Research in Astronomy and Astrophysics, 24:075011 (13pp), 2024 July https://doi.org/10.1088/1674-4527/ad4fc4

UWLPIPE: Ultra-wide Bandwidth Low-frequency Pulsar Data Processing Pipeline

Ya-Zhou Zhang[1,2], Hai-Long Zhang[1,2,3,4], Jie Wang[1,4], Jian Li[1,2], Xin-Chen Ye[1,2,4], Shuang-Qiang Wang[1], Xu Du[1,2], Han Wu[1,2], Ting Zhang[1,2], and Shao-Cong Guo[5]
[1] Xinjiang Astronomical Observatory, Chinese Academy of Sciences, Urumqi 830011, China; zhanghailong@xao.ac.cn
[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
[5] Southeast University, Nanjing 211189, China
Received 2024 April 16; revised 2024 May 14; accepted 2024 May 21; published 2024 July 8

Abstract

For real-time processing of ultra-wide bandwidth low-frequency pulsar baseband data, we designed and implemented an ultra-wide bandwidth low-frequency pulsar data processing pipeline (UWLPIPE) based on the shared ringbuffer and GPU parallel technology. UWLPIPE runs on the GPU cluster and can simultaneously receive multiple 128 MHz dual-polarization VDIF data packets preprocessed by the front-end FPGA. After aligning the dual-polarization data, multiple 128M subband data are packaged into PSRDADA baseband data or multi-channel coherent dispersion filterbank data, and multiple subband filterbank data can be spliced into wideband data after time alignment. We used the Nanshan 26 m radio telescope with the *L*-band receiver at $964 \sim 1732$ MHz to observe multiple pulsars. Finally, we processed the data using DSPSR software, and the results showed that each subband could correctly fold out the pulse profile, and the wideband pulse profile accumulated by multiple subbands could be correctly aligned.

Key words: (stars:) pulsars: general – methods: data analysis – techniques: miscellaneous

1. Introduction

With the rapid development of key technology, the performance requirements for digital backend system-related equipment in astronomical observation are also continuously increasing. Backend systems need to perform high-speed sampling, real-time analysis, and data preprocessing under wider bandwidths, higher time resolution, and higher frequency resolution. The deployment of PAF (van Cappellen et al. 2022; Zhang & Duan 2022) and multi-beam receiving systems (Yang & Han 2022) has doubled the amount of information obtained from astronomical observations. The high-speed data streams produced during transmission, storage, and real-time processing on heterogeneous platforms are urgent problems to be solved during the operation of various radio observation devices. Wider sampling bandwidths, higher digital signal bit widths, and more array antennas lead to exponential growth in the amount of data that need to be processed. Due to the performance limitations of storage devices, massive amounts of astronomical signals can only be processed and analyzed in real-time, posing a significant challenge to computing hardware and software (Wei 2019).

To address the high-speed transmission and processing of data, a viable solution is to create the shared ringbuffer in memory for real-time caching of data, and transfer the data to the GPU for immediate processing. Internationally, the use of CPU+GPU heterogeneous computing platforms for real-time processing of pulse signals has become mainstream (Wei et al. 2023). Utilizing the computational units of the GPU to process data can reduce the load on the CPU and significantly enhance the system's data stream processing efficiency. With the continuous development of digital backend technology in radio astronomy, real-time signal processing poses severe challenges to traditional computing technologies. The powerful computational capabilities of GPU clusters offer a feasible solution for handling the massive data streams generated during radio astronomical observations.

Currently, mainstream digital backend systems often adopt a hybrid architecture of Field Programmable Gate Array (FPGA), CPU, and GPU. Heterogeneous systems require data exchange between different devices. After preprocessing by FPGA, the data is transmitted via the network to the CPU memory of the data processing server, and then copied to the GPU memory for further processing. One of the technical challenges of a data processing system based on a heterogeneous platform is how to achieve high-speed data circulation between FPGA and CPU, and between CPU and GPU. In radio astronomy digital backend systems, the program that achieves high-rate real-time transmission and preprocessing of data packets is called the pipeline (Paine & Lee 2014), which includes several steps such as reception, data buffering, data preprocessing, and packaging in astronomical standard formats.

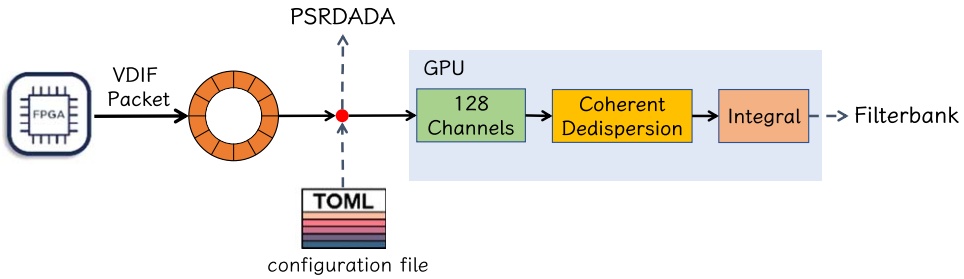


Figure 1. UWLPIPE data flow.

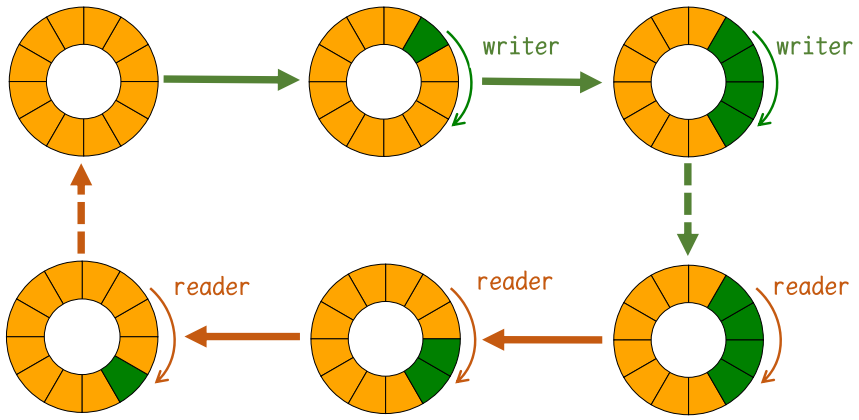


Figure 2. Ringbuffer cyclic read and write operations.

To address the quasi-real-time or real-time high-speed transmission and preprocessing of radio astronomical data, several pipeline softwares have been developed both domestically and internationally. GUPPI_daq (Data acquisition software for GUPPI)[6] is the data acquisition software for the Green Bank Telescope pulsar digital backend Green Bank Ultimate Pulsar Processing Instrument (GUPPI). It operates in a multi-thread mode, such as network data receiving threads and data processing threads, which transmit data through a shared memory ringbuffer (DuPlain et al. 2008). After preprocessing the real-time received network data stream, GUPPI_daq stores it as PSRFITS (Hobbs 2021) data format in fold or search mode based on configuration.

High Availability SHared PIPeline Engine (HASHPIPE)[7] is an efficient data processing framework based on shared memory. The core of HASHPIPE is the shared ringbuffer, and applications written based on HASHPIPE are created as shared library "plugins," realizing the circulation and sharing of data among multiple threads. The shared memory buffer between threads is controlled by the semaphore to manage tasks, achieving mutual exclusion between read and write threads and preventing data read and write errors (MacMahon et al. 2018).

Niu et al. (2019) designed the Tianlai Disk Array Correlator based on Reconfigurable Open Architecture Computing Hardware-2 (ROACH2) and GPU, and implemented UDP network data reception and subsequent data processing in GPU servers based on HASHPIPE. Pei et al. (2022) developed the HRBF_HASHPIPE, based on HASHPIPE for the PAF. After FPGA collected the simulated signals, the data with a bandwidth of 256 MHz was output to a GPU server through four 10 Gigabit Ethernet (10GbE) links. The server ran four instances of HRBF_HASHPIPE, each receiving data at a bandwidth of 64 MHz and invoking a GPU card for beamforming calculations. According to the future plan of QiTai radio Telescope (QTT; Wang et al. 2023), Pei et al. (2023) designed UWB_HASHPIPE to multithreadedly receive and store multiple subband data in parallel, which could be flexibly configured for data distribution links based on IP addresses and port numbers. UWB_HASHPIPE was tested in the Nanshan 26 m radio telescope (NSRT) pulsar observation experiment, where the collected data bandwidth was 512 MHz, sent to two servers via eight links. Each server ran four instances of UWB_HASHPIPE for data processing and encapsulation. The test results showed that the signal

[6] https://github.com/demorest/guppi_daq
[7] https://github.com/david-macmahon/hashpipe

acquisition system had high precision, good data integrity, and the signal-to-noise ratio of the merged pulsar profile was superior to that of single subband data.

PSRDADA (Distributed Acquisition and Data Analysis for Radio Astronomy) is capable of flexibly managing a shared ringbuffer, which is used for the distributed recording and processing of pulsar baseband data (van Straten & Jameson 2021). Based on PSRDADA, it is possible to implement read-write clients that write data into the ringbuffer and read data from it. The configuration and control of coherent processes are initiated through a web-based interface that launches scripts. Data recording backend such as the ATNF Parkes Swinburne Recorder (APSR), Berkeley Parkes Swinburne Recorder (BPSR), and CASPER Parkes Swinburne Recorder (CASPSR) at the Parkes radio telescope are all developed based on PSRDADA.

The Parkes ultra-wide bandwidth low-frequency (UWL) receiving system, employs an FPGA+GPU architecture for the preprocessing and recording of observational data. It divides the 704–4032 MHz wideband signal into three analog RF bandwidths for sampling. During the FPGA preprocessing stage, the Polyphase Filter Bank (PFB; Harris & Haines 2011) channelization technique is used to further divide it into 26 continuous subbands, each with a bandwidth of 128 MHz. The FPGA packages the data in VLBI Data Interchange Format (VDIF; Whitney et al. 2010), and transmits it to various GPU nodes via the UDP protocol. To cope with high-speed network streams, a high-speed ringbuffer is established using PSRDADA on each GPU node to temporarily store data. Subsequently, the data is copied from the buffer to the GPU memory for further processing. This can include operations such as pulsar folding, pulsar searching, Fast Radio Burst searching, and spectral line observations (Hobbs et al. 2020).

```
[Pulsar]
name = "J0332+5434"
dm = 26.7641

[Telescope]
name = "nanshan"
receiver = "UWL"
dec = 471500.0
ra = 43700.0

[Observation]
nband = 3
npol = 2
otime = 600.0  # total observation time (seconds)
bandwidth = 128.0
cfreq = [128.0, 256.0, 384.0]
calfreq = 1.0
observer = "zhangyazhou"

[Storage]
# Packaged file types, such as dada, psrfits, filterbank and etc.
filetype = ["dada","filterbank"]
```

Figure 3. Observation configuration file.

```
[Network]
port = 60000
ip = ["192.168.1.114", "192.168.2.115", "192.168.3.116"]

[RingBuffer]
key = [0xdada, 0xdadb, 0xdadc]
nbuf = 8
bufsize = 1073741824 # 2e30

[Node]
index = 0
outdir = ["/nvme0", "/nvme1", "/nvme2"]
```

Figure 4. Machine configuration file.

2. UWLPIPE

Ultra-wide bandwidth low-frequency pulsar data processing pipeline (UWLPIPE) is a simple and efficient pulsar backend processing pipeline software. It can quickly configure the current observation by reading the configuration information from the toml[8] file. For example, it can obtain observation source information (pulsar source name, DM, observation bandwidth, observation center frequency, etc.) and configure the GPU server network (IP and port, etc.). As shown in Figure 1, by receiving dual-polarization VDIF packets and unpacking them according to the VDIF header information, the data is rearranged based on the polarization timestamp information. Afterwards, the "write" client is enabled to write the data into the shared ringbuffer. The "read" client reads the data from the buffer, copies it to the GPU memory for processing, and finally encapsulates it into the standard astronomical data format.

[8] https://toml.io

Shared memory can serve as a communication medium between processes for synchronous data exchange. The write client process writes data into the shared memory, while the read client reads data from the shared buffer, processing it differently according to various research needs. As shown in Figure 2, the ringbuffer is essentially linear memory that is logically connected at the head and tail, facilitating cyclic read and write operations on the buffer.

From the perspective of buffer operations, processes can be roughly divided into two modes: reading and writing. In the writing mode, the client receives high-speed network data and writes it into the shared buffer through rearrangement. There might be various types of reading clients, such as the baseband data encapsulation process that reads data from the shared buffer, adds header information, and directly stores it on the disk. Another example is the filterbank channelization process, which reads data from the shared buffer and copies it to the GPU memory for subsequent pipeline processing, such as

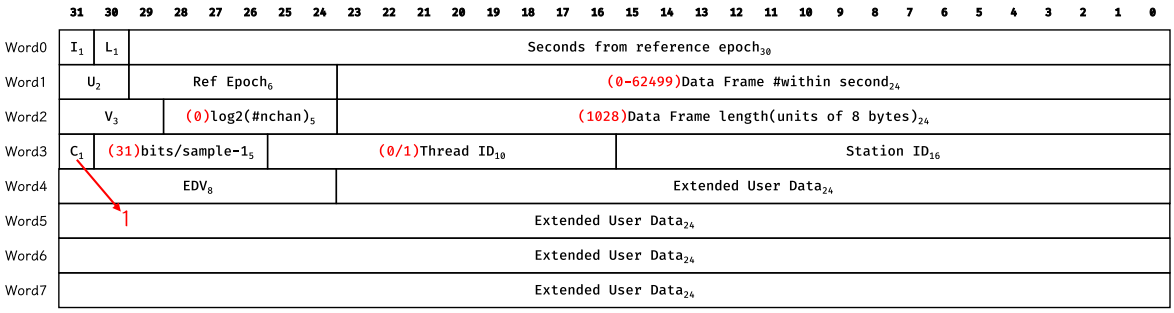


Figure 5. Information of VDIF Data Frame Header Transmitted in UWL.

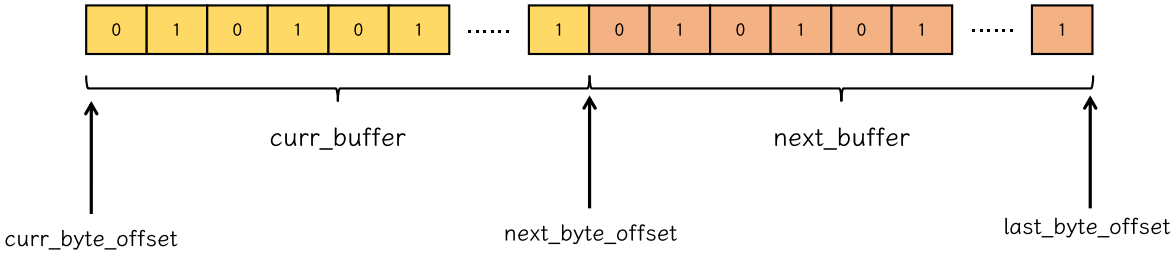


Figure 6. Double buffer structure initialization.

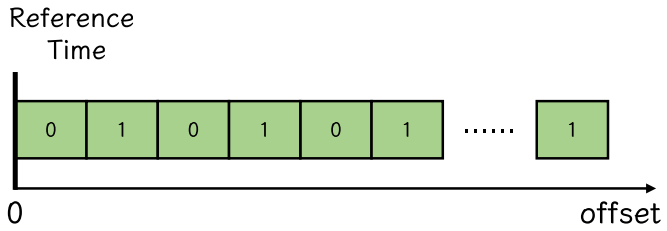


Figure 7. Calculate the offset from the reference time based on the VDIF timestamp.

channelization, coherent dedispersion, integration, and folding. Other operations include RFI mitigation, pulsar searching, and VLBI. The pulsar data processing pipeline based on shared memory allows for modular code writing according to specific functions, ensuring good extensibility.

2.1. UWLPIPE Configuration

Currently, UWLPIPE configures the observation targets and GPU nodes by reading the configuration files Observation.toml and Machine.toml. As shown in Figure 3, Observation.toml includes information such as the name of the pulsar and the dispersion value, the name of the telescope, the name of the receiver, the number of observation subbands, the bandwidth size, the center frequency of each subband, and the storage format.

The Machine.toml is shown in Figure 4, which allows the configuration of ip and port for receiving subband network data, as well as the key values, number, and size of the circular buffer. Since baseband data storage can be selected, there are certain requirements for the speed of the storage medium. The data of each subband is stored on a separate high-speed solid-state disk.

2.2. High-speed Network Data Reception and Unpacking Algorithm

Traditional TCP/IP technology, in the process of handling massive astronomical data packets, has to go through the operating system and other software layers, which requires a large amount of server resources and memory bus bandwidth. Data is copied and moved back and forth between system memory, processor cache, and network controller cache, imposing a heavy burden on the server's CPU and memory, exacerbating the effect of network latency.

To reduce the system overhead brought about by the transmission of massive astronomical data, we reconstructed the high-speed network unpacking program using the VMA library.[9] The VMA library utilizes RDMA-enabled network cards for direct hardware access and advanced polling techniques to achieve kernel bypassing. This allows the VMA library to bypass the kernel's network stack for all IP network traffic transmission and reception socket API calls, reducing CPU usage, alleviating memory bandwidth bottlenecks, and enhancing bandwidth utilization efficiency.

[9] https://github.com/Mellanox/libvma

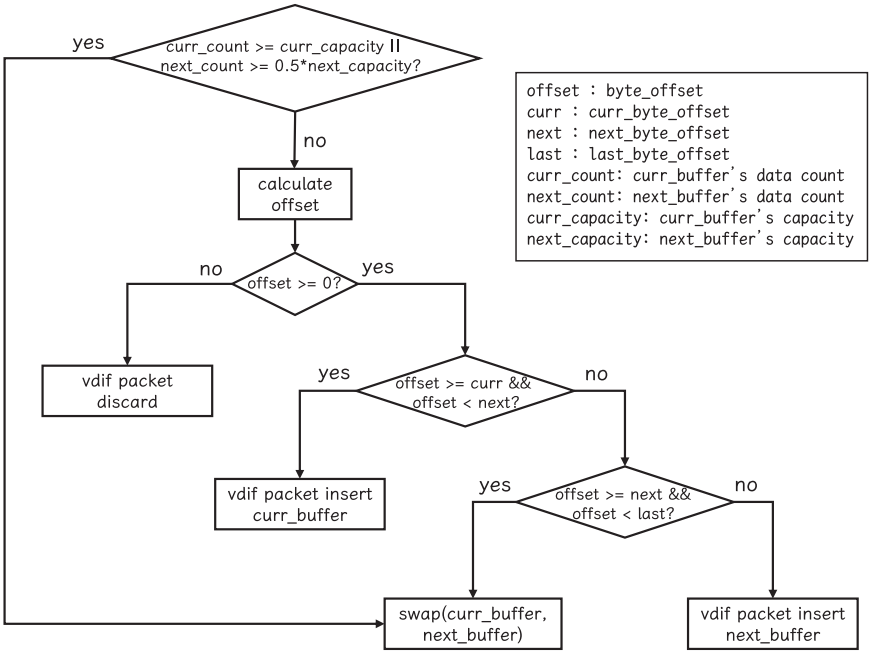


Figure 8. LDRP algorithm flow.

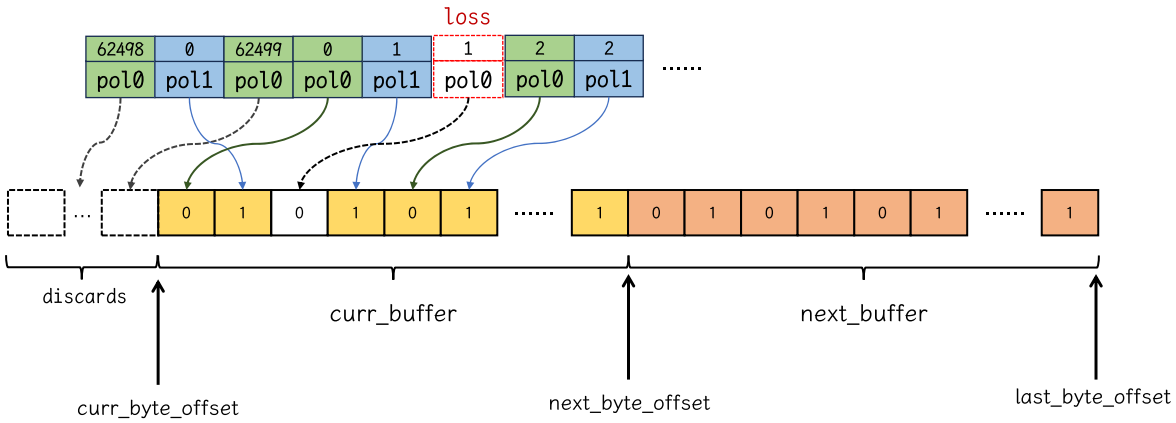


Figure 9. VDIF packets are rearranged in a double-buffered structure.

The future planning of QTT's UWL receiving system is the same as Australia's Parkes, with a bandwidth range of 704–4032 MHz (Wang et al. 2023). During the FPGA preprocessing stage, the PFB technology is used to divide the 704–4032 MHz bandwidth signal, totaling 3328 MHz, into 26 dual-polarization subband signals of 128 MHz bandwidth each. The data quantization precision is 32 bit (real part 16 bit + imaginary part 16 bit). As shown in Figure 5, the VDIF header information is depicted, with key and fixed information highlighted in red. Other channel numbers are set to 1, with $\log 2(1) = 0$. Each VDIF data frame size is 8224 bytes (header 32 + data 8192), set to $8224/8 = 1028$. The complex flag bit is set to 1, with a quantization precision of 32 bit, set to 31. Dual-polarization data is distinguished by Thread ID, where 0 represents polarization 0 and 1 represents polarization 1. The number of data frames per second is 62,500 (0–62,499), calculated according to Equation (1), where *fs* represents the sampling rate of 128 MHz and *nbit* represents the quantization precision of 32.

$$nframe = \frac{fs * nbit}{8 * 8192} = \frac{128 * 1e6 * 32}{8 * 8192} = 62,500 \quad (1)$$

After packaging each channel of dual-polarization subband data into VDIF data format, FPGA transmits it to the GPU

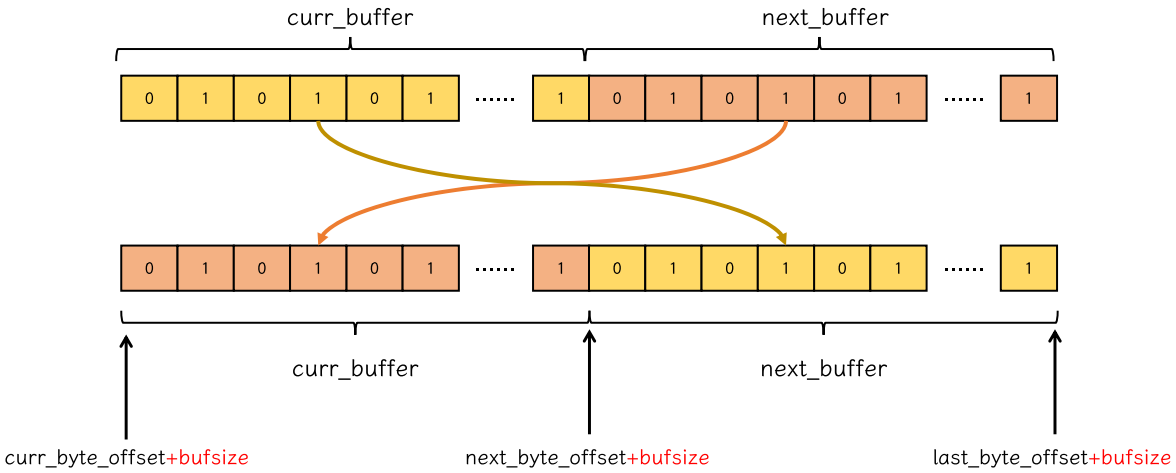


Figure 10. Double buffer structure swap.

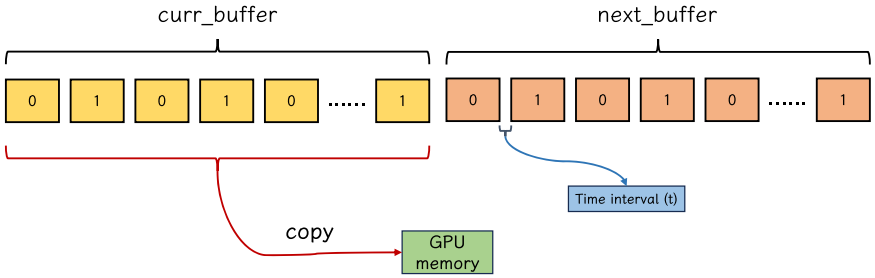


Figure 11. Buffer data copy.

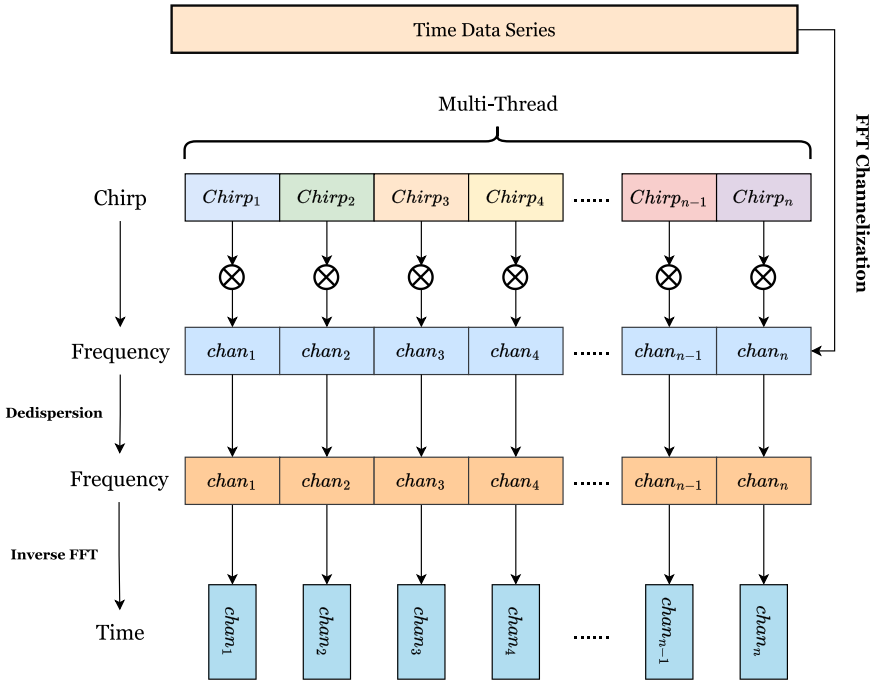


Figure 12. Multichannel coherent dispersion process.

Table 1
Based on the Dada Baseband Data Generated by PFB/OPFB

Key	Value
BW	···
CFREQ	···
HDR_SIZE	4096
HDR_VERSION	1.0
NBIT	16
NCHAN	1
NDIM	2
NPOL	2
RECEIVER	UWL
TSAMP (μs)	···
UTC_START	yyyy-MM-dd-HH:mm:ss

server via a high-speed network. Each channel of data is sent to the corresponding 100GbE network card on the GPU, and the GPU server receives data by listening to IP and specific ports. Since the UDP transmission protocol is adopted, targeted processing is required for lost, disordered, and duplicate packets; otherwise, phase information will be missing. For pulsar data, packet loss can severely affect the prediction of pulsar periods and folding. Astronomical observation has high requirements for time accuracy, so it is necessary to align dual-polarization data. To solve the above problems, we proposed a Linked-list Double-buffer Receive Packet (LDRP) algorithm.

LDRP is a dual-polarization VDIF packet alignment algorithm based on a double buffer structure for alternate storage. As shown in Figure 6, two consecutive segments of memory are allocated as buffers, named curr_buffer and next_buffer. Three byte offset pointers are set up as curr_byte_offset, next_byte_offset, and last_byte_offset. Their initial values are set as shown in Equation (2), pointing to the start of curr_buffer, the end of curr_buffer (the start of next_buffer), and the end of next_buffer, respectively

$$\begin{aligned} curr\_byte\_offset &= 0 \\ next\_byte\_offset &= curr\_byte\_offset + bufsize \\ last\_byte\_offset &= next\_byte\_offset + bufsize. \end{aligned} \tag{2}$$

Before performing polarizations alignment, it is necessary to set a reference start time. The header of the VDIF data frame contains an accurate timestamp and counter, which can be further converted into the byte offset from the reference start time. When starting to receive network data packets, unpack the first VDIF packet received for polarization 0 to obtain the timestamp seconds of the data packet. If the packet's counter is greater than 0, add 1 to the timestamp seconds as the reference start time, with the aim of starting to receive data at the beginning of an entire second. Subsequently received VDIF packets can calculate the byte offset from the reference start time using Equation (3), as shown in Figure 7

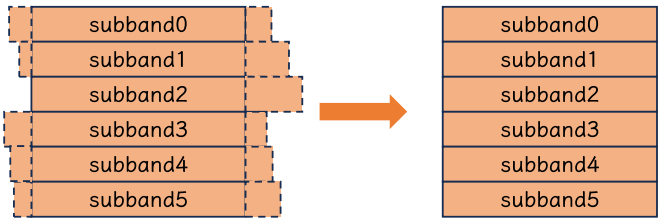


Figure 13. Multiple subbands are combined into one broadband signal.

Table 2
GPU Server Configuration

Key	Information
Operating System	Debian10
CPU	2xIntel Xeon Gold 5317
GPU	3xNVIDIA GeForce RTX 3090
Memory Size	512GiB
Storage Size	3xNVME SSD(8TB)

$$\begin{aligned} byte\_offset &= (offset * bytes) \\ &+(i * frames) + (thread\_id * packets). \end{aligned} \tag{3}$$

Among them, offset is equal to the difference between the timestamp seconds of the VDIF package and the reference start time, bytes represents the number of bytes sent per second, iframe is the counter number of the VDIF package, frames is twice the number of data bytes in the VDIF package, thread_id is equal to 0 for polarization 0, equal to 1 for polarization 1, packets is the size of the VDIF package bytes. The LDRP algorithm flow is shown in Figure 8. The detailed processing of VDIF packages is shown in Figure 9, which is separately processed in several cases based on byte_offset.

1. When the offset is less than 0, the timestamp of the VDIF data packet is before the reference start time, discard it directly, as shown in the dotted white box in Figure 9.
2. When the byte_offset is greater than or equal to curr_byte_offset and less than next_byte_offset, it indicates that the VDIF data packet belongs to the curr_buffer buffer. Calculate the difference between byte_offset and curr_byte_offset as the offset position of the VDIF data block relative to the starting position of curr_buffer, and copy the data to the corresponding position in curr_buffer, and increment the counter of curr_buffer by 1.
3. When the byte_offset is greater than or equal to next_byte_offset and less than last_byte_offset, it indicates that the VDIF data packet belongs to the next_buffer buffer. Calculate the difference between byte_offset and next_byte_offset as the offset position of the VDIF data block relative to the starting position of

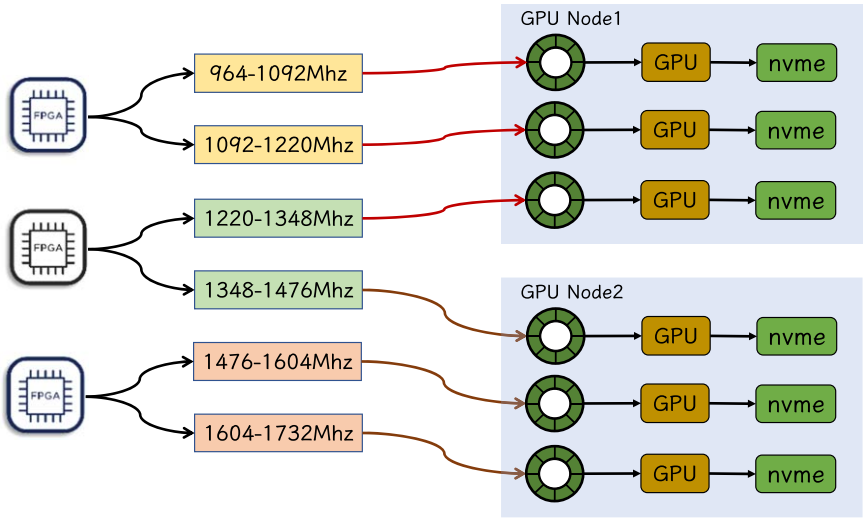
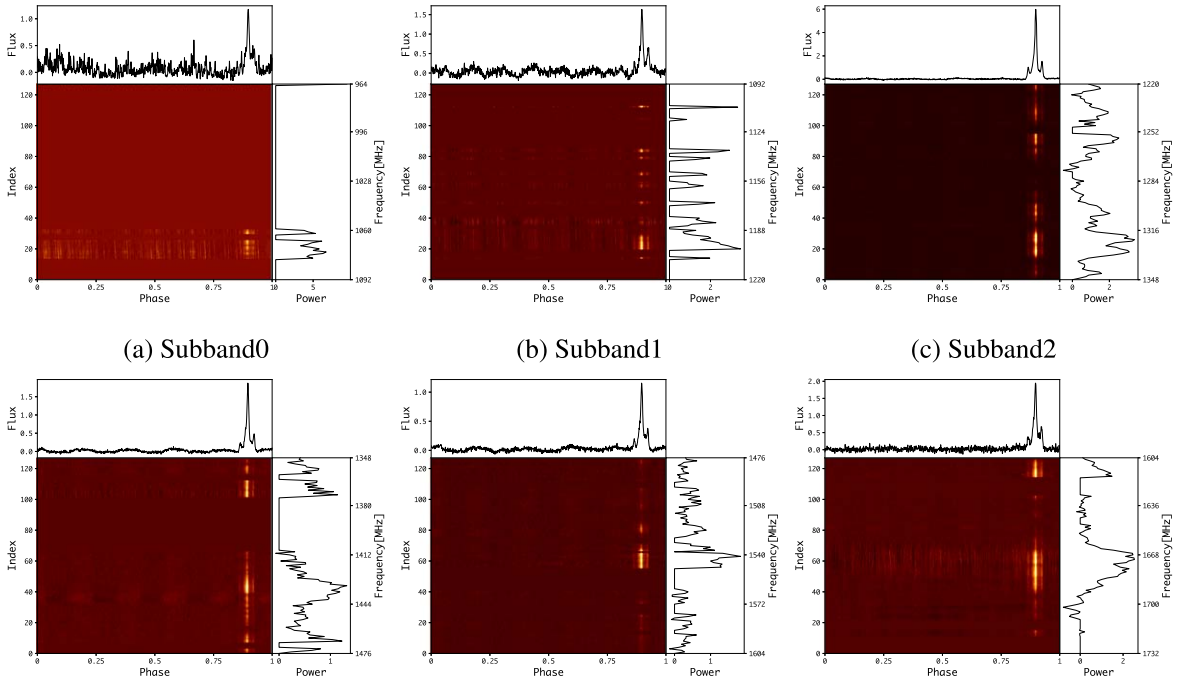


Figure 14. NSRT test architecture.



(a) Subband0 (b) Subband1 (c) Subband2

(d) Subband3 (e) Subband4 (f) Subband5

Figure 15. J0332+5434's subband Dynamic Spectrum Diagram (PFB).

next_buffer, and copy the data to the corresponding position in next_buffer, and increment the counter of next_buffer by 1.

4. When the byte_offset is greater than or equal to last_byte_offset, it indicates that there is an abnormality in the network data (high system load or other network issues), and the VDIF packet has exceeded the range of double buffer. In response to this situation, as shown in Figure 10, it is necessary to immediately update the pointers of the double buffer structure to avoid further packet loss.
5. When the curr_buffer buffer is filled with data (the counter of curr_buffer is equal to the buffer capacity), it

indicates that the data in the curr_buffer buffer is complete and can be processed by subsequent steps. At this time, it is necessary to update the double buffer offset pointers and swap buffers, as shown in Figure 10.

6. When the counter of next_buffer is greater than or equal to half of the capacity of next_buffer, it indicates that the data in the curr_buffer buffer is not filled, indicating that there is a VDIF data packet loss. To avoid further packet loss, it is necessary to update the double buffer offset pointers and swap buffers, as shown in Figure 10.

For ultra-wide bandwidth pulsar baseband data, tens of thousands of VDIF data packets are received per second, and the time interval t between VDIF data packets is very small, as shown in Figure 11. When one of the double buffers is filled with data, it needs to be copied to a shared buffer or directly to the GPU memory for subsequent processing. If the copy time is greater than the time interval t, it will cause the loss of subsequent VDIF data packets due to the inability to receive them in time. To address this issue, we employ multi-threaded asynchronous copying. Specifically, when the buffer is filled with data, a new thread is initiated to copy the data to the location required for subsequent steps, while the main receiving thread continues to receive VDIF data packets. Both processes proceed in parallel without interfering with each other.

2.3. Pulsar Data Processing and Astronomical Data Format Packaging Technology

During the FPGA preprocessing stage, the wideband signal is divided into multiple narrowband signals using PFB/Oversampled Polyphase Filter Banks (OPFB; Zhang et al. 2023b), which are then transmitted to various nodes of the GPU cluster for processing. UWLPIPE is capable of handling subband data from both PFB and OPFB channel division modes. For OPFB (with a $4/3$ oversampling factor), the data bandwidth size for each subband transmission is $128 \times \frac{4}{3}$ MHz. After performing FFT operations, it is necessary to discard data corresponding to the first and last $\frac{1}{6}$ of the bandwidth, retaining the useful data within the central 128 MHz bandwidth. Subsequent steps only process this 128 MHz data, ultimately encapsulating it in an astronomical data format. However, if encapsulated as DADA baseband data, to reduce computation time, the data is stored directly at $128 \times \frac{4}{3}$ MHz to minimize computational time.

2.3.1. Baseband Data Record

UWLPIPE, based on the configuration information, is capable of directly packaging data into dada baseband data format. This dada baseband data is identical to the baseband data generated by the Parkes Medusa backend and can be processed by DSPSR (van Straten & Bailes 2011). For PFB/OPFB sub-channel technology, the key header information for the generated dada baseband is shown in Table 1.

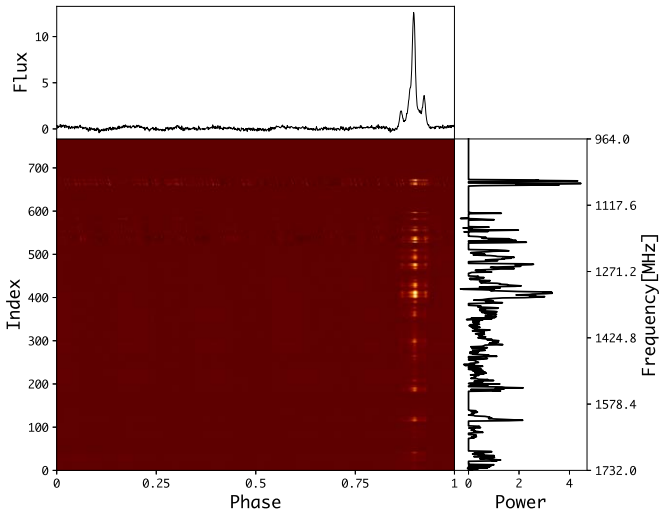


Figure 16. Dynamic spectrum of J0332+5434 (PFB).

Among them, BW represents the bandwidth size, FREQ represents the center frequency, NBIT represents the quantization accuracy of the recorded data, NDIM is 1 for real numbers and 2 for complex numbers, NPOL represents the number of polarizations, RECEIVER represents the type of receiver, here set to UWL, DSPSR can recognize it as baseband data recorded by ultra-wide bandwidth receiver backend, TSAMP represents the interval time of sampling points, and UTC_START is the starting time of the baseband record.

The data portion needs to be encoded using the offset binary, as shown in Equation (4)[10]

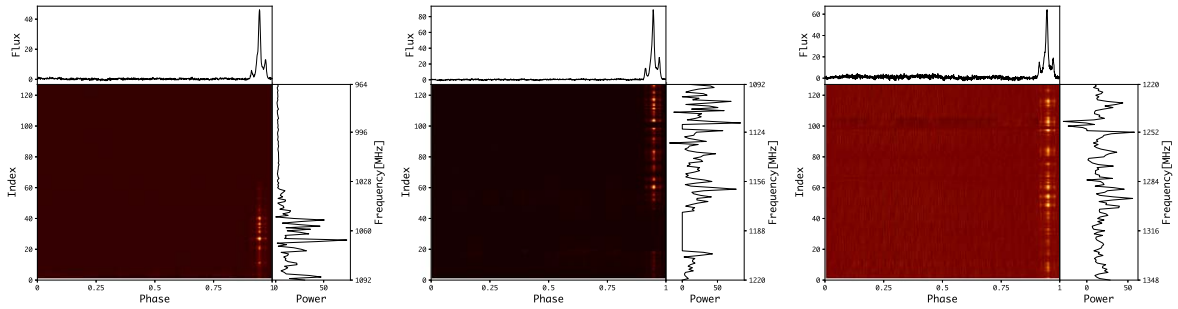
$$\text{out} = \texttt{data} \oplus \texttt{0x8000}. \tag{4}$$

Among them, $\oplus$ represents the XOR operation symbol, input represents the input data, and output represents the encoded data. For bipolarized data, since the data part in the VDIF data frame only contains one polarized data, and the size is 8192 bytes, under the 16 bit quantization precision, the two polarizations in the baseband data are alternately stored every 2048 numbers.

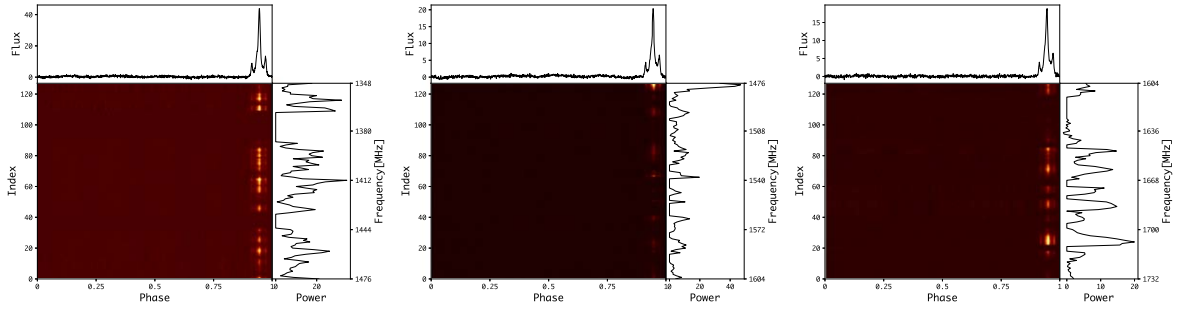
2.3.2. Filterbank Data Record

UWLPIPE can be packaged into filterbank format according to the configuration. For the filterbank format, it is necessary to further divide each subband into channels, into 128 subbands (each subband has a bandwidth of 1 MHz), and perform coherent dispersion on each subband. As shown in Figure 12, after performing FFT on the time data sequence, it is further channelized into multiple subbands. According to the center

[10] https://sourceforge.net/p/dspsr/code/ci/master/tree/Kernel/Formats/uwb/dsp/UWBUnpacker.h



(a) Subband0 (b) Subband1 (c) Subband2



(d) Subband3 (e) Subband4 (f) Subband5

Figure 17. J0332+5434's subband Dynamic Spectrum Diagram (OPFB).

frequency, bandwidth, and dispersion value (DM) of each subband, the chrip coefficient (the inverse function of the interstellar medium transfer function) is generated (Zhang et al. 2023a), as shown in Equation (5)

$$\text{chirp} = \exp\left[-i\frac{2\pi C}{(f_{\text{ref}} + \Delta f)f_{\text{ref}}^2}\text{DM}(\Delta f)^2\right]. \quad (5)$$

After coherent dedispersion, in order to reduce the amount of data, the data is integrated (32us) and finally stored as filterbank data (Lorimer 2011).

UWLPIPE will eventually output multiple filterbank subbands with a bandwidth of 128 MHz. Due to the network latency of different GPU nodes, the starting record times of each subband may vary. In order to synthesize a wideband signal, it is necessary to align the times of multiple subbands. As shown in Figure 13, find the maximum start time among the multiple subbands, use it as the effective reference start time for synthesizing the wideband, skip a certain number of bytes in other subbands to align with the reference time, and synthesize the final wideband signal.

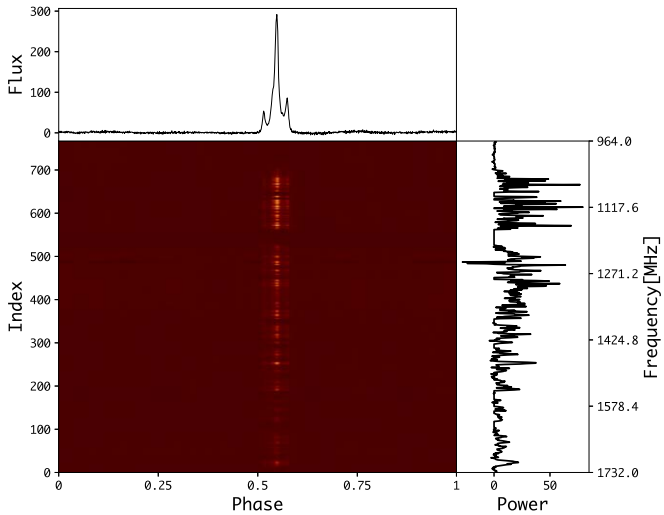
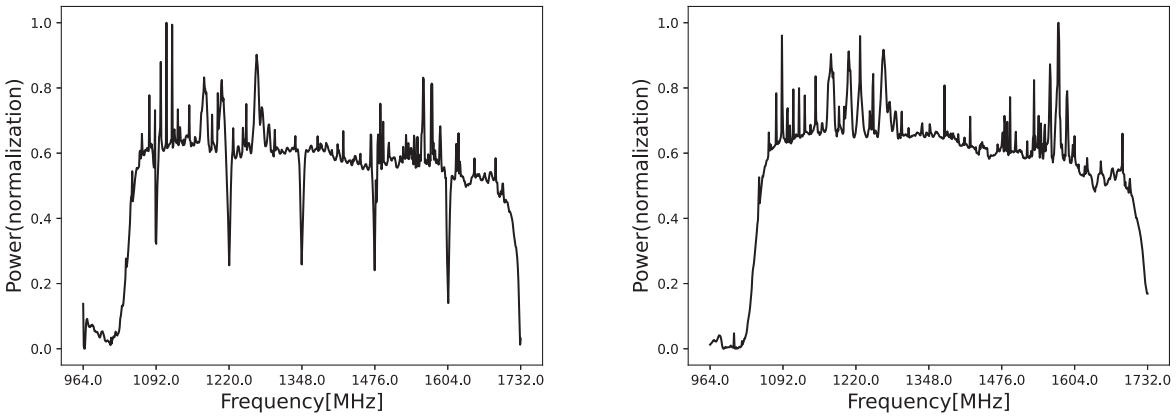


Figure 18. Dynamic spectrum of J0332+5434 (OPFB).

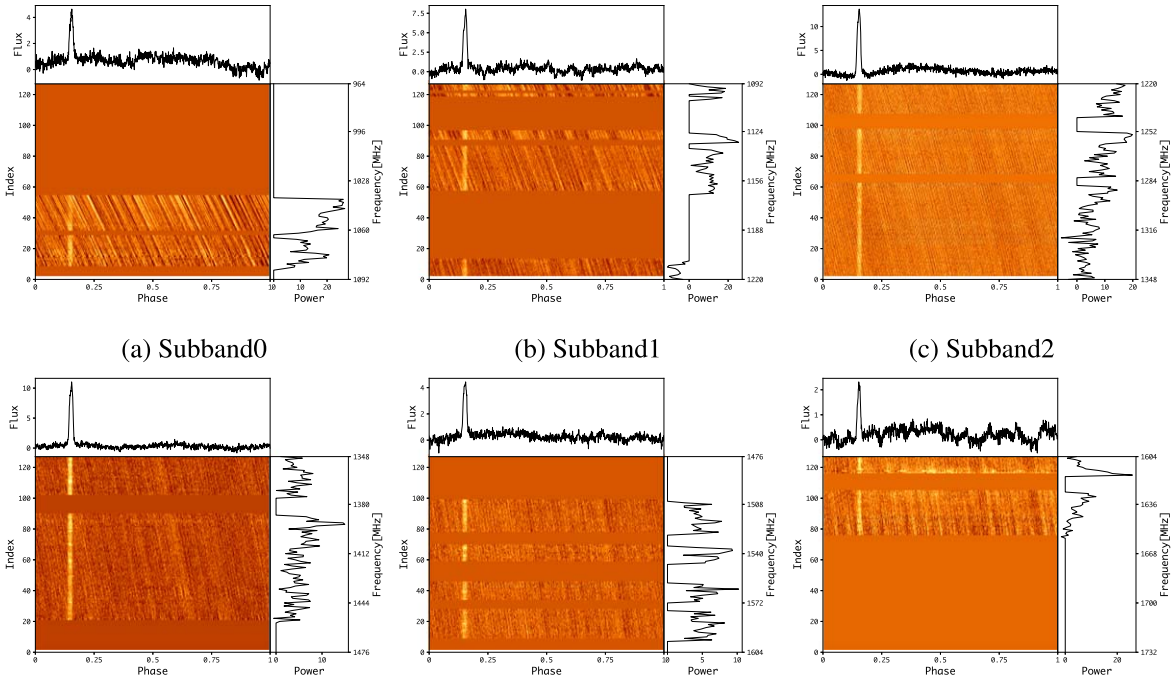
3. Results

The UWLPIPE was tested using the L-band 1G bandwidth receiver of NSRT. The test scheme is shown in Figure 14, which used 3 FPGA development boards. The back-end uses two GPU servers for data processing, Table 2 shows the configuration of each GPU server. Each development board outputs two 128 MHz bandwidth dual-polarized subband data, with a total bandwidth range of 964 ∼ 1732 MHz, covering a bandwidth size of 768 MHz.



(a) PFB (b) OPFB

Figure 19. J0332+5434 frequency spectrum.

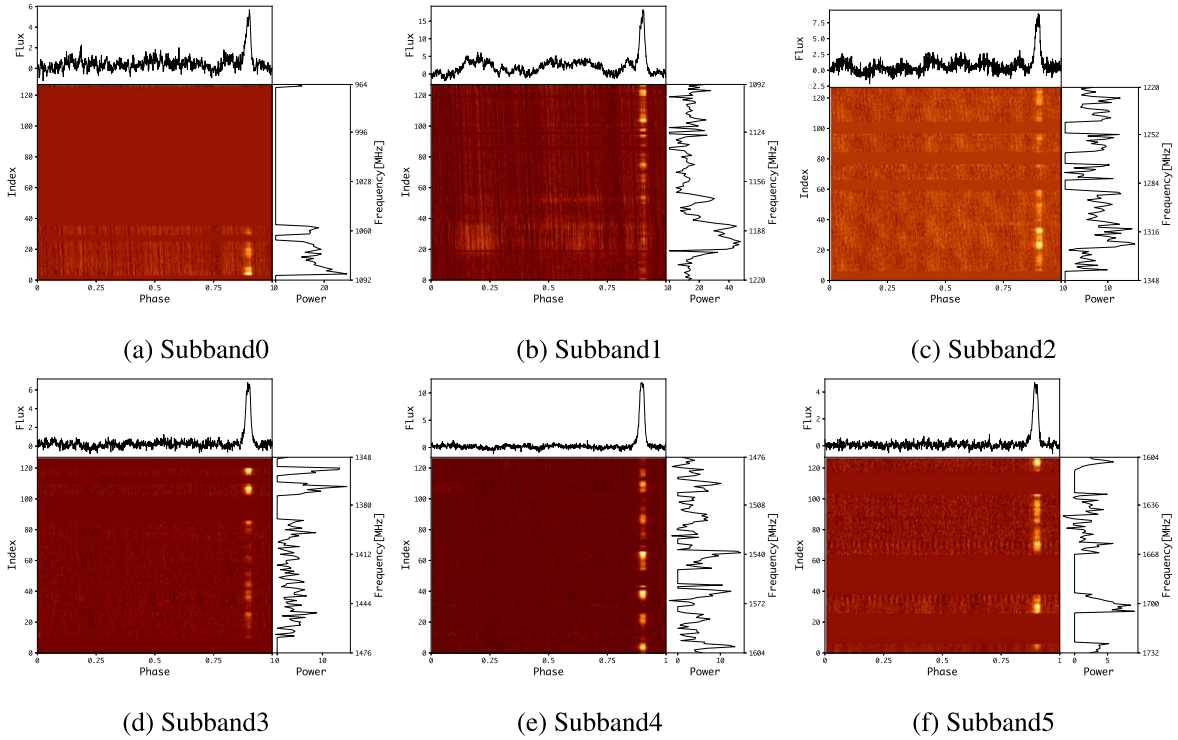


(a) Subband0 (b) Subband1 (c) Subband2

(d) Subband3 (e) Subband4 (f) Subband5

Figure 20. J1935+1616's subband Dynamic Spectrum Diagram (OPFB).

Based on the PFB subband technique, UWLPIPE observed J0332+5434 for 5 minutes. After processing with DSPSR, the dynamic spectrum of the six subbands is shown in Figure 15. The final synthesized wideband signal's dynamic spectrum after processing is shown in Figure 16. Each subband correctly folds out the pulse profile, and the final synthesized wideband signal's processed pulse profile aligns accurately. The frequency spectrum of the six subbands is shown in Figure 19(a), which clearly shows significant attenuation between subbands. Using the OPFB (Orthogonal Polyphase Filter Banks) subband technique, UWLPIPE also observed J0332+5434 for 5 minutes. The dynamic spectrum of the six subbands is shown in Figure 17, and the final synthesized wideband signal's dynamic spectrum after processing is shown in Figure 18. The frequency spectrum of the six subbands is shown in Figure 19(b), and when compared to Figure 19(a), it is evident



(a) Subband0 (b) Subband1 (c) Subband2

(d) Subband3 (e) Subband4 (f) Subband5

Figure 21. J2022+5154's subband Dynamic Spectrum Diagram (OPFB).

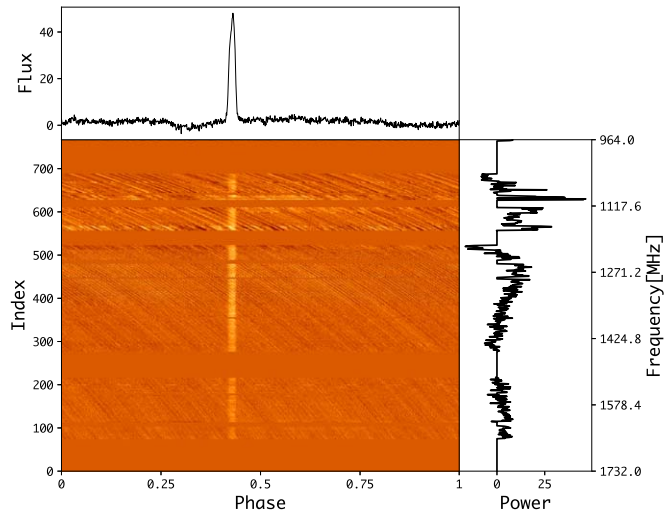


Figure 22. Dynamic spectrum of J1935+1616 (OPFB).

that there is no attenuation between OPFB subbands, which does not affect the continuity of the pulsar signal.

Furthermore, UWLPIPE was used to observe J1935+1616, and J2022+5154. The dynamic spectra of the processed six subbands are shown in Figures 20 and 21, respectively. The final synthesized wideband signals' dynamic spectra after processing are shown in Figures 22 and 23, respectively. The results demonstrate that UWLPIPE can correctly receive subband data from both PFB and OPFB channels, verifying the effectiveness of UWLPIPE in data processing.

4. Summary

To address the real-time processing requirements for ultra-wide bandwidth pulsar data, we design and implement UWLPIPE based on GPU parallel technology. UWLPIPE can configure the current observation source and GPU node parameters by reading the toml configuration file information. UWLPIPE demonstrates proficiency in real-time parallel reception of multi-subband dual-polarization pulsar signals. It facilitates multi-channel parallel real-time coherent dedispersion processing, enables ultra-wide bandwidth data synthesis, and packages the data into formats suitable for scientific data processing needs. The LDRP algorithm is designed and implemented to unpack the header information of the dual-polarization VDIF data packets, extracting the timestamp information. By calculating the offset of each VDIF data packet relative to the reference time, the data is placed in the correct position within a double buffer structure to align the dual-polarization data.

UWLPIPE is capable of packaging data into PSRDADA baseband data and filterbank data astronomical formats, that is,

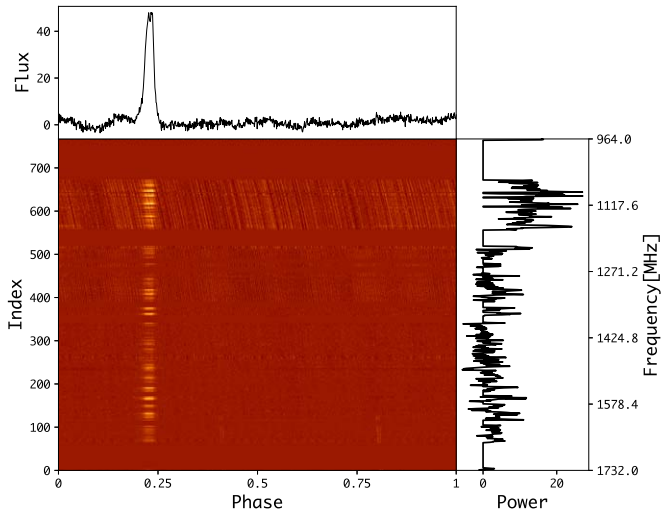


Figure 23. Dynamic spectrum of J2022+5154 (OPFB).

directly saving the aligned dual-polarization data as baseband data, or storing filterbank data after performing operations such as channel separation, coherent dedispersion and integration on the GPU.

We tested UWLPIPE using the NSRT with an *L*-band bandwidth range of $964 \sim 1732$ MHz. During the FPGA preprocessing stage, we used PFB and OPFB channel separation techniques respectively, and conducted a 5 minutes-observation of J0332+5434. Comparing the spectra across six subbands, it is evident that the OPFB-based channel separation technique yields a smoother spectrum compared to PFB, eliminating the spectral decay effect between subbands.

We also observed J1935+1616 and J2022+5154, obtaining six subbands of 128 MHz data. After performing folding integration on each subband, we obtained the correct pulse profile. Finally, we merged the six subbands to generate wideband data and performed folding integration again. The generated subband profiles were correctly aligned. The above experiments verified the accuracy of UWLPIPE's data processing and laid the foundation for real-time processing of ultra-wide bandwidth pulsar data.

Acknowledgments

This work is supported by the National Key R&D Program of China Nos. 2021YFC2203502 and 2022YFF0711502; the National Natural Science Foundation of China (NSFC) (12173077); the Tianshan Talent Project of Xinjiang Uygur Autonomous Region (2022TSYCCX0095 and 2023TSYCCX0112); the Scientific Instrument Developing Project of the Chinese Academy of Sciences, grant No. PTYQ2022YZZD01; China National Astronomical Data Center (NADC); the Operation, Maintenance and Upgrading Fund for Astronomical Telescopes and Facility Instruments, budgeted from the Ministry of Finance of China (MOF) and administrated by the Chinese Academy of Sciences (CAS); Natural Science Foundation of Xinjiang Uygur Autonomous Region (2022D01A360).

ORCID iDs

Ya-Zhou Zhang https://orcid.org/0000-0001-6046-2950
Hai-Long Zhang https://orcid.org/0000-0002-8951-7094
Jie Wang https://orcid.org/0000-0003-0380-6395
Xu Du https://orcid.org/0000-0001-6448-0822

References

DuPlain, R., Ransom, S., & Demorest, P. 2008, Proc. SPIE, 7019, 70191A
Harris, C., & Haines, K. 2011, PASA, 28, 317
Hobbs, G. 2021, pfits: PSRFITS-format Data File Processor, Astrophysics Source Code Library, ascl:2104.013
Hobbs, G., Manchester, R. N., Dunning, A., et al. 2020, PASA, 37, e012
Lorimer, D. R. 2011, SIGPROC: Pulsar Signal Processing Programs, Astrophysics Source Code Library, ascl:1107.016
MacMahon, D. H. E., Price, D. C., Lebofsky, M., et al. 2018, PASP, 130, 044502
Niu, C.-H., Wang, Q.-X., MacMahon, D., et al. 2019, RAA, 19, 102
Paine, D., & Lee, C. P. 2014, in 2014 IEEE 10th International Conf. e-Science, Vol. 1 (Sao Paulo: IEEE), 231
Pei, X., Li, J., Duan, X., & Zhang, H. 2023, PASP, 135, 075003
Pei, X., Wang, N., Werthimer, D., et al. 2022, RAA, 22, 045016
van Cappellen, W. A., Oosterloo, T. A., Verheijen, M. A. W., et al. 2022, A&A, 658, A146
van Straten, W., & Bailes, M. 2011, PASA, 28, 1
van Straten, W., Jameson, A., & OsÅowski, S. 2021, PSRDADA: Distributed Acquisition and Data Analysis for Radio Astronomy, Astrophysics Source Code Library, ascl:2110.003
Wang, N., Xu, Q., Ma, J., et al. 2023, SCPMA, 66, 289512
Wei, D. 2019, Research on Key Technologies in Radio Astronomical Data Realtime Processing, PhD thesis, University of Chinese Academy of Sciences
Wei, J., Zhang, C., Zhang, Z., et al. 2023, SSPMA, 53, 229506
Whitney, A., Kettenis, M., Phillips, C., & Sekido, M. 2010, in Sixth Int. VLBI Service for Geodesy and Astronomy. Proc. 2010 General Meeting, ed. R. Navarro et al., 192
Yang, J., & Han, W.-l. 2022, ChA&A, 46, 309
Zhang, H.-L., Zhang, Y.-Z., Zhang, M., et al. 2023a, RAA, 23, 015023
Zhang, M., Zhang, H.-L., Zhang, Y.-Z., et al. 2023b, RAA, 23, 085012
Zhang, X., & Duan, R. 2022, Proc. SPIE, 12190, 1219032