CASA Next Generation Infrastructure

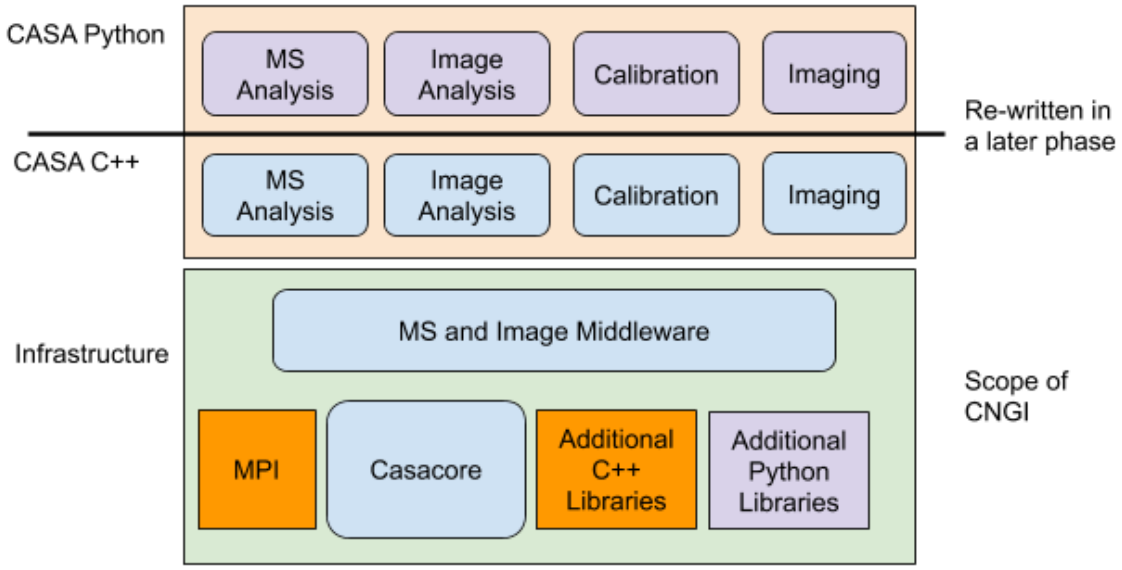
Ryan Raba

Origins

- CASA 2015 Strategic Plan called for the modernization of casacore and data formats
  - Better scalability to large data sizes
  - Robust hardware configurations
  - Global support of a diverse range of instruments and uses
- NRAO internal study kicked off in 2019 to look at potential options for rebuilding casacore and the base processing infrastructure of CASA
- One groundrule of the study is that the next generation infrastructure may not be backwards compatible with the current infrastructure
- This leads to a broader Next Generation CASA Program to subsequently rebuild CASA overtop of a new infrastructure

Scope

- A complete top-down overhaul of the entirety of CASA is not feasible as there are no requirements specifications, detailed design, architecture or definition of correct output.
- Instead, the scope of this project is to condense and replace the CASA data processing software infrastructure and casacore code base only with a new and functionally equivalent package
- Subsequent Next Gen CASA definition and migration comes later



Next Generation Motivations

- Maximize Performance
  - Fast experimentation
  - Linear scalability to terascale computing with thousands of cores on commodity/cloud hardware
  - Fully utilize all cores, memory, and disk I/O simultaneously
  - Support everything from single user laptops to high-end clusters in enterprise datacenters

- Minimize engineering overhead
  - Simplify scientific coding
  - Easy implementation of science algorithms without software engineering complexity
  - Eliminate explicit manual storage I/O, memory paging, and parallel processing execution
  - Reduce development time of new features and maintenance overhead of existing features

Plan and Schedule

- CNGI ~ 1.5 year effort
  - Focus on just casacore and support infrastructure first
  - Produce a standalone package for MS and Image manipulation, independent of CASA
  - 3 months: Software Trade Study ✓
  - 3 months: Framework Trade Study ✓
  - 6 months: Prototype development – 0.1 release
  - 6 months: Complete initial development – 1.0 release
- Next Gen CASA
  - After CNGI, evaluate and plan full CASA migration to new infrastructure
  - 6 months: Prototyping of key science/processing alongside CNGI
  - ~1 year: Initial set of methods migrated – 0.1 release
  - ~2-3 years: Phased migration of more complex CASA components
- Next Gen Visualization

Trade Studies

- Software Trade Study
  - Evaluate the field of modern Software Engineering
  - Downselect promising candidates for large scale scientific data processing and analytics
  - Looking at languages, parallelization schemes, and programming paradigms
    - Python Map/Reduce, vector mathematics, linear algebra
    - Rust
    - Seamless CPU/GPU execution: Tensorflow
- Framework Trade Study
  - Using the results of the first study, focus more specifically on what off-the-shelf packages exist to satisfy the motivations of CNGI
  - Build some small toy examples in each framework
  - Select a promising candidate for prototyping
    - Apache Spark, Dask
    - Pandas Dataframe structure in memory, Apache Parquet or HDF5 on disk

Results to Date

- Dask is the final selection for future prototyping
  - https://dask.org/
  - Built in scheduler, DAG execution, memory/process/storage control
  - Feels a lot like Pandas and Numpy, supports Numba
  - Additional Python libraries for numpy-like GPU processing still on the table for execution within Dask environment
- Dataframe model
  - https://pandas.pydata.org/
  - Dask extension of Pandas Dataframe standard
  - Map operations rather than looping
  - Groupby operation for selection
- Apache Parquet file format
  - https://parquet.apache.org/
  - Suitable for MS, not yet decided for Image
  - First level partition by DDI
  - Second level partition by row-chunks

Next Steps

- API
  - Define functions of CNGI prototype as standalone package
  - Begin to assess impact to CASA and next gen Visualization
- Examples
  - ms_to_parquet.py converter
  - Dataframe tutorial:
    - `ms.groupby(['ANTENNA1', 'ANTENNA2'])`
    - `ms[cont] = ms.apply(lambda row: row[chans].mean())`
    - `ms.filter(lambda row: row[chans].mean() < row[cont])`
  - Simple gridder
  - Scalability demonstration ~100 to ~1000 cores
- Prototype Development