# HDF5-Based I/O Optimization for Extragalactic HI Data Pipeline of FAST

Yiming Ji[1], Ce Yu[1], Jian Xiao[1](✉), Shanjiang Tang[1], Hao Wang[1], and Bo Zhang[2]

[1] College of Intelligence and Computing, Tianjin University, Tianjin, China
{jiym,yuce,xiaojian,tashj,imwh}@tju.edu.cn

[2] CAS Key Laboratory of FAST, NAOC, Chinese Academy of Sciences, Beijing, China
zhangbo@nao.cas.cn

**Abstract.** The Five-hundred-meter Aperture Spherical Radio Telescope (FAST), which is the largest single-dish radio telescope in the world, has been producing a very large data volume with high speed. So it requires a high performance data pipeline to covert the huge raw observed data to science data product. However, the existing solutions of pipelines widely used in radio data processing cannot tackle this situation efficiently. The paper proposes a pipeline architecture for FAST based on HDF5 format and several I/O optimization strategies. First, we design the workflow engine driving the various tasks efficiently in the pipeline; second, we design a common radio data storage specification on the top of HDF5 format, and also developed a fast converter to map the original FITS format to the new HDF5 format; third, we apply several concrete strategies to optimize the I/O operations, including chunks storage, parallel reading/writing, on-demand dump, and stream process etc. In the experiment of processing 700 GB of FAST data, the results show that HDF5 based data structure without other optimizations was 1.7 times faster than original FITS format. If chunk storage and parallel I/O optimization are applied, the overall performance can reach 4.5 times as the original one. Moreover, due to the good expansibility and flexibility, our solution of FAST pipeline can be adapted to other radio telescopes.

**Keywords:** FAST · FITS · HDF5 · Pipeline in parallel · High performance I/O

## 1 Introduction

Data pipeline, a key procedure for modern observational astronomy, is to convert the raw observed data to science product, which astronomers can use directly to make new discoveries and theoretical testing. As the continuous improvement of capacity and resolution of telescopes, the observed data volume explosively increases. So the performance of astronomical data pipeline becomes one of the

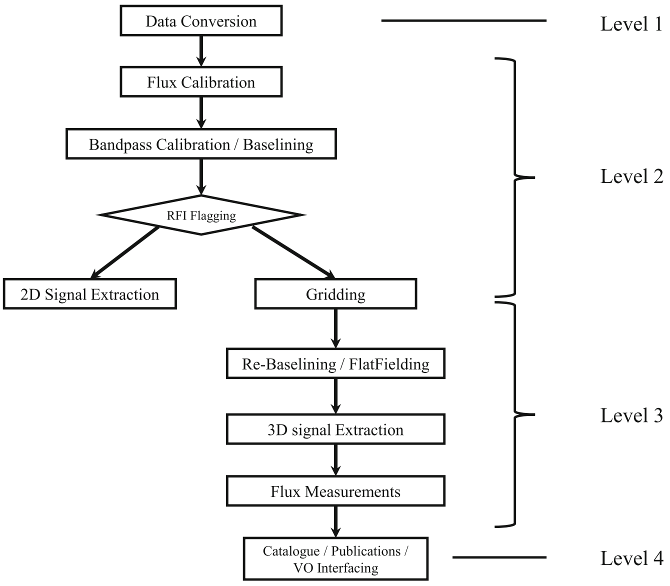


**Fig. 1.** The main steps of pipeline

biggest challenges for modern large telescopes. For example, the Five-hundred-meter Aperture Spherical Radio Telescope (FAST) [15] in Guizhou, China, is the largest single-dish radio telescope in the world. Currently, typical data rate of FAST is as high as 3 GB/s. FAST will output about 10 TB of raw data per hour. Its tasks include neutral hydrogen survey and pulsar detection, which will record a huge amount of data. Meanwhile, the 19-beam receiver, the most frequently used receiving system of FAST [23], will produce 19 times data size as the single beam receiver. These observed data provides more opportunities to new scientific discoveries, meanwhile it also brings unprecedented pressure to the traditional solution of data pipeline.

Figure 1 shows a classical work flow of radio data pipeline. It is usually divided into four levels. Level 1 is the data format conversion, converting the original format to an inner format throughout the whole pipeline. Level 2 is the necessary calibrations to eliminate the noise from the universe, the earth and the device itself to a tolerable level, including flux calibration, bandpass correction, baseline subtraction, and radio frequency interference (RFI) mitigation [8] etc. The main purpose of level 3 is to generate the data cube for extracting signals [18]. Level 4 refers to publish the data product to astronomers for further research. For FAST, the process flow is almost the same as Fig. 1. In terms of the overall architecture of pipeline, data exchange between different steps is the efficiency bottleneck to process data.

On the other hand, unlike optical telescopes, the radio telescope have a very wide frequency coverage, so each radio telescope has its unique physical features, though they can share a relative common process flow, but the details of each steps may be quite different. Therefore, there is none existing universal solution or mature software for various radio telescopes. For example, CASA [14] aims to adapt all radio telescopes, but is used mainly by VLA and ALMA until now, which both are radio array or similar structure instead of single dish. CLASS [21] is designed for (sub-)millimeter single dishes. CLASS does not support parallel processing because it only serves small and medium telescopes. Therefore, it is not suitable for processing tasks with large amounts of data.

There are several research works aiming to improve the data pipeline efficiency. Data Activated Liu Graph Engine (DALiuGE) is an execution framework for processing large astronomical datasets at a scale required by the Square Kilometre Array Phase 1 (SKA1) [27]. It can be considered as one representation of next generation pipeline architectures, but is more like a task deployment and scheduling tool, lack of consideration of data layout and exchange. In addition, AST3 daemon [10,28] is a light-weight pipeline engine for Antarctic Schmidt Telescopes, and it has been running in Antarctic dome A for 5 years normally. While AST3 also lacks of support for cluster environment, which is necessary for FAST.

At present, most of existing solutions still use FITS file as the default data format during the data processing [16,26]. FITS is a traditional astronomical data format with fixed format specifications and complex content packaging. According to our experiment, FITS related I/O operations take roughly 25%–30% of the whole process. Obviously the I/O efficiency has become one of bottlenecks of the modern radio data pipelines. Faced with rapidly increasing volume of data, the FITS performance seems to be inadequate. Motivated by data volumes, Hierarchical Data Format version 5 (HDF5) [9] has been implemented for the LOFAR radio telescope [1], the CCAT telescope [22], and the CHIME pathfinder [13]. Compared with FITS format, HDF5 format has more concise structure. So it's possible to improve I/O efficiency by changing the layout of data in HDF5 format. In addition, the pipeline has separate processing steps and the intermediate results are saved frequently. The large volume of FAST's data makes I/O cost become a vital factor impacting the overall performance. Therefore, reducing I/O time can improve efficiency significantly.

So in order to process FAST's continuous huge output data with high speed and accuracy, an end-to-end solution including parallel pipeline engine, fast inner data exchange format, high speed I/O interface, and deeply optimized algorithm of each steps is necessary. In this paper, we focus on the common aspects of building the qualified pipeline for FAST, that is providing an optimized pipeline framework with high performance I/O support. Various calibration algorithms, RFI mitigation and data cube generation modules can be easily integrated into the pipeline through the interface, and all of them can exchange data with high efficiency within the pipeline. The work content of this paper is described as follows. We propose the data layout specification for pipeline on the top of HDF5 format, and also develop a fast converter to map the original FITS format to the new HDF5 format. In addition, we apply several concrete strategies to optimize the I/O operations, including chunks storage, parallel reading/writing, on-demand dump, and stream process etc. Furthermore, we implement the FAST pipeline engine, which supports concurrency, real-time monitoring and error reporting, in-memory execution, and asynchronous storage of resulting data. In the experiment of processing 700 GB of FAST data, the results show that merely HDF5-based data structure is 1.7 times faster than original FITS format. With chunk storage and parallel I/O optimization, the overall performance can be 4.5 times as the original one. The key contributions are:

– We design a parallel data pipeline engine to tackle FAST huge data volume, and it can efficiently drive the pipeline to process large intermediate result in stream style.
– We define a general data layout for single dish radio data based on HDF5, which is used for fast data exchange between various tasks of pipeline.
– We explore several optimization strategies such as chunks storage, parallel I/O, etc. We also integrate main processing steps into the pipeline, and make a comprehensive evaluation based on real observed data.

This article is organized as follows. In Sect. 2, we provide some related work, including the characteristics of FITS and HDF5 file formats, the performance of some radio telescope pipelines, and I/O method of pipelines. In Sect. 3, we present a description of our overall work; we also present a mapping of FAST's raw data (FITS) into intermediate data (HDF5) and the details of the pipeline. Section 4 gives the experimental results to verify the efficiency of our proposed format and the concrete implementation of pipeline. Some general summaries, practical applications for FAST and possible future extensions are given in Sect. 5.

## 2 Related Work

In this section, we discuss related work from three aspects: the format of astronomical data, the pipeline of radio telescopes and file I/O.

### 2.1 Data Format

**FITS Model.** The Flexible Image Transport System (FITS) has enjoyed several decades of usage among the field of Astronomy [16], and it was stipulated

**Table 1.** Example of FITS HDU

| Key = | Value/comment |
|---|---|
| SIMPLE = | T/File does confirm to FITS standard |
| BITPIX = | 16/ Number of bits per data pixel |
| NAXIS = | 2/Number of data axes |
| NAXIS1 = | 320/Number of pixels along the fastest changing axis |
| NAXIS2 = | 512/The number of pixels along the sub-fast changing axis |
| END | |

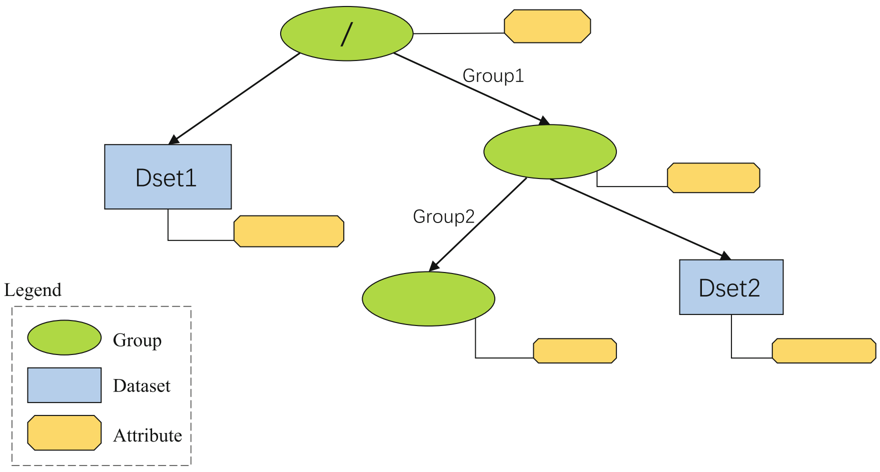


**Fig. 2.** Example of HDF5 object [17]

as the unified standard format for data transmission and exchange between different observatories established by the International Astronomical Union (IAU) in 1982.

FITS file consists of a set of header data units (HDUs), which are ASCII headers followed by consecutive blocks of data (binary or ASCII encoding). HDUs contain some descriptive variables, whose format is "Key = Value/Comment". There are some indispensable keywords listed in Table 1, such as SIMPLE, BITPIX, NAXIS, NAXISn and END. In general, there are other keywords indicating related information of observation data, such as date, telescope, observer and so on. In astronomy, FITS is a classical format for pictorial data and spectral data.

**HDF5 Model.** The hierarchical data format 5 (HDF5) is the latest among the series. HDF5 consists of data format specification and library implementation. Compared with the old version of HDF, Its related support has been extended. In addition, its hierarchical structure and supported libraries can reflect the advantages of storage, reading and writing [7].

HDF5 format file is organized in a hierarchical structure, which contains three main elements: Dataset, Group and Attribute.

- Dataset: Multidimensional arrays of data elements and support for metadata.
- Group: The grouping structure contains instances of zero or more objects, groups or datasets; it's supported for metadata.
- Attribute: User-defined metadata information, which can be attached to Datasets and Groups.

The Group is the root of HDF5 object. A separate HDF5 object can perform as its substructure. The dataset mainly stores data array, whose dimensions can be set from 1 to N (any positive integer). At the same time, the attributes can be linked to different nodes of HDF5 object. Attributes annotate temperature, time and other information defined by users. Figure 2 shows hierarchical structure of HDF5 object and relation of Groups, Datasets and Attributes.

### 2.2 Pipelines Introduction

Pipeline is one kind of dataflow computation model, which was initially proposed [19] to express programs as Directed Acyclic Graphs (DAG), where the vertices are the stateless computational tasks that compose the program, and edges connect the output of one task with the input of another. In astronomy, the pipeline implements the data processing algorithm in the order of dataflow. Each step of the pipeline is independent. The output of the previous step will flow into the next step.

At present, many telescopes all over the world have their own data processing pipelines. Davis [5,6] proposed the ALMA prototype science pipeline in 2004. As of 2014, it had already supported distributed processing. The shooting speed of ALMA determines the pipeline speed to be 6M/s, which is not suitable for massive data processing of FAST.

The Transients Key Science Project (TKP) [24] is developed for LOFAR and can study all variable sources detected by LOFAR. Its functions include the study of transient and variable low-frequency radio sources with an extremely broad science case ranging from relativistic jet sources to pulsars, exoplanets, flare stars, radio bursts at cosmological distances, the identification of gravitational wave sources and even SETI. As Lofar and FAST have different research goals, TKP's approach cannot be transplanted into FAST's supporting environment.

The Very Large Telescope (VLT) [3] is a collection of eleven instruments. For each of them, European Southern Observatory (ESO) provides automatic data reduction facilities in the form of instrument pipelines developed in collaboration with the instrument consortia. The Multi Unit Spectroscopic Explorer (MUSE, Bacon et al. [2]) is one of four second generation instruments being built for the ESO VLT. MUSE can process 150 GB of raw data per night and support two modes, online and offline. Online mode focuses on timeliness, while offline mode tries to optimize results and minimize user interaction.

### 2.3 File I/O

At present, the data file performs as the I/O unit of FAST pipeline. To optimize the I/O of FAST pipeline, this paper provides several studies that concentrate on optimizing the parallel file-I/O performance of HPC applications. Y.Chen and R.Thakur proposed libraries and parallel file systems respectively, and exploit advancements in storage technologies [4,25]. This system has some constraints on files. It require data to be stored in its own file system. The method is not suitable for FAST pipeline because specific file systems or libraries are not suitable for FAST neutral hydrogen data. Other optimization techniques include exploiting access patterns to assist file system prefetching, data sieving, and caching [12], overlapping computation with I/O [11], and employing asynchronous prefetching [20]. FAST pipeline is a fixed process that does the same execution to each file and the same file is not read repeatedly. Thus data sieving, caching, overlapping computation and so on cannot provide FAST pipeline with improvement.

From preceding part of this section, both of the formats are used in the astronomy field. By contrast, the simplicity and flexibility of the HDF5 format show an advantage. With HDF5 being used as intermediate data format to replace FITS in the pipeline, I/O overhead will be reduced. As a result, overall efficiency is improved. In addition, concurrent processing of pipeline provides favor to process massive FAST data. By the analysis of the existing work, no method can fully satisfy the performance demand of FAST data processing. Based on the technical environment, the paper proposes the implementation of workflow engine and optimization strategy for FAST pipeline.

## 3 HDF5-Based I/O Optimization in Pipeline

The Fig. 3 shows the architecture of the FAST data pipeline with HDF5-based I/O optimization. The whole solution includes a work flow engine to manage the execution of pipeline, a highly optimized I/O interface to read/write intermediate data during the process, the HDF5 data model of FAST, and a dedicated mapper for converting raw data from FITS to HDF5 format.

The data produced by the FAST is recorded in FITS files. Since the FITS format is a multi-layer encapsulated table structure, the process of parsing FITS files takes much time. HDF5 format is chosen as the alternative to reduce the overhead of parsing FITS. In order to maximize compatibility with existing astronomical software and share data, we keep the archived raw files in FITS format and use HDF5 format as an intermediate format to improve I/O performance. HDF5 has two distinct characteristics: hierarchical structure and attributes. Hierarchical structure highlights the hierarchy and ownership among different parts. In our proposed HDF5 specification, the data is grouped according to the scanning sequence or polarization number, which makes it easier for programmers and astronomers to clarify the data content. It's straightforward to determine the location of the target data according to the hierarchy. In addition, the attributes is added to the specified parts of the HDF5 object. This combination determines its self-explanatory advantages for each grouping pair. By adding attributes, it's visual to specify the content of the object. There are some descriptive variables like time, shooting equipment, shooting status and so on, which need to be recorded as float or string variables. It's unnecessary to create new datasets. Attributes work in this situation, which indicate these variables in the corresponding location. In this way, the efficiency has been saved and the relationship has been described clearly. Compared with the traditional format FITS, our proposed data layout based on HDF5 performs better structurally.

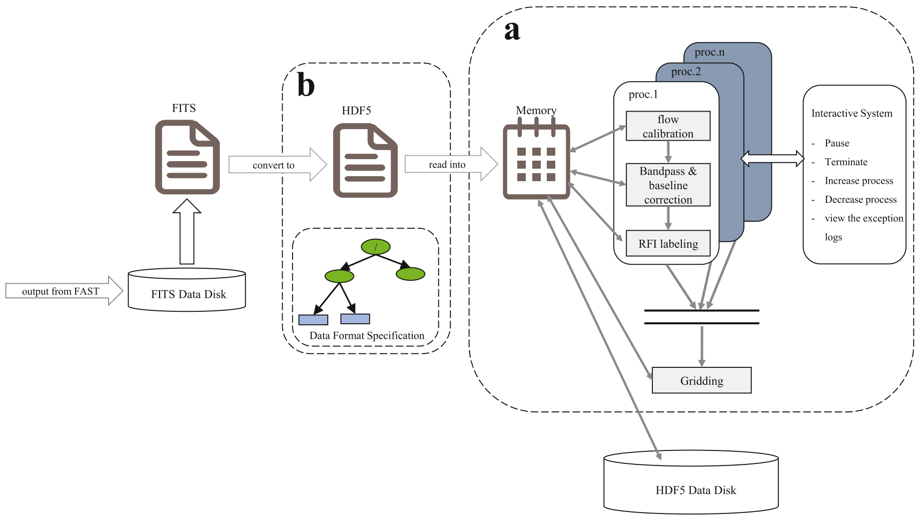


**Fig. 3.** An overview of the design, divided into two parts – **a** and **b**

## 3.1 Transformation and Mapping from FITS to HDF5

For all the content contained in the FITS file, we design the transformation and mapping from FITS to HDF5, showed in Fig. 4. In the raw FITS file, PrimaryHDU stores the descriptive information of FAST data, and the observed content are independently stored in binTableHDUs. The size of binTableHDU is 2048(rows) × 21(columns). Each row is the scanning serial number according to time. The first 20 columns record the time, coordinates, external conditions, etc. The 21st column named "DATA" is a two-dimensional array (65536 × 4). 65536 is the number of channels, and 4 is the polarization serial number. As shown in the Fig. 4, the entire HDF5 object contains only one group as its root directory. The original FITS file's PrimaryHDU is transformed into a set of attributes directly connected to the root directory, which includes BITPIX, NAXIS, EXTEND, ORIGIN, DATE. Then 2048 datasets are corresponding to the "DATA" columns of FITS binTableHDUs. Each binTableHDU's first 20 columns are turned into a set of attributes for each dataset. In the intermediate data, attributes contain the following information: frequency of observation center, bandwidth, number of spectral channels, start/end time of data recording in the file, Angle of the telescope (azimuth Angle and zenith Angle) after the correction of heliocentric system, and information about the definition standard of coordinate reference system.

## 3.2 HDF5 Optimization Strategies

Outside of the transformation from FITS to HDF5, there are some improvement on the underlying field. HDF5 data is stored linearly in memory by default, so

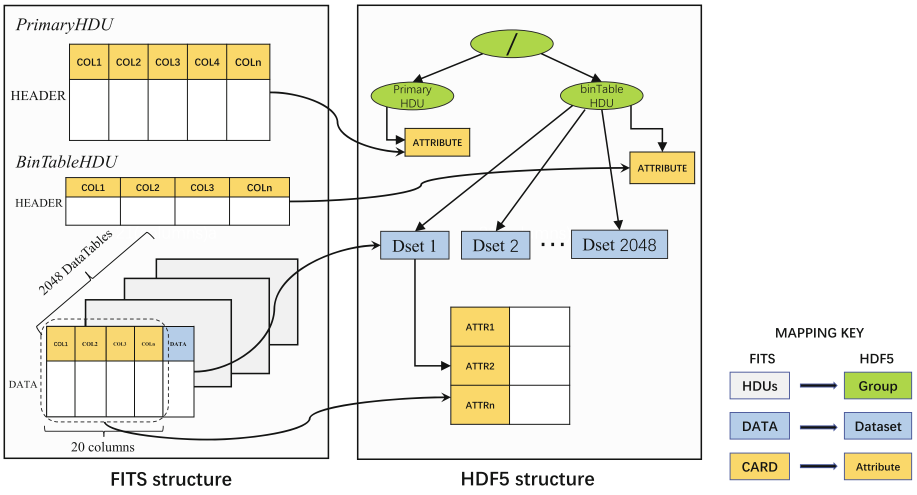


**Fig. 4.** The details of mapping FITS to HDF5

the reading process has to go through all the content to find the target. If a smaller sub-block is the target, accessing parts outside the block is invalid. In order to avoid these invalid operations, the high frequency accessed areas are supposed to be obtained directly.

In the process of converting data from FITS to HDF5, we specify the dataset as a two-dimensional float array. Each dataset in HDF5 data uses a type system similar to the Numpy module in Python. At the time of reading and storing, the concrete array is processed as the Numpy array, and the datatype has been mapped to the dtype of Numpy.

**Chunks Storage.** A two-dimensional array has two dimensions in the mathematical sense, but virtually all the dimensional data in the computer's memory is stored in a linear continuum. Sequential storage is suitable for reading all data at one time. However, when we only need to read one or more blocks in many cases, such as reading the sub-dataset [2048 : 8192, 0 : 2] from an array of 65536 $\times$ 4, the program will read from beginning to end under sequential storage. There will be a lot of overhead outside the target region. It indicates that sequential storage does not match most sub-block reading directly. By default, the HDF5 datasets are sequentially stored, which causes unnecessary I/O cost for pipeline.

For the datasets, the N-dimensional shape is specified in the chunks storge, which fits the access mode best. When data needs to be written into disk, it will be split into blocks of the specified shape and written into memory in blocks. These blocks are stored in the file whose coordinates are indexed by a B-tree. Because pipeline involves a large number of array readings, and the size of sub-block is determined by the algorithm in the pipeline. The pipeline proposed in this paper supports both the default size and manually specifying the size.

**In-memory Cache.** Pipeline involves many times data reading and writing. If each reading or writing is directly sent to disk, the overall running time will be spent primarily on reading and writing rather than computing. To overcome this disadvantage, the data file is kept in the memory until it is processed completely. By this strategy, the pipeline can fetch target data from memory and every step can exchange the data directly. As an HDF5 object is generated, new space in memory will be created to maintain it. As the in-memory file is closed, its contents are saved to disk. As long as the entire file is put into memory, the pipeline process only needs to read and write to the disk once per process. Subsequent data reading and writing, attributes creation, and other operations needn't occupy disk I/O at all.

**Data Operation in Parallel.** According to the above data specification, each file contains 2048 datasets, and each dataset is an array of 65536 $\times$ 4. Parallel I/O for data of this size is a great way to increase efficiency. Common parallel operations are multi-threaded and multi-process. In our early exploration experiments, thread-level concurrency for HDF5 object takes a lot of technical development. This will shift the focus of our research and it's uncertain whether we can meet our expectations by this way. And using HDF5 in multi-threaded programs does not improve efficiency. If multi-process is used directly to manipulate a single HDF5 file, it is easy to conflict with the process of the pipeline hierarchy and the structure appears redundant. Considering the above points, Message Passing Interface (MPI) is a superior choice. In the MPI program environment, one HDF5 object can be accessed by multiple processes. This method supports frequent communication between processes and collection of final results. The process created for each HDF5 file is coordinated by the MPI library and does not conflict with the upper process. In the process, we specify the MPI driver and an MPI communicator. The MPI communicator is responsible for communication between different processes. For example, the process keeping a single data file creates 4 sub-processes. Each sub-process is responsible for calculating one chunk of data, so that four sub-blocks are computed simultaneously. In theory, the performance of this scheme is four times better than that of the serial scheme with the same computing power.

## 3.3 The Implementation of FAST Pipeline

Our work is based on the real neutral hydrogen data of FAST. FAST pipeline in parallel is implemented based on the proposed optimization details above. The main function of the pipeline is to process FAST raw data and archive processed data. The sub-systems of the pipeline can be described as follows:

- Data processing system: the pipeline starts a specified number of processes, each responsible for one dataflow.
- Data I/O system: the pipeline's data read-write system is responsible for reading data files into memory and writing them to disk. When the raw data

files are read for the first time, they will be converted to the designed HDF5 specification.
- Fault-tolerant system: the pipeline's fault-tolerant system logs all exceptions occurring in the middle of the process and re-executes from this step.
- Interactive system: the pipeline's interactive system displays the current total amount of unprocessed tasks in real time, and the current task progress percentage of each process.

As showed in part **a** of Fig. 3, the result of FAST surveying is stored in FITS files disk in real time. When the pipeline starts, it creates a certain number of processes to monitor FITS files disk. If the processes are more than the files, the existing files enter the corresponding number of processes, and other processes are idle. Then the newly generated files enter the waiting processes. If the processes are less than the files, all the processes start working and some files enter the task queue. Processes in working mode are locked. The only access to the processes is the exception handling and enforcement commands issued by the interacting system. When the task is finished, the process will be unlocked and load the next task.

In each independent process, flow calibration, bandpass correction, baseline correction and RFI labeling are conducted in turn. When processed data files reach to a certain amount, Gridding is executed. The input to the first step is an in-memory HDF5 data file, and the input to the next step is the output from the previous step. The pipeline's data stream is executed entirely in memory. Typically, each data file requires only one time disk reading into memory, and I/O of subsequent steps are memory-based. Since the size of a single HDF5 file is 2 GB, the total memory consumption of the pipeline is 2n GB when the number of processes is n. This trade-off for time efficiency at the expense of physical memory consumption is reasonable in large scientific projects. According to the requirements of FAST staff, the results of the intermediate steps should be backed up and saved. The size and dimensions of the stored data files are the same as the proposed HDF5 specification. After each step in the process ends, saving the intermediate results and processing the next step data are executed in parallel.

If an exception occurs in the pipeline, the exception information will be written into the log file. And the pipeline will be re-executed from the appropriate step. Supposing the same exception occurs three times in a row, the task of the data file will be cleared from the process. Then it loads a new file from the task queue.

During the execution of the pipeline, the interactive system can display the total number of tasks in the queue and the schedule of all processes in real time. At the same time, interactive systems support command operations, such as: pause, terminate, increase or decrease the process and view the exception logs.

### 3.4 Functional Module of the Pipeline

To be used for research, the raw files need to be processed. The pipeline implements four main FAST data processing algorithms, which are introduced in the form of modules.

**Spectral Data Flow Calibration.** The raw data recorded by FAST is presented in the form of mechanical records, which does not have direct physical significance. The purpose of flow calibration is to correspond mechanical records to physical units so that one can measure the flow of celestial sources. There are two steps among the transformation: first, the mechanical records are converted into the source's bright temperature $T_{source}$ (K), and then the bright temperature is converted into the source's physical flow (Jy) through the gain coefficient.

**Bandpass and Baseline Correction.** The ultimate goal of spectral observation is to extract spectral signals from celestial bodies and to measure the physical information of spectral lines. After the calibration mentioned previously, the observed data have been corresponded with the actual flow density of the celestial body. But it have still not met the demand of spectral data observation, which can be explained by two reasons. Firstly, even for ON/OFF observations in tracking mode, the calibrated data will inevitably contain the information of continuous spectrum radiation in the sky; secondly, the frequency response and time evolution properties of the device itself need to be considered. We can introduce two vital concepts, bandpass and baseline. Bandpass indicates the frequency response of the observation instruments. Baseline is the influence of background continuous spectrum. Correcting baseline and bandpass is to minimize the impact of these two factors.

**RFI Labeling.** Radio astronomical observation faces a huge challenge from the ubiquitous radio frequency transmission in modern society, such as radio, mobile communications, satellite signal raking, navigation, military/civilian, various daily electronic equipment and so on. These radio frequency radiation levels are often much stronger than the celestial signals measured. Therefore, in order to map the large-scale sky survey data accurately in the later stage, screen and mark the radio frequency interference is of great importance during the preprocessing period.

**Gridding.** As a data product obtained from sky survey, the three-dimensional data cube (three dimensions are the right ascension, declination, and the frequency or spectral line velocity) for final analysis needs to have uniform spacing between the right longitude and the right latitude. Radio data processing process usually requires Gridding to convert raw data from irregular sampling space to regular grid space with uniform spacing, so as to carry out scientific research using scanning data. In the process of FAST data pre-processing, it often needs to accumulate multiple scans of data before Gridding. When the data can cover a large area of sky, the Gridding will be carried out.

## 4 I/O Performance Evaluation

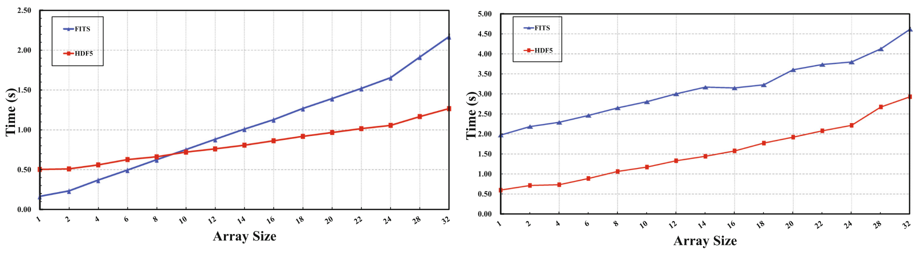
The experimental data was generated during the trial operation of FAST sky survey on September 18, 2018. The total volume of data is 700 GB, and a single data file is 2 GB. The experimental environment is a 4-core 8 GB Tencent Cloud Server.

### 4.1 Efficient Data Format Conversion

The first step of our approach is transforming FAST raw data from FITS format to HDF5 format. The size of a single Fast raw data file is 2 GB, and the converted HDF5 file is 2 GB. The conversion time is 2.91 s, which accounts for a small proportion in the whole pipeline. Thus the conversion step is assumed to have no effect on overall efficiency.

### 4.2 FITS and HDF5 Reading-Writing Comparisons

This paper provides the results for the read-write performance with native FITS and HDF5. FITS file contains 2048 arrays (65536, 4). The corresponding HDF5 file contains the 2048 datasets to store these arrays. Figure 5a shows the performance of reading sub-blocks from it.



(a) Reading arrays of different sizes from the FITS and HDF5 files directly (b) Writing arrays of different sizes into the FITS and HDF5 files directly

**Fig. 5.** FITS and HDF5 read-write comparisons without other optimizations

The horizontal axis denotes the size factor of the selected block and the vertical axis denotes the time. The size factor is n, which denotes the actual size of the array is (2048 × n, 4). As shown in Fig. 5a, when the size of the read block is small(n < 9), the speed of FITS-based method is faster than that of HDF5; when n = 9, the reading performance of both them is equal; when the block is larger (n > 9), the HDF5-based method is faster. When the whole arrays (65536 × 4) are read, the speed of HDF5-based method is 1.7 times as that of FITS. HDF5 performs better than FITS in large arrays reading. FITS format encapsulates data in a more complex way. The tree structure of HDF5 format is beneficial to the retrieval of intersection components.

Figure 5b shows the performance of writing 2048 different size arrays into FITS and HDF5. The sizes of new arrays are (2048 × n, 4). The horizontal axis denotes n and the vertical axis denotes time. As shown in Fig. 5a, HDF5-based method consistently performs better than FITS-based when writing new arrays, whose size is from (2048 × 4) to (65536 × 4). When the size is (65536 × 4), the speed of HDF5-based method is 1.7 times as that of FITS.

### 4.3 HDF5 Performance with Different Chunks

Based on the data layout we specified, this paper studies the performance of different chunks storage schemes with the file driver of HDF5. Figure 6 shows the test results of reading different size of blocks under four chunks schemes. The four schemes are respectively chunks-free, automatic chunks (chunks = TRUE), chunks = (4096, 2), and chunks = (32768, 2). These four schemes set the size of chunks of dataset in memory. It has been proved that in this experiment, the scheme of automatic chunks splits the dataset into chunks of size (4096, 1).

When chunks of the dataset are not specified (the line marked with triangles in the figure), the time of reading an array of (x, 4) is less than that of (x, 1), (x, 2) or (x, 3). This is because the default sequential storage is on the basis of row order. When chunks = True, the system automatically sets the chunks of dataset to (4096, 1). Reading sub-blocks of (4096,1), (8192, 1), (32768, 1) and (65536, 1) performs better than others. When chunks = (4096, 2), reading sub-blocks of (4096, 2), (8192, 2) performs better than others. When chunks = (32768,2), reading sub-blocks of (32768, 2) and (65536, 2) performs better than others. Where chunks = (4096, 2) goes the same way as chunks = (32768, 2), because (32768, 2) is an integer multiple of (4096, 2). When chunks are set up largely, the performance of reading small sub-blocks will be sacrificed because it needs to shred the whole chunk of storage. The scheme of chunks = (32768, 2) performs worst in reading sub-blocks smaller than (32768, 1). The four columns of FAST data respectively represent four polarizations. The relation between the polarizations in calculation is lower than that of the rows, so the connection between columns is not the main concern. Combining the curves of each scheme, we choose the schemes of chunks = True as the best strategy.

### 4.4 HDF5 Performance with MPI

Selecting the most suitable chunks scheme, we apply it to three main algorithms in pipeline for evaluation. The three algorithms are used for flow calibration, bandpass correction and RFI marking. The RFI marking uses the classic algorithm SumThreshold. Figure 7 shows the comparison results in FITS format, HDF5 format, and MPI-based HDF5. It illustrates performance gaps among the FITS-based method, HDF5-based method and parallel hdf5-based method.

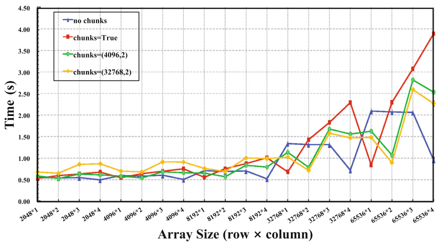


**Fig. 6.** HDF5 reading performance for different chunks storage strategies

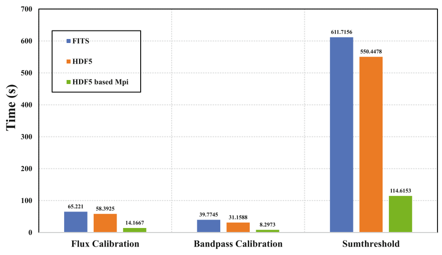


**Fig. 7.** Performance comparisons of different strategies applied to the three algorithms

In three modes, three algorithms are implemented respectively to record the time. The time of HDF5-based is slightly less than that of FITS in three algorithms, but the extent of the improvement is not obvious. There was a significant improvement with a data file is operated in parallel based on MPI. In the experiment, there are 4 processes working simultaneously (all four cores of the CPU in the experimental environment are fully operational). In flux calibration, the speed of HDF5-based with MPI driver is 4.60 times faster than that of FITS-based; in bandpass calibration, the speed of HDF5-based with MPI driver is 4.79 times faster than that of FITS-based; in Sumthreshold, the speed of HDF5-based with MPI driver is 5.34 times faster than that of FITS-based. As experimental environment configuration is limited, this paper only implements four processes to work simultaneously. While the field environment will achieve a larger amount of calculation. This paper shows that the performance has been improved in existing experimental environment.

## 5 Conclusion and Future Work

In view of the problem that FITS format cannot meet the performance requirements of FAST neutral hydrogen data pipeline, this paper proposes the method of using HDF5 as an intermediate format to optimize data I/O. We propose the efficient file format conversion scheme, in which we use the dataset of HDF5 to store the data in FITS's binTables, and use the attributes of HDF5 to store FITS variables in form of $\langle key, value \rangle$. In addition, we implement the workflow engine for extragalactic HI data pipeline of FAST. HDF5 format performs better than FITS in the pipeline. Without any drivers, HDF5's performance is 2.1 and 2.5 times as that of FITS in reading and writing respectively. Furthermore, we improve the performance of HDF5 through chunks storage and MPI driver, and improve the performance of flux calibration, bandpass correction and RFI marking by 4 times, 5 times and 6 times respectively.

In addition, the strategies of converting FITS to HDF5 may also work well with other telescope pipelines to improve I/O performance. At the same time, the MPI driver of our approach can support the distributed system conveniently.

**Acknowledgement.** This work is supported by the Joint Research Fund in Astronomy (U1731125, U1731243, 11903056) under cooperative agreement between the National Natural Science Foundation of China (NSFC) and Chinese Academy of Sciences (CAS), the National Natural Science Foundation of China (11573019). BZ is supported by Open Project Program of the Key Laboratory of FAST, NAOC.

## References

1. Anderson, K., Alexov, A., Baehren, L., Griessmeier, J.M., Renting, A.: LOFAR and HDF5: toward a new radio data standard. Astron. Data Anal. Softw. Syst. XX **442**, 53–56 (2010)
2. Bacon, R., et al.: The second-generation VLT instrument muse: science drivers and instrument design. In: Proceedings of SPIE - The International Society for Optical Engineering, pp. 1145–1149 (2004)
3. Ballester, P., et al.: Data reduction pipelines for the very large telescope. Proc. SPIE - Int. Soc. Opt. Eng. **22**(2), 85–98 (2006)
4. Chen, Y., Winslett, M., Yong, C., Kuo, S.W.: Automatic parallel I/O performance optimization in Panda. In: Proceedings of Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 108–118 (1998)
5. Davis, L.E.: An overview of the ALMA pipeline system. In: Astronomical Data Analysis Software and Systems XVIII ASP Conference Series, vol. 411, p. 306 (2009)
6. Davis, L.E., Glendenning, B.E., Tody, D.: The ALMA prototype science pipeline. Astron. Data Anal. Softw. Syst. XIII **314**, 89 (2004)
7. Folk, M., Heber, G., Koziol, Q., Pourmal, E., Robinson, D.: An overview of the HDF5 technology suite and its applications. In: EDBT/ICDT Workshop on Array Databases, pp. 36–47 (2011)
8. Fridman, P.A., Baan, W.A.: RFI mitigation methods in radio astronomy. Astron. Astrophys. **378**, 327–344 (2001)
9. Group, H.: The board of trustees of the University of Illinois: "introduction to HDF5" (2006). http://web.mit.edu/fwtools_v3.1.0/www/H5.intro.html
10. Yan, J., et al.: Optimized data layout for spatio-temporal data in time domain astronomy. In: Ibrahim, S., Choo, K.-K.R., Yan, Z., Pedrycz, W. (eds.) ICA3PP 2017. LNCS, vol. 10393, pp. 431–440. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65482-9_30
11. Ma, X., Jiao, X., Campbell, M.T., Winslett, M.: Flexible and efficient parallel I/O for large-scale multi-component simulations. In: International Parallel and Distributed Processing Symposium (2003)
12. Madhyastha, T.M., Reed, D.A.: Exploiting Global Input/Output Access Pattern Classification. In: Supercomputing, ACM/IEEE Conference (1997)
13. Masui, K., et al.: A compression scheme for radio data in high performance computing. Astron. Comput. **12**, 181–190 (2015)
14. McMullin, J.P., et al.: CASA architecture and applications. In: Astronomical Data Analysis Software and Systems XVI, Vol. 376 (2007)
15. Nan, R.: Five hundred meter aperture spherical radio telescope (FAST). Sci. China **49**(2), 129–148 (2006)
16. Pence, W.D., Chiappetti, L., Page, C.G., Shaw, R.A., Stobie, E.: Definition of the flexible image transport system (FITS), version 3.0. Astron. Astrophys. **524**, 10 (2010)

17. Price, D.C., Barsdell, B.R., Greenhill, L.J.: HDFITS: porting the FITS data model to HDF5. Astron. Comput. **12**, 212–220 (2015)
18. Luo, G., et al.: HyGrid: a CPU-GPU hybrid convolution-based gridding algorithm in radio astronomy. In: Vaidya, J., Li, J. (eds.) ICA3PP 2018. LNCS, vol. 11334, pp. 621–635. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05051-1_43
19. Rodrigues, J.E., Rodriguez Bezos, J.E.: A graph model for parallel computation. Massachusetts Institute of Technology (1969)
20. Sanders, P.: Asynchronous scheduling of redundant disk array. IEEE Trans. Comput. **52**(9), 1170–1184 (2000)
21. Bardeau, S., Pety, J.: CLASS: continuum and line analysis single-dish software, a GILDAS software. https://www.iram.fr/IRAMFR/GILDAS/doc/html/class-html/. Accessed 21 Nov 2006
22. Schaaf, R., Brazier, A., Jenness, T., Nikola, T., Shepherd, M.: A new HDF5 based raw data model for CCAT. Eprint Arxiv (2014)
23. Smith, S., Dunning, A., Bowen, M., Hellicar, A.D.: Analysis of the five-hundred-metre aperture spherical radio telescope with a 19-element multibeam feed. In: IEEE International Symposium on Antennas and Propagation, pp. 383–384 (2016)
24. Swinbank, J.D., et al.: The lofar transients pipeline. Astron. Comput. **11**, 25–48 (2015)
25. Thakur, R., Gropp, W., Lusk, E.: Data sieving and collective I/O in ROMIO. In: Symposium on the Frontiers of Massively Parallel Computation (1999)
26. Wells, W.D., Greisen, E.W., Harten, R.H.: FITS-a flexible image transport system. Astron. Astrophys. Suppl. Ser. **44**, 363 (1981)
27. Wu, C., et al.: DALiuGE: a graph execution framework for harnessing the astronomical data deluge. Astron. Comput. **20**, 1–15 (2017)
28. Zichao, Y., et al.: An energy efficient storage system for astronomical observation data on dome A. In: International Conference on Algorithms and Architectures for Parallel Processing, pp. 33–46 (2015)