Developing a Real-Time Processing System for HERA

Paul La Plante, Peter K. G. Williams, and Joshua S. Dillon

Abstract – The Hydrogen Epoch of Reionization Array (HERA) is a radio telescope in the Karoo desert of South Africa endeavoring to observe Cosmic Dawn and the Epoch of Reionization. When fully constructed, it will consist of 350 antennas and generate over 60 TB of data each night. In order to keep pace with the relatively large rate of data, we have developed a real-time processing system for HERA. The system is responsible for inspecting data for acceptable levels of quality and flagging unusable data, as well as providing an initial calibration solution. It consists of the pipeline management system, called the hera_opm package, and the data management system, called the librarian. Though the systems have been developed in the context of analyzing HERA data, they feature public and open-source software and have been designed to be adapted and used in other contexts as necessary.

1. Introduction

As current- and next-generation telescopes are constructed, they are projected to generate more data than ever before. This fact is particularly true for the field of radio astronomy, where the volume of interferometric data sets is among the largest in astronomy. Part of the reason for the large data comes from the availability of modern digital signal-processing technology, where the number of array elements and spectral resolution can quickly lead to total data volumes orders of magnitude larger than in, for example, optical astronomy. The raw data sets are known as *visibilities*, and encode the degree of correlation observed by a pair of antennas at a given time of observation in the form of a complex-valued quantity. For a given array of N_{ant} antennas, one can construct $N_{\text{ant}}(N_{\text{ant}}+1)/2$ unique pairs of antennas (including autocorrelations). Each of these baselines produces a spectrum containing N_{freq} frequency channels every t_{corr} s. A spectrum is produced for each instrumental polarization element, and combined to generate N_{pol} spectra. For a given observation of length T_{obs}, the number of snapshots generated in time is $N_{\text{times}} = T_{\text{obs}}/t_{\text{corr}}$, so the total number of spectra is $N_{\text{times}}N_{\text{pol}}N_{\text{ant}}(N_{\text{ant}}+1)/2$. Visibilities are typically stored as single-precision floating-point numbers for both the real and imaginary components. Thus, the total volume of data V produced is

$$V = 4N_{\text{ant}}(N_{\text{ant}}+1)N_{\text{freq}}N_{\text{pol}}N_{\text{times}} \text{ bytes} \quad (1)$$

When fully constructed, the Hydrogen Epoch of Reionization Array (HERA; https://reionization.org) [1] will consist of 350 antennas, produce spectra with 6144 frequency channels, generate four instrumental polarizations, and have a correlation sampling rate of 2 s. Thus, for a typical observation of 12 h, this leads to a raw data volume of nearly 260 TB. The actual data volume produced by HERA is expected to be smaller than this by several factors, primarily by effectively averaging data longer for shorter baselines using techniques proposed in the literature [2]. Nevertheless, the nightly volume is projected to be over 60 TB. Obviously, analyzing, calibrating, and cataloging such a large volume of data manually each night would be difficult, and storing the raw data indefinitely would require a tremendous amount of storage.

To accommodate the large volume of data, the HERA collaboration has developed a real-time processing (RTP) system, designed to keep pace with data-acquisition rates and reduce the data to a reasonable volume in a short amount of time. The goal is to provide data-quality metrics, calibration solutions, and visualizations of data in near-real time, and subsequently reduce the volume of data to a quantity that is acceptable for long-term storage. Once the data have been stored on site, we also want to transfer the files to remote processing clusters, which can make the data widely available to the broader collaboration. Throughout all of these operations, we also desire to have a robust automatic system that is not sensitive to poor-quality data and provides monitoring tools for overall telescope operation as well as metrics about individual array elements. The science goals of HERA—the detection and characterization of the first stars and galaxies—are exquisitely sensitive to sources of systematic errors, and so having real-time feedback on the performance of the telescope is crucial. These separate but interrelated tasks are accomplished primarily through the work of two packages: the hera_opm package (https://github.com/HERA-Team/hera_opm), responsible for running analysis on data, and the librarian (https://github.com/HERA-Team/librarian), responsible for storing data and transferring it to other sites. In the following discussion, we provide a brief overview of the RTP system and showcase some of its features. In Section 2 we discuss the processing performed by the hera_opm package. In Section 3 we describe the librarian. Finally, in Section 4 we provide

Manuscript received 28 August 2020.

Paul La Plante and Joshua S. Dillon are with the University of California, Berkeley, CA 94720, USA; e-mail: plaplant@berkeley.edu, jsdillon@berkeley.edu.

Peter K. G. Williams is with the Harvard-Smithsonian Center for Astrophysics, Cambridge, MA 02138, USA; and the American Astronomical Society, Washington, DC 20006, USA; e-mail: pwilliams@cfa.harvard.edu.

some general discussion and mention planned future upgrades. All of the software and tools used throughout this manuscript are free and open-source software licensed under the BSD 2-clause License, with the code bases freely available on GitHub. The packages are designed to be flexible enough to accommodate the needs of other telescopes, and more generally any data-reduction pipelines.

2. Processing

A single visibility data file for HERA contains data corresponding to all baselines, frequencies, and polarizations, for several consecutive times. The data are saved in a custom HDF5-based format known as UVH5, and are generally interacted with using the pyuvdata package (https://github.com/RadioAstronomySoftwareGroup/pyuvdata) [3]. In general, we wish to perform the same fixed set of analysis routines on each data file. Furthermore, there are several steps that are effectively prerequisite for a subsequent step. For example, an initial calibration solution is determined using a redundant calibration technique [4], after which a sky-referenced calibration is performed [5]. The redundant calibration must be performed before the absolute calibration, though only for each data file specifically. On the other hand, there are some analysis techniques, such as smoothing of calibration solutions in time and frequency, that require an earlier step to be completed for all files. Finally, there are some steps that must operate on several files consecutive in time as a single unit, such as identifying data contaminated by radio frequency interference (RFI), and stride through the data files operating on chunks. From a user perspective, the goal is to provide a list of tasks to run for each file, enumerate any necessary prerequisites for the file (including ones adjacent in time), and describe a method for chunking data into steps. Once the workflow and input data files are provided, the user should be able to dispatch the workflow to appropriate compute resources automatically, without having to individually run the steps for each file and keep track of dependencies by hand. Such an approach helps automate the analysis process, and provides a reliable and reproducible way to perform complicated workflows on a large volume of data.

The primary means by which these goals are accomplished in HERA is through the hera_opm package, which provides the online processing module (OPM). There are several steps required for using this package. The user first defines a *configuration file*, which contains the workflow of steps to perform on each file. For each step, the user lists any prerequisites, time chunking, options, and specific arguments to pass to shell scripts. In addition, the user writes a series of *task scripts*, which carry out the actual analysis steps. These are generally shell scripts, which may in turn call other scripts or programs (Python or Perl scripts, compiled C analysis programs, etc.). For several steps in HERA, these tasks execute dynamically generated Jupyter Notebooks (https://jupyter.org/), which are automatically uploaded to GitHub for wider consumption. Some steps also include automatically making multifrequency synthesis images using CASA (https://casa.nrao.edu) [6], which provide valuable cross-checks of instrument functionality. (Data files are first converted from UVH5 to UVFITS using pyuvdata, at which point CASA can import them using the IMPORTUVFITS task.) The tools in hera_opm convert the configuration file and a list of input files into a series of wrapper scripts. In addition, a file is generated that is suitable for input to the makeflow program [7], part of the cctools package (https://github.com/cooperative-computing-lab/cctools). The makeflow file generated contains all of the rules necessary to execute the workflow as defined by the user in the configuration file. Furthermore, makeflow has the ability to either interface with local execution on a user's personal machine or deploy execution to a cluster resource manager. For on-site processing in HERA, we opt for the latter and make use of the Slurm scheduler (https://slurm.schedmd.com/) installed on the local processing cluster. However, makeflow allows for straightforward use of different schedulers, which increases portability. For example, HERA performs further postprocessing analysis at National Radio Astronomy Observatory (NRAO) facilities using TORQUE (https://adaptivecomputing.com/cherry-services/torque-resource-manager/), which is accomplished by changing a single line in the configuration file.

As part of on-site operations, a monitoring script waits for new data to be generated. Once data collection is complete, a matching makeflow input file and accompanying wrapper scripts are generated, which enables the launch of a new workflow. The workflow is automatically launched, and runs by dispatching jobs to the Slurm cluster on-site. As part of the workflow, data products such as calibration solutions and flags corresponding to poor-quality data are generated. Some of these products, as well as some of the raw data, are saved as part of our on-site cluster storage management system and transported to remote processing clusters. These actions are handled by the librarian package, which we turn to in Section 3.

3. Data Storage

In addition to processing data on-site, we want to be able to store raw data, calibration solutions, data-quality metrics, and derived data products indefinitely. Although simple indexing and cataloging schemes may be able to keep pace for several thousand files and straightforward organizational structures, the problem quickly becomes intractable when applied to millions of files across hundreds of nights of observation, each with unique qualities that may affect the overall results in different ways. For HERA, the 60 TB of data each night will be divided into roughly 50,000 files, each of which will have unique products and associated metadata. To accommodate such a tremendous proliferation of files,

we have developed the librarian package, which handles tracking data files and transporting them to remote clusters. The underlying architecture also scales well with the total number of files to track, so that the data can be added and accessed without significant slowdown even as the number of files becomes large. We now briefly describe some of the main features of this package and how its functionality enables the RTP system to provide data and data products to users working at remote clusters.

The librarian consists of two components: a *librarian server*, which catalogs and stores data files, and the *librarian client*, which supports access in various ways. The server runs a Tornado-based web server (https://github.com/tornadoweb/tornado) and makes use of a backing PostgreSQL relational database (https://www.postgresql.org/). The server also requires access to large storage drives (typically RAID arrays) for saving data, called *stores*; however, these storage drives need not be on the same machine as the librarian server. Additionally, the librarian can make use of multiple stores across different machines. As part of the internal bookkeeping, the librarian keeps track of which store a file is saved to, so that it is able to access it again in the future. The librarian client supports web-based access to the Tornado server as well as a command-line interface for programmatic use of librarian utilities. There is also a Python-based application programming interface for incorporating librarian access into analysis scripts. The client access, either web-based or command-line-based, provides search capabilities for files known to the librarian server, as well as the ability to copy data to a local storage location or remote machine.

When a file is added to the librarian, the server reads some of the metadata to determine which observation session the file corresponds to. The server also performs file-level checking, such as computing the file's checksum and inferring the file type. These bits of metadata are saved along with the actual data and copied to a store. The librarian automatically groups multiple observations that are contiguous in time into a single observing session, which is treated as a "night" of data. When searching for data, users can specify the desired file name itself (or include wild-card characters to match multiple files), file type, individual observation identifier, or observation-session identifier. The librarian also supports searching for combinations of these identifiers, allowing for straightforward access based on a variety of features.

The librarian also supports communication with other librarian servers installed at other locations. For example, librarian servers run both on-site and at the NRAO processing facility that HERA uses. These librarian servers are loosely coupled, and are able to transfer metadata of individual files between each other. The general problem of syncing files is rendered tractable by requiring that files have globally unique names and that both these names and the file contents be immutable. In practice, the unique-name constraint is easy to adhere to by using the Julian date of observation as part of the file name. The inter-librarian communication also supports automatic transfer of files meeting certain criteria between servers. For example, all new data can be transferred from the on-site librarian to the NRAO server, or just files corresponding to calibration solutions, or only site-specific maintenance data, etc. The transfer of files between librarian servers supports both simple transferring using rsync as well as more sophisticated transfer protocols such as Globus (https://www.globus.org/).

As an additional feature, the librarian installation at NRAO features what is known as "staging" data from the long-term storage nodes to short-term scratch space for user interaction. The user performs a search for specific files, after which the user requests that the data be transferred to the user's personal scratch space. The librarian automatically fetches the data from the appropriate NRAO store and copies it to the user's scratch space, after which the librarian notifies the user that the data were successfully transferred (or of any errors that may have occurred). Accordingly, the user need not worry about manually copying data or about implementation details, such as files being saved across different local stores. In many cases, these high-level access patterns facilitate user access to data without requiring a cumbersome and error-prone series of command-line calls, allowing users to focus on analyzing data rather than hampering their access to it.

4. Discussion

Successful operation of the RTP system on-site requires the coherent and interconnected operation of both the processing and data-storage services. There are also a series of monitoring and control utilities that help ensure continued operation of all of these services, though we defer detailed discussion of these processes to future work. In this section, we briefly touch on the way in which the processing and data-management services interact with the raw data on-site, and explore future planned improvements. We also highlight ways in which these tools may be adapted to operation of other telescopes or data-reduction pipelines.

Figure 1 shows a schematic of how the different processes interact on-site. The main head node of the cluster runs the librarian server process, as well as the monitoring service waiting for acquisition of new data, represented in the diagram as the "Raw Data Storage" location. The storage disk on this machine is mounted via NFS protocol to the head node as well as all of the compute nodes, which allows for simple access without needing to explicitly copy data to local storage. As part of the workflow that processes the raw data, several steps include uploading files to the librarian server running on-site. Currently, the data volumes for HERA are sufficiently modest that all raw data and derived products can be uploaded and transferred, though as the number of active antennas rapidly expands in the coming year, this will no longer be possible. When new data are uploaded to the librarian, they are automatically

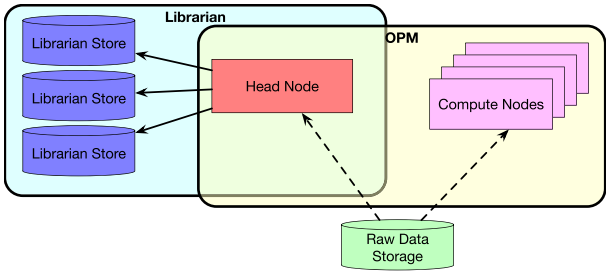

Figure 1. A schematic of how the RTP system operates on-site as part of the HERA processing cluster. Both the librarian and hera_opm systems make use of the head node, as well as auxiliary machines. See Section 4 of the text for additional discussion of how the systems interact.

transferred to the store with the most free space available, to avoid overfilling a single store. Transfer of uploaded data files to the NRAO librarian is handled automatically, and proceeds as data are available. At present, the system is able to keep up with processing requirements in real time, though as the data volume grows, additional testing and benchmarking will be required to ensure that the RTP remains real time and can keep pace with the larger computational load required.

In practice, the current HERA system records about 2 TB of data per night. The on-site system is able to perform basic quality checks of individual antennas, flag all data files for RFI, and perform an initial redundant calibration on about 10% of the data files for diagnostic purposes. The data are transferred in near-real time (within about a day) to the NRAO, where calibration on the remaining files is performed as well as imaging. Given that the amount of data currently recorded is roughly 2% of the total planned, testing and benchmarking of the system are required to ensure that the computational load can be satisfied. For instance, we have shown that we likely require multiple nodes to record the full data rate in real time and then ensure that there is no I/O bottleneck when accessing the data. Additionally, given the scaling of calibration algorithms, we may be limited to using a restricted set of baselines for redundant calibration to keep pace with the raw data-recording rate. To mitigate some of these issues, we have developed calibration routines and RFI-flagging algorithms that can make use of graphics processing units to significantly speed up computation. We plan to perform more robust tests as the data volume increases.

In the future, we plan to make use of other computational resources available to HERA beyond those which are located on-site. In South Africa, the Inter-University Institute for Data Intensive Astronomy (IDIA) hosts computational resources that are available to high-performance astronomy applications. HERA has computational resources available as part of the IDIA community, and plans to run some portion of its processing load on IDIA machines. Based on the size of the compute resources available at IDIA, we believe that we will be able to perform redundant calibration for the full data volume in near-real time, with sufficient accuracy to coherently average data at the same local sidereal time from different days. Nevertheless, future development and additional benchmarking will be required to leverage these resources efficiently and fold the different steps at different resources into a single, coherent workflow. Nevertheless, given the dramatic increase in data volume and associated computational requirements in the near future, HERA will need to make use of all available resources.

Finally, we would like to emphasize that these processing tools are freely available, and can be adapted for use as part of other astronomy applications and workflows. Although the hera_opm and librarian packages were initially developed to work in the context of HERA processing and data management, they are written in a sufficiently general manner to work in other contexts. For example, the librarian is currently being adapted for use in the Simons Observatory (https://simonsobservatory.org/), an experiment observing the cosmic microwave background. Similarly, the core functionality of the hera_opm package is not specialized to HERA data, and can be used generally for any software pipeline applied to a series of input files. The software powering these packages is free and open-source, is available on GitHub, and has an active team of developers behind it. It is our hope that other groups find them useful and can use them for their own data-processing needs. For questions and comments about the use of these systems, please reach out to us via email or GitHub.

5. References

1. D. R. DeBoer, A. R. Parsons, J. E. Aguirre, P. Alexander, Z. S. Ali, et al., "Hydrogen Epoch of Reionization Array (HERA)," *Publications of the Astronomical Society of the Pacific*, **129**, 974, April 2017, p. 045001.
2. S. J. Wijnholds, A. G. Willis, and S. Salvini, "Baseline-Dependent Averaging in Radio Interferometry," *Monthly Notices of the Royal Astronomical Society*, **476**, 2, May 2018, pp. 2029-2039.
3. B. J. Hazelton, D. C. Jacobs, J. C. Pober, and A. P. Beardsley, "pyuvdata: An Interface for Astronomical Interferometric Datasets in Python," *The Journal of Open Source Software*, **2**, 10, February 2017, p. 140.
4. J. S. Dillon, M. Lee, Z. S. Ali, A. R. Parsons, N. Orosz, et al., "Redundant-Baseline Calibration of the Hydrogen Epoch of Reionization Array," arXiv:2003.08399, November 2020.
5. N. S. Kern, J. S. Dillon, A. R. Parsons, C. L. Carilli, G. Bernardi, et al., "Absolute Calibration Strategies for the Hydrogen Epoch of Reionization Array and Their Impact on the 21 cm Power Spectrum," *The Astrophysical Journal*, **890**, 2, February 2020, p. 122.
6. J. P. McMullin, B. Waters, D. Schiebel, W. Young, and K. Golap, "CASA Architecture and Applications," Astronomical Data Analysis Software and Systems XVI, Tucson, AZ, October 2006, pp. 127-130.
7. M. Albrecht, P. Donnelly, P. Bui, and D. Thain, "Makeflow: A Portable Abstraction for Data Intensive Computing on Clusters, Clouds, and Grids," SWEET '12: Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, New York, NY, May 2012, pp. 1-13.