Distributed Data-Processing Pipeline for Mingantu Ultrawide Spectral Radioheliograph

F. Wang,1,2,3 Y. Mei,1 H. Deng,1 W. Wang,4 C. Y. Liu,1 D. H. Liu,4 S. L. Wei,1,2,3 W. Dai,1,2,3 B. Liang,1 Y. B. Liu,1,2,3 X. L. Zhang,1 and K. F. Ji1

ABSTRACT. The Mingantu Ultrawide Spectral Radioheliograph (MUSER) is a synthetic-aperture radio interferometer built in Ming’antu, Inner Mongolia, China. As a solar-dedicated interferometric array, the MUSER can produce high-quality radio images in a frequency range of 400 MHz–15 GHz with high temporal, spatial, and spectral resolution. Implementing of the data processing system for the MUSER is a major challenge to performing high-cadence imaging in wideband and obtaining more than two orders of higher multiple frequencies. There is an urgent need to build a pipeline for processing the massive amount of MUSER data generated each day. In this article, we present a high-performance distributed data processing pipeline (DDPP) built on the OpenCluster infrastructure for processing MUSER observational data, including data storage, preprocessing, image reconstruction, deconvolution, archiving, and real-time monitoring. We comprehensively elaborate the system architecture of the pipeline and the implementation of each subsystem. The DDPP is automatic, robust, scalable, and manageable. The processing performance under a parallel CPU/GPU hybrid system meets the requirements of MUSER data processing.

Online material: color figures

1. INTRODUCTION

Solar activities such as coronal mass ejections (CMEs), flares, and solar energetic particles (SEPs) have a critical influence on space weather because of their sudden energy release, particle acceleration, and/or transportation processes of the solar magnetic field. Observations of radio bursts are an important diagnostic approach to reveal parameters related to the various solar activities, such as magnetic fields, electron density, and plasma temperature.

The Mingantu Ultrawide Spectral Radioheliograph (MUSER), originally known as the Chinese Spectral Radioheliograph (CSRH), is a synthetic-aperture radio interferometer that is capable of observing radio bursts and producing high-quality radio images at frequency from 400 MHz to 15 GHz with high temporal, spatial, and spectral resolution. The goal of MUSER is to better understand coronal dynamics.

The MUSER consists of 100 radio antennas spirally distributed in Ming’antu, Inner Mongolia, China (see Fig. 1). The RF signal of MUSER in 0.4–15 GHz is divided into 40 antennas of 4.5 m at 0.4–2 GHz (subarray MUSER-I), and 60 antennas of 2 m at 2–15 GHz (subarray MUSER-II) bands. These antennas have been successfully installed, and the first light image was obtained in February, 2013. Table 1 shows the final specifications of the MUSER, as driven by a scientific goal (Yan et al. 2004, 2009, 2010, 2011).

The construction of a data processing system (DPS) for the MUSER is a big challenge. Ideally, the DPS should deal with all the data produced by the MUSER in real time, including data acquisition, storage, processing, and publication. However, as a radio synthesis aperture telescope with high temporal resolution, high spectral resolution, and high spatial resolution, the MUSER will produce massive observational data each day, which is a challenge to process promptly. Every 3 ms, the digital receivers of MUSER-I and MUSER-II will generate a data frame that includes the autocorrelation and cross-correlation data of 16 channels, and will then transfer the frame to the specified computer, respectively. MUSER-I will output about 31 MB data per second and about 1.05 TB in an observational day of 10 hr. The amount of data produced by MUSER-II would be approximately 2.146 TB per day. In one month, the size of all observational data will be approximately 100 TB. Meanwhile, monitoring the operation of the full telescope system requires generating images as quickly as possible to determine the status of the telescopes.

In this scenario, setting up an automatic data processing system, i.e., a pipeline, is one of the most significant issues in the construction of the MUSER. In this study, we present...
a distributed data-processing pipeline (DDPP) built on a distributed computing infrastructure of our own design named OpenCluster. After a brief introduction of the MUSER, we concentrate on the design of the high-performance data-processing pipeline. The rest of this article is organized as follows. § 2 discusses the system architecture of the MUSER. The requirements of MUSER data processing are discussed in § 3. We introduce the OpenCluster in § 4 and present the key techniques of MUSER pipeline in § 5. § 6 lists DDPP performance. Finally, § 7 and § 8, respectively, provide discussions and a short summary.

2. MUSER ARCHITECTURE

2.1. The Architecture of MUSER

Figure 2 shows an architecture diagram of the MUSER. The outdoor equipment consists of antennas, wideband feeds, low-noise amplifiers, optic transmitters, optic fibers, and control units, as well as such things as power supplies. Optical fibers transmit the radio frequency signal bands to the indoor unit. The array configuration is a self-similar spiral geometry (Wang et al. 2011).

The indoor equipment includes optic receivers, analogous receivers, analog-to-digital (A/D) converters, digital correlation receivers, local oscillators, monitors, and computers (see Fig. 2). The signal of each band is processed digitally, using a 1 Gsps analog digital converter. The sampled signals then go through a digital spectral analyzer of 16 channels with 25 MHz as the bandwidth of each channel for MUSER. Then, the signals are correlated at a point frequency within each channel. This procedure is repeated to cover the whole frequency. The signal from each 25 MHz will be correlated with signals from other antennas. We estimated the delay compensation bank to be in the range of 10 μs with a step of 1 ns.

2.2. The Architecture of Data Processing Environment

A computer cluster environment has been built for MUSER data processing (see Fig. 2). The system can efficiently realize load balance, high compatibility, and good scalability. The number of the servers can be seamlessly expanded to obtain more computing power when computational load gets high.

So far, the computer cluster consists of eight servers. Each server has two-way Intel Xeon E5-2650 v2 CPUs, 2.6 GHz, 16 cores, 32 GB memory, and a 1 TB hard disk. Considering the processing requirements of massive data, the computer cluster is divided into two subclusters. One subcluster, i.e., Cluster-C (including C1–C4), is mainly for file processing. Another subcluster, i.e., Cluster-G (including G1–G4), is mainly for high-performance imaging, and all servers in the Cluster-G are equipped with an NVIDIA Tesla C2050 graphics process unit (GPU) card, respectively.5

A network attached storage (NAS) system is deployed for the MUSER-I/II observational data archive. The capacity of the NAS is about 300 TB and will be expanded to 1 PB in the near future.

All servers are connected to a 10 Gb Ethernet switch using 10 Gb links. The NAS system is also connected to the 10 Gb Ethernet switch using multiple 10 Gb aggregated links to guarantee communication performance.

In addition, there are four types of dedicated servers:

1. The acquisition server receives observational data from the digital receiver every 3 ms and further stores the data to the storage system. Meanwhile, the acquisition server repeatedly forwards the observational data to monitoring servers for real-time monitoring. The data forward frequency can be adjusted according to the monitoring requirements. There are two acquisition servers for MUSER-I and MUSER-II, respectively.

5 See http://www.nvidia.com.
Each acquisition server has a stand-alone storage area network (SAN) device to temporarily store observational data. The capacity of the SAN can store the observational data in a month.

2. The database server stores all parameters such as instrument parameters, telescope status, and weather data. A MySQL database is installed on the server.

3. The weather server is a gateway that acquires the weather data from a Vantage Pro weather station. It will acquire data every 1 minute and save the weather information to the database server. The Vantage Pro2 weather station currently installed measures barometric pressure, temperature, humidity, rainfall, wind speed and direction, and UV/solar.

4. A monitoring server monitors running states via the visualization method. A display adapter with four Digital Visual Interface (DVI) output ports is installed on the server so as to connect four LCD monitors simultaneously. Two monitoring servers are in charge of status display for MUSER-I and MUSER-II, respectively.

3. REQUIREMENTS ANALYSIS

The data processing of synthetic aperture radio interferometry has been described in detail in previous literature (Thompson et al. 2008; McMullin et al. 2007). Figure 3 shows the flow chart of MUSER data processing.

1. Data storage and data distribution. The storage of observational data is the premise of the data processing. Current MUSER observational data are saved to the SAN system first. However, due to the performance limitations of the SAN, it is hard to synchronously read observational data from the SAN while writing data. Therefore, we cannot read data from the SAN while in MUSER observation. The observational data must be read and processed in the batch after the observation each day.

2. Data archive format. The data format of the observational data for archiving is a worthwhile problem in MUSER data storage. Due to the amount of observational data, the different data storage format will significantly affect the hardware configuration of the storage system. The available space of the storage system is closely related to the archive format. In addition, the high-performance index is another important issue while...
designing the high-performance data processing system of the MUSER, or the data retrieval will be a bottleneck of MUSER data processing.

3. **High-performance imaging.** The MUSER is capable of observing 64 channels in MUSER-I and 528 channels in MUSER-II. The high-performance imaging is an urgent demand for data processing, publication, and monitoring. However, the deconvolution manipulation for dirty images is a very time-consuming calculation procedure. It is necessary to develop a high-performance imaging technique to improve the scientific output of the MUSER.

4. **Customizable workflow.** With the change of a scientific research goal, the data processing flow will also change. The DDPP should support customizable workflow in data processing so as to dynamically adjust the data-processing flow and satisfy the requirements of astronomical scientists.

5. **Data reprocessing.** Data reprocessing is a critical requirement of the MUSER. Due to the change of requirements, observational data often needs to be processed in different ways by scientific research. For example, to improve the spatial resolution, scientists need to integrate multiframe data in any given period of observational time.

Obviously, all the procedures in Figure 3 should be implemented by the MUSER pipeline. Meanwhile, referring to related studies (Jenness & Economou 2014; Freudling et al. 2013; Shamir & Nemiroff 2008; Hummel et al. 2006) on the pipeline design of modern telescopes, the requirements listed as follows are critically considered and designed because of the specific features of the MUSER.

4. **OPENCLUSTER—DISTRIBUTED COMPUTING INFRASTRUCTURE**

To design and develop the DDPP, we first developed a novel distributed-computing infrastructure, OpenCluster, which is a wholly self-designed software for quickly designing a scientific data-processing pipeline.

Referring to the design and operation principles of stream computing (Neumeyer et al. 2010; Buck et al. 2004), OpenCluster simplifies the technical implementation of stream computing and further adds greater design capability and many additional features for astronomical data processing.

Figure 4 shows a conceptual diagram of OpenCluster. OpenCluster regards a data-processing pipeline as a data-processing factory. The factory is in charge of data processing and undertakes all data-processing tasks. In the factory, there are many task managers who manage a group of workers. The task managers obtain the tasks from the factory or other task managers, schedule their subordinate workers to run the tasks, and finally, collect the task results from the workers. In addition, there are several external service windows that open for all workers and task managers in the factory to provide specified services.

It is easy to design an astronomical data-processing pipeline using the OpenCluster infrastructure. OpenCluster is written with Python language, which is widely used in astronomy. Many mature packages such as PyFits (Barrett & Bridgman 1999) and Pyro4 are used in software development. OpenCluster is a pure Python application that can be installed on any operating system. Figure 5 shows the class diagram of OpenCluster.

The current OpenCluster edition has encapsulated all of the complicated concepts of distributed computing, e.g., task

---

3 See https://pythonhosted.org/Pyro4/.
5. IMPLEMENTATION OF THE DDPP

According to the data-processing flow of the MUSER (see Fig. 3), we created six task modules and four services that inherited from the base classes of the OpenCluster.

Figure 6 shows a structure diagram of the DDPP. The six task modules include the fundamental functions of data preprocessing, image reconstruction, image deconvolution, raw observational data UVFITS file generation, FITS file generation for final data products, and observation monitoring. Meanwhile, three global services are designed to provide the services of ephemeris computing, instrumental status query, and weather information query.

5.1. Workflow Tag Design

OpenCluster provides some level of support for workflow technology. According to the data tag, OpenCluster will schedule the proper managers to process the data and to obtain scientific data production. The DDPP is a stand-alone computing platform that supports processing MUSER-I and MUSER-II observational data simultaneously. Therefore, the design of workflow tag is important for the DDPP. Meanwhile, these tags can be expanded according to future scientific requirements.

Table 2 lists all possible tags including, currently, six data sources, five process modes, and four data publication modes. It is easy to understand the definition of the tag. For example, the data tag of <1:5:3> means that the input data include real-time observational data from MUSER-I; these information should be first preprocessed and then generated to the dirty images. Deconvolution can then be performed. Finally, these images will be published with the FITS file format.

5.2. Data Acquisition and Distribution

To acquire the massive data produced by MUSER, the acquisition server receives the observational data encapsulated as a
frame from digital receivers through a 1.25 GB optical fiber every 3 ms. A frame includes the observational data of 16 channels with one polarization. Therefore, the acquisition server has to receive multiple consecutive frames to acquire all channels and all polarizations. For MUSER-I, the acquisition server should receive eight consecutive frames to generate a full frame (64 channels and two polarizations) and will take 25 ms $(3 \times 8 + 1 \text{ ms data read out})$. The data of MUSER-II is close to MUSER-I. The major difference is that MUSER-II has 528 channels. Hence, the size of a frame in MUSER-II is 204,800 bytes and the 66 total frames are treated as a full frame. The acquisition period is 206.25 ms.

Figure 7 shows the raw data format definition of MUSER-I. All observational parameters such as polarization, band, and channel are stored and can be easily readout according to the predefined byte stream offset. The MUSER-II format is close to the MUSER-I format.

To monitor the instrumental status, we have to periodically extract parts of observational data and send these to the monitoring system via TCP protocol. The monitoring server acts as the TCP server.

Considering the requirements of observational monitoring, 5 s in MUSER-I and 15 s in MUSER-II are setup as the sampling period. In every sampling period, the data acquisition server will send 16 frames (MUSER-I) and 132 frames (MUSER-II) to the monitoring server. The monitoring server will separate a full frame from these frames for subsequent data processing.

### 5.2.1. Observational Data Storage and File Format

After comparing the advantages and disadvantages of each file format, we determined to archive the observational data with a MUSER raw data format. Actually, most modern telescopes use a specified data format to store observational data. For example, the ALMA and JVLA projects store data with a common archival science data model (ASDM) format (Glendenning & Raffi 2008), and have jointly developed the software to fill this data into CASA (McMullin et al. 2007). In the ASDM format, the bulk of the data is contained in large binary data format (BDF) tables, with the metadata and ancillary information in XML tables. Meanwhile, due to the wide applications of CASA
software, a measurement set (MS) format is also a common file format in radio astronomy.

Storing with an ASDM or MS format will bring more convenience for further data sharing and utilizing, but it would occupy much additional storage space. For a frame acquired in every 3 ms of MUSER-I, the size is 100,000 bytes. Table 3 lists the sizes with different file formats, respectively. Obviously, due to the use of XML and the metadata definition, the size of ASDM and UVFITS files would be significantly increased and further increase the expenditure of the storage system. For MUSER and its massive observational data, it is a huge stress burden because of limited funds.

5.2.2. Data Archive and High-Performance Index

5.2.2.1. Observational Data Archive

The data index technique is critically significant for the MUSER to retrieve observational data quickly from the massive data of the MUSER. So far, all observational data are saved into the storage system in file form. About 1.152 million observational data frames in a day, and about one billion frames in a month, will be output by the MUSER-I. To retrieve a specified frame and locate the corresponding file quickly, relational database technology has been widely used to manage the index information of the observational data. However, based on our preliminary experiment, we realize that it is difficult to archive so many observational data files and meet the performance requirements of subsequent data processing (see Fig. 8). The query performance of MySQL under more than 0.1 billion records would take more than 60 s to fetch a record. Obviously, this result would critically limit the processing performance of the DDP.

We create indexes for observational data by using the FastBit (Wu 2005; Wu et al. 2009; Liu et al. 2014) technique. FastBit is very well suited for managing massive data because of its bit index technique. As an open-source data-processing library following the spirit of the NoSQL movement, FastBit offers a set of searching functions supported by compressed bitmap indexes (Wu et al. 2001). It treats user data in a column-oriented manner.

The FastBit indexes for each observational data file are automatically created while transferring the observational data from SAN to NAS. The main index fields include file name and location, observational date and time, polarization, and band and frame byte offset. Frame byte offset means that the byte offsets of a specified frame from the beginning of the observational file. In data processing, it is easy to retrieve the information of the file name, location, and frame byte offset with the query parameters of observational date, time, polarization, and band. The subsequent program can open the observational file with the retrieved file name, skip the bytes defined by the

### Table 3

| File format     | File size (bytes) |
|-----------------|-------------------|
| Raw data        | 100,000           |
| UVFITS          | 200,000           |
| FITS-IDI        | 368,000           |
| Measurement set | 2,200,100         |
| ASDM            | 324,000           |

![Diagram of the MUSER data format.](image_url)
frame byte offset, and directly read the observational data needed.

5.2.2.2. Parameter Data Archive

In addition to archiving of observational data, it is necessary to separately record all parameter data, such as weather conditions, instruments status (i.e., antenna, receivers, etc.), instruments parameters (i.e., the position of each antenna and the length of each optical fiber) in a time frame. To guarantee the correspondence of MUSER observational data, the data must be permanently stored and can be retrieved according to the observational date and time.

All parameter data are stored in a MySQL database. Four tables, such as instrumental status, optical fiber length, antenna position, and weather, are created (see Fig. 9): (1) The instrumental status table records the status of the telescope, especially the availability of each antenna that can be used to flag the observational data in data processing. (2) An optical fiber length table records the length of each optical fiber between the outdoor and indoor devices that will be used in computing RF signal transfer delay. Although the length rarely varies, it is necessary to record for high-precision computing. (3) The antenna position table stores all the locations and the altitude of all antennas. The center position of the MUSER is (0., 0.). The deviation values from the center position of the each antenna are stored in the table, respectively. (4) A weather conditions table records the weather information.

There are three steps to retrieve the observational data and related parameters (see Fig. 9). In step 1, the DDPP gets the processing date and the time from the users. In step 2, according to the processing date and time, the DDPP retrieves the observational data by using a FastBit index and then locates the file directory. In step 3, the DDPP retrieves the parameters from four tables, according to the date and time, respectively.

![Fig. 8.—Comparison between FastBit and the MySQL database.](image)

5.3. Task Modules

5.3.1. Data Preprocessing

Data preprocessing is a significant part of MUSER data processing. The goal of data preprocessing is to correct, flag, compensate and calibrate the observational data. Meanwhile, it is possible to overlap the data among several continuous frames in data preprocessing to generate an integral image with higher resolution.

5.3.1.1. Delay Compensation and Fringe Stop

The delay compensation and the fringe stop must be conducted in MUSER data processing. The digital receiver of the MUSER will encapsulate the values of transmission delay into the frame and output to the acquisition server. Therefore, to correct the transmission delay and conduct the fringe stop, the following steps are implemented in data preprocessing:

1. Obtain the correlated visibility data from the observational data, including the real and imaginary parts; define as $Aej\phi$, where $A$ is the amplitude and $\phi$ is the phase.
2. Obtain the delay parameter ($dt_{\text{raw}}$) from the observational data encapsulated by the digital receiver.
3. Subtract the delay parameter from the delay skew ($dt_{\text{trans}}$) caused by different lengths of the optical fiber. These delay skews are recorded in the database.
4. Compute the delay between antenna $i$ and $j$: $dt_{ij} = dt_j - dt_i$, where $dt = dt_{\text{raw}} - dt_{\text{trans}}$. The corresponding correlation value is $A_{ij}e^{j\phi}$.
5. Obtain the observation frequency ($F_{rf}$) and intermediate frequency ($F_{if}$) of each channel from the observational data.
6. Compute fringe stopping. The phase, which can be compensated on the complex correlation value $A_{ij}e^{j\phi}$ between antenna $i, j$, can be defined as $\Phi_{ij} = 2\pi[F_{rf} \times dt_{ij} - F_{if} \times (dt_j - dt_i)]$; then, we can compute the complex correlation.
value by subtracting the initial phase from the phase of fringe stopping \( A_{ij} e^{i\phi} = A_{ij} e^{i(\phi - \phi_{fsij})} \).

### 5.3.1.2. Satellite Calibration

The MUSER observes the satellite to calibrate the phase of each channel. The calibration steps are listed as follows:

1. Obtain the correlated visibility function from the observational data, including real and imaginary parts; define as \( A_{\text{sun}} e^{i\phi_{\text{sun}}} \), where \( A_{\text{sun}} \) is the amplitude and \( \phi_{\text{sun}} \) is the phase.
2. Obtain the correlated visibility data from the observational data, including real and imaginary parts, defined as \( A_{\text{satellite}} e^{i\phi_{\text{satellite}}} \), where \( A_{\text{satellite}} \) is the amplitude and \( \phi_{\text{satellite}} \) is the phase.
3. Obtain the result: \( A_{\text{sun}} e^{i\phi_{\text{sun}}} = A_{\text{sun}} e^{i(\phi_{\text{sun}} - \phi_{\text{satellite}})} \).

### 5.3.1.3. Data Flagging

Radio frequency interference (RFI) is a disturbance that affects an electrical circuit due either to electromagnetic conduction or electromagnetic radiation emitted from an external source. Especially at low radio frequencies of MUSER, stray electromagnetic transmissions often interfere with the incoming radiation from a source, corrupting the data being recorded. Therefore, flagging involves the identification and masking of RFI-affected data points, and is an inevitable step in standard data analysis.

The DDPP supports two main data flagging approaches. One is using hardware information acquired from the instrumental status to flag bad data directly. Another is using “flagcal,” an automatic RFI identification and flagging algorithm that have been integrated into CASA software (Urvashi et al. 2003).

### 5.3.2. Monitoring

Two servers are specifically deployed for monitoring MUSER-I/II observations in real time, respectively. After data preprocessing, the results will be transferred to the monitoring task module. So far, to monitor the status of each antenna and its corresponding parameters, we refer to the monitoring functions of other interferometers and design two types of diagrams for real-time monitoring. One is a power and phase scatter diagram, and the other is a histogram diagram of autocorrelation.

The power and phase scatter diagram [see Fig. 10 (left panel)] is for monitoring the baseline and its visibility data. The \( x \)- and \( y \)-axis is the number of each antenna. The power spectrum of each baseline is plotted in the bottom left corner of the diagram, and the phase is plotted in the upper right corner. Obviously, if an error occurs during observation, the power spectrum of the corresponding baseline should be unusual. Therefore, a black line would be displayed in the power and phase scatter diagram.

The autocorrelation diagram is a histogram diagram [see Fig. 10 (right panel)], which monitors the autocorrelation variations of each baseline. The \( x \)-axis is the number of the antenna and the \( y \)-axis is the power spectrum of the autocorrelation.

![Fig. 10.—MUSER monitoring diagrams. See the online edition of the PASP for a color version of this figure.](image-url)

2015 PASP, 127:383–396
5.3.3. Distributed UVFITS/FITS-IDI File Generation

It is necessary to generate UVFITS/FITS-IDI format files when the MUSER needs to share observational data with other scientific groups. The raw observational data cannot be directly processed by other groups because it has no corresponding information of observation. For example, the antenna status should be used to flag the observational data.

The DDPP should be capable of generating the UVFITS and FITS-IDI formats. According to the official definition of the UVFITS format (Wells et al. 1981; Greisen 2012), the DDPP writes four binary tables (i.e., Primary HDU, AIPS FQ, AIPS AN, and AIPS SU) to the UVFITS file. For the FITS-IDI format file, the DDPP will write five binary tables such as ANTENNA, FREQUENCY, SOURCE, ARRAY_GEOMETRY, and UV_DATA besides the primary HDU (Greisen 2011).

The implementation of UVFITS/FITS-IDI file generation module is very simple. However, to generate a large amount of files quickly, the DDPP uses a multiprocessing technique and can schedule several workers to generate files at the same time so as to improve generation performance. Figure 11 shows the flow chart of UVFITS generation.

5.3.4. High-Performance Imaging and Deconvolution

MUSER data processing uses the Högbom CLEAN algorithm (Högbom 1974), and other CLEAN algorithms are under development. Due to the low performance of the CLEAN algorithm, it is necessary to improve its performance.

The migration of the CLEAN algorithm from CPU to GPU is feasible. Both the gridding and CLEAN kernels were parallelized by pixels in the UV and image plane, respectively. Meanwhile, once the maxima has been located in the dirty image, the convolution is parallel for both building the CLEANed image and subtracting from the dirty image.

Referring to an implementation of the Högbom Clean algorithm under the GPU CUDA platform, we migrate the MIRIAD (Sault et al. 1995) and its implementations from the CPU to GPU environment. We have implemented standard algorithms for both these tasks on the GPU, achieving speedups of $\sim 5$ for gridding and $\sim 50$ for CLEANing. The program is supported by PyCuda and Scikits.cuda.

5.4. Global Services

5.4.1. High-Precision Ephemeris Calculation

High-precision ephemeris calculation is the fundamental issue of the MUSER processing system. Whether in observation or data processing, the position of the target, e.g., the Sun or an artificial satellite, is always an important parameter. Many processing procedures in MUSER data processing are seriously dependent on the high-precision target position.

Based on the evaluation of the MUSER baseline, the largest baseline is about 3200 m, which means that the position accuracy should be superior to 1 mas while the observational frequency is 15 GHz (Yan et al. 2009).

5.4.1.1. JPL DE405 and Ephemeris Calculation

To obtain such an accuracy level, the precise ephemeris must be considered. In the study, we selected the JPL DE405 planet ephemeris (Standish et al. 1998; Newhall et al. 1983; Charlot et al. 1995) to provide a high-precision fundamental ephemeris, and we selected Naval Observatory Vector Astrometry Software (NOVAS) (Kaplan et al. 2012) to construct a high-precision ephemeris program for the MUSER.

The computing procedures are as follows:

1. Compute the $X$, $Y$, and $Z$ coordinates of the observation station in the ECEF coordinate system based on the observational time (UTC), longitude ($L$), latitude ($B$), and the altitude ($H$) of the observation station.

2. Calculate the position and velocity of the observational target under the International Celestial Reference System (ICRS) and the J2000.0 mean equatorial system of coordinates, respectively. Meanwhile, the distance between the target and the observational station is also calculated.

3. Calculate the position of the observational target at the observational time.
4. Calculate the local apparent sidereal time of the topocentric coordinates. Further calculate the right ascension and the declination under the topocentric coordinate systems.

5. Calculate the Greenwich sidereal time and further calculate the local apparent sidereal time (LAST) by using the geographic longitude.

5.4.1.2. Automatic IERS Data Updating

In ephemeris calculations, TAI-UTC and three Earth-orientation parameters (i.e., x, y, and UT1-UTC) are necessary for high-precision position calculation. To guarantee the precision of the ephemeris calculation, we must update these parameters from an official organization. We selected the data from the International Earth Rotation and Reference Systems Service (IERS). The IERS was established in 1987 by the International Astronomical Union and the International Union of Geodesy and Geophysics. The IERS provides data on Earth orientation, on the International Celestial Reference System/Frame, on the International Terrestrial Reference System/Frame, and on geophysical fluids. It maintains also conventions containing models, constants and standards.

The IERS publishes four bulletins: Bulletin A contains rapid determinations for earth orientation parameters, Bulletin B contains monthly earth orientation parameters, Bulletin C contains announcements of the leap seconds in UTC, and Bulletin D contains announcements of the value of DUT1. To retrieve the necessary parameters, we chose Bulletin A as the data source.

A Linux daemon, IERSSync, runs on the background of the server to automatically maintain the IERS data. The IERSSync is similar to a search engine crawler and will visit the IERS Web site every day. After the retrieval of the Web site pages, the IERSSync can analyze the contents of the HTML pages and try to locate the new IERS Bulletin A file URL. If a new Bulletin A file is published, the IERSSync will download the text file automatically, search the information from the text file, and finally save the parameter data into the MySQL database.

5.4.1.3. High-Performance Interpolation Computing

The ephemeris calculation is a time-consuming task, and the computing performance of ephemeris calculation is far below the expectation of real-time processing. Even on a high-end server with Intel Xeon 16 × 2.60 GHz cores and 32 GB memory, the calculation speed of one planet is about 20 ms. Obviously, it will cost a large amount of time when processing a single frame data of every 3 ms.

To guarantee the precision of the final position and obtain the maximum performance of computing, we used the interpolation method. We used NOVAS to calculate a series of accurate positions, e.g., the right ascension and the declination in UTC time—0:30, 0:00, 0:30,...,10:00, etc., of the observational target in a day, and interpolate the right ascension and the declination to the given time. Obviously, the interpolation method should bring faster performance than the NOVAS.

We conducted several preliminary experiments to assess the availability and performance upon two interpolation methods such as linear interpolation and three-point parabolic interpolation. We assumed that the final precision of interpolation will be greater than 0.001°. The preliminary experiments results show that both interpolations methods can meet the requirements of the computing performance. In the condition of the precision 0.001°, the linear interpolation method needs 49 accurate values (every 30 minutes) to compute the planet position by any given time in a day, and the parabolic interpolation method needs only 25 (every 1 hr) real values.

The DDPP uses a parabolic interpolation method to obtain high-precision position of the target. We collected the statistical time overhead of two interpolation methods and the related data initialization, respectively, and chose the three-point parabolic interpolation method. Actually, the computing performances of the two methods were very close in the high-end computer server. The main difference is the time overhead of data initialization. Therefore, the calculation of 25 real values by NOVAS only takes about 500 ms.

5.4.2. Weather and Instrumental Status Query

Due to the requirements of data flagging, the DDPP provides two global services to query weather information and instrumental status respectively. As mentioned in the previous section, we store this information in a MySQL database. Hence, the implementation of these two services is quite simple. According to the data and time, the service would compose a SQL statement and submit to the MySQL database. The query results would send back to the invokers.

In general, weather conditions will not affect the observation of the MUSER. The DDPP only considers the two conditions that would affect the observational data, such as strong wind and heavy rain.

6. SYSTEM DEPLOYMENT AND APPLICATION

The DDPP has been deployed on each server for MUSER instrument testing and observation. All servers are installed on the CentOS 64 bit operation system. We used version 2.9.7 of Python. All source codes of the DDPP are stored in a directory.

To test the availability of the DDPP, we focused on two aspects. One is the correctness of the DDPP. Another is data process performance. It is easy to verify the correctness of the DDPP because there is much mature and standard data-processing software for synthetic aperture interferometer such as CASA and MIRIAD. We generate the UVFITS file of the observational data, import the file to CASA, and finally compare the results between the DDPP and the CASA.

11 http://www.iers.org.
Another significant issue of the DDPP is computing performance. We carefully tested the time overheads of each processing task that runs under one process; Table 4 lists the results.

According to Table 4, it is easy to estimate the time overhead of the tasks. For example, to generate the UVFITS file with one process should take at least 0.72 s (the time overhead of data preprocessing + UVFITS generation).

To further improve the performance, multithreaded and multiprocess technologies have been used in the DDPP, which is a multithreaded application that permits more threads to run tasks. According to the hardware configuration of the servers, we setup the number of the threads as the number of CPU cores. At least 32 threads started in one server to improve the processing performance. For example, if four servers are in the cluster work in parallel, there are a total of at least 128 threads to parallel generate the UVFITS files. Therefore, under the current hardware environment, about 178 files will be generated in 1 s.

7. DISCUSSION

The DDPP is a distributed parallel-computing pipeline for the MUSER. Although the much construction of the DDPP has been completed, some issues still need to be discussed and further improved.

7.1. OpenCluster

OpenCluster is a new lightweight infrastructure for designing an astronomy data-processing pipeline. Objectively, it is risky to build the DDPP on the OpenCluster infrastructure for the MUSER. Many mature traditional techniques such as Message Passing Interface (MPI) (Gropp et al. 1999), Hadoop,12 and Storm13 have been widely used in data processing, and have many of successful cases. For example, according to the documents of SKA, a stream computing technique will be deployed in the SKA’s high-performance storage and data processing. MPI technology is widely used in high-performance image processing for many modern telescopes.

We ultimately developed OpenCluster for use in place of these mature technologies because, although the mature systems provide several useful features to easily construct high-performance distributed-computing programs, it is difficult to implement the MUSER data-processing pipeline by using these systems:

1. MUSER data processing has many different and variable requirements of data reduction and production. For example, MUSER data productions will be in different formats, such as raw data with data preprocessing only, UVFITS, FITS-IDI, dirty image, or deconvolution image. For traditional technologies, especially MPI, it is very difficult to process different tasks in parallel, which would lead to losing the advantage of data parallel processing.

2. It is difficult for traditional distributed-computing infrastructures, such as MPI, to support the data-driven mode that the MUSER urgently demands.

3. These infrastructures are difficult and complex for astronomers to construct their pipeline system because the astronomers have to learn many profound theories such as process, thread, mutex, and semaphores. Meanwhile, these astronomers also need to master the programming skills on distributed-computing programming.

7.2. Advantages and Disadvantages of the DDPP

The DDPP is the first high-performance distributed astronomical data-processing system in China. After continuous system tests and improvements, the DDPP testifies the preferable improvement of the reliability and availability of the equipment with continuous operation in a period of time. According to user feedback, the DDPP has the following distinguished advantages:

1. Expandable. The DDPP is a loose-coupled system. All processing components are encapsulated into the stand-alone services and deployed upon the network. It is easier to build more service components and deploy to expand the functions of the DDPP. Meanwhile, deadly errors in a service would not interrupt the operation of the DDPP.

2. Robust. Referring to the current mature systems, message queue (MQ) technology is used for data and control message transferring among services. MQs provide an asynchronous communications protocol, meaning that the sender and receiver of the message do not need to interact with the MQ at the same time. Messages placed onto the queue are stored until the

| Task                      | Subtask                  | TO  |
|---------------------------|--------------------------|-----|
| cmData preprocessing      | Analyzing one frame      | 0.188 |
|                          | Delay compensation       | 0.183 |
|                          | Computing UVW            | 0.072 |
| UVFITS generation         | Creating tables          | 0.251 |
|                          | Writing a UVFITS file    | 0.026 |
| Integral                  | One frame overlay        | 0.031 |
|                          | Average value computing  | 0.039 |
| Gridding and dirty map    | One 1024 × 1024 image    | 1.307 |
|                          | One 512 × 512 image      | 1.005 |
|                          | One 256 × 256 image      | 0.860 |
| CLEAN with one iteration  | One 1024 × 1024 image    | 1.650 |
|                          | One 512 × 512 image      | 0.945 |
|                          | One 256 × 256 image      | 0.937 |
| Services                  | Ephemeris calculation    | ~0.001 |
|                          | Weather retrieval        | ~0.002 |
|                          | Instrument status retrieval | ~0.001 |

NOTE.—TO, time overhead (s).

12 http://hadoop.apache.org/.
13 http://storm.apache.org.
recipient retrieves them. Therefore, the DDPP is a robust system that can provide reliable operation without any maintenance.

3. Hybrid computing support. The DDPP has integrated distributed-computing technology and GPU technology for high-performance data processing. Actually, it is difficult for a single technique to meet the requirements of high-performance data processing of the MUSER. Although the GPU is suitable for high-performance image processing, it is difficult to deal with situations of massive data communication and transfer. Traditional parallel computing technology such as MPI has significant disadvantages in communications between each cluster node. Communications would create a large overhead while processing the massive MUSER data.

However, current performance is a considerable problem of the DDPP. Using the Python programming language brings more advantages for the DDPP, such as good scalability and portability, especially with the availability of many scientific computing packages. Python is becoming a mainstream computer language in current scientific data processing. There are many mature scientific data packages, such as SunPy, AstroPy, and NumPy, that can improve development performance and guarantee the correctness of data reduction. However, according to the results shown in Table 4, the computing performance of each processing task that is written in the Python language is not very high in comparison with the performance of C/C++ language implementation. In some computing tasks, the performance of C/C++ would be at least twice as fast as that of Python.

In addition, the performance of image deconvolution is still a big problem for current systems. Due to the limitations of the Högbom CLEAN algorithm, multiple iterations would lead to a very time-consuming CLEAN.

Improving computing performance is one of the most significant tasks in the future. Meanwhile, with the quick decrease of the computer hardware price, the program written by Python can also meet the requirements of high-performance data processing by purchasing more computers.

8. CONCLUSION

The MUSER will be an important synthetic-aperture radio interferometer for obtaining high-quality radio images from 400 MHz–15 GHz with high temporal, spatial, and spectral resolution. To meet the requirements of MUSER data processing, we have implemented and deployed a DDPP that has many distinguished features. Meanwhile, we have proposed key techniques in detail, such as a raw data archive, data index creation, high-precision target position calculation, and high-performance FITS file generation.

In summary, the successful application of the DDPP proves that the open-source distributed-computing infrastructure (i.e., our self-developed OpenCluster, which can be downloaded online\(^1\)) is robust, reliable, and scalable. The distributed-computing technology should be a trend for developing a high-performance data-processing pipeline for modern telescopes. Our study presents a valuable reference for other radio telescopes, especially aperture synthesis telescopes, and also gives a valuable contribution to current and future data-intensive astronomical observations.

This work is supported by the National Natural Science Foundation of China (No. U1231205, 61462053,11263004, 11203011, 11163004, and 11103005) and the Natural Science Foundation of Yunnan Province (No. 2013FA013, 2013FA032, 2013FZ018). We do appreciate the suggestions from Prof. Monique Pick of Paris Observatory and Prof. Hiroshi Nakajima of National Astronomical Observatory of Japan. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers.

\(^1\) At https://github.com/astroitlab/opencluster.

REFERENCES

Barrett, P., & Bridgman, W. 1999, in ASP Conf. Ser 172, Astronomical Data Analysis Software and Systems VIII, eds. D. M. Mehringer, R. L. Planit, & D. A. Roberts (San Francisco: ASP), 483
Buck, I., et al. 2004, ACM Trans. Graphics, 23, 777–786
Charlot, P., Sovers, O., Williams, J., & Newhall, X. 1995, AJ, 109, 418
Freudling, W., Romaniello, M., Bramich, D. M., Ballester, P., Forchi, V., García-Dabló, C. E., et al. 2013, A&A, 559, 93
Glendenning, B., & Raffi, G. 2008, Proc. SPIE, 701902, 701902
Greisen, E. W. 2011, AIPS Memo Ser. 114r (Socorro, NM: NRAO)
Högbom, J. 1974, A&AS, 15, 417
Hummel, W., Modigliani, A., Szeifert, T., & Dumas, C. 2006, NewA Rev., 50, 412
Jennness, T., & Economou, F. 2014, Astron. Comput., 9, 40
Kaplan, G., et al. 2012, Astrophysics Source Code Library (ascl/1202.003)
Liu, Y.-B., et al. 2014, J. Korean Astron. Soc., 47, 115
McMullin, J., Waters, B., Schiebel, D., Young, W., & Golap, K. 2007, in ASP Conf. Ser. 376, Astronomical Data Analysis Software and Systems XVI, eds. R. Shaw, F. Hill, & D. Bell (San Francisco: ASP), 127
Neumeyer, L., Robbins, B., Nair, A., & Kesari, A. 2010, in Proc. 2010 IEEE Int. Conf. on Data Mining Workshops (New York: IEEE), 170
Newhall, X., Standish, E. M., & Williams, J. G. 1983, A&A, 125, 150
Sault, R. J., Teuben, P. J., & Wright, M. C. 1995, in ASP Conf. Ser. 77, Astronomical Data Analysis Software and Systems IV, eds. R. A. Shaw, H. E. Payne, & J. J. E. Hayes (San Francisco: ASP), 433
Shamir, L., & Nemiroff, R. J. 2008, Appl. Soft Comput., 8, 79

2015 PASP, 127:383–396
Standish, E. 1998, JPL IOM 312.F-98 (La Cañada Flintridge: JPL)
Thompson, A. R., Moran, J. M., Swenson, G. W., Jr. 2008, Interferometry and Synthesis in Radio Astronomy (New York: Wiley)
Urvashi, R., Rao, A. P., & Pune NCRA 2003, Automatic RFI Identification and Flagging, Vol. 22, Tech. Rep., NRAO
Wang, W., Yan, Y., Liu, D., Chen, Z., & Liu, F. 2011, URSI GASS, Istanbul, Turkey
Wells, D., Greisen, E., & Harten, R. 1981, A&AS, 44, 363
Wu, K. 2005, in J. Phys. Conf. Ser. 16 (Bristol: IOP), 556
Wu, K., Otoo, E. J., Shoshani, A., & Nordberg, H. 2001, Tech. Rep. (Berkeley: Lawrence Berkeley National Laboratory)
Wu, K., et al. 2009, in J. Phys. Conf. Ser. 180 (Bristol: IOP), 012053
Yan, Y., Huang, J., Chen, B., Liu, Y., & Tan, C. 2010, Adv. Space Res., 46, 413
Yan, Y., et al. 2011, in Proc. IEEE General Assembly and Scientific Symp. XXXth URSI (New York: IEEE), 1
Yan, Y., Zhang, J., & Huang, G. 2004, in Proc. IEEE Radio Science (New York: IEEE), 391
Yan, Y., et al. 2009, Earth Moon Planets, 104, 97