Ref. Ares(2024)1579696 - 29/02/2024

RADIOBLOCKS

Project ID: 101093934

DASK-accelerated interface to analyse radio astronomy data formats

Deliverable:	D5.1
Lead beneficiary:	EPFL
Submission date:	29 February 2024
Dissemination level:	Public

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No *101093934*

Introduction

Upcoming astronomical instruments such as the Square Kilometre Array (SKA) will produce massive datasets that will need to be processed and analysed with novel imaging and machine learning techniques. However, contemporary astronomical data formats are not suited for Big Data analysis.

Data produced by contemporary interferometers and stored in the current Casacore Table Data System (CTDS)-based formats (van Diepen 2015) can be processed by CASA and other packages on single nodes (McMullin 2007). However, neither CASA nor the CTDS were designed with distributed computing and storage concerns in mind (Emonts 2018). In fact, many design choices made in the CTDS design, such as explicit file locking, mitigate the limitations of single-node POSIX file systems. The sheer quantity of data produced by SKA will require first-class distributed computing support. Therefore, a new generation of software and storage formats are required.

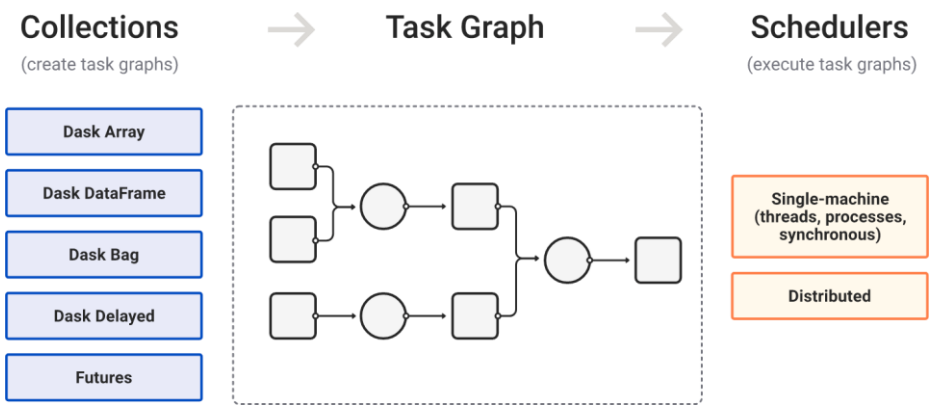
The contemporary CTDS-based data formats can present two challenges to distributed computing:

1. All data for a single column is stored in one file (the file may contain data for other columns). Access to the column data is controlled by a per-process file lock which severely degrades performance on distributed systems.

2. Corruption to parts of the Measurement Set during the reduction of an observation leads to the loss of the entire dataset, necessitating running the reduction from the beginning. This is undesirable given that reductions are computationally expensive and the need for reproducibility.

Thus new more modern storage format is required.

The PyData ecosystem provides a rich set of packages that enable Data Science Applications. In particular, Dask provides a parallel programming framework that scales computation with NumPy, SciPy, and Pandas, from single nodes up to 1000 node clusters. The following diagram taken from the Dask website illustrates the components of the chosen parallelism framework:



This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No *101093934*

Additionally, newer, generic distributed computing formats with interfaces to the PyData ecosystem have been developed, most notably Apache Arrow, Zarr and TileDB, which support Data Versioning and enable rollback of data. The PyData ecosystem therefore provides a compelling platform for developing distributed Data Science applications and formats for Radio Astronomy.

In order to support this paradigm, dask-ms is currently being developed to expose CASA Columnar Data as Datasets of Dask Arrays by Rhodes University. Dask-ms serves as a translation layer between Dask Arrays and advanced PyData formats. Both dask and XArray are first-class citizens in the PyData ecosystem, and XArray in particular is widely used in Earth Sciences. The advantage of exposing data in these packages is that they are easily interfaced with other PyData ecosystem packages. A number of new tools and packages have been developed using dask-ms, including pfb-clean, quartical and shadems.

A similar library is also under development by NRAO and SKAO called xradio. Xradio is a new prototype package of the next-generation CASA. While it does not currently support phased array instruments such as LOFAR, it provides a similar xarray interface to CASA columnar data.

Development in RADIOBLOCKS

A dask-enabled xarray interface to radio astronomy datasets will be the foundation of ongoing work in WP5 to refactor data analysis pipelines. We have reviewed the sdask-ms and xradio libraries, implemented initial dask data I/O prototypes using existing libraries, and benchmarked them against their standard implementations. The two libraries that we have developed dask-ms data reading are:

BIPP: Bluebild Imaging++, an HPC implementation of the Bluebild algorithm for image synthesis in radio astronomy. The Bluebild algorithm offers a novel approach to image synthesis, leveraging fPCA to decompose the sky image into distinct energy eigenimages.

AOFlagger: a flagger framework that implements several methods to deal with radio-frequency interference (RFI)

We have collected our examples using these libraries with dask-ms at: https://git.astron.nl/radioblocks/workpackage-5.1/dask-ms-examples. This repository contains three Jupyter Notebooks that:

1. use dask-ms to run aoflagger baseline-wise

2. use dask-ms to run aoflagger on blocks of baselines (all ANT2s available for each ANT1)

3. use dask-ms to preprocess the visibilities for BIPP.

The README.md contains some instructions for installing the necessary environment and running the examples, and the codes are annoted with explanatory comments.

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No *101093934*

We also provide a comparison between approaches 1. and 2. to show the effect of the grouping in dask-ms on overall performance. 1. is based on an intuitive but inefficient use of dask-ms that generates xarray dataset for each baseline. As each dataset is backed up by deferred calls to casacore MS reading functions this appears to be pretty inefficient. A better approach is taken in 2., where each dataset contains all baselines for a given antenna. Then AOFlagger is applied locally on each baseline, but the number of calls to casacore is hence driven by the number of stations and not the number of baselines.

The performance of methods 1. and 2. when run on a MWA dataset with 124 stations, 28 integration times and 768 frequency channels, using 20 physical CPU cores of a single node of EPFL GPU cluster Izar (Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz) are shown in Table 1 and Figure 1. In Table 1, we compare the timing of two methods, 1. using dask-ms to run aoflagger baseline-wise and 2. using dask-ms to run aoflagger on blocks of baselines. The blocking in Method 2 results in better performance. Scripts which implement both methods are available at https://git.astron.nl/radioblocks/workpackage-5.1/dask-ms-examples

	dask-ms_aoflagger_bsl	dask-ms_aoflagger
Dask cluster	2.3 s	4.1 s
xds_from_ms	145.4 s	3.6 s
dataset length	7750	124
Dask futures	24.8 s	0.5 s
Processing time	178.2 s	25.9 s
Fraction of time reading data	85.8 %	75.4 %
Total time	**350.7 s**	**34.1 s**

Table 1: The performance of dask-ms is highly sensitive to the data reading strategy.

Dask allows for trivial parallelization but the scaling is far from optimal, as shown in Figure 1. In-depth profiling is required to understand the limitations of the current setup, and the MS format itself may be a major blocker to parallelization, motivating alternative data formats for radio astronomy.

We plan to extend this example repository to include additional software and pipelines that would benefit from some parallelization with Dask.

RADIO BLOCKS

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No *101093934*

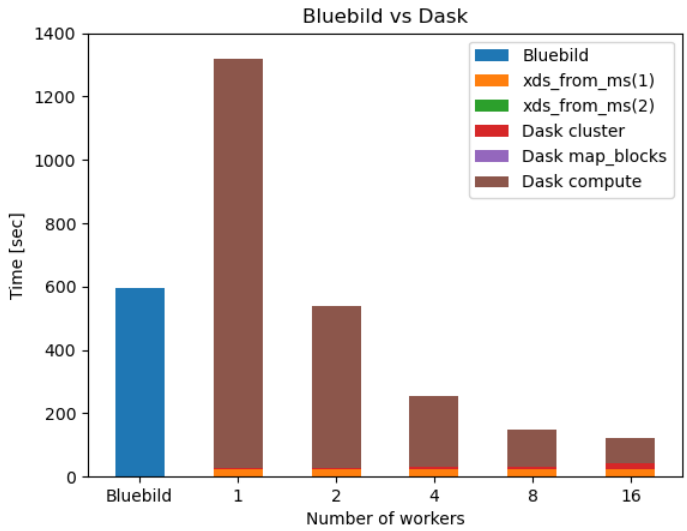


Figure 1: Performance of reading data with dask-ms vs using python casacore.

References

van Diepen, G.N.J., "Casacore Table Data System and its use in the MeasurementSet", in *Astronomy and Computing*, Volume 12, 2015.

Emonts, B., *CASA Memo 5: CASA Performance on Lustre: serial vs parallel and comparison with AIPS*. (2018) https://casa.nrao.edu/casadocs/casa-6.1.0/memo-series/casa-memos

McMullin, J. P., Waters, B., Schiebel, D., Young, W., and Golap, K., "CASA Architecture and Applications", in *Astronomical Data Analysis Software and Systems XVI*, 2007.

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No *101093934*