SQUARE KILOMETRE ARRAY : PROCESSING VOLUMINOUS MEERKAT DATA ON IRIS

Priyaa Thavasimani  
Faculty of Science and Engineering  
The University of Manchester  
Manchester, UK  
priyaa.thavasimani@manchester.ac.uk  

Anna Scaife  
Faculty of Science and Engineering  
The University of Manchester  
Manchester, UK  
anna.scaife@manchester.ac.uk  

June 1, 2021

ABSTRACT

Processing astronomical data often comes with huge challenges with regards to data management as well as data processing. MeerKAT telescope is one of the precursor telescopes of the World’s largest observatory Square Kilometre Array. So far, MeerKAT data was processed using the South African computing facility i.e. IDIA, and exploited to make ground-breaking discoveries. However, to process MeerKAT data on UK’s IRIS computing facility requires new implementation of the MeerKAT pipeline. This paper focuses on how to transfer MeerKAT data from the South African site to UK’s IRIS systems for processing. We discuss about our RapifXfer Data transfer framework for transferring the MeerKAT data from South Africa to the UK, and the MeerKAT job processing framework pertaining to the UK’s IRIS resources.

Keywords  Big Data · MeerKAT · Square Kilometre Array · Telescope · Data Analysis · Imaging

Introduction

The Square Kilometre Array (SKA) will be the world’s largest telescope, but once built it comes with its huge data challenges. The SKA instrument sites are in Australia and South Africa. This paper focuses on the South African SKA precursor MeerKAT, in particular on its data processing, calibration, and imaging using the UK IRIS resources. MeerKAT data are initially hosted by IDIA (Inter-University Institute for Data-Intensive Astronomy; [1]), which itself provides data storage and a data-intensive research cloud facility to service the MeerKAT science community in South Africa. Data transfers from IDIA onto IRIS resources are themselves a challenge for the terabyte-scale MeerKAT datasets, making traditional transfer methods like scp, sftp, or rsync prohibitively slow. In this paper we introduce our data transfer framework from IDIA to IRIS, combining the open source Globus transfer service [2] and gfal2 [3] to improve transfer speeds. This paper also discusses the pipeline implementation for processing the MeerKAT data on IRIS. The original processMeerKAT pipeline [4] was implemented for processing data on the ilifu (IDIA research cloud) SLURM cluster. However, to process the MeerKAT data on the IRIS systems requires a revised implementation of this pipeline.

Related Work

SKA and its Data Rate:

The Square Kilometre Array (SKA) is an international project to build the world’s largest radio telescope [5] with a collecting area of one million square metres, i.e. a square-kilometre. It is a global science project with requirements derived from thirteen high-priority science objectives [6]. The SKA Phase 1 will consist of two telescopes, SKA1-MID located in South Africa’s Karoo region, and SKA1-LOW in Western Australia’s Murchisonshire. It is estimated that the data rate between the antennas and the correlator will be 23 Tb/s whereas the data rate between the correlator and the HPC (provided by the Science Data Processor) is 14 Tb/s, which is equivalent to 12,000 PB/month. For comparison the world’s total IP traffic in 2011 was 27,000 PB/month [7]. Australian Square Kilometre Array Pathfinder (ASKAP) is a low frequency (700-1800 MHz) interferometer, which is capable of 200 Tb/day of raw data [8].
IRIS resources for SKA

UKRI (UK Research and Innovation) plays a leading role in many global science projects, including the Square Kilometre Array. Currently, the UK government has invested £100m in the construction of the SKA and the SKA Headquarters, as a core member of the project through the Science and Technology Facilities Council (STFC); [9]. IRIS is a coordinating body that provides digital research infrastructure by working with providers [10] to support STFC science projects. Worldwide LHC Computing Grid (WLCG) [11] is a global provider of computing infrastructure which is supported by associated national and international grids across the world. WCLG was originally established to tackle data challenges by projects including (ATLAS, CMS, LHCb, and ALICE). UK’s involvement in WLCG is called Grid for UK Particle Physics (GRIDPP) collaboration. The main objective of GridPP was to provide computing resources to process and manage Physics data coming from LHC. As one of the IRIS partners, GridPP [12] provides a number of services to perform large scale computing for research, originally created for LHC experiments and has been expanded for use by other research communities [13].

DiRAC (Distributed Infrastructure with Remote Agent Control) [12] is a pilot based framework which was developed as a workflow management (WMS) and data management system (DMS) [14], where WMS provides service to run JDL (Job Description Language) jobs on the grid and DMS provides store/retrieve access to large scale storage systems. An extensive survey was conducted on pilot job systems in the article [15]. Job Description Language (JDL) is a high-level, user-oriented language [16]. The JDL scripts provide specific information about the job itself which is necessary for the WMS to handle the request. DiRAC’s WMS facilitates the submission of a wide variety of jobs including single-core, multi-core, MPI, parametric tasks. The Data Management System (DMS) comprises 3 main components including Storage Element (SE), File Catalogues, and File Transfer Service [17].

SKA makes use of these services through IRIS for submitting jobs on the grid as well as managing files on storage systems through successful authentication of the user’s X.509 certificate. SKA has its dedicated high-mem Machines including 1.5 TB and 3 TB RAM machines at Manchester to run its jobs.

MeerKAT Data Rate

MeerKAT is the precursor instrument to SKA1-MID and will be integrated into Phase 1 of SKA. Consisting of 64 antennas in the Northern Cape of South Africa, it is estimated that the MeerKAT telescope will have a total Science Data Processor (SDP)’s input data rate of about 4 terabits per second (4000 gigabits per second) [18].

Data Transfer Protocols

SSH-based file transfer commands including SCP, Rsync, and SFTP are all preferred options for transfers within the same network. SCP (Secure Copy Protocol) can be used to transfer data from one device to another over a network. SFTP (Secure File Transfer Protocol) is like FTP, which provides an interactive file transfer service where all the operations are carried over an encrypted transport. As the name suggests, Rsync is more appropriate for transferring and synchronising files with complex hierarchies. It facilitates checking the timestamps and size of files. If the data size is as small as 2 GB, the choice of the transfer protocol is not important. However, for very large datasets, like those from MeerKAT or the SKA, the choice of transfer protocol plays a crucial role.

Globus Connect Personal is an open-source GridFTP service that provides secure and reliable research data management. Globus offers services including data transfer between systems, file sharing with others through authentication and verification of email addresses, application development, and providing gateways that facilitate automation of workflows.

Scientific Reproducibility

Scientific reproducibility is achieved by the virtualisation of the computation environment through the use of Workflow Management Systems and container virtualisation. Docker and Singularity are the most commonly used container-based technologies, however, complications can arise when using Docker on shared high-performance systems because the container and hence the user needs to have root access, which is typically not possible due to security concerns. Singularity is crafted to support HPC environments, and the authors [19] discuss the advantages of singularity over the other container technologies. The native support for MPI/Parallel applications when using a singularity container is also an added advantage for the execution of complex scientific workflows.

MeerKAT pipeline or workflow is separated into individual scripts or workflow blocks which helps in scientific reproducibility.

The CERN Virtual Machine File System (CVMFS) [20] not only facilitates data distribution, content de-duplication, and software preservation, but also reproducible, ready-to-use environments. CVMFS can be configured to continuously check container images for updates, if there is an updated version, it will be reflected in CVMFS. The advantages of using CVMFS for distributed computing are briefly discussed in the paper [21].
To facilitate further reproducibility, we intend to use Common Workflow Language \[22\], which helps to merge workflows written in different languages and also CWL prov helps track of execution provenance. Execution provenance helps to understand the results of reproduced workflows. Extracting and understanding the provenance for tracking the changes in the reproduced workflows is widely discussed and demonstrated in the papers \[23\] and \[24\].

**Fundamental steps in Radio Astronomy**

Three fundamental steps in radio astronomy data processing include data preparation, data calibration, and imaging. The data from the telescope are not always clean and hence it is essential to remove contaminated data samples that are not of scientific benefit.

During the data preparation step, data that are corrupted by interference from mobile phones, satellite signals, or similar are removed.

The main goal of calibration is to remove any instrumental intervention that might have affected the measurements.

The imaging step is the final step that produces science-ready data products.

**CASA Library**

The Common Astronomy Software Applications (CASA) software package was developed to meet the data processing and analysis needs of the ALMA and EVLA telescopes \[25\]. In addition to more common functionalities such as data flagging, calibration, and imaging, CASA also provides functionalities for interactive GUI-enabled operations such as a viewer, plotter, logger, and table browser. CASA 6 is the most recent version of the package and offers a more modular approach.

The Cube Analysis and Rendering Tool for Astronomy (CARTA) \[26\] is an alternative to the CASA viewer and provides a more advanced functionality including visualization of large cubes through tile rendering and optimized memory use.

**Contributions**

1. We propose a novel framework for transferring the large (> 1 TB) MeerKAT datasets from the IDIA Ilifu cloud to IRIS grid storage.

2. We introduce a framework for MeerKAT data processing using the available SKA-IRIS computing resources.

3. We outline how the MeerKAT data can be processed efficiently on IRIS resources.

**’RapidXfer’ - MeerKAT Data Transfer Framework**

The primary data products \[27\] intended to be generated by the SKA’s Science Data Processors (SDPs) will not necessarily be suitable for immediate scientific analysis, and during instrument commissioning it is also envisaged that the SDP calibration and imaging functionality will need to be replicated at external computing facilities. For these purposes, it will be necessary to send data sets to a network of worldwide SKA Regional Centres (SRCs) for further processing and imaging. End users in the astronomy community will rely on SRCs rather than the telescope sites for their data storage, processing and access \[28\]. Consequently, SRCs form an intrinsic part of SKA operations \[29\].

Here we focus on how MeerKAT data can be transferred to IRIS resources for processing. Within South Africa, data from the SKA-SA (SKA South Africa) CHPC (Centre for High Performance Computing) is transferred to the IDIA Regional Science Data centres using a dedicated Globus endpoint \[28\]. In order to process data on the IRIS infrastructure, it is essential to transfer the data onto grid storage and register it within the DiRAC filecatalogue, where each data file is given a dedicated logical file name.

To do this the data from IDIA are transferred to a local grid UI machine at the Manchester Tier 2 using a dedicated Globus endpoint.

A sample transfer of 1.3 TB data from IDIA to Manchester UI machine had 14 data rate of MB/s. The Grid File Access Library, gfal, is then used to transfer data from the Manchester UI node to physical storage where each file is given a “Physical File Name”.

To copy a 1.3 TB MeerKAT data set from Manchester UI to LFN using gfal-copy took approximately 45 minutes. As a whole, the transfer time from IDIA to IRIS is reduced to half compared to using traditional transfers like SCP (SCP took nearly 3 days to transfer the 1.3 TB dataset) or direct addition of data from IDIA onto grid storage using DiRAC’s inbuilt function dirac-dms-add-file. DiRAC breaks due to time-out and security issues. This process \[30\] is illustrated in Figure 1 and we propose that these transfer timings can be used as a base reference for the development of a full transfer framework for future SRC operations.
Once the data is copied to grid storage using `gfal-copy`, the data can be registered in place using the inbuilt DiRAC command `register` which creates an entry for the dataset within the DiRAC File Catalogue. This registration then allows one to make as many replicas as needed depending on the preferred Storage Element. Each file registered will be assigned a Logical File Name, which then can be used as an input data location in DiRAC processing jobs. The Logical File Catalogue (LFC) contains the mappings of Logical File Names (LFN) to physical files, along with any associated replicas on the grid.

![Figure 1: Data Transfer Framework for MeerKAT](image1)

![Figure 2: MeerKAT Job Processing Framework on IRIS](image2)
MeerKAT Job Processing on IRIS

Once data is registered using our framework, the Logical File Name of the data stored on File Catalogue can be used in JDL (Job Description Language) scripts to submit jobs with the specific Input Data. Two types of job requests can be made through JDL: simple jobs and DAG which represents a Direct Acyclic Graph of dependent jobs. Figure 2 shows how the data is processed on IRIS. The following explains each component which core aspects of our MeerKAT job processing framework.

1 INPUT SANDBOX: Input Sandbox is a list of filenames that are in local system. These files are transferred from the local system to the WMS node and then to the Worker Node for execution. With respect to the MeerKAT data execution, we include “processMeerKAT.sh” and the configuration file “config.txt” in the Input Sandbox. The “config.txt” includes all the configuration needed to run the pipeline including where is measurement data set i.e. MS, column number in the input MS to identify the essential fields and the self calibration values. These “myconfig.txt” will be updated along the way during the execution of independent MeerKAT workflow blocks. The “processMeerKAT.sh” is a shell script that essentially includes singularity calls to the individual python scripts of the “MeerKAT Pipeline”.

2 DFC: DFC represents “DiRAC File Catalogue”, which actually holds the input Measurement Set (MS) dataset, which is passed as an input to the “MeerKAT Pipeline”. During the execution of each step of the “MeerKAT pipeline”, a lot of intermediate files are generated including the “MMS”, “Tables”, “More Tables”, Target MMS, “Calibrator 1 MMS”, “Calibrator 2 MMS”, and “images”. Though all of the intermediate datasets are not required, the “Target MMS”, “Calibration 1 MMS”, “Calibration 2 MMS” from “split.py”, and “images” from “quick_tclean.py” are the required Output Data, which will be permanently stored on the “DFC”.

3 GRID NODE(S): JDL script with specified input and the executable file to be executed on GRID nodes. The node on which the script to be run can be specified in the “Tag” field, with the specification of the maximum number of processors based on the need. For the MeerKAT pipeline, the data preparation and calibration scripts need less memory and so can be executed on low-memory nodes, whereas the imaging scripts consumes high-memory and so it is worth to execute those scripts on high-memory machines.

4 MeerKAT Pipeline: MeerKAT pipeline is originally developed by IDIA pipelines Team. It is then extended to adapt to the IRIS resources in the UK. “validate_input” script is used to identify and validate the fields in Measurement Set (MS). “partition.py” is to partition the input Measurement Set (MS) to Multi Measurement Set(MMS) which facilitates parallel processing. The reference antenna calculation is done in “calc_refant.py”. “flag_round_1.py” is pre-calibration flagging, which helps to get rid of the worst RFIs (Radio Frequency Interferences). “setjy.py”, “xx_yy_solve.py”, and “xx_yy_apply.py” perform parallel hand calibration. “flag_round.py” performs post-calibration flagging with tighter RFI thresholds. “setjy.py”, “xy_yx_apply.py”, and “xy_yx_solve.py” perform cross hand calibration. In “split.py”, each field in the Multi Measurement Set (MMS) is split out and optionally averaged in time and frequency. “quick_tclean.py” is the imaging step that uses CASA’s tclean functionality to create quick images. This imaging step is high memory intensive and should require high memory Grid Nodes to be run. Finally, the “plot_solutions.py” plots the solutions that are applied

5 CVMFS: A singularity container “meerkat.simg” is built which packages all the necessary CASA libraries, and other supporting libraries for running the pipeline. CERN Virtual Machine File System (CVMFS) is a distributed file system designed to provide scalable and reliable file distribution service. To enhance reproducibility, the container “meerkat.simg” is stored on our centralised system called “CVMFS”. The singularity containers are stored on CVMFS as sandbox, facilitating the addition of newer libraries without having to rebuild again.

6 OUTPUT SANDBOX: Output Sandbox is a list of filenames that are generated by the job, which the user wants to retrieve locally after the successful job execution. With respect to the “MeerKAT processing”, the expected outputs are “PLOTS” generated out of “plot_solutions.py”, updated configuration file “config.txt”, “StdOut”, and “StdErr”. In this “StdOut” and “StdErr” are helpful in understanding the Standard Output and Standard Error of the submitted job on the GRID.

Execution of MeerKAT pipeline on IRIS

Figure 3 shows the memory consumption of the MeerKAT pipeline on a sample dataset approximately of size 12 GB. The entire pipeline was executed on our high memory modes at Manchester. The figure illustrates that the data preprocessing and calibration took less memory compared to imaging. There is a sudden rise in memory usage during the imaging step. The data preparation and calibration steps do not need high memory whereas the imaging step required high memory. To utilise the available resources efficiently, the pipeline is divided into 3 sub tasks where the data
Figure 3: Memory profiling for MeerKAT data processing

preparation, calibration, plotting solutions were run on low memory node and the imaging task i.e. “quick-telean.py” was run on high-memory node and the memory usage by the “data preparation and calibration”, “imaging” and “plotting solutions” are presented in Figures 4a, 4b and 4c respectively.

Figure 4: Memory profiling for MeerKAT pipeline

Plotting Solutions on IRIS

In the MeerKAT pipeline represented in Figure 2, the plot_solutions.py needs an in-built display to plots the solutions. However, unlike IDIA machines, IRIS machines do not have display hardware or input devices. Xvfb is an X server that is used to provide display functionality to the machines that have no display hardware or physical devices by emulating dumb frame buffer using the virtual memory. Thus the meerkat singularity container which was used to run the MeerKAT pipeline on IDIA machines can not be used and so the Xvfb software environment is added to the singularity containers to run specifically on IRIS machines.
MPI Processing in Imaging step

Some of the CASA tasks can be executed in parallel. Different tasks follow different parallelisation mechanisms. For example, Flagging is parallelised over scans whereas imaging is parallelised over the channel. CASA achieves parallelisation by Message Passing Interface (MPI), which is a message-passing industrial standard that facilitates the execution of parallel programs on HPC platforms. In the MeerKAT pipeline, the parallelisation of imaging is achieved for imaging task tclean i.e. “quick_tclean.py” using the command “mpicasa”.

```
mpicasa -n <number_of_processes> path_to_casa/casa
```

Parametric Jobs and Memory profiling for MeerKAT XMMLSS Dataset 1.3 TB

Large datasets of MeerKAT varies from 1 TB - 1.5 TB and so it is essential to optimise the MeerKAT processing further and the pipeline can be only run on very high memory machines. For the processing of the 1.3 TB MeerKAT XMMLSS dataset, we used our high memory machine i.e. 3 TB machine at Manchester with 32 processors. Figure 5 refers to the splitting of the MeerKAT pipeline into 3 subworkflows: i.e. Pre calibration pre-cal, run calibration run-cal and self calibration slf-cal for efficient processing of MeerKAT pipeline for larger datasets that are greater than 1 TB. The partition.py in the pre-cal partitions the voluminous 1 TB Measurement dataset into 12 partitions (i.e Multi Measurement Sets (MMS)) for each frequency range or spectral windows (i.e. 880.0 930.0MHz, 930.0 980.0MHz, 980.0 1030.0MHz, 1030.0 1080.0MHz, 1080.0 1130.0MHz, 1130.0 1180.0MHz, 1280.0 1330.0MHz, 1330.0 1380.0MHz, 1380.0 1430.0MHz, 1430.0 1480.0MHz, 1480.0 1530.0MHz, 1630.0 1680.0MHz). The step run calibration can be optimally executed by running the different frequency ranges in parallel, thus reducing the execution time even further. DiRAC [17] provides the provision of submitting a set of jobs by specifying parameters for each job. We take the advantage of the DiRAC’s parametric job functionality for running the run-cal step of the
MeerKAT pipeline thus one parameteric job submitted for each spectral window i.e. the frequency range. The volume of data per spectral window will depend on RFI flagging. Therefore the maximum memory usage for the spectral window varies. The range for XMMLSS between 5.5 GB - 12 GB, one of the frequency range’s memory profiling is represented in Figure 6b. The memory usage and execution time for the Pre-calibration, Run-calibration (for one spectral window), Concatenation tasks are depicted in Figures 6a, 6b and 6c respectively. Partitioning during the Pre-calibration stage has consumed maximum memory up to 100MB Virtual Memory than the other steps, which includes validating inputs and uploading input data. The Concatenation step concatenates, compresses, and upload the images and plots into Output Sandbox and so most of the tasks in this step are data IO related which required less memory.

![Figure 6: Memory profiling for MeerKAT XMMLSS Dataset](a) Pre calibration, (b) Run Calibration and (c) Concatenation](b)

**Conclusion and Future Work**

In this paper, we have discussed about Square Kilometre Array’s precursor telescopes: ASKAP and MeerKAT telescope and their data rates. We outlined the list of available IRIS resources to process the highly voluminous MeerKAT data. Discussion on various file transfer protocols was presented. Transferring of MeerKAT data is itself a huge challenge and we explained our data transfer framework for transferring MeerKAT from the South African IDIA site to UK site IRIS. We have also shown the importance of scientific reproducibility for complex scientific workflows like “MeerKAT”. A Brief discussion on the fundamental steps in Astronomy, “CASA” Library, and its functionalities was presented. “MeerKAT Job Processing Framework on IRIS” was illustrated in detail. We have also demonstrated how the IRIS resources can be efficiently used by splitting the workflow into ‘low memory-intensive’ and ‘high-memory intensive’ sub-workflows. Furthermore, We have illustrated the efficient processing of a voluminous XMMLSS dataset of approximately 1.3 TB by parametric jobs for each frequency range. Our future work will focus on MPI processing for the partitioning and imaging steps to reduce the execution time further. The pre-cal and slf-cal steps for the XMMLSS 1.3 TB dataset could be fine-tuned further for efficient use of high-memory resources.

**References**

[1] Inter-university institute for data intensive astronomy. https://www.idia.ac.za/about-idia/, 2015.

[2] Globus. Data transfer with globus. https://www.globus.org/data-transfer/.

[3] Andrey Krysanov, Alejandro Alvarez Ayllon, and Oliver Keeble. Fts3 / webfts ,Ã­ a powerful file transfer service for scientific communities. *Procedia Computer Science*, 66:670 – 678, 2015. 4th International Young Scientist Conference on Computational Science.

[4] Calibration in processmeerkat. https://idia-pipelines.github.io/docs/processMeerKAT/calibration-in-processmeerkat/.

[5] SKA. The ska project. https://www.skatelescope.org/the-ska-project/.

[6] P. E. Dewdney, P. J. Hall, R. T. Schilizzi, and T. J. L. W. Lazio. The square kilometre array. *Proceedings of the IEEE*, 97(8):1482–1496, 2009.

[7] Jodrell Bank Centre for Astrophysics. The square kilometre array (ska). http://www.jodrellbank.manchester.ac.uk/research/research-centres/ska-project/.
[8] C. M. Trott. Impact of station size on calibration of ska-low. In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5659–5663, 2015.

[9] Science and Technology Facilities Council. Square kilometre array (ska). https://stfc.ukri.org/research/astronomy-space-science/astronomy-space-science-programme/ska/, 2019.

[10] STFC-IRIS. Cutting edge science needs cutting edge digital infrastructure. https://www.iris.ac.uk/.

[11] Worldwide LHC Computing Grid. Worldwide lhc computing grid. "https://wlcg-public.web.cern.ch/".

[12] The GridPP Collaboration 2020. Gridpp: Distributed computing for data-intensive research. https://www.gridpp.ac.uk/, 2020.

[13] D Bauer, D Colling, R Currie, S Fayer, A Huffman, J Martyniak, D Rand, and A Richards. The GridPP DIRAC project - DIRAC for non-LHC communities. Journal of Physics: Conference Series, 664(6):062036, Dec 2015.

[14] David Britton, A.J. Cass, P.E.L. Clarke, Jeremy Coles, David Colling, A.T. Doyle, N.I. Geddes, John Gordon, R.W.L. Jones, David Kelsey, S.L. Lloyd, R.P. Middleton, G. Patrick, Robert Sansum, and Sarah Pearce. Gridpp: The uk grid for particle physics. Philosophical transactions. Series A, Mathematical, physical, and engineering sciences, 367:2447–57, 07 2009.

[15] Matteo Turilli, Mark Santcroos, and Shantenu Jha. A comprehensive perspective on pilot-job systems. ACM Comput. Surv., 51(2), April 2018.

[16] F. Pacini and A. Maraschini. Job description language (jdl) attributes specification. Technical report, EGEE Consortium, 2006.

[17] A. Tsaregorodtsev, V. Garonne, and I. Stokes-Rees. Dirac: a scalable lightweight architecture for high throughput computing. In Fifth IEEE/ACM International Workshop on Grid Computing, pages 19–25, 2004.

[18] SARAO. Breakthrough listen to incorporate the meerkat array in its existing search for extraterrestrial signals and technosignatures. https://www.sarao.ac.za/media-releases/breakthrough-listen-to-incorporate-the-meerkat-array-in-its-existing-search-for-extraterrestrial-signals-and-technosignatures/, 2018.

[19] Gregory M. Kurtzer, Vanessa V. Sochat, and Michael Bauer. Singularity: Scientific containers for mobility of compute. PLoS ONE, 12, 2017.

[20] CernVM Team. Cernvm-fs documentation. https://buildmedia.readthedocs.org/media/pdf/cvmfs/latest/cvmfs.pdf

[21] M. Alhowaidi, B. Ramamurthy, B. Bockelman, and D. Swanson. The case for using content-centric networking for distributing high-energy physics software. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pages 2571–2572, 2017.

[22] Common Workflow Language. Common workflow language. https://www.commonwl.org/.

[23] Sehrish Kanwal, Farah Zaib Khan, Andrew Lonie, and Richard Sinnott. Investigating reproducibility and tracking provenance – a genomic workflow case study. BMC Bioinformatics, 18, 07 2017.

[24] P. Thavasimani, J. Cala, and P. Missier. Why-diff: Exploiting provenance to understand outcome differences from non-identical reproduced workflows. IEEE Access, 7:34973–34990, 2019.

[25] S. Jaeger. The Common Astronomy Software Application (CASA). In R. W. Argyle, P. S. Bunclark, and J. R. Lewis, editors, Astronomical Data Analysis Software and Systems XVII, volume 394 of Astronomical Society of the Pacific Conference Series, page 623, August 2008.

[26] B. Emonts, R. Raba, G. Moellenbrock, S. Castro, C. E. Garcia-Dabo, J. Donovan Meyer, P. Ford, R. Garwood, K. Golap, J. Gonzalez, W. Kawasaki, A. McNichols, D. Mehriinger, R. Miel, F. Montesino Pouzols, T. Nakazato, S. Nishie, J. Ott, D. Petry, U. Rau, C. Reynolds, D. Schiebel, N. Schweighart, J. W. Steeb, V. Suoranta, T. Tsutsumi, A. Wells, S. Bhatnagar, P. Jagannathan, Joe Masters, and K. S. Wang. The casa software for radio astronomy: status update from adass 2019, 2019.

[27] A. M. M. Scaife, R. Joshi, T. M. Cantwell, A. O. Clarke, and A. Project. Compute and storage for ska regional centres. In 2019 URSI Asia-Pacific Radio Science Conference (AP-RASC), pages 1–1, 2019.

[28] David Aikema, Brad Frank, Robert Simmonds, Yan Grange, Susana Sánchez-Expósito, Séverin Gaudet, and Sharon Goliath. Data Delivery Architecture for MeerKAT and the SKA. In Pascal Ballester, Jorge Ibsen, and Keith Solar, Mauricio and Shortridge, editors, Astronomical Data Analysis Software and Systems XXVII, volume 522 of Astronomical Society of the Pacific Conference Series, page 331, April 2020.
[29] AENEAS. Ska regional centres. https://www.aeneas2020.eu/project/regional-centres/, 2020.

[30] Anna Scaife Priyaa Thavasimani. Rapidxfer - proposed data transfer framework for square kilometre array. https://indico.egi.eu/event/5000/contributions/14329/.

[31] CASA. Task tclean description. https://casa.nrao.edu/casadocs/latest/global-task-list/task_tclean/about.