Performance Optimization Methods for Low-Frequency Radio Pulsar Search

Authors: Wei Jianwen, Zhang Chenfei, Zhang Zhongli, Ting Yu, Lin Xinhua, An Tao, Zhongli Zhang

Date: 2022-06-28T00:00:00+00:00

Abstract

With the construction and operation of large scientific facilities such as the Square Kilometre Array (SKA) radio telescope, and the emergence of innovative platforms for big data and high-performance computing, the interconnection between astronomy and high-performance computing is becoming increasingly profound. Astronomical computing, particularly pulsar search as one of the primary scientific objectives of SKA, is characterized by large data volumes and computationally intensive workloads. This paper presents a scheme for accelerating the pulsar search pipeline based on OpenMP multi-threading and multi-processing techniques, proposes a method for addressing load imbalance issues, and successfully deploys the optimized pipeline on both x86 and ARM compute nodes of the China SKA Regional Centre prototype. Through pulsar search tests utilizing observation data from the Murchison Widefield Array (MWA), the pipeline achieves speedups of 10.4–12.2 and 24.5–27.6 times compared to the original single-threaded method on the respective platforms. The ARM platform demonstrates 1.1–1.3 times faster computation than the x86 platform, revealing its substantial potential for SKA data processing. The optimized pulsar search pipeline deployed on the China SKA Regional Centre prototype will be primarily applied in the near future to low-frequency pulsar searches for the MWA South Galactic Rapid Two-metre Survey (SMART) project, to satisfy various scientific requirements including gravitational wave detection timing arrays.

Full Text

Parallel Optimization of the Pulsar Search Pipeline on the China SKA Regional Centre Prototype

Jianwen Wei[1], Chenfei Zhang[1], Zhongli Zhang[2,3,4*], Ting Yu[2,5], James Lin[1], Tao An[2]

[1]Shanghai Jiao Tong University, Shanghai 200240, China
[2]Shanghai Astronomical Observatory, Chinese Academy of Sciences, Shanghai 200030, China
[3]Key Laboratory of Radio Astronomy, Chinese Academy of Sciences, Nanjing 210008, China
[4]School of Astronomy and Space Science, University of Chinese Academy of Sciences, Beijing 100049, China

*Corresponding author, E-mail: zzl@shao.ac.cn
Received: 2022-06-28; Accepted: 2022-07-xx

Funding: This work is supported by the China SKA Program (No. 2020SKA0120200), the National Key R&D Program of China (No. 2018YFA0404603), the National Natural Science Foundation of China (Nos. 12041301, 11873079), and the Youth Innovation Promotion Association of the Chinese Academy of Sciences (No. 2021258).

Abstract

With the construction and operation of major scientific facilities such as the Square Kilometre Array (SKA) and the proposed innovative platforms for big data and high-performance computing, the connection between astronomy and high-performance computing is becoming increasingly close. Astronomical computation, particularly pulsar search as one of the primary scientific directions of the SKA, is characterized by massive data volumes and intensive computational requirements. This paper presents an approach to accelerate the pulsar search pipeline based on OpenMP multithreading and multiprocessing techniques, proposes a method to solve the load imbalance problem, and successfully deploys the optimized pipeline on both x86 and ARM compute nodes of the China SKA Regional Centre Prototype (CSRC-P). Through search tests using pulsar observation data from the Murchison Widefield Array (MWA), the pipeline achieves speedups of 10.4–12.2× and 24.5–27.6× compared to the original single-threaded method on x86 and ARM platforms, respectively. The ARM platform demonstrates 1.1–1.3× faster computation than the x86 platform, revealing its tremendous potential for SKA data processing. The optimized pulsar search pipeline deployed on CSRC-P will soon be applied to low-frequency pulsar surveys for the MWA Southern-sky Rapid Two-metre (SMART) program to meet various scientific needs, including pulsar timing arrays for gravitational wave detection.

Keywords: Square Kilometre Array, pulsar, pulsar search, high performance computing, parallel optimization
PACS: 47.27.-i, 47.27.Eq, 47.27.Nz, 47.40.Ki, 47.85.Gj

1. Introduction

Pulsars are rapidly rotating neutron stars with strong magnetic fields, typically having radii of about 10–12 km and masses approximately 1.4 times that of the Sun [1]. First discovered in low-frequency radio observations in 1967 [2], pulsars hold significant value in time-domain astronomy, dense object structure studies, interstellar medium modeling, and gravitational wave detection. Observational studies show that pulsar radiation originates from magnetic poles, and when the rotation axis is inclined relative to the magnetic axis, a lighthouse effect produces periodic pulse signals that can be stably received by observers on Earth.

The tremendous scientific value of pulsars in astronomy and physics is built upon large sample sizes, and pulsar search efforts have been conducted globally. Examples include the Arecibo L-band Feed Array pulsar survey (PALFA) [3] using the Arecibo telescope in the United States and the High Time Resolution Universe pulsar survey (HTRU) [4, 5] using the Parkes telescope in Australia. China' s Five-hundred-meter Aperture Spherical radio Telescope (FAST) [6–8] has discovered over 350 new pulsars to date (e.g., [9–12]), along with more than 200 pulsar candidates from the FAST CRAFTS survey [13, 14]. Meanwhile, low-frequency pulsar surveys below a few hundred MHz are also underway. For instance, the GBNCC survey [15] in the United States uses the Green Bank telescope at 350 MHz to search the entire sky north of -40° declination, having discovered 195 pulsars. The Arecibo 327 MHz survey has discovered 96 pulsars since 2010 [16, 17]. The LOFAR low-frequency radio telescope in the Netherlands has discovered 73 pulsars at even lower frequencies of 135 MHz [19, 20]. To date, over 3,300 radio pulsars have been discovered (ATNF catalog [21]), most of which are distributed near the Galactic plane, with only a small fraction at high Galactic latitudes, consistent with the theory that pulsars are produced by supernova explosions.

The first phase of the Square Kilometre Array (SKA1), representing 10% of the full array, is expected to discover more than 18,000 pulsars—over five times the currently known number [22]. Searching for more pulsars requires longer integration times, and computational demand grows with the cube of observation time, making it urgent to improve pulsar search speeds. Several efforts have utilized modern multi-core processors to optimize pulsar searches. Yu et al. [23] tested the effects of hyperthreading and different thread counts on the performance of the PRESTO pulsar search tool [24], though further code optimization is possible. Dimoudi et al. [25] optimized pulsar searches on GPUs using the Fourier Domain Acceleration Search (FDAS) algorithm, but this GPU implementation has not been integrated into PRESTO. Wang et al. [26] used FPGA chips to accelerate pulsar searches, but FPGA chips are not widely used by astronomers for scientific data processing and do not integrate well with PRESTO. In contrast, multi-core CPUs based on x86 or ARM architectures support over 95% of workloads, making them ideal platforms for accelerating pulsar searches.

Due to the steep spectra of pulsars, they are theoretically easier to detect at lower frequencies, making low-frequency radio observations crucial for finding new pulsars [22]. Additionally, low-frequency telescopes have larger fields of view (FoV). However, low-frequency pulsar detection is challenging due to interference from the low-frequency environment and scattering effects from the interstellar medium. With the operation of SKA-low precursor projects [27], low-frequency pulsar search techniques continue to advance. SKA-low will become the most advanced low-frequency pulsar detection telescope.

Before SKA construction is complete, many SKA pathfinder instruments are already in operation. The Murchison Widefield Array (MWA) in Western Australia [28] is a precursor to SKA-Low, with a field of view that includes the Galactic center region where pulsars are most densely distributed. The MWA Voltage Capture System (VCS) [29] covers a frequency range of 70–300 MHz, subdivided into 3,072 frequency channels across a total bandwidth of 30.72 MHz, achieving a frequency resolution of 0.1 kHz. With 0.1 ms time sampling, each hour of observation generates 28 TB of data. Despite the enormous data volume posing significant computational challenges, MWA is expected to detect 230 new pulsars [30], while future SKA1-Low is expected to discover approximately 11,000 pulsars [22].

In the SKA era, the limitations of single-dish radio telescopes in sensitivity and survey efficiency will be completely overcome. SKA is expected to achieve unprecedented high sensitivity, large field of view, and high survey speed while generating massive observational data [27, 31]. Upon completion of SKA1, the data generation rate could reach terabytes per second [32, 33]. SKA1 will require at least 300 Pflops of floating-point operations per second for data processing [31], nearly equivalent to the capability of the world's fastest supercomputer (e.g., Fugaku' s maximum performance is 442 PFlops [34]). Considering transmission and computational efficiency, the actual data processing requirements will far exceed this theoretical estimate.

Successful processing and storage of massive data volumes is a prerequisite for SKA' s normal operation. Therefore, SKA member countries plan to establish their own Regional Centres (SRCs) to perform deep scientific data analysis, long-term data product storage, and support global scientific users [35]. Distributed SRCs will form a global collaborative and coordination network responsible for SKA scientific data products [36]. The design, development, operation, and maintenance of the global SRC network will meet the needs of the SKA Observatory and national/regional strategic development.

The Shanghai Astronomical Observatory is leading the development of the China SKA Regional Centre Prototype System (CSRC-P), which has built the world' s first SRC prototype [37, 41] and its software system [39]. Upon completion of the SKA Regional Data Centre, it will be based on MWA while also expanding cooperation with ASKAP, MeerKAT, and LOFAR to conduct scientific pre-research and technological development for SKA precursor projects, including low-frequency radio continuum survey observations [40] and the low-

frequency radio pulsar search research presented in this paper. Before full SRC construction, prototype development and testing are crucial. CSRC-P undertakes tasks including data processing technology verification, data challenge testing, data processing pipeline development, and data analysis for SKA precursor and pathfinder telescopes. CSRC-P adopts a hybrid heterogeneous computing architecture using both x86 and ARM [42] processor clusters, making it the first SKA data processing system to use ARM architecture. As demonstrated at the 2019 SKA Engineering Conference, ARM compute nodes showed excellent overall performance in benchmarking tests. CSRC-P is equipped with multi-core processors and excellent storage and network systems, providing an ideal test platform for improving the pulsar search pipeline.

This paper introduces optimization methods implemented for the pulsar search pipeline on modern multi-core processor platforms, particularly for CSRC-P design, and presents search test results on four incoherent sum datasets from MWA-VCS. The paper is organized as follows: Section 2 describes our pulsar search pipeline, Section 3 presents our optimization methods for hotspot steps, and Sections 4 and 5 provide performance evaluation and summary, respectively. Notably, this pipeline will soon be applied to low-frequency pulsar surveys for the MWA SMART [43] program.

---

2. Pulsar Search Pipeline

PRESTO [24], developed by Scott Ransom, is the most widely used pulsar search pipeline. PRESTO was originally designed to efficiently search for millisecond pulsar binaries from long observations of globular clusters. To date, over 600 pulsars have been discovered using PRESTO, including more than 230 millisecond pulsars and millisecond pulsar binaries [44]. The PRESTO code is primarily written in ANSI C, with many routines in Python. The pulsar search pipeline using PRESTO is shown in Figure 1 [Figure 1: see original paper] and consists of six steps: (1) Remove narrowband radio frequency interference (RFI); (2) De-dispersion; (3) FFT and remove red noise; (4) Accelerate search; (5) Candidate folding; and (6) Single pulse search. Final candidate selection for periodic and single-pulse signals currently employs manual methods, though artificial intelligence screening programs will be inserted in subsequent steps based on different observational data and requirements. The following sections describe the methods for the first six steps and the parallel optimization schemes designed for these steps.

2.1. Radio Frequency Interference Removal

RFI primarily originates from artificial radio sources and can be easily misidentified as pulsar candidates, while genuine pulsar signals may be overlooked due to interference. RFI severely impacts pulsar search and observation, significantly increasing the workload of subsequent processing steps.

PRESTO provides the rfifind.py program to identify and remove strong narrowband RFI and short-duration broadband RFI. The algorithm divides the time domain into intervals of several seconds to meet the interference removal needs of most observational data (set to 12 seconds in our tests). rfifind scans the entire data using these intervals to identify RFI, marks it, and generates a .mask file. Another type of interference to search for is periodic signals. Since radio signals are inevitably affected by the interstellar medium, signals with dispersion measure (DM) of 0 do not exist and can therefore be considered RFI. PRESTO performs a DM=0 search to identify signals with zero dispersion and records the search information in a .bird file to eliminate such interference from subsequent searches.

2.2. De-dispersion

Since electromagnetic waves of different frequencies propagate at different speeds through the interstellar medium, signals at different frequencies received by radio telescopes have time delays, causing pulse broadening. Wider bandwidth radio telescopes achieve higher sensitivity in pulsar observations, but the dispersion effect caused by the interstellar medium also becomes more significant. We must perform de-dispersion processing on the observed data.

We use the PRESTO-based prepsubband program for de-dispersion processing. The prepsubband program performs de-dispersion as follows: according to a specific dispersion measure DM, signals from different channel groups are shifted in time to generate a series of time series with different dispersion measures. The de-dispersion operation removes the RFI marked in the previous step. For relatively bright pulsars, pulse signals become visible after de-dispersion. If pulsar signals with dispersion measures within the search range exist in the data, their periods can be identified in subsequent search steps.

2.3. Fourier Transform and Red Noise Removal

After obtaining time series data with different dispersion measures, we perform discrete Fourier transforms on the sequences and conduct searches in the frequency domain. This paper uses the Fast Fourier Transform (FFT) algorithm, which can be efficiently implemented on computers, to greatly improve pulsar search efficiency.

Due to ubiquitous Brownian motion of particles in the environment, collected information is mixed with red noise, causing the low-frequency end of the frequency domain signal to rise. Therefore, for .fft files after Fourier transformation, we need to execute PRESTO' s red noise removal program to whiten the signal and generate new .fft files with red noise removed.

2.4. Acceleration Search

In the frequency domain, a real pulsar signal does not appear at only one frequency but manifests as a fundamental frequency and a series of harmonics.

To fully utilize harmonics, we use "harmonic summing" techniques to enhance pulsar signals in the frequency domain and obtain pulsar candidates.

The accelsearch program in PRESTO provides pulsar acceleration search algorithms that can search for both normal pulsars and pulsars in binary systems. The -numharm parameter can set the maximum number of harmonic summations. Theoretically, higher settings are more beneficial for searching, but the computational cost also increases exponentially. In this paper, this parameter is set to 16. The -zmax parameter in accelsearch refers to the maximum drift in the frequency domain caused by orbital motion in binary systems within the search parameter space. Larger zmax values result in longer computation times. In this paper, zmax is set to 200.

2.5. Candidate Folding

For candidates selected by the program, we fold their time series at the searched dispersion measure DM and period P and analyze their folded profiles.

PRESTO' s prepfold has two methods for folding pulse profiles: one performs a small-range search around the input DM and P values before folding to find more precise values, while the other directly uses the input DM and P values for folding. Since the optimization goal in this paper is to prepare for large-scale survey searches, we adopt the direct folding method for each candidate.

2.6. Single Pulse Search

Some pulsar signals do not exhibit obvious periodicity, such as pulsars with nulling or giant pulse phenomena. Pulsars with nulling show no pulse signals for periods of time, while giant pulses can be 100–1000 times stronger than average pulses. Both types are difficult to detect in the frequency domain. Single pulse search typically uses time-domain search algorithms. PRESTO provides single pulse search programs that can identify relatively prominent single pulse signals in time series with different dispersion measures.

---

3. Optimization Methods

3.1. Hotspot Analysis of the Pulsar Search Pipeline

The time consumption percentages of each pulsar search step are shown in Figure 2 [Figure 2: see original paper]. The hotspots are narrowband RFI removal, dedispersion, acceleration search, candidate folding, and single pulse search, which have high runtime costs and large optimization potential. FFT and red noise removal consume little time and have small optimization potential, requiring no further optimization.

Based on analysis of PRESTO' s code hotspots and programming languages, OpenMP multithreading and Python multiprocessing methods are appropriate

optimization approaches for the following reasons: First, most code runs single-core, and utilizing modern multi-core compute nodes is expected to achieve several-fold speedup. Second, pulsar multi-channel data has no dependencies, enabling better parallelization to further reduce runtime. Third, compared to multi-node parallel optimization, multithreading or multiprocessing parallel optimization is easier to implement and verify for correctness, making it a good starting point for future optimizations.

The parallel techniques used in this paper are multiprocessing, OpenMP, and load balancing methods, applied to the steps shown in Figure 1 [Figure 1: see original paper]. **Multiprocessing** is a process-level parallel technology supporting multiple computer languages, implemented using Python' s multiprocessing library [45], which uses subprocesses rather than threads to fully utilize multi-core performance on clusters. **OpenMP** [46] is a shared-memory standard for parallel programming on multi-core processors, where different cores can obtain required data from shared memory for processing. **Load balancing** [47] addresses the issue that not all cores finish work simultaneously—when some cores stop running, others may still be working, and idle cores waste performance. Therefore, we need to balance the load across all cores to avoid core idleness as much as possible.

3.2. OpenMP Optimization for Narrowband RFI Removal

We use OpenMP to optimize the narrowband RFI removal process, which consists of two steps: searching for RFI and writing .mask files. The original program calls rfifind to divide the time series into intervals of fixed seconds and searches for RFI in each interval (Algorithm 1). Since there are no data dependencies between these intervals, we add OpenMP pragma directives to the C source code of rfifind to parallelize the RFI search (Algorithm 2). The process of writing search results to mask files consumes little time and requires no further optimization.

3.3. Multiprocessing Parallelization with Python

After RFI removal, each subsequent hotspot step has a similar computational pattern: using Python scripts to input multiple independent files and output to independent files. Therefore, they can all use parallel optimization as described below. This paper uses Python' s multiprocessing library to evenly divide all files to be processed into N portions, with each portion handled by a Python process bound to a CPU core (Algorithm 3). Based on the de-dispersion scheme, the parallel processes for acceleration search and single pulse search number in the thousands, while candidate folding typically has about a hundred parallel processes. We detail each step based on parallel processing below.

3.4. Parallelization and Load Balancing in De-dispersion

Using the de-dispersion scheme for low-frequency MWA incoherent summed data shown in Table 1 , we unroll the de-dispersion loop into two nested loops and optimize with OpenMP. The hotspot in the de-dispersion process is the loop that removes dispersion from observed data: first, reading a piece of data without narrowband RFI and generating preprocessed data (GetData in pseudocode), then using data from the previous loop (previous data) and current loop (current data) to remove dispersion (dedisp in pseudocode). Since previous data does not exist during the first run, the current data is used twice in the first loop.

In the de-dispersion loop, data dependencies exist because previous data is needed for the current data' s de-dispersion. To solve this problem, we unroll the loop into two separate loops and parallelize each. The first loop preprocesses data and loads them into arrays, while the second loop reads the preprocessed data and performs de-dispersion operations (Algorithm 5).

However, the strategy of assigning one CPU core per prepsubband command in de-dispersion can lead to load imbalance. As shown in Figure 3 [Figure 3: see original paper], the time to complete prepsubband ranges from 300 to 1500 seconds. CPU cores that finish tasks early must wait for the slowest core to complete before proceeding to the next task. Table 1 lists the differences in runtime caused by variations in individual computational tasks, where Low-DM and High-DM represent the lower and upper limits of the de-dispersion operation, dDM represents the DM step size, Nsteps represents the number of steps to execute, Downsamp represents the downsampling factor, and nsub represents the number of frequency sub-channels. Each prepsubband command executes 100 Nsteps, and there are 28 commands total for de-dispersion. Larger nsub values result in longer command execution times.

To balance prepsubband runtime, we designed a load balancing algorithm for de-dispersion operations using OpenMP to allocate threads proportionally to each task' s workload. Table 2 shows the predicted execution time for commands given nsub and thread count. Based on these predicted times, we generate locally optimal process-thread configurations to balance process loads. The configuration algorithm includes the total number of processes, the commands contained in each process, and the corresponding thread count, where a process contains several commands that execute serially with the same thread count.

The specific steps are as follows (see Algorithm 6): (1) According to the de-dispersion scheme, assuming C commands need to be executed, initialize C processes, each containing one de-dispersion command with thread count set to 1; (2) Based on predicted command execution times, let T be the maximum time consumption among all processes; (3) If the current thread count is less than or equal to the number of CPU cores on the node, find the most time-consuming process and increase its thread count by 1. Otherwise, find the two least time-consuming processes, merge them into one process, and set the thread count to

the larger of the two original processes; (4) Update T and compare with the original value to verify convergence conditions. If convergence is satisfied and the current thread count is greater than or equal to the number of CPU cores on the node, the process-thread configuration algorithm ends and outputs the de-dispersion configuration scheme. Otherwise, return to the previous step.

---

4. Performance Evaluation

We selected four MWA-VCS observations in incoherent mode with a central frequency of 185 MHz as test cases (Table 3). The incoherent mode of MWA-VCS has a large visible sky area of about 600 square degrees, making it very suitable for the large-area shallow census in the early stages of the SMART survey. In 2019, Gong et al. [48] published northern sky pulsars found in the sidelobes of 50 MWA-VCS incoherent data surveys.

We ran search experiments for these four observations on both x86 and ARM compute nodes of CSRC-P (specific configurations in Table 4). The hardware and software configurations of these two node types are basically comparable, making them suitable for comparative analysis of experimental results. We compared runtime, result correctness, and optimization effectiveness of different strategies on x86 and ARM platforms before and after optimization.

4.1. Correctness Verification

The pipeline is designed to find pulsar candidates from time series data. After the search process completes, information about pulsar candidates is stored in candslist.txt, including the dispersion measure and period of each candidate. We found that the optimized search pipeline obtained exactly the same results as the original pipeline regardless of the node type, verifying the correctness of our optimized pulsar search pipeline.

4.2. OpenMP Parallel Performance Analysis

We tested the OpenMP parallel performance of narrowband RFI removal on single x86 and ARM nodes. The test results are shown in Figure 4 [Figure 4: see original paper]. Table 5 shows the speedup S_p and parallel efficiency _p for narrowband RFI removal on different nodes with different data, calculated as S_p = T_1/T_p and _p = T_1/(pT_p), where p is the number of cores on the compute node, T_1 is the time for single-core execution, and T_p is the time for p-core execution.

For different test data, ARM and x86 show similar speedup. Individual ARM nodes typically have more CPU cores than x86 nodes, so narrowband RFI removal runs faster on ARM clusters. However, the speedup ratio between ARM and x86 clusters is not exactly proportional to the core count ratio because

OpenMP multithreading performance is limited by memory bandwidth, causing parallel efficiency on ARM clusters not to scale linearly with CPU core count.

4.3. Multiprocessing Parallel Performance Analysis

Based on the multiprocessing parallel method described earlier, we tested the parallel performance of single pulse search, acceleration search, candidate folding, and de-dispersion using multiprocessing on single x86 and ARM nodes.

Single pulse search used the PRESTO-based Python script single_{pulse}_{search}.py. Performance comparison is shown in Figure 5 [Figure 5: see original paper], with speedup and parallel efficiency in Table 6 . Acceleration search used accelsearch from PRESTO, with harmonic summing limit numharm set to 16 and maximum frequency drift zmax set to 200. Table 7 and Figure 6 [Figure 6: see original paper] show speedup and parallel efficiency for different data on different nodes. Both single pulse search and acceleration search achieve high parallel efficiency because they process thousands of files with no dependencies between file processing, making multiprocessing highly effective.

For candidate folding, we ran prepfold from PRESTO with n set to 128 points after folding. Table 8 and Figure 7 [Figure 7: see original paper] show speedup and parallel efficiency for different data on different nodes. Candidate folding executes multiple prepfold commands with different parameters, and these commands have the same execution time. Observation data case1 requires executing 50 prepfold commands, which runs 28 commands first on the 28-core x86 node, then the remaining 22 commands, leaving idle CPU cores and reducing parallel efficiency. On the 96-core ARM node, 46 CPU cores remain idle, resulting in lower parallel efficiency. Observation data 1091309464 requires executing 88 prepfold commands, where ARM' s parallel efficiency improves due to the increased number of commands. On x86, the increased command count raises the proportion of time when all CPU cores are occupied, also improving parallel efficiency.

Finally, for de-dispersion, we ran prepsubband from PRESTO with parameters set according to the de-dispersion scheme for MWA low-frequency incoherent data. Speedup and parallel efficiency for different data on different nodes are shown in Table 9 and Figure 8 [Figure 8: see original paper]. De-dispersion parallel efficiency is also relatively low because, according to the de-dispersion scheme' s parameter space, there are large runtime differences between prepsubband commands when parallelized on multi-core compute nodes, causing computational resource waste and reduced parallel efficiency. In the next section, we adopt a combined OpenMP+multiprocessing approach on ARM nodes to eliminate load imbalance and fully utilize each node' s 96 cores.

4.4. Load Balancing Optimization

Based on the runtime distribution of de-dispersion commands shown in Figure 2 [Figure 2: see original paper], we can see that commands 1-11 correspond to nsub = 4, commands 12-16 to nsub = 8, commands 17-21 to nsub = 16, commands 22-26 to nsub = 32, and commands 27-28 to nsub = 64. Commands 16, 21, and 26 have Nsteps values of 57, 15, and 35, respectively, resulting in shorter runtimes.

For the x86 single node' s 28 CPU cores, the optimized parallel strategy obtained by the load balancing generation algorithm uses multiprocessing to run 14 processes, each serially executing 1-3 prepsubband commands, with each command using 2 OpenMP threads, thus fully utilizing all 28 CPU cores on the x86 single node and maximizing load balance per process. The correspondence between commands and processes is: (1,2,21), (3,16,26), (4,25), (5,24), (6,23), (7,22), (8,20), (9,19), (10,18), (11,17), (12,15), (13,14). Each parenthesis represents one process, with numbers indicating serially executed command IDs within that process.

Due to pipeline algorithm design limitations, the currently enabled 28 processes cannot fully exploit the 96-core computational capability of ARM nodes. We designed a two-level parallel method combining multiprocessing and OpenMP multithreading, allocating 2-10 OpenMP threads to different processes to fully utilize the 96-core computational capacity. Each process is allocated different OpenMP thread counts based on the nsub value in the executed commands, with the correspondence shown in Table 10 .

Through this load balancing strategy, we compared performance before and after further optimization, with results shown in Figure 9 [Figure 9: see original paper] and Table 11 . The speedup effect is more pronounced on ARM because before load balancing optimization, the ARM single node only used 28 processes for de-dispersion, while after load balancing, each process uses multiple OpenMP threads, fully utilizing all 96 CPU cores. In contrast, x86 already used all 28 CPU cores before and after load balancing, so the improvement is less significant.

The overall speedup for the four test datasets is shown in Figure 10 [Figure 10: see original paper] and Table 12 . Compared to the single-threaded approach, we achieved 10.4-12.2× speedup on the x86 platform and up to 27.6× speedup on the ARM platform. For our test data, the ARM platform runtime was 1.1-1.3× shorter than the x86 platform due to ARM nodes having more computational cores.

---

5. Summary

Based on OpenMP multithreading and multiprocessing parallel methods, we successfully implemented multi-core parallelization for hotspots in the pulsar search pipeline and designed an automatic load balancing generation algorithm. Using

a combined multiprocessing+OpenMP approach, we solved the load imbalance problem in the de-dispersion algorithm, further improving overall parallel efficiency. The optimization methods for multi-core architectures in this paper are applicable to both x86 and ARM platforms, with search tests conducted on four incoherent sum datasets from MWA-VCS at 185 MHz center frequency on both platforms. The conclusions are as follows:

- Compared to the original single-threaded method, runtime on a 28-core x86 single node was reduced by 10.4–12.2× with 36% parallel efficiency; on a 96-core ARM single node, runtime was reduced by up to 27.6× with 31% parallel efficiency.
- Executing the same parallel optimization techniques on x86 and ARM compute nodes demonstrates the portability of optimizations from x86 to ARM platforms. Due to the massive parallelism of pulsar search tasks, the ARM platform benefits from more computational cores per node and is 1.1–1.3× faster than the x86 platform in our tests, showing great potential for ARM nodes in SKA data processing tasks.
- Our optimization algorithm reduces pulsar search time consumption by an order of magnitude, which will effectively advance pulsar research.

Moreover, performance optimization of data processing pipelines not only solves the time-consuming problem of pulsar search pipelines but can also be extended to other application pipeline optimizations, greatly benefiting astronomical research at the China SKA Regional Centre (e.g., [48–55]).

In the coming years, our work will focus on further improving parallel efficiency and vectorizing the PRESTO pipeline, conducting larger-scale experiments using SMART survey beamformed data from MWA-VCS to verify ARM platform suitability. For the upcoming SKA1 data, we will be well-prepared to meet the challenges of pulsar searching, with ARM platforms playing a pivotal role and impact.

References

1. Sturrock, P. A. "A model of pulsars." *The Astrophysical Journal* 164 (1971): 529.
2. Hewish, A., Bell, S. J., Pilkington, J. D. H., Scott, P. F., & Collins, R. A. Observation of a Rapidly Pulsating Radio Source. *Nature*, 1968, 217: 709.
3. Cordes, J. M., Freire, P. C. C., Lorimer, D. R., et al. Arecibo Pulsar Survey Using ALFA. I. Survey Strategy and First Discoveries. *The Astrophysical Journal*, 2006, 637: 446.
4. Keith, M. J., Jameson, A., van Straten, W., et al. The High Time Resolution Universe Pulsar Survey - I. System configuration and initial discoveries. *Monthly Notices of the Royal Astronomical Society*, 2010, 409: 619.

5. Barr, E. D., Champion, D. J., Kramer, M., et al. The Northern High Time Resolution Universe pulsar survey - I. Setup and initial discoveries. *Monthly Notices of the Royal Astronomical Society*, 2013, 435: 2234.
6. Nan, R., Li, D., Jin, C., et al. The Five-Hundred Aperture Spherical Radio Telescope (FAST) Project. *International Journal of Modern Physics D*, 2011, 20: 6.
7. Jiang, P., Tang, N.-Y., Hou, L.-G., et al. The fundamental performance of FAST with 19-beam receiver at L band. *Research in Astronomy and Astrophysics*, 2020, 20: 064.
8. Qian, L., Yao, R., Sun, J., et al. 2020, *The Innovation*, 1, 100053.
9. Cameron, A. D., Li, D., Hobbs, G., et al. An in-depth investigation of 11 pulsars discovered by FAST. *Monthly Notices of the Royal Astronomical Society*, 2020, 495: 3.
10. Qian, L., Pan, Z., Li, D., et al. The first pulsar discovered by FAST. *Science China Physics, Mechanics, and Astronomy*, 2019, 62: 5.
11. Pan, Z., Qian, L., Ma, X., et al. FAST Globular Cluster Pulsar Survey: Twenty-four Pulsars Discovered in 15 Globular Clusters. *The Astrophysical Journal Letters*, 2021, 915: 2.
12. Han, J.-L., Wang, C., Wang, P.-F., et al. The FAST Galactic Plane Pulsar Snapshot survey: I. Project design and pulsar discoveries. *Research in Astronomy and Astrophysics*, 2021, 21: 107.
13. 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). *IEEE Microwave Magazine*, 2018, 19: 3.
14. Li, D., Dickey, J. M. & Liu, S. Preface: Planning the scientific applications of the Five-hundred-meter Aperture Spherical radio Telescope. *Research in Astronomy and Astrophysics*, 2019, 19: 2.
15. Stovall, K., Lynch, R. S., Ransom, S. M., et al. The Green Bank Northern Celestial Cap Pulsar Survey. I. Survey Description, Data Analysis, and Initial Results. *The Astrophysical Journal*, 2014, 791: 67.
16. J. S. Deneva et al., "New discoveries from the Arecibo 327 MHz drift pulsar survey radio transient search" , *The Astrophysical Journal*, 2016, Volume 821, Number 1.
17. http://www.naic.edu/ deneva/drift-search/
18. van Haarlem, M. P., Wise, M. W., Gunst, A. W., et al. LOFAR: The LOw-Frequency ARray. *Astronomy and Astrophysics*, 2013, 556: 2.
19. Sanidas, S., Cooper, S., Bassa, C. G., et al. The LOFAR Tied-Array All-Sky Survey (LOTAAS): Survey overview and initial pulsar discoveries. *Astronomy and Astrophysics*, 2019, 626: A104.
20. Tan, C. M., Bassa, C. G., Cooper, S., et al. The LOFAR Tied-Array all-sky survey: Timing of 21 pulsars including the first binary pulsar discovered with LOFAR. *Monthly Notices of the Royal Astronomical Society*, 2020, 492: 5878.
21. Manchester, R. N., Hobbs, G. B., Teoh, A., & Hobbs, M. The Australia Telescope National Facility Pulsar Catalogue. *The Astronomical Journal*, 2005, 129: 1993.

22. Keane, E., Bhattacharyya, B., Kramer, M., et al. A Cosmic Census of Radio Pulsars with the SKA. *Advancing Astrophysics with the Square Kilometre Array (AASKA14)*, 2015: 40.
23. Qiuyu Yu. (2020) A PRESTO-based parallel pulsar search pipeline and pulsar signal de-noising, https://kns.cnki.net/KCMS/detail/detail.aspx?dbname=CMFD202101&filenam
24. Ransom, S. M. New search techniques for binary pulsars. Ph.D. Thesis, 2001.
25. Sofia Dimoudi, Wesley Armour. Pulsar Acceleration Searches on the GPU for the Square Kilometre Array, https://arxiv.org/abs/1511.07343.
26. Haomiao Wang and Prabu Thiagaraj and Oliver Sinnen. FPGA-based Acceleration of FT Convolution for Pulsar Search Using OpenCL. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2019, 11(4): 1-25.
27. Dewdney, P. E., Hall, P. J., Schilizzi, R. T., & Lazio, T. J. L. W. The Square Kilometre Array. *IEEE Proceedings*, 2009, 97.
28. Tingay, S. J., Goeke, R., Bowman, J. D., et al. The Murchison Widefield Array: The Square Kilometre Array Precursor at Low Radio Frequencies. *Publications of the Astronomical Society of Australia*, 2013, 30: e007.
29. Tremblay, S. E., Ord, S. M., Bhat, N. D. R., et al. The High Time and Frequency Resolution Capabilities of the Murchison Widefield Array. *Publications of the Astronomical Society of Australia*, 2015, 32, 5.
30. Beardsley, A. P., Johnston-Hollitt, M., Trott, C. M., et al. Science with the Murchison Widefield Array: Phase I results and Phase II opportunities. *Publications of the Astronomical Society of Australia*, 2020, 37, 14.
31. An, T. Science opportunities and challenges associated with SKA big data. *Science China Physics, Mechanics, and Astronomy*, 2019, 62: 989531.
32. S. Ratcliffe, F. Graser, B. Carlson, O. B, SKA1 LOW SDP - CSP Interface Control Document, SKA Project Document number 100-000000-002 1 (1).
33. S. Ratcliffe, F. Graser, B. Carlson, O. B, SKA1 MID SDP - CSP Interface Control Document, SKA Project Document number 300-000000-002 1 (1).
34. S. Yamamura et al., "A64FX: 52-Core Processor Designed for the 442PetaFLOPS Supercomputer Fugaku," 2022 IEEE International Solid-State Circuits Conference (ISSCC), 2022, pp. 352-354, doi: 10.1109/ISSCC42614.2022.9731627.
35. Peter Quinn, Michiel van Haarlem, Tao An, et al., "SKA Regional Centres - A White Paper by the SKA Regional Centre Steering Committee" V1.0, 2020.
36. The SKA Regional Centre Steering Committee, SKA Regional Centres White Paper, v1.0, 2020 May.
37. T. An, X. P. Wu, and X. Y. Hong, SKA data take centre stage in China, *Nature Astronomy*, 3, 1030 (2019).
38. T. An, X. C. Wu, B. Q. Lao, et al. Status and progress of China SKA Regional Centre prototype. arxiv:2206.13022.
39. B. Q. Lao, Y. K. Zhang, T. An, et al. Software Platform on China SKA Regional Center Prototype System (in Chinese). ChinaXiv:202206.00173.
40. J. W. Wei, C. F. Zhang, B. Q. Lao, et al. Optimization of parallel process-

ing of Square Kilometre Array low frequency imaging pipeline (in Chinese). ChinaXiv:T202206.00292.
41. An, T., Wu, X. C., Lao, B. Q., et al. Status and progress of China SKA Regional Centre prototype. arxiv:2206.13022.
42. ARM architecture, URL https://en.wikipedia.org/wiki/ARM_{architecture}, 2021.
43. Bhat, N. D. R., Swainston, N. A., McSweeney, S. J., et al. The Southern-sky MWA Rapid Two-metre (SMART) pulsar survey: System overview, pulsar census, and first pulsar discoveries, *Publications of the Astronomical Society of Australia*, 2022, in preparation.
44. Ransom, S. M. PRESTO Home, https://www.cv.nrao.edu/ sransom/presto/.
45. Multiprocessing library, Python Software Foundation, https://docs.python.org/2/library/multiprocessing
46. Chandra, Rohit, et al. Parallel programming in OpenMP. Morgan Kaufmann, 2001.
47. Tabirca, Tatiana, et al. "A static workload balance scheduling algorithm." *Proceedings. International Conference on Parallel Processing Workshop.* IEEE, 2002.
48. Gong, H., Zhang, Z., Xue, M., et al. Search and detection of northern pulsars in the side lobes of the murchison wide-field array. *Scientia Sinica Physica, Mechanica & Astronomica*, 2020, 50: 109501.
49. Lao, B. Q., An, T., Yu, A., & Guo, S. G. "Research on Parallel Algorithms for uv-faceting Imaging" . *Acta Astronomica Sinica*, 2019, 60: 12.
50. Lao, B., An, T., Yu, A., et al. "Parallel implementation of w-projection wide-field imaging" . *Science Bulletin*, 2019, 64.
51. Lao, B., An, T., Wang, A., et al. "Artificial intelligence for celestial object census: the latest technology meets the oldest science" . arXiv e-prints, 2021: arXiv:2107.03082.
52. An, T., Mohan, P., Zhang, Y., et al. Evolving parsec-scale radio structure in the most distant blazar known. *Nature Communications*, 2020, 11: 143.
53. Gu, J., & Wang, J. Direct parameter inference from global EoR signal with Bayesian statistics. *Monthly Notices of the Royal Astronomical Society*, 2020, 492: 4080.
54. Wang, R., Wicenec, A., & An, T. SKA shakes hands with Summit. *Science Bulletin*, 2020, 65: 337.
55. Wang, Y., Tuntsov, A., Murphy, T., et al. ASKAP observations of multiple rapid scintillators reveal a degrees-long plasma filament. *Monthly Notices of the Royal Astronomical Society*, 2021, in press.

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