Automation and parallelization scheme to accelerate pulsar observation data processing postprint

Authors: Xingnan Zhang, Minghui Li, Minghui Li

Date: 2025-07-25T10:18:45+00:00

Abstract

Previous studies aiming to accelerate data processing have focused on enhancement algorithms, using the graphics processing unit (GPU) to speed up programs, and thread-level parallelism. These methods overlook maximizing the utilization of existing central processing unit (CPU) resources and reducing human and computational time costs via process automation. Accordingly, this paper proposes a scheme, called SSM, that combines "Srun job submission mode", "Sbatch job submission mode", and "Monitor function". The SSM scheme includes three main modules: data management, command management, and resource management. Its core innovations are command splitting and parallel execution. The results show that this method effectively improves CPU utilization and reduces the time required for data processing. In terms of CPU utilization, the average value of this scheme is 89%. In contrast, the average CPU utilizations of "Srun job submission mode" and "Sbatch job submission mode" are significantly lower, at 43% and 52%, respectively. In terms of the data-processing time, SSM testing on the Five-hundred-meter Aperture Spherical radio Telescope (FAST) data requires only 5.5 h, compared with 8 h in the "Srun job submission mode" and 14 h in the "Sbatch job submission mode". In addition, tests on the FAST and Parkes datasets demonstrate the universality of the SSM scheme, which can process data from different telescopes. The compatibility of the SSM scheme for pulsar searches is verified using 2 days of observational data from the globular cluster M2, with the scheme successfully discovering all published pulsars in M2.

Full Text

Preamble

Astronomical Techniques and Instruments, Vol. 2, July 2025, 226–238
Article Open Access
Automation and parallelization scheme to accelerate pulsar observation data processing

Xingnan Zhang, Minghui Li*
State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China
*Correspondence: limh@gzu.edu.cn

Received: February 27, 2025; Accepted: April 22, 2025; Published Online: May 13, 2025
https://doi.org/10.61977/ati2025022; https://cstr.cn/32083.14.ati2025022

Citation: Zhang, X. N., Li, M. H. 2025. Automation and parallelization scheme to accelerate pulsar observation data processing. *Astronomical Techniques and Instruments*, 2(4): 226−238. https://doi.org/10.61977/ati2025022.

Abstract: Previous studies aiming to accelerate data processing have focused on enhancement algorithms, using the graphics processing unit (GPU) to speed up programs, and thread-level parallelism. These methods overlook maximizing the utilization of existing central processing unit (CPU) resources and reducing human and computational time costs via process automation. Accordingly, this paper proposes a scheme, called SSM, that combines "Srun job submission mode", "Sbatch job submission mode", and "Monitor function". The SSM scheme includes three main modules: data management, command management, and resource management. Its core innovations are command splitting and parallel execution. The results show that this method effectively improves CPU utilization and reduces the time required for data processing. In terms of CPU utilization, the average value of this scheme is 89%. In contrast, the average CPU utilizations of "Srun job submission mode" and "Sbatch job submission mode" are significantly lower, at 43% and 52%, respectively. In terms of the data-processing time, SSM testing on the Five-hundred-meter Aperture Spherical radio Telescope (FAST) data requires only 5.5 h, compared with 8 h in the "Srun job submission mode" and 14 h in the "Sbatch job submission mode". In addition, tests on the FAST and Parkes datasets demonstrate the universality of the SSM scheme, which can process data from different telescopes. The compatibility of the SSM scheme for pulsar searches is verified using 2 days of observational data from the globular cluster M2, with the scheme successfully discovering all published pulsars in M2.

Keywords: Astronomical data; Parallel processing; Pulsar Exploration and

Search Toolkit (PRESTO); CPU; FAST; Parkes

1. Introduction

In the 21st century, large-scale survey projects such as the Large Synoptic Survey Telescope (LSST) and the Square Kilometer Array (SKA) are ushering astronomy into a new data-intensive era [1,2]. It is expected that these projects will lead to a substantial increase in astronomical data, with volumes measured in petabytes (PB) and even exabytes (EB). Consequently, astronomy has entered the realm of big data.

Astronomy data are growing exponentially as a result of the development of several major radio telescopes and the implementation of extensive sky survey projects. For example, the 305-m Arecibo Radio Telescope in the United States generated approximately 1 PB of data during its operational lifetime [3]. The High Time Resolution Universe (HTRU) is a collaborative survey project between the Parkes Observatory in Australia and the Effelsberg Observatory in Germany designed to search for pulsars and radio transients across the entire sky [4]. Although the total data volume from the Parkes multibeam survey amounts to approximately 5 TB, with its enhanced time and frequency resolution, has significantly increased the data scale, reaching 875 TB [5].

FAST, a radio telescope independently developed by China, operates across a frequency range from 70 MHz to 3 GHz [6–8]. With enhanced sensitivity and comprehensive detection capabilities, it features 19 beams and an ultra-wideband receiver offering superior temporal and spectral resolution. The single-beam sampling frequency has been upgraded from 1,024 Hz to 4,096 Hz. When all 19 beams operate concurrently at a temporal resolution of 100 μs and a frequency bandwidth of 400 MHz, the system generates approximately 100 TB of data daily using 8-bit recording [9,10]. From July 2017 to May 2018, the Commensal Radio Astronomy FAST Survey (CRAFTS) [11] collected multi-science-target observational data across two frequency bands (0–1 GHz and 1–2 GHz), accumulating several PB of systematic pulsar survey data. This includes approximately 0.8 PB of ultra-wide pulsar drift scan data and 2 PB of 19-beam survey data. Including intermediate data generated during processing and analysis, the total exceeds 3 PB. Projections indicate that FAST will accumulate between 10 PB and 100 PB of data in the field of radio pulsar research [12].

Many studies have focused on three aspects to improve the efficiency of pulsar searches: enhancement algorithms, using the GPU to speed up programs, and thread-level parallelism. The main goal of algorithm optimization is to improve the efficiency of search operations by refining execution methods and reducing the computational load. The utilization of the fast Fourier transform (FFT) enables the acceleration of the conversion of time-domain data to frequency-domain processing, thereby reducing the computational requirements [13]. The deployment of rapid search algorithms allows for the circumvention of implausible pulsar periods, thereby reducing the search range and enhancing

the efficiency of the search process [14]. The implemented adaptive integration times permit real-time modification of the integration period, contingent on the strength of the pulsar signals and the quantity of background noise. This facilitates the discernment of signals from noise [15]. For example, Fu [16] targeted time-consuming algorithms in single-pulsar searches. They developed an optimized solution. This approach drastically cut the computational workload. It also enhanced the search efficiency and overall data-processing performance. The optimized algorithm demonstrates a speedup of approximately 1.4–2 times compared with the original program across diverse datasets and various computing configurations. Although algorithmic optimization is excellent at improving the efficiency of search operations, there are some limitations; for example, over-optimization can cause the algorithm to perform poorly when dealing with unconventional data.

The utilization of GPU acceleration has led to significant advancements. By making full use of the parallel computing capabilities of GPUs, researchers have been able to transfer computational tasks to GPUs for parallel processing, thereby greatly accelerating search speeds. For example, investigating de-dispersion algorithms in pulsar search processes using GPUs results in a reduction of the data-processing time by 50% or more [17]. Van Straten and Bailes utilize coherent dispersion removal algorithms on both the CPU and the GPU, achieving results that exceed real-time processing [18]. They utilize the GPU to enhance the removal of dispersion and periodic search codes, achieving nearly a 50-fold speedup. In the field of single-pulsar searches, parallel research has also been undertaken, including the creation of the GPU-based single-pulsar search software Heimdall and the GPU-based single-pulsar search pipeline (GSP) developed by You et al. [19]. The GSP is primarily built on the open-source pulsar exploration and analysis toolkit PRESTO (https://github.com/scottransom/presto) and achieves GPU-accelerated dispersion removal, demonstrating a remarkable speedup of 120 times compared with the original PRESTO and is 60 times faster than the MPI-parallelized version of PRESTO [20]. Although utilizing GPU acceleration brings significant advantages, there are also some drawbacks. First, the programming and tuning of GPUs are relatively complex, resulting in increased development costs. In addition, GPUs are not as efficient as CPUs at handling certain types of computing tasks, especially when the tasks frequently require accessing memory.

The implementation of thread-level parallelization significantly enhances the efficient processing of pulsar search tasks. By partitioning data into smaller units for parallel processing, it achieves a substantial reduction in search times, thereby improving the overall efficiency of the search process. For example, the use of parallel techniques on CPUs for single-pulsar searches allows for the distribution of tasks across multiple processors for simultaneous computation. This method effectively leverages the resources of multi-core processors, thereby improving the speed and efficiency of data processing. The results show that enhanced parallel single-pulsar search algorithms surpass the original search pro-

grams across various datasets and computing configurations. This is supported by speed-up factors of approximately 5–16 times [16]. These methods overlook maximizing the utilization of existing CPU resources and reducing human and computational time costs through process automation.

Section 1 of this paper provided an overview of the research background, emphasizing the inherent challenges associated with pulsar search and the necessity for parallel processing. Section 2 presents a comprehensive analysis of the fundamental principles of pulsar search, the script theory for parallel processing, and optimization methods, providing a detailed account of the implementation steps. Section 3 evaluates the performance of SSM in terms of the data-processing efficiency, CPU utilization, and generality and tests it on datasets from different telescopes. Section 4 tests the capabilities of SSM in actual pulsar searches using observational data from the globular cluster M2 on two separate days.

2. Methods

This section provides the detailed design and implementation methods used in the automated parallel pulsar search program. It starts with an overview of the key elements that support the program, followed by a comprehensive explanation of the theoretical foundations. Then, the methods, including command splitting and parallel execution, to improve the pulsar search efficiency are analyzed and discussed. These methods not only improve the data-processing efficiency but also optimize the utilization of computing resources in the cluster environment. By integrating these strategies, the program can process large datasets more efficiently, significantly improving the data-processing efficiency.

2.1. General Principle

The PRESTO software generates independent commands for different distributed metrics. This approach allows each command to run in parallel. Given this context, this paper proposes a scheme, called SSM. SSM aims to improve the overall efficiency of pulsar searches.

The SSM scheme includes modules for data management, command management, and resource monitoring. This method uses a dynamic resource monitoring function that checks the system every 2 s. When idle resources are detected, the system is designed to dispatch a predetermined number of jobs; specifically, the number of jobs is twice the number of idle cores; this multiple can be modified by the user. The primary program runs continuously, enabling it to submit and execute all jobs sequentially over time. This strategy ensures that the system can efficiently utilize available resources without overwhelming the nodes. By continuously monitoring the resource availability and dynamically adjusting the number of job submissions, SSM enhances the utilization of the computational resources, thereby accelerating the processing of pulsar search tasks.

Most scheduling systems use static job submission strategies, which struggle to

effectively adapt to real-time fluctuations in resources and load changes. For example, traditional scheduling algorithms such as the fair scheduler and the capacity scheduler often face challenges in efficiently allocating computational resources within heterogeneous computing environments. This can lead to instances of resource idleness and delayed responses, as highlighted in previous studies [19]. In contrast, SSM can dynamically adjust the number of jobs based on real-time monitoring of the resource usage and node load, ensuring stable system performance even under high-load conditions. This adaptability enables SSM to exhibit greater efficiency and responsiveness when processing large-scale pulsar data.

The core innovation of SSM lies in its comprehensive integration of parallel processing and resource optimization. By breaking down jobs into smaller units for parallel execution, SSM not only enhances the overall processing speed but also effectively reduces the performance overhead of individual jobs on the system. Many recent studies have focused on scheduling strategies based on swarm intelligence optimization algorithms; these methods often excel in specific scenarios but struggle to adjust resource allocation in real time to accommodate rapidly changing loads. Thus, SSM's innovative parallel mechanism, combined with dynamic resource management, offers a more efficient and scalable solution for big-data processing.

2.2. Script Theory

The objective of this study is to develop an automated pulsar search scheme. The scheme mainly automates the pulsar search steps and then adds data management, command management, and resource monitoring.

The left half of Fig. 1 [Figure 1: see original paper] illustrates the primary components of the study and details five essential steps and three optional steps in the pulsar data search process. The essential steps are as follows:

1. **Radio Frequency Interference (RFI) Signal Suppression (rfifind)**: This step identifies and mitigates RFI signals to improve the data signal-to-noise ratio (SNR).
2. **Data Preprocessing (prepdata/prepsubband)**: This process removes the time delay between channels caused by the dispersion effect, generates time series for various dispersion measures (DMs), and enhances the accuracy of signal reconstruction.
3. **Forward Fast Fourier Transform (realfft -fwd)**: This step transforms time-domain signals into frequency-domain representations and facilitates the analysis of the frequency components.
4. **Periodic Signal Search (accelsearch)**: This process involves searching for periodic signals within the time-domain data to identify potential pulsar signals.
5. **Pre-folding (prepfold)**: This step aligns and folds the time-domain data according to the periods of potential pulsar signals and creates phase-

averaged profiles. This enhances the signal visibility by consolidating data from multiple cycles and allows for a more precise analysis and detection of pulsar characteristics.

The optional steps are as follows:

1. **Red Noise Removal (rednoise)**: This step further eliminates red noise during data processing to enhance signal clarity.
2. **Inverse Fast Fourier Transform (realfft -inv)**: This process performs an additional inverse FFT to ensure thorough data analysis.
3. **DM-SNR Comparison Plot**: This step generates and analyzes DM-SNR comparison plots to provide comprehensive statistical support.

The upper right panel of Fig. 1 illustrates the comprehensive framework of SSM, which includes data management, command management, and resource monitoring. The data management module is designed to guide users in setting the parameters required for the pulsar search process via an intuitive interface, thereby improving the user experience. The module has a fixed configuration file, and users can specify the directory for the original FITS files by modifying this configuration file. The uploaded data must follow a specific format: `./ClusterName/ObservationDate/Beam/SpecificObservationFitsData`.
The initial level of the hierarchy displays the cluster name, the second level reflects the observation date, the third level represents the number of beams for that observation date, and the fourth level displays the number of specific FITS data files for each beam. When the data are processed, they are moved to a directory designated by the user. Therefore, the data in the original FITS storage directory comprise unprocessed data. The module checks the original FITS storage directory every 30 min to determine if there are any data files currently not being processed. Upon discovering new data, it generates a report that provides detailed information concerning the data source, observation date, beam counts, and FITS file counts. It also reminds the user to adjust the configuration parameters and submit processing instructions. After the processing instruction is approved, the system creates an empty directory to store the results. The folder structure adheres to the following format: `./ClusterName/ObservationDate/Beam/GeneratedFilesDuringProcessing`.

The command management module begins with the decomposition of the command line for the pulsar search tasks. When the necessary commands have been entered, the relevant information is extracted, including the input file, output file, and the required number of worker threads. After the parsing process is complete, the module's command split function creates independent short commands for parallel processing based on the user's configuration parameters. For example, for the globular cluster M2 data, if the user sets the DM range from 0 to 1,000 with a step size of 0.5, it will produce 2,000 short commands. It then ensures that the appropriate amount of computing power is allocated to the task at hand.

The resource monitoring module is an automatic resource monitoring system

that checks the availability of resources every 2 s. It has been designed so that users can easily access information related to each node, including its designation, operational status, and the number of available cores. To obtain the necessary data in real-time, simply enter the command `info -N -o %.18n %.10t %.18c`. The `-N` flag enables the non-interactive mode, disabling interactive features such as paged output or user prompts, and is commonly used in script automation scenarios, while the `-o` option specifies the output format via a template syntax where `%.18n` defines a fixed-width 18-character name field (n, padded with spaces or truncated as needed), `%.10t` sets a 10-character time field (t), and `%.18c` designates an 18-character classification field (c), ensuring structured and aligned output for consistent parsing or display, although the actual behavior may depend on the specific tool implementation. When idle cores are detected on one or more nodes, the system begins gathering relevant information, such as partition names, node names, and core counts. Based on these data, the system generates multiple snatch command texts, typically twice the number of available cores. This method optimizes resource utilization. The header information included in the sbatch text serves to ensure that all relevant configuration parameters are accurately specified for the particular job in question. When the commands are generated, the system submits them for resource requests and utilizes srun to execute tasks in parallel. This is advantageous because it allows the execution of multiple commands on a single node, thereby enhancing the overall efficiency.

Furthermore, the ability to submit two srun jobs on a single core is advantageous because it enables optimal utilization of the CPU. The resource monitoring module can function independently, initiate tasks without manual intervention, process jobs in batches, and promptly release cores upon job completion. In the event of insufficient resources, the jobs will remain waiting, preventing a lengthy queue from forming and ensuring efficient resource utilization and the expeditious processing of jobs.

2.3. Optimizing Pulsar Search with Command Splitting and Parallel Execution

Developing a parallel pulsar search program utilizing PRESTO software is well established. The core innovations of this program are command splitting and multi-process parallel execution. Command splitting is an effective technique that facilitates the division of a single command into smaller, more manageable units. First, the original long command that requires serial execution is split into multiple short commands based on DM values, where each DM value corresponds to several short commands such as rffind, prepdata, …, accelsearch, and prepfold. Second, the sequential short commands associated with a DM value (rffind→prepdata→…→accelsearch→prepfold) are configured as an independent processing pipeline to request computing cores and execute tasks. A schematic representation of this segmentation mechanism is illustrated in Fig. 2A [Figure 2: see original paper], while Fig. 2B details the corresponding command

submission workflow within the command management module.

1. Real-time and comprehensive monitoring of the availability of computational resources in the cluster to ensure the timely submission of tasks.
2. When there are idle cores, use the sbatch command to request these available resources.
3. The system automatically assigns multiple short commands to these cores via "Srun job submission mode", allowing each core to handle a large number of tasks simultaneously.

3. Performance Test on SSM

This section presents the SSM performance via three aspects: the data-processing efficiency, CPU utilization, and generalizability. The experiments are conducted in an environment equipped with a 100-core Intel Xeon processor, ensuring robust computational resources for high-performance analyses. First, based on the FAST globular cluster M2-20230410 dataset, the data-processing efficiency and CPU utilization of SSM are compared with two other methods, providing insights into its effectiveness when handling large datasets. Subsequently, the Parkes PMPS01 dataset is used to verify the generalizability of the SSM program between different datasets.

3.1. Practical Implementation and Usage

Based on PRESTO's excellent pulsar search ability, this study further designed the SSM program with a fast data-processing ability.

3.1.1. Environment setup The hardware foundation of this experiment is based on a high-performance computing cluster environment. Regarding the CPU resources, the system employs multiple models of Intel Xeon processors, with single-node core-counts ranging from 48 to 56, achieving a total CPU core count of 3,752. Some nodes are equipped with Intel 6338 Xeon processors (30 nodes, 48 cores per node), while others utilize Intel 6331 Xeon processors (20 nodes, 56 cores per node), providing robust support for large-scale parallel computing. Regarding the memory configuration, the maximum memory capacity of a single node reaches 2 TB, based on the Intel 6336 Xeon processor, ensuring the efficient processing of large datasets.

The software environment is built on the CentOS 7.9 operating system, with various configurations implemented to efficiently execute the proposed solution. The Python environment uses version 3.8.5, independently deployed through Anaconda3-2020.11, with upgrades to setup tools and compatibility optimizations for dependency libraries. Within the astronomical computing toolchain, the system integrates the PRESTO pulsar search framework, along with FFTW (supporting single-precision floating-point optimization), CFITSIO (for astronomical I/O), TEMPO/TEMPO2 (pulsar timing software). Custom compilation ensures seamless integration of library paths (e.g.,

LD_{{{LIBRARY}}_{{PATH}}}) with Python bindings (PYTHONPATH). For system-level dependencies, complete MPI implementations (including OpenMPI and MPICH), the GCC compiler suite, and scientific computing libraries (such as BLAS, LAPACK, and ATLAS) were installed via yum, with compatibility issues of libpng12 resolved to support cross-component collaboration. PGPLOT (graphics library) was also installed.

Before utilizing the SSM program, users must ensure that PRESTO and its dependent libraries (such as NumPy and SciPy) are fully installed and configured. SLURM [21] scheduler workload manager is required for job scheduling and resource management in this cluster environment. Additionally, the cluster must be properly configured to guarantee recognition and interconnectivity among all computational nodes. Node-to-node communication should be validated using commands such as ping or SSH to ensure seamless collaboration during distributed processing tasks.

These configurations collectively provide highly reliable hardware support and a full-stack software ecosystem for pulsar data processing, signal searching, and astronomical numerical simulations.

3.1.2. Data preparation This part of the work is done by the data management module. Users should organize the data into a structured format. Directory configuration: Create a base directory structure in a configuration file, specifying paths such as `./ClusterName/ObservationDate/Beam/SpecificObservationFITSData`. Ensure that the directory is accessible and that the system can traverse it to locate new data files. Initial data filtering: A pre-processing report should be generated using scripts to analyze the volume and structure of the incoming data. This stage serves to guarantee that the data are free from any contamination and are thus fit for processing. Utilize the `ls` command to enumerate the files and ascertain their formats, thereby ensuring compliance with the anticipated format.

3.1.3. User configuration and customization Users are encouraged to personalize their configuration files, which will allow them to specify particular parameters that are relevant to their search. Configuration file: The user configuration file should list the reference values of the parameter settings and their contents, including the DM value, ZMax value, and NUMBER value. Users can adjust the parameters according to their specific research requirements.

3.1.4. Command splitting and job submission This is done by the command management module, which recommends using the following methods for submitting jobs.

Command splitting: The processing pipeline should be divided into distinct commands based on the different DM values. The commands should be created as follows: `- rfifind -ignorechan 655:819 -time 2.0 -o globular cluster M2_{20230410}2.0/groups/g9800070/home/username/globular`

```
cluster M2/20230410/M01/*.fits    -    prepdata -ignorechan 655:819
-zerodm -dm 43.52 -nobary -mask /groups/g9800070/home/username/globular
cluster M2/test/new test5 SRB/20230410/M01/*.mask -o globular M2
username 20230410_{DM43}.52/groups/g9800070/home/share/GC/collated
data/username test/globular cluster M2/globular cluster M2_{20230410}/20230410/M01/*.fits
cluster    -    realfft -fwd /group/g9800070/home/username/globular
cluster M2 test/new test/20230410/M01/globular cluster M2 username
20230410 DM43.52.dat - accelsearch -sigma 1.5 -zmax 50 -numharm 32
/groups/g9800070/home/username/globular cluster M2 test/new/20230410/M01/globular
cluster M2 username 20230410 DM 43.52.fft - prepfold -noxwin -topo
-nosearch -n 64 -npart 128 -dm 43.52 -nsub 64 -o Pms 2.000000
globular cluster M2 username 20230410 DM43.52 -accelcand 3 -accelfile
20230410 globular cluster M2 DM43.52 ACCEL 20.cand 20230410
globular cluster M2 DM43.52.date
```

Job submission: Use the SLURM scheduler sbatch command to submit jobs for each DM value. For example: - `sbatch job-cpu1-task54-20240619162508.sh` - `sbatch job-cpu10-task26-20240619162516.sh` - `sbatch job-cpu12-task48-20240619162519.sh`

3.1.5. Resource monitoring Set up a monitoring function (Fig. 3 [Figure 3: see original paper]) to continuously check the available computing resources. This monitoring function includes two parts: real-time acquisition of idle resource information and idle information analysis. It runs commands to query the status of each node and obtain real-time information concerning idle cores, allowing for a comprehensive overview of the computing environment.

Use the following command to query for idle resource information: `sinfo -N -o %.18n %.10t %.18c`.

The script's real-time monitoring function analyzes resource utilization every 2 s to identify idle cores and submit tasks. Tasks are allocated using a queuing technique to enhance resource efficiency and ensure maximum computing power. The script's real-time monitoring function analyzes the resource utilization every 2 s to detect which cores are idle, allowing jobs to be submitted when idle cores are available. It then analyzes the identified idle core information and automatically extracts the corresponding number of commands from the total command text, generating the required small command text and jobs, which are then submitted via the sbatch command. Notably, the program employs a queuing technique when submitting jobs, enabling multiple tasks to be submitted on each core, with the number of jobs determined by a set of predefined parameters.

Furthermore, real-time monitoring enhances the system's responsiveness to resource fluctuations. By continuously acquiring the idle resource information, the monitoring function can adaptively allocate tasks, ensuring that idle cores are utilized promptly. It minimizes idle time and maximizes throughput, leading to faster data processing. Additionally, the queuing technique not only

optimizes core usage but also balances the workload across available resources. Overall, this comprehensive resource monitoring strategy significantly improves the efficiency of task operations.

3.1.6. Executing the pulsar search After the data are organized according to specific requirements and the command has been submitted, the user can start searching for pulsars using the PRESTO software. The SLURM Scheduler is used to monitor the status of the job to identify and correct any issues that arise in real time, ensuring that the program runs more stably and minimizing potential downtime.

3.2. Test Results

The test consists of two parts, which are tested based on the FAST dataset and Parkes dataset, respectively. Tests based on the FAST data focus on the time consumed by data processing and CPU utilization during processing. The results are compared and analyzed to highlight the differences between the SSM method and the two other methods, i.e., "Srun job submission mode" and "Sbatch job submission mode". Section 3.2.2 tests the universality of the SSM scheme, based on the Parkes data.

3.2.1. Test result on FAST data The SSM method was introduced in detail in Section 2; its command submission mode is based on the "Srun job submission mode" and "Sbatch job submission mode", incorporating a monitoring function and queue technology.

The tests were performed in a CPU Normal Partition configured with approximately 100 cores of an Intel Xeon CPU. The globular cluster M2-20230410 dataset has a file size of approximately 2.189 TB, a bandwidth of 500 MHz consisting of 2048 channels, a channel width of 0.244140625 MHz, and an observing time of 7,158 s. To facilitate data processing, several parameters were established: the DM parameters range from 43.000 to 44.500 with a step size of 0.001, the ZMax parameter is set to 20, the NUMBER parameter (accelsearch -numharm) is set to 32, and the CHANNELS (prepfold -n) parameter is set to 64. In terms of the CPU utilization, the three methods are illustrated in Fig. 4 [Figure 4: see original paper]. Fig. 4A shows the CPU utilization of the "Srun job submission mode". It can be seen the CPU utilization remains relatively low, peaking at approximately 13%. There are two reasons for the observed low CPU utilization. First, there is a job scheduling delay. When a large number of jobs are submitted, the scheduling system requires time to allocate resources. The earlier jobs are gradually scheduled onto the available computing nodes, resulting in waiting times. Second, the jobs require a significant amount of I/O operations at the start, waiting for the same batch of FITS data to be loaded, leading to competition among jobs and consequently reducing CPU utilization. Note that between 18:00 and 22:00, the utilization rate rises significantly, exceeding 80%. This is due to the end of job competition; each job can read

data, sequentially increasing the CPU utilization. Despite the subsequent peak indicating that the resource utilization is effective, the earlier low levels suggest that there may still be room for improvement in the workload allocation.

Fig. 4B shows the CPU utilization of the "Sbatch job submission mode". During periods of high computational demand, the average utilization of the "Sbatch job submission mode" reaches 51%, with a maximum utilization of 80%. Compared with the "Srun job submission mode", the "Sbatch job submission mode" employs a sequential submission approach rather than submitting jobs in bulk at the same time, which helps avoid competition issues caused by jobs reading FITS data simultaneously. However, the maximum utilization of the "Sbatch job submission mode" is only 80%, because the "Sbatch job submission mode" submission method still follows the principle of one job per core, and a single job does not fully utilize all the resources of a core, resulting in resource idle issues.

Fig. 4C shows the CPU utilization of the SSM method. It achieves 100% utilization at the beginning and maintains above 80% utilization for an extended period. This is due to the method using a sequential job submission approach similar to the "Sbatch job submission mode" while utilizing queueing techniques, allowing multiple jobs to run on a single core, thereby making full use of the core resources. Because the jobs are completed one after another, resource utilization also shows a linear decline.

In terms of data-processing duration, the "Sbatch job submission mode" takes the longest at 14 h because of the limited availability of cores, which allows a maximum of only 56 tasks to run on a single node. Although this test was conducted manually across three nodes with simultaneous job execution, the absence of real-time resource monitoring led to significant inefficiencies and wasted resources. Meanwhile, the "Srun job submission mode" improves the processing time, reducing it to 8 h, but it still faces limitations with a maximum of 100 jobs running simultaneously. In contrast, the SSM method effectively addresses these challenges by utilizing advanced cross-node and real-time task submission technologies, completing the same data processing in just 3.5 h, showcasing its superior efficiency and resource utilization. This remarkable reduction in processing time not only enhances productivity but also allows for a better allocation of computational resources, ultimately leading to more effective and timely data analyses.

This test compares the "Srun job submission mode" and "Sbatch job submission mode" with the SSM method. The test results show that SSM is superior to traditional methods with respect to data-processing efficiency and resource utilization. The SSM method completes the dataset testing in 5.5 h, which represents 39.29% of the 14-hour runtime required by the "Sbatch job submission mode". The average CPU utilization for the SSM method is 89%, while the "Srun job submission mode" and the "Sbatch job submission mode" have average utilizations of 42.78% and 51.88%, respectively. Note that, for 87% of the time, the CPU utilization of the SSM method exceeded 80%.

3.2.2. Test result on Parkes data Section 3.2.1 tested the SSM processing efficiency and CPU utilization with FAST datasets, indicating that the SSM method can accelerate the processing speed of astronomical observation data. To further test the universality of the SSM method in processing data from different telescopes, the test environment used in this section also utilized an Intel Xeon system and 100 cores in the CPU normal partition.

The PMPS01 dataset comes from Parkes, with a size of 0.365 TB and an observation duration of 1,203 s. The DM is set to a minimum of 2.00 and a maximum of 200.00, with a step size of 0.10; the ZMax parameter is set to 50, the NUMBER parameter is set to 32, and the CHANNELS parameter is set to 64. These parameters generate 1,980 individual commands, which are then submitted using SSM.

As shown in Fig. 5 [Figure 5: see original paper], the CPU utilization of SSM in this test remained above 80% from 14:40 to 14:48, indicating that SSM can effectively utilize core resources on a new dataset. However, note that the maximum core resource utilization is 83% and does not reach 100%. This is because the amount of data processed by a single task decreases, which leads to a reduction in the core resources required for the task. In other words, running only two tasks per core does not fully occupy the core resources, and it is suggested that more than two tasks should be run per core to achieve 100% resource utilization. However, to ensure consistent parameters in both experiments, only two tasks were run per core in this test.

Tests on the Parkes data demonstrated that SSM can handle observational data from different telescopes. In addition, testing datasets with different amounts shows a nonlinear relationship between the data-processing time and the data size. The efficiency is 456 MB/s when processing the Parkes dataset (Efficiency=Total data volume/Total processing time). As the dataset size increases, the processing time decreases. It is reasonable to infer that this is caused by the constraint on the upper bound of I/O reads. When the amount of data to be read is too large, the upper limit of I/O reading is reached, and the task cannot be provided with all the data at once; consequently, it has to wait in a queue, resulting in a longer time.

3.3. Tests on Pulsar Searches

The performance of SSM was well demonstrated in Section 3. However, to further test SSM's ability to search for actual pulsars, this section describes a new test using observational data from the globular cluster M2 on 2 days, January 4, 2020, and April 10, 2023. The main purpose of this test is to verify that the SSM scheme works well for pulsar searches.

3.3.1. Data and test The dataset used in this test was derived from an observation of the globular cluster M2 conducted by FAST on January 4, 2020, and April 10, 2023. Detailed information is provided in Table 1 . The hardware

environment for this test is consistent with that described at the beginning of Section 3. To effectively use the SSM program, it is essential for users to first ensure that the necessary software packages, such as PRESTO and its dependencies (including NumPy and SciPy), are properly installed.

According to the DM range and step size, 1,500 commands were generated from the observation data in 1 day, which were submitted to the SSM, and the results were verified by screening to test the SSM's ability to actually search for pulsars. To improve the efficiency of the data processing, the original data storage address, the result file storage address, the ZMax value, and the NUMBER value were written in the same configuration file, allowing the program to automatically process the data.

Table 1. Observation parameters for different dates

Parameter	January 4, 2020	April 10, 2023
File size/TB		
Observation time/s		
DM Min/cm^{-3} pc		
DM Max/cm^{-3} pc		
Step size		
Number of commands		

3.3.2. Results The total running time of the test was 7 h and 40 min, and the average CPU utilization was 92%. Ten pulsar candidates were selected from the result, and the analysis charts of these pulsar candidates are shown in Fig. 6 [Figure 6: see original paper].

These 10 pulsar candidates are compared with the published DM and P-value of the known pulsars of globular cluster M2 (Table 2). Within the acceptable error range of DM, the equation for calculating the harmonics based on the P-value is:

$$P_2 = P_{11} n; \quad (1)$$

where P_1 is the period of the known pulsar, P_2 is the period of the suspected pulsar found in the test, and n is the harmonic number. Thus, the calculation using the P-values helps determine Fig. 6A, F, G, I, and J.

In conclusion, by processing 2 days of globular cluster M2 observation data, this test confirmed that the SSM method can significantly improve the data-processing efficiency and CPU utilization and successfully search all known pulsars in the globular cluster M2. This is possible because the original pulsar search program is compatible with SSM.

Table 2. Pulsars in M2 [22] (NGC 7089)

Released pulsars in Globular Cluster M2	Information from this test
Pulsars	J2133–0049A J2133–0049B J2133–0049C J2133–0049D J2133–0049E J2133–0049F J2133–0049G J2133–0049H J2133–0049I J2133–0049J
Period/ms DM/(cm^{-3} pc)	43.3 ± 0.24 43.8 ± 0.080 44.1 ± 0.10 43.6 ± 0.010 43.8 ± 1.3 43.4 ± 0.80 43.3 ± 2.6 43.2 ± 0.12 44.1 ± 1.6 43.5 ± 2.1
Period/ms DM/(cm^{-3} pc)	
Harmonic characteristics	9th Harmonic 3rd Harmonic 2nd Harmonic 2nd Harmonic 2nd Harmonic

4. Discussion and Conclusions

This study presented a novel approach to process-level parallel computation based on PRESTO. This method is designed to deal with the rapid growth of astronomical data and uses the CPU's parallel processing ability to decompose complex astronomical data-processing tasks into manageable parallel sub-tasks, which optimizes the computational efficiency and resource utilization. To deal with large datasets more efficiently, the SSM scheme consists of three modules: data management, command management, and resource management. The data management module is mainly for the real-time monitoring of new data, and when new data are found, it reminds the user to fill in the configuration file to further process these new data. The most important part of the command management module is command splitting and submission. The most important part of the resource management module is the monitoring of idle resources in real time and its interactions with the command management module once idle resources appear, so that jobs can be quickly submitted to idle cores.

Based on the 20230410 dataset of globular cluster M2 from FAST, we compared the testing efficiency and CPU utilization of the SSM method with two other common methods. The results show that SSM effectively improves the CPU utilization and reduces the time required for data processing. In terms of the CPU utilization, the average value of this scheme is 89%, while the average CPU utilizations of "Srun job submission mode" and "Sbatch job submission mode" are significantly lower, at 43% and 52%, respectively. The Parkes PMPS01 dataset was used to verify the universality of SSM on different datasets.

To further test SSM's ability to search for pulsars, we performed a new test using observations of the globular cluster M2 over 2 days, January 4, 2020, and April 10, 2023. All known pulsars in globular cluster M2 were found by the SSM

scheme. This test verifies the SSM scheme's performance in a pulsar search. Future research can further refine this approach by integrating advanced machine-learning techniques for pulsar detection, optimizing task-scheduling algorithms to accommodate varying workloads, and extending the methodology to other domains requiring high-performance computing solutions. In addition, future research can investigate the expansibility of this approach on larger datasets or more complex scene applications.

Acknowledgements

This work is supported by the National Nature Science Foundation of China (12363010). Minghui Li is supported by the Guizhou Provincial Basic Research Program (Natural Science) (ZK[2023]039) and the Key Technology R&D Program ([2023] 352). This work made use of the data from FAST. Thanks to associate researcher Zhichen Pan of the National Astronomical Observatories, Chinese Academy of Sciences, among others. We thank for the computing support of the State Key Laboratory of Public Big Data, Guizhou University.

AI Disclosure Statement

During the preparation of this study, the DeepSeek artificial intelligence tool was utilized to assist with English grammar checking and text polishing.

Tool name and version: ChatGPT-4, DeepSeek R1.
Purpose and implementation: The tool was applied throughout the manuscript to correct grammatical errors.
Scope of impact: The AI tool did not influence the study's core conclusions, experimental data, or methodological design.
Author responsibility: The authors carefully reviewed, edited, and revised the texts to their own preferences, assuming ultimate responsibility for the content of the publication.

Author Contributions

Minghui Li participated in manuscript review and editing, contributed to drafting the original version, resource acquisition, methodological design, investigation, and conceptualization. Xingnan Zhang undertook software development, assisted in manuscript writing, and performed data curation. All authors read and approved the final manuscript.

Declaration of Interests

The authors declare no competing interests.

References

1. Feigelson, E., Babu, G. 2012. Big data in astronomy. *Significance*, 9(4): 22−25.
2. Zhang, Y. X., Zhao, Y. H. 2015. Astronomy in the big data era. *Data Science Journal*, 14.
3. Huang, J. S. 2021. Big Data Astronomy. *China Public Science*, (5): 16−17. (in Chinese)
4. Olausen, S. A., Kaspi, V. M. 2014. The McGill Magnetar catalog. *The Astrophysical Journal Supplement Series*, 212(1): 6.
5. Zhang, X. 2018. Pulsar search acceleration based on distributed computing and research and implementation of feature extraction and identification of pulsar candidates. Master thesis, Guizhou Normal University. (in Chinese)
6. Nan, R. D., Li, D., Jin, C. J., et al. 2011. The Five-Hundred-Meter Aperture Spherical Radio Telescope (FAST) project. *International Journal of Modern Physics D*, 20(6): 989−1024.
7. Jiang, P., Tang, N. Y., Hou, L. G., et al. 2020. The fundamental performance of FAST with 19-beam receiver at L band. *Research in Astronomy and Astrophysics*, 20(5): 064.
8. Qian, L., Yao, R., Sun, J. H., et al. 2020. FAST: Its scientific achievements and prospects. *The Innovation*, 1(3): 100053.
9. Zhang, H., Xie, X. Y., Li, D., et al. 2021. An accelerated method and system for PB-scale pulsar data processing in FAST. *Astronomical Research and Technology*, 18(1): 129−137. (in Chinese)
10. Yan, Z., Shen, Z. Q. 2017. FAST: A sharp tool to do pulsar studies. *Science & Technology Review*, 35(24): 16−19. (in Chinese)
11. Li, D., Pan, Z. C. 2016. The Five-hundred-meter Aperture Spherical Radio Telescope project. *Radio Science*, 51(7): 1060−1064.
12. Li, D., Wang, P., Qian, L., et al. 2018. FAST in space: considerations for a multibeam, multipurpose survey using China's 500-m Aperture Spherical Radio Telescope (FAST). *IEEE Microwave Magazine*, 19(3): 112−119.
13. Yang, C. 2015. Research and design of parallel memory system for FFT algorithm. Master thesis, National University of Defense Technology (NUDT). (in Chinese)
14. Hu, Q. C. 2023. Research and implementation of pulsar phase characteristic search method for heterogeneous computing platform. Master thesis, Southwest University of Science and Technology. (in Chinese)
15. Yang, D. L. 2010. Research on application of CMOS image sensor in two dimensional photoelectric autocollimation. Master thesis, Graduate University of Chinese Academy of Sciences (Xi'an Institute of Optics and Precision Mechanics). (in Chinese)
16. Fu, Z. M. 2023. Research on algorithm optimization and parallelization based on PRESTO single pulse search. Master thesis, Guizhou Normal University. (in Chinese)
17. Morello, V., Barr, E. D., Cooper, S., et al. 2019. The high time resolution

universe survey–XIV. Discovery of 23 pulsars through GPU-accelerated reprocessing. *Monthly Notices of the Royal Astronomical Society*, 483(3): 3673−3685.
18. van Straten, W., Bailes, M. 2011. DSPSR: Digital signal processing software for pulsar astronomy. *Publications of the Astronomical Society of Australia*, 28(1): 1−14.
19. You, S. P., Wang, P., Yu, X. H., et al. 2021. A GPU based single-pulse search pipeline (GSP) with database and its application to the Commensal Radio Astronomy FAST Survey (CRAFTS). *Research in Astronomy and Astrophysics*, 21(12): 314.
20. Ransom, S. 2025. Presto: Open source pulsar search and analysis toolkit, v. 5.0.3, GitHub, license: GPL-2.0.
21. Moll, F. 2018. Slurm Overview. Available from https://slurm.schedmd.com/ [Accessed 2025−04−21].
22. Pulsars in Globular Clusters. Available from https://www3.mpifr-bonn.mpg.de/staff/pfreire/GCpsr.html [Accessed 2025−04−21].

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