pawsey

Migrating Complex Workflows to the Exascale: Challenges for Radio Astronomy

Dr. Pascal Jahan Elahi
Pawsey Supercomputing Research Centre
CUG 24

Australian Government
NCRIS National Research Infrastructure for Australia
CSIRO
Curtin University
Murdoch University
Government of Western Australia
ECU
The University of Western Australia

Australian Square Kilometre Array Pathfinder

- Consists of 36 12-m dish antennas working together as one giant interferometer collecting radio signals from the sky.
  - 0.7-1.8 GHz frequency range, 0.3 GHz bandwidth split into 16384 channels
- Provides a wide field of view of the sky, roughly 30x larger than conventional radio telescope receiver.
- Located in a radio-quiet zone, "Inyarrimanha Ilgari Bundara" or Murchison Radio-astronomy Observatory (MRO), WA, the traditional lands of the Wajarri Yamaji people.

ASKAP Radio Telescope

ASKAP Compute

Interferometers measure *visibilities*, (i.e., amplitude & phase of cross-correlated signals between pairs). They require significant computational resources to be operational.

ASKAP has

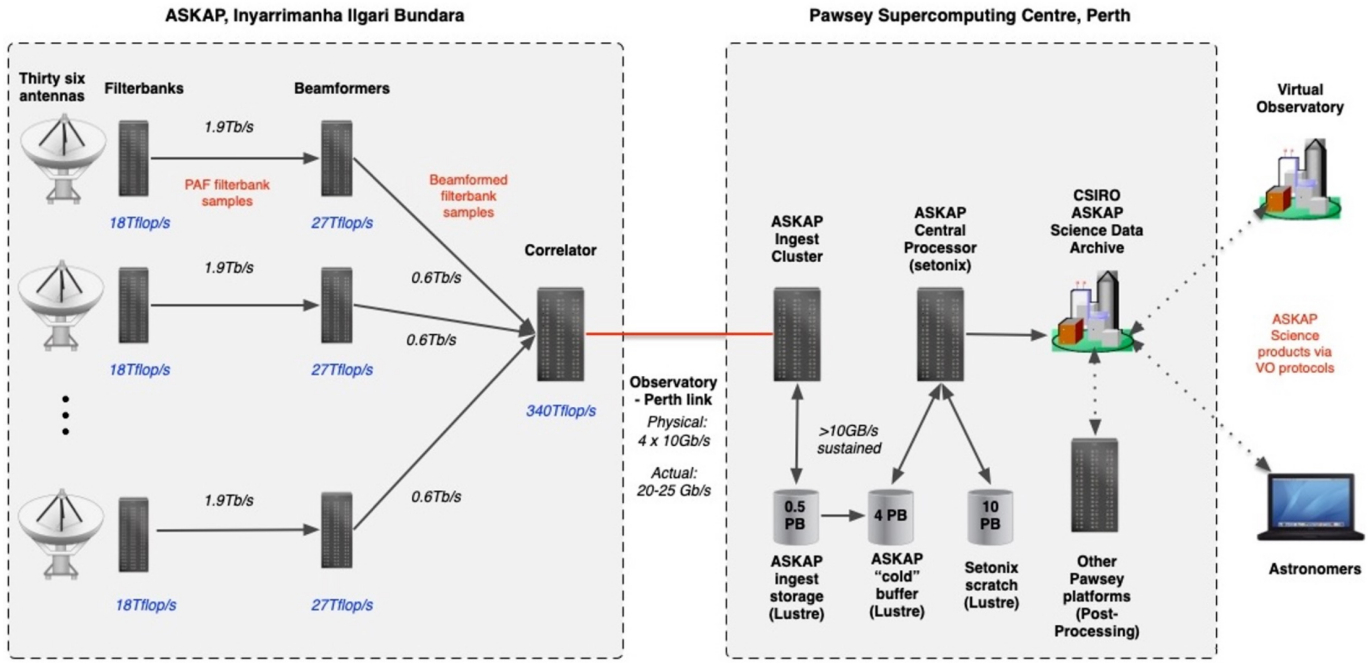
- onsite large deployment of FPGA's to do initial processing (correlation)
- ASKAP ingest cluster composed of 18 nodes for direct ingestion of data and scheduling of further science processing.
- 180 CPU-nodes of Setonix for science processing

ASKAP FPGA Correlator

ASKAP Radio Data Processing

Consists of several components

- `askap-cpmanager:` Communicates with TOS (Telescope Operating System) to start observation raw data ingest. Coordinates/orchestrates CP services
- `askap-tosconnector:` Listens for scheduling block state notifications, emits data movement and schedules processing.
- `askap-datamanager:` Manages two copy queues (high and standard priority) to move data
- **`askap-processingmanager:`** Launches initial calibration processing. Manages calibration observations which have been processed and the queue of science observations which require processing. Launches science processing (ASKAP SDP pipeline using ASKAPSOFT [C++, Python, MPI, OpenMP]) when the required matching calibration observations are available.
- `askap-arwen:` Diagnostic plots



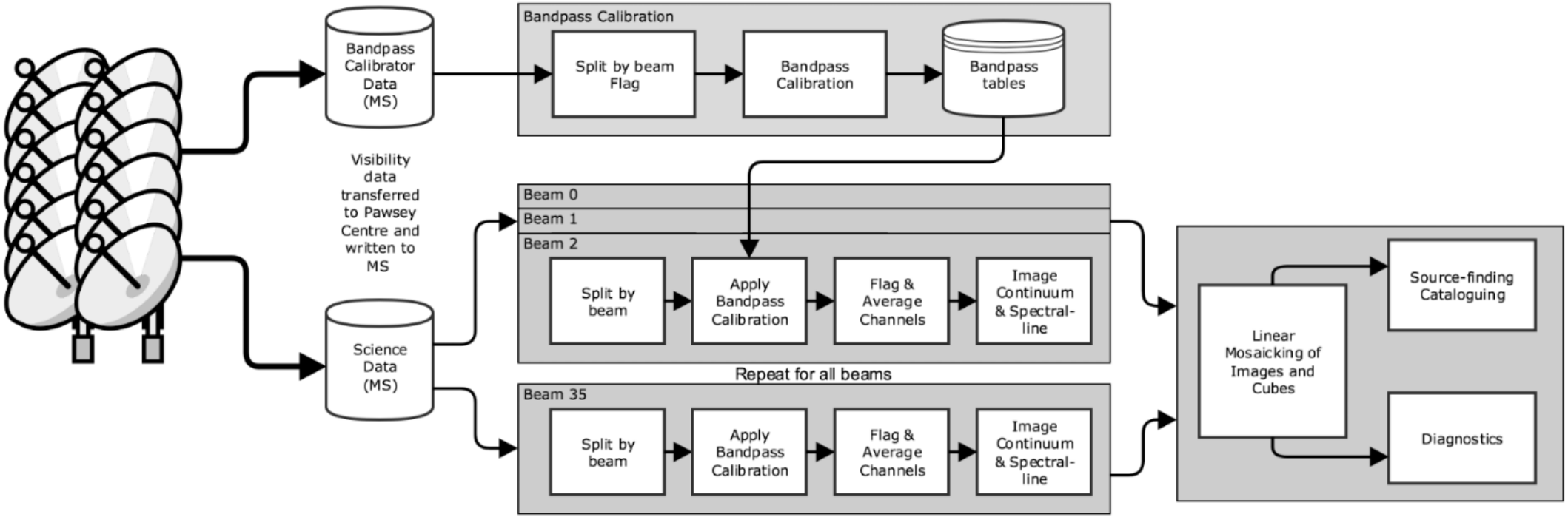
Science Data Processing (SDP) Pipeline

Processing path taken depends on science goal & input data quality. Resource footprint can vary from

- 100-node MPI jobs reading/writing TBs of data with large computational and memory imbalances
- Large number of small cpu+mem shared-memory jobs, which can quickly stress job scheduler (`slurm`).
- Jobs can be IO heavy with lots of throughput or lots of small file read/write access.
- Jobs contain iterative components (for “cleaning/calibrating”), thus runtime depends on input data.

Supernova remnant processed on Setonix (& Setonix)

Science Data Processing (SDP) Pipeline



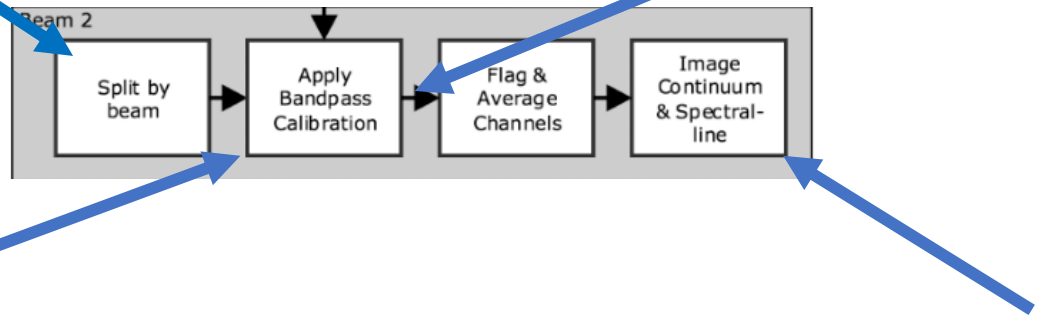
Example: *Imaging* which generates several groups of multi-node MPI tasks, one for each beam (36). Memory footprint & computational time to solution per beam not uniform. Processing involves large number of IO operations.

Each rectangle corresponds to a different slurm task in current pipeline.

Zooming In to SDP

Each slurm job is an MPI-enabled, large memory footprint job where, for this example, the ~16000 frequency channels of the input data are split amongst the ranks with rank 0 being the orchestration process of the processing.

Often, different processes are stitched together with IO, reading/writing numerous files.

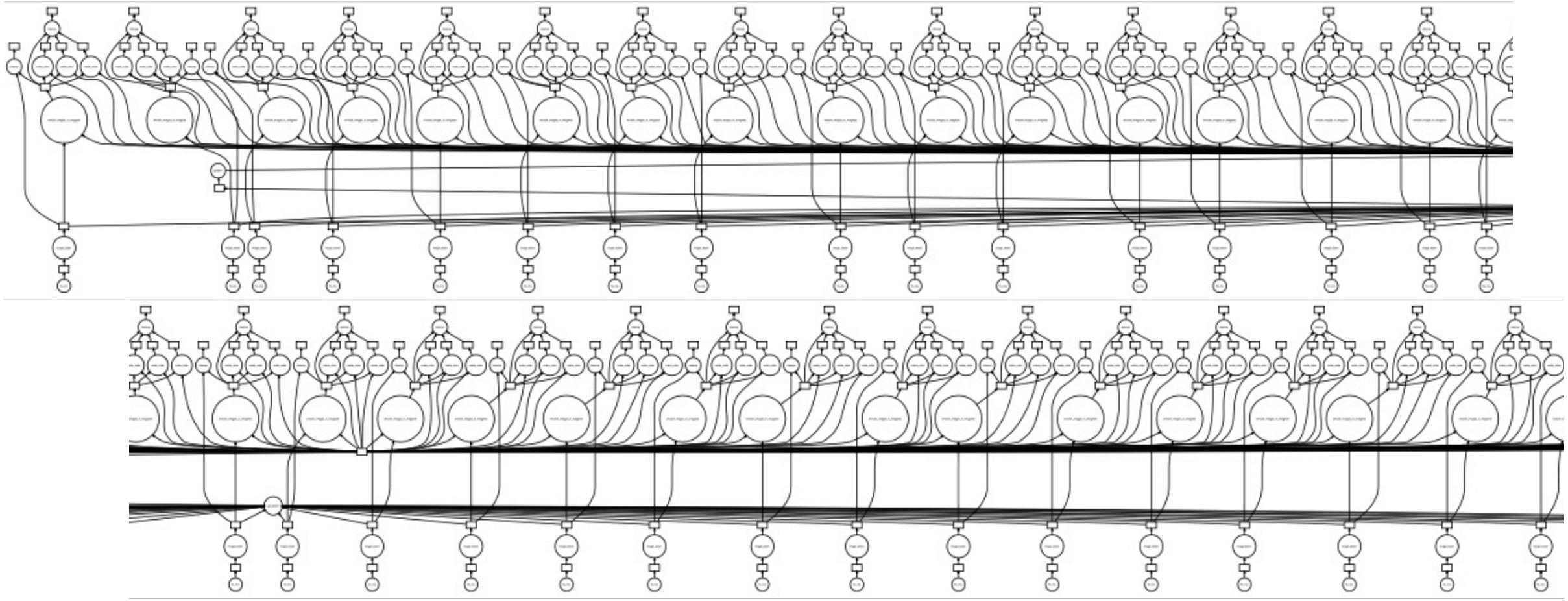


To have useable data, raw data calibrated by determining the complex gains (amplitude and phase), the frequency response (bandpass) and flux scale for each antenna.

MPI heterogeneous and jobs resource requirements vary :

- 2 nodes, 1 process on one node, 36 processes with 16 threads on the other (1 supervisor + 36 workers)
- 1 node, 33 process (1 s + 32 w)
- 2 nodes, 145 processes (1 s + 144 w)
- 2 nodes, 193 processes (1 s + 192 w)

Zooming in on job dependency



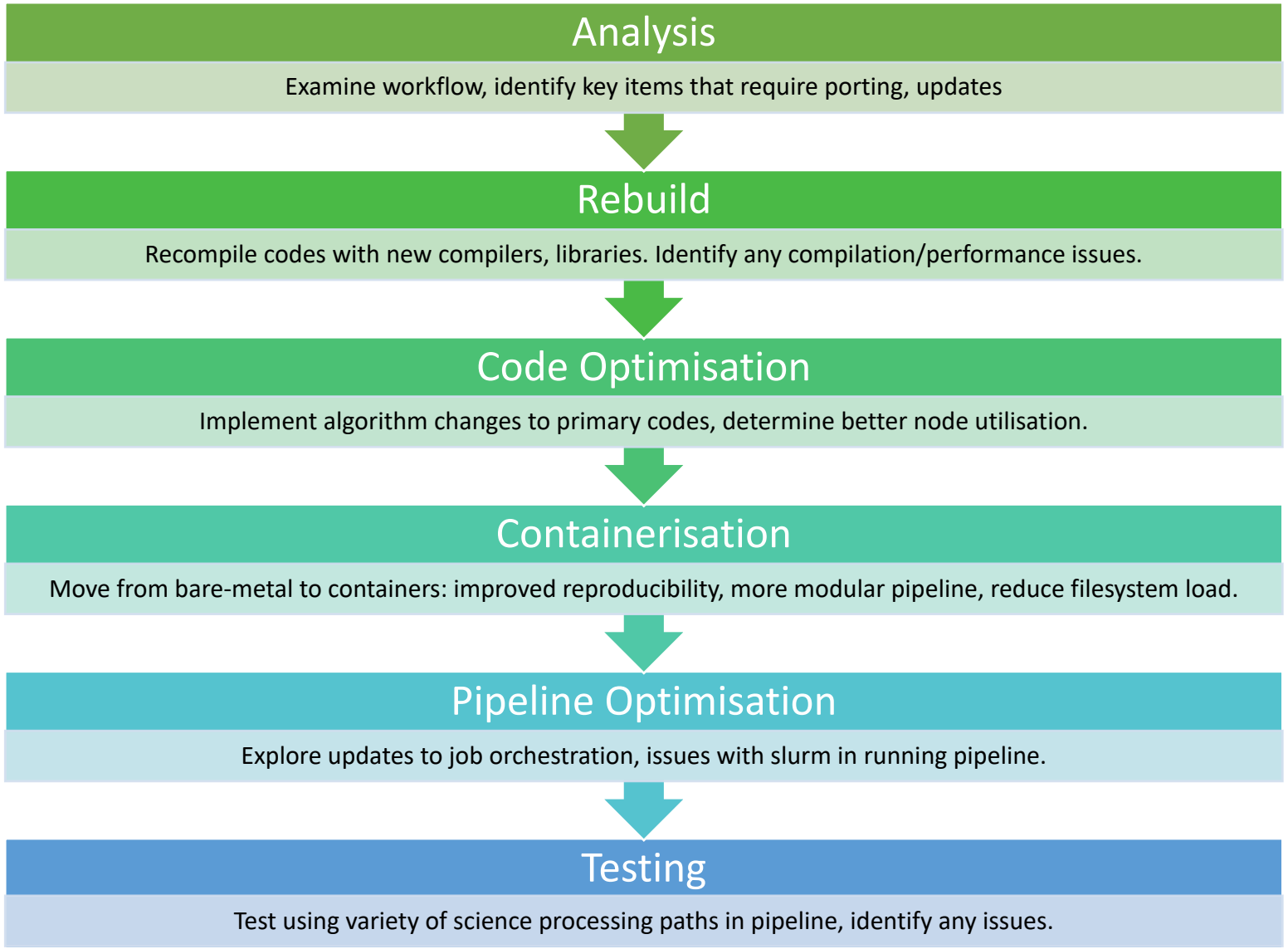
Example: *SPICE RACS* (polarisation) workflow DAG for a particular type of processing for input observation, showing the complexity of the job dependency. Only a portion of the DAG is shown.

Migration Process

Pawsey Supercomputing Research Centre & ASKAP Operations teams took an AGILEish approach: weekly meetings, sprints, early access to the CPU architecture used on Setonix, access HPE Cray EX Test and Development System (TDS).

Migrating a pipeline consisting of many subtasks, where path taken depends on input data and desired science output.

Standard "profiling" not the whole picture and too many components to profile in detail.



Challenges

Migration presented some daunting issues not just related to the pipeline or software stack running on new CPUs. We encountered issues

- Running MPI jobs [severity: **blocker**]
  - Number of MPI-related bugs missed in standard acceptance testing uncovered by running pipeline. Blocker: codes hang when there was large time between associated MPI send & receives
- Developing and running MPI-enabled Containers [severity: severe]
  - Little guidance on how to inject host libraries into container to enable MPI, MPI-IO.
- With IO & Lustre filesystem [severity: severe/**blocker**]
  - Numerous Lustre bugs exposed by IO stress of pipeline.
- With Job Orchestration using Slurm [severity: medium]
  - Some science workflows produce too complex a DAG for Slurm.

Solutions & Workarounds

MPI

- Workarounds were identified with environment variables
- Solution relied on upgrading libfabric library.
- Resulted in new sets of MPI tests.

Containers

- Developed container recipes to build lustre in container, Lustre-aware MPI and explore all MPI dependencies to preload libraries.
- Documentation provided to HPE.

Job Orchestration

- Current pipeline uses bash scripts and relies on slurm to enforce job dependencies. The complexity will increase. Job scheduler is not well designed as job orchestration too.
- Initial work to migrate some workflows to Prefect and on-going work to improve current pipeline.

Solutions & Workarounds

IO

- Initially, file locks were used, generating high (almost unmanageable) load. Also exposed several Lustre bugs, which are are not fully patched. Removed need for file locks.
- Some pipelines can produce millions of small files. Quota halts this but there was a bug with quotas, resulting in file systems going down.
- Working on integration of compressed filesystems and new IO with ADIOS2.
- Still issues with current pipeline and scratch performance slowing/halting processing [severity: medium/severe]

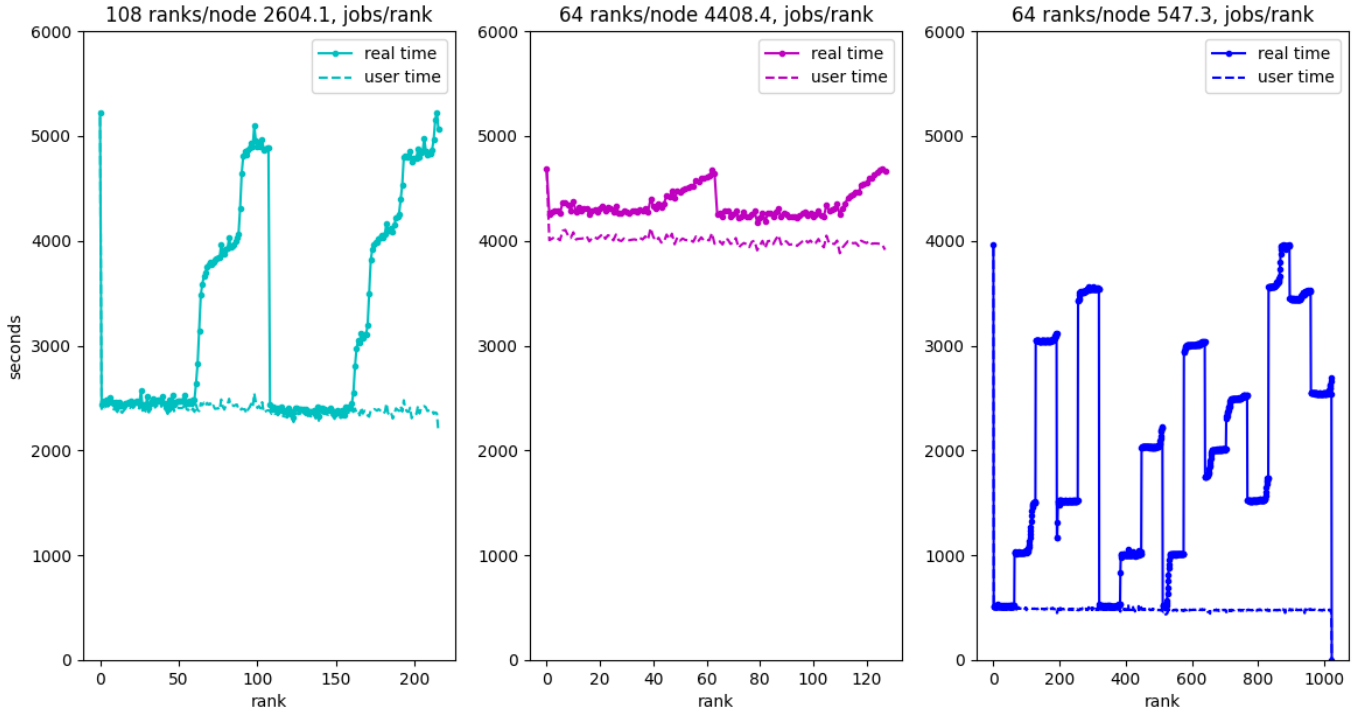
Solutions & Workarounds

IO

- Workflow's can still be very negatively impacted by IO performance that is a result of underlying read/write patterns in software. Require changing updating code [hard] or restructuring portion of workflow [also tricky].

Time taken to run some processing, where there are small read/writes during iteration.

This is in one tool among many.
Identifying where slow performance occurs takes a while.



Lessons Learned

Initial analysis is more involved and require teams to work together.

- Analysis will need to be deep dives into not only codes but a full workflow.
  - What tool(s) are used to run the workflow? Workflows now can consist of lots of steps with a complex inter-dependency. This might require a mix of tools as no single one might be fit-for-purpose.
- Profiling where to spend time not limited to just code.
  - Bottlenecks will not just be computational but could be a multitude of IO issues, issues with efficient node utilisation requiring heterogeneous jobs
- Diagnosing issues far more complex than segfaults.
  - With a full workflow, identifying underlying issues is far more involved and error could have occurred in one portion of the workflow and gone unnoticed till later.

Lessons Learned

Updates can possibly consist of:

- Updates to IO operations across entirety of workflow. Portions maybe well suited to parallel filesystems, while others are optimal worst case for such file systems.
  - Example: Incorporation of ADIOS2 into ASKAPSOFT and integration of SquashFS into the python heavy diagnostic workflow of ASKAP. Other possibilities stitching through libraries like capio.
- Code development of computationally heavy tools.
  - Example: Offloading specific routines to GPUs. Can be complex to integrate in workflow.
- Updating workflow orchestration tool itself.
  - Often older workflow tools would rely on Slurm. We recommend investigating tools like Prefect, Nextflow and prototyping a pipeline.
- Investigation of a variety of deployment technologies.
  - Bare-metal builds typical of HPC may not be ideal. Containers offer a means of solving dependency hell and reproducibility.

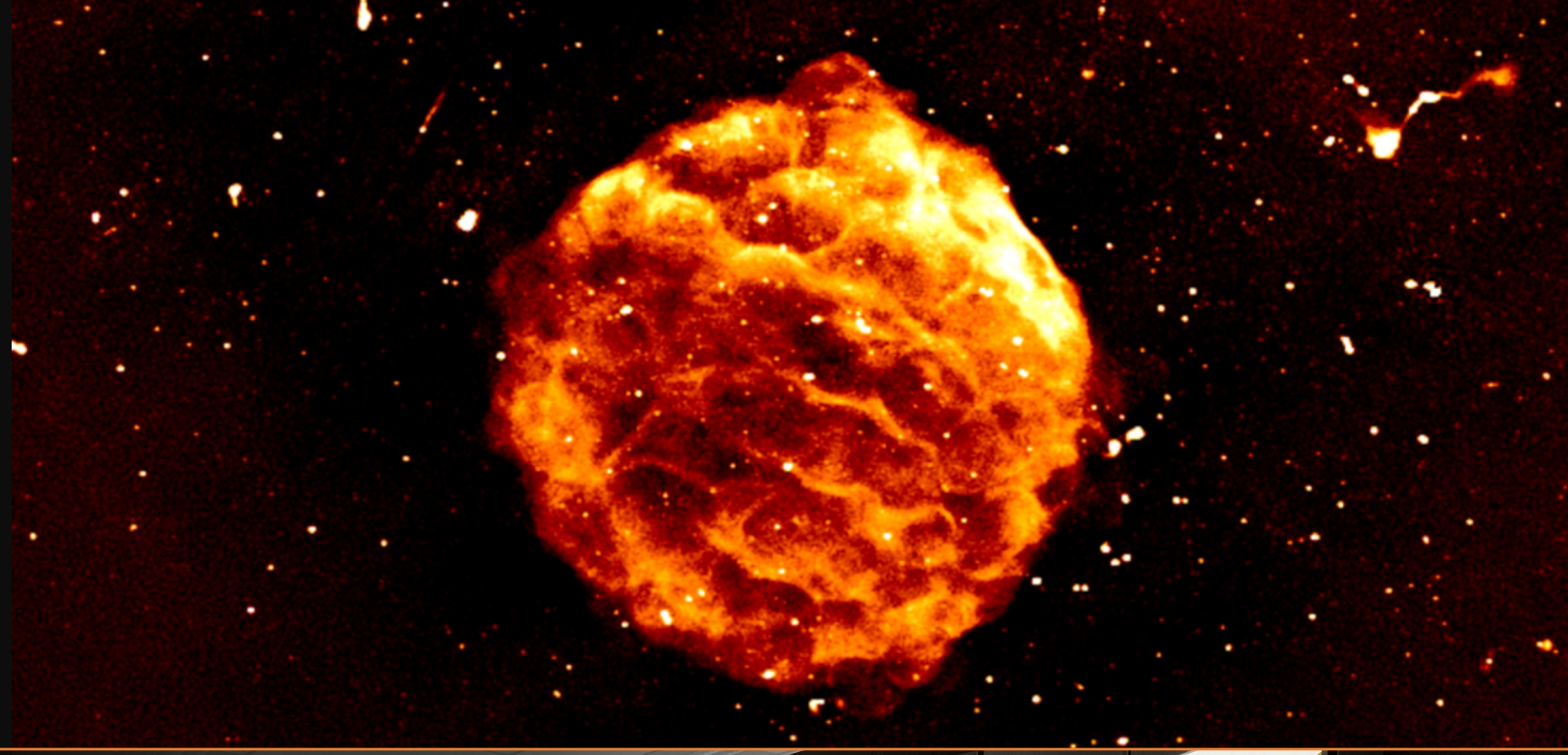
Future & SKA

- Square Kilometre Array (SKA) is an international collaboration involving 15 countries, which will be an order of magnitude larger in scope and complexity than current telescopes like ASKAP. SKA workflow is still in design phase.
- ASKAP migration demonstrates the need for domain scientists, HPC experts working tightly together to incorporate variety of technologies and approaches is the only path forward
- ASKAP migration and radio astronomy not outliers. Other domains like bioinformatics also have complex workflows.

Need holistic approach to be ready for exascale workflows and not just exascale codes.

SKA-Low Radio Telescope

Questions?



Migration to Exa-workflows: Radio Astronomy