URSI GASS 2021, Rome, Italy, 28 August - 4 September 2021

ASKAP Processing

Maxim A. Voronkov* [(1)], Wasim Raja[(1)], Daniel Mitchell[(1)], Stephen M. Ord[(1)], Matthew T. Whiting[(1)], Mark H. Wieringa[(1)], Eric Bastholm[(2)], and Juan Carlos Guzman[(2)],
(1) CSIRO Astronomy & Space Science, PO Box 76, Epping, NSW 1710, Australia
(2) CSIRO Astronomy & Space Science, PO Box 1130, Bentley, WA 6102, Australia

The Australian Square Kilometre Array Pathfinder (ASKAP) is a wide field of view (30 square degrees) low frequency (700-1800 MHz) interferometer, which began pilot surveys in 2019. It routinely generates over 100 Tb/day of raw data and is capable of producing spectral cubes up to 70 Tb in size per observation (if all the data are used). This fact makes ASKAP an excellent test-bed for the Square Kilometre Array (SKA) project, which is facing even greater processing challenges.

To address the principal limitation of our inability to store the raw data long term, the original design was to produce science-ready images automatically in near real-time and dispose the raw data shortly after processing. Moreover, to minimize the disk I/O, which is often the dominant factor in the spectral line case, we aimed at processing using a single iteration over such data. This has implications for what is possible in terms of the processing algorithms and adopted approximations. The so-called continuum case deals with about a factor of 50 smaller data sizes (due to smaller number if spectral channels) and has limitations of a different kind: the image quality in the wide field of view regime (largely a research topic) and the desire for a greater code unification with the spectral line case to make the system less complex.

A specialised suite of software called ASKAPsoft (later evolved into YandaSoft) was effectively co-designed with the rest of the telescope, including the requirements for the HPC platform and data distribution architecture. For example, the number of ranks in the processing job was tied down to the physical number of independent data chunks. The main factors driving the development of the imaging part of the software were the available memory (both total and per core), memory bandwidth and disk I/O. In the course of several years we tried different approaches and converged on partitioning the data both by beam and by splitting the bandwidth. In addition, by the original design, the beam pattern is stabilised on the sky with the help of the unique 3-axis telescope mount reducing the need to deal with time-variable direction-dependent calibration effects during data reduction. Also, some commissioning effort has been directed towards understanding the effects normally absorbed into self-calibration solutions, which resulted in a better phase stability of the instrument and a more relaxed requirement for the cadence of calibration.

In reality, the perfect match between the data distribution patterns designed into ASKAPsoft, the science requirements and the dedicated Cray XC30 "Galaxy" supercomputer made available for data reduction did not eventuate. This necessitated further compromises and changes in software. Moreover, the diversity in the complexity of the fields as well as the science goals revealed by the early science phase made it difficult to continue to follow a "one size fits all" strategy with on-the-fly calibration assumed by the original design. To aid with this flexibility requirement, we developed a suite of scripts wrapping around the original ASKAPsoft tasks, effectively implementing the traditional offline processing model where disposing of the raw data is the operational choice rather than an inherent feature of the algorithm. While this process was essential for commissioning of the instrument as a whole, for refining the image quality and for progressive enhancement in the processing efficiency, it significantly (by more than a factor of 5) increased our disk I/O load and temporary storage requirements (due to multiple iterations over data). In addition, it led to workarounds adding complexity to our code as well as significant technical debt. An example would be the split-by-time strategy (on top of the beam and bandwidth split mentioned above) that had to be implemented to make the end-to-end processing feasible in near real-time (albeit offline). We review the current state of the data reduction software and the challenges encountered, both in terms of the code and algorithms as well as the reality of living in an HPC environment shared amongst commissioning scientists and day-to-day operations thus limiting the overall efficiency and turn-around times.