timing residuals are obtained, with results shown in Figure 6 [Figure 6: see original paper].

3.3 Scientific Results

The data volume generated by FAST observations is enormous and cannot be processed on desktop computers. Following the deployment of the container framework on the pulsar data processing server, researchers have utilized the rapidly generated data processing environments to process FAST data and have already produced scientific results.

The fourth plasma lensing phenomenon discovered in black widow pulsars [18]. PSR J1720-0533 was discovered during drift-scan observations in the Commensal Radio Astronomy FAST Survey (CRAFTS) [19]. Researchers used the data processing architecture deployed in this paper to process observational data of PSR J1720-0533 and found that this pulsar exhibits obvious eclipsing phenomena, a typical characteristic of black widow pulsars. Using the graphical user interface, multiple pulsar data processing environments can be quickly generated, and running different data processing scripts in each container can improve the efficiency of comparative analysis. Further analysis of the data processing results by researchers revealed plasma lensing phenomena during the pulsar' s entry into eclipse, representing the fourth case of plasma lensing discovered in the black widow pulsar population.

Conclusion

This paper encapsulates the pulsar data processing environment using container technology and establishes a private image registry, achieving rapid deployment of pulsar data processing environments. We have implemented a container-based data processing architecture on the pulsar data processing server, designed and developed a graphical user interface, and optimized container management functions according to scientific data processing needs, reducing the learning curve for researchers using containers for data processing. Comparative tests using physical machines, virtual machines, and containers for pulsar data processing demonstrate that container performance is comparable to physical machines and superior to virtual machines. In multi-task concurrent scenarios, container environments exhibit better load balancing capability and stability. The currently deployed data processing server has been applied to pulsar data processing work and has produced relevant scientific results.

References

[1] BETHAPUDI S, DESAI S. Separation of pulsar signals from noise using supervised machine learning algorithms[J]. Astronomy and Computing, 2018, 23(1): 15-26.

[2] VAN STRATEN W, BAILES M. DSPSR: digital signal processing software for pulsar astronomy[J]. Publications of the Astronomical Society of Australia, 2011, 28(1): 1-14.

[3] LUO J, RANSOM S, DEMOREST P, et al. PINT: a modern software package for pulsar timing[J]. The Astrophysical Journal, 2021, 911(1): 45.

[4] HOBBS G B, EDWARDS R T, MANCHESTER R N. TEMPO2, a new pulsar-timing package–I. An overview[J]. Monthly Notices of the Royal Astronomical Society, 2006, 369(2): 655-672.

[5] VAN STRATEN W, MANCHESTER R N, JOHNSTON S, et al. psrchive and psrfits: definition of the stokes parameters and instrumental basis conventions[J]. Publications of the Astronomical Society of Australia, 2010, 27(1): 104-109.

[6] XU Y F, FAN D W, CUI C Z, et al. Investigation and analysis of core functional requirements of China VO[J]. Astronomical Research & Technology, 2020, 17(1): 111-120.

[7] BERTOCCO S, DOWLER P, GAUDET S, et al. Cloud access to interoperable IVOA-compliant VOSpace storage[J]. Astronomy and Computing, 2018, 24(1): 36-44.

[8] MERKEL D. Docker: lightweight linux containers for consistent development and deployment[J]. Linux Journal, 2014, 2014(239): 2.

[9] POTDAR A M, NARAYAN D G, KENGOND S, et al. Performance evaluation of docker container and virtual machine[J]. Procedia Computer Science,

2020, 171: 1419-1428.

[10] GERLACH W, TANG W, WILKE A, et al. Container orchestration for scientific workflows[C]//Proceedings of the 2015 IEEE International Conference on Cloud Engineering. 2015: 377-378.

[11] TAGHIZADEH-POPP M, KIM J W, LEMSON G, et al. SciServer: a science platform for astronomy and beyond[J]. Astronomy and Computing, 2020, 33: 100412.

[12] YAO K, DAI W, YANG Q P, et al. Automatic deployment method of astronomical application software based on container technology[J]. Astronomical Research & Technology, 2019, 16(3): 321-328.

[13] MORRIS D, VOUTSINAS S, HAMBLY N C, et al. Use of Docker for deployment and testing of astronomy software[J]. Astronomy and Computing, 2017, 20: 105-119.

[14] MOLENAAR G, MAKHATHINI S, GIRARD J N, et al. Kliko—the scientific compute container format[J]. Astronomy and Computing, 2018, 25: 1-9.

[15] GIOMMI P, BRANDT C H, DE ALMEIDA U B, et al. Open universe for blazars: a new generation of astronomical products based on 14 years of Swift-XRT data[J]. Astronomy & Astrophysics, 2019, 631: A116.

[16] TIBURZI C. Pulsars probe the low-frequency gravitational sky: pulsar timing arrays basics and recent results[J]. Publications of the Astronomical Society of Australia, 2018, 35: e013.

[17] STAIRS I H. Observations of binary and millisecond pulsars with a baseband recording system[D]. Princeton: Princeton University, 1998.

[18] WANG S Q, WANG J B, WANG N, et al. Unusual emission variations near the eclipse of black widow pulsar PSR J1720−0533[J]. The Astrophysical Journal Letters, 2021, 922(1): L13.

[19] LI D, WANG P, QIAN L, et al. FAST in space: considerations for a multibeam, multipurpose survey using china' s 500-m aperture spherical radio telescope (FAST)[J]. IEEE Microwave Magazine, 2018, 19(3): 112-119.

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