Data post-processing with CASA package

Francesco Bedosti

IRA – INAF

Bologna

The ALMA Telescope

The Atacama Large Millimeter/sub-millimeter Array (ALMA) is an astronomical interferometer of radio telescopes in the Atacama desert of northern Chile.

- Largest and most sensitive instrument in the world at millimetre and sub-millimetre wavelengths
- Coordinated and operated by the Joint ALMA Observatory (JAO)
- Three Regional Centers (Europe, North America and East Asia)
- Chajnantor, Atacama, Chile, 5000 metres altitude
- 66 movable antennas spread over distances of up to 16 km
- 0.3 to 9.6 mm wavelength range.

European ALMA Regional Center

- Central node @ ESO
- Italian, German, Dutch, U.K., Nordic, IRAM and Czech nodes.
- Provides an interface between the ALMA project and the European science community.
- Supports its users mainly in the areas of proposal preparation, observation preparation, data reduction, and data analysis.

Italian node activities...

- node expertises: Data handling/GRID-technology, Mosaicing, Coordinating surveys/key-projects, Polarimetry
- Face2Face support
- Schools and dissemination
- Science verification and software beta-testing
- Software development
- Web site and community building
- ARC Cluster

CASA

- Developed by NRAO, ESO, NAOJ, CSIRO, ASTRON
- Based on AIPS++
- Used for calibration and imaging of interferometric and single-dish radio astronomical data.
- 2.4 GB package of binaries, interfaces, libraries, ancillary data and documentation
- Interface in IPython, number crunching in C++
- Input, scratch and output data in the same folder.
- Some algorithms inerently not parallelizable
- Parallel algorithms not production-ready, yet

Typical workload

- A sequence of 10-15 different algorithms
- High resolution data plotting
- Input dataset size up to 4 TiB
- Up to 6 TiB of temporary data
- Up to 10 GiB output data
- Up to 24h running time

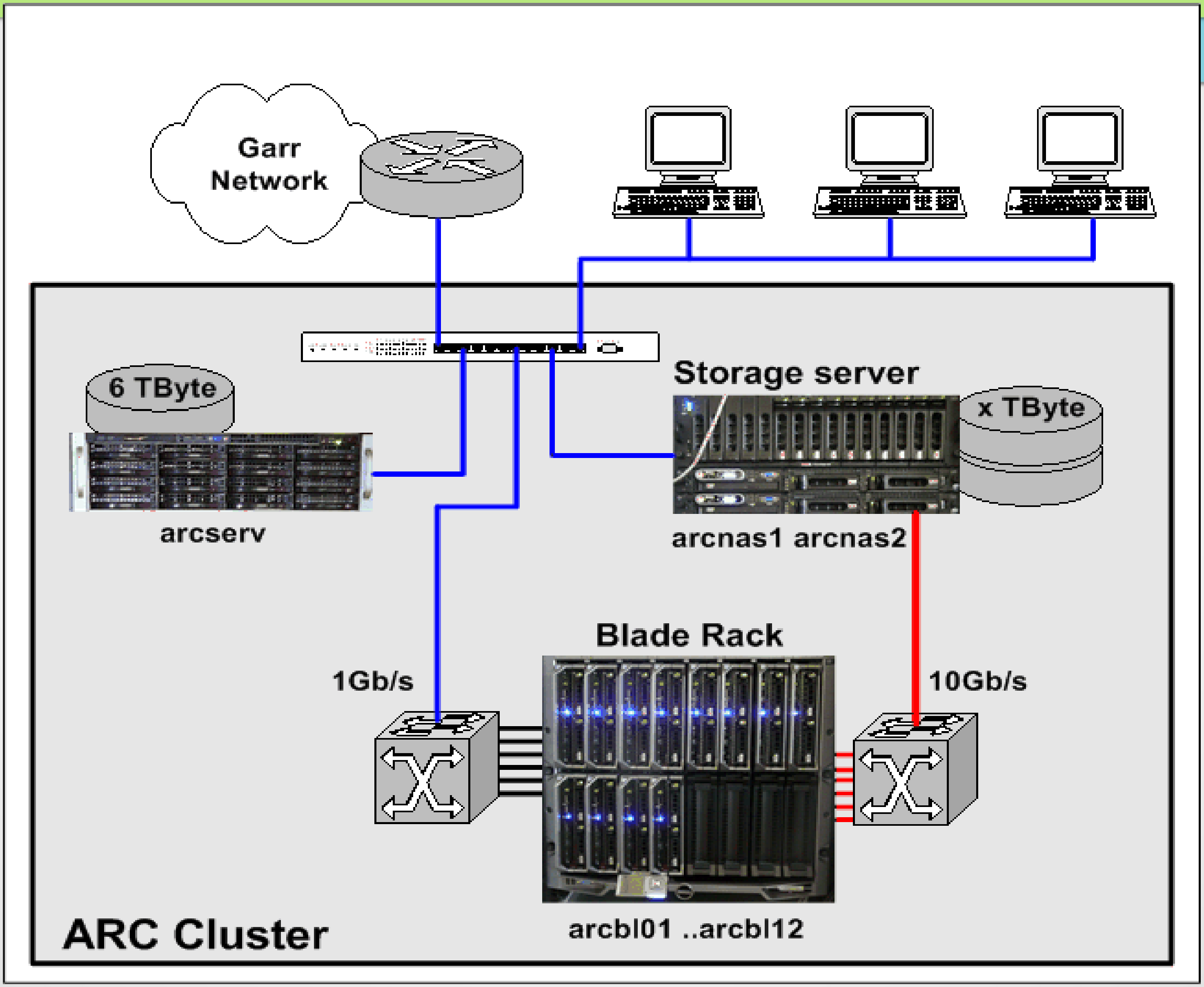
Users expectations

- Interactive workflow
- Graphical interface
- Transparent access to data
- Transparent acces to computational resources

Our ARC cluster

- 44 TiB Shared storage (90TiB soon..)
- 12 x 8 Opteron cores, 8 GiB RAM blade nodes
- 1 x 4 Opteron cores, 16 GiB RAM blade node
- 10 Gbps Jumbo Frames data network
- 1 Gbps control network

The ARC Cluster @ IRA – INAF



Storage types - Network attached

- Uses Lustre filesystem over 10Gbit/s dedicated network.
- Has huge latency & low random seek time.
- Has good concurrency capability.
- Fast in sequential workloads.
- Is shared among all workstations/cluster. Avoids to load all data on computer before processing.
- Can grow smoothly to peta-scale sizes.
- Its bandwidth is shared among all working nodes.

Storage types – Local

- Two 15 SAS Disks in RAID-0
- Uses EXT4 linux-native filesystem.
- Low latency & fast random seek time.
- One filesystem for one computer.
- Slow in sequential workloads.
- Hard to scale to more than 2TiB.

Access to resources

Computational

- Interactive sessions dynamic scheduling
- PBS jobs
- Grid jobs

Storage

- WebDAV
- NFS
- Lustre
- SshFs

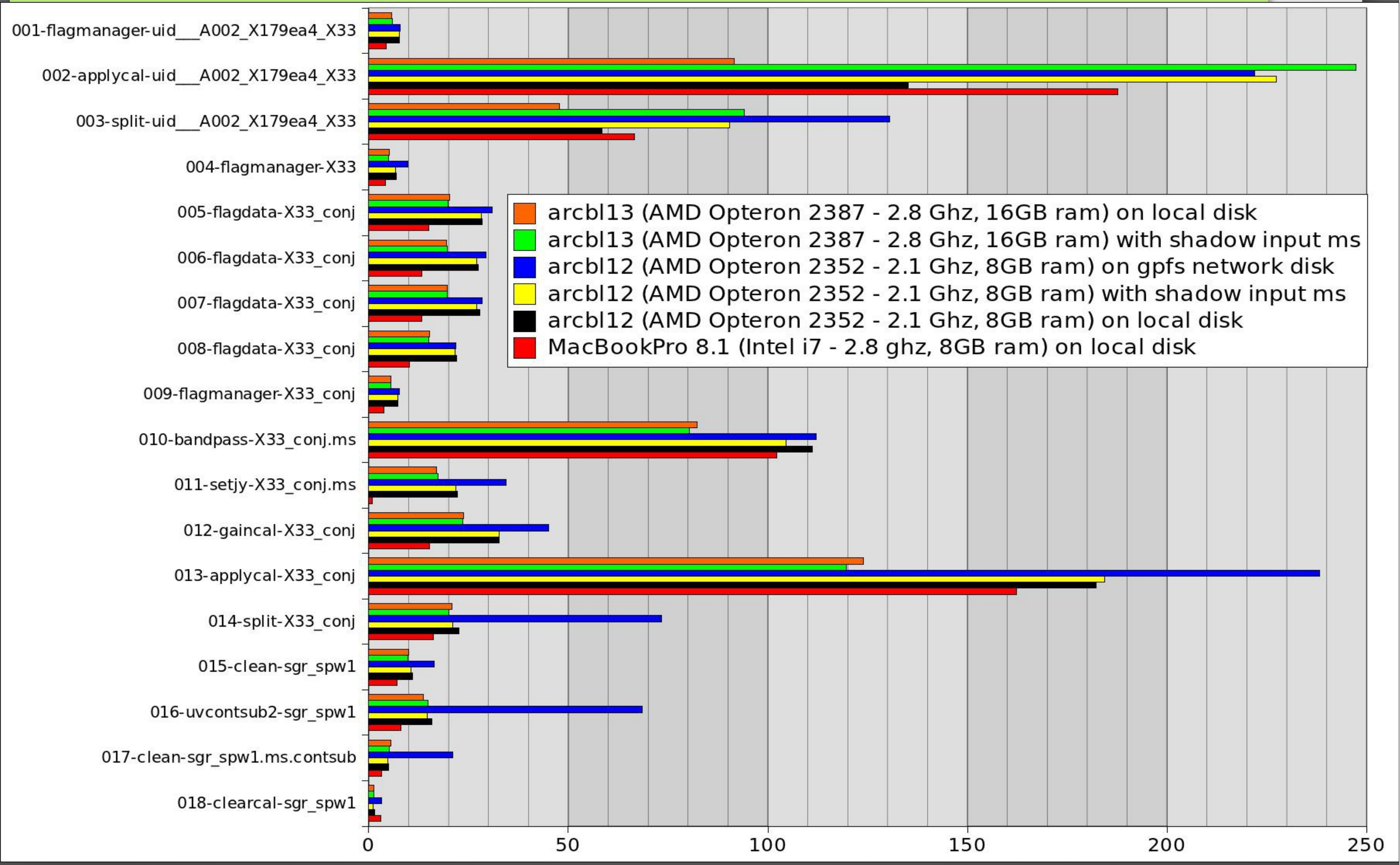
The mixed way - Shadow

Working on **local** disk accessing input files stored on **network attached** disks via symbolic links. To create a **shadow** dataset we:

- create a directory-tree identical to the input dataset one.
- populate this tree with symbolic links referring to real input dataset files.

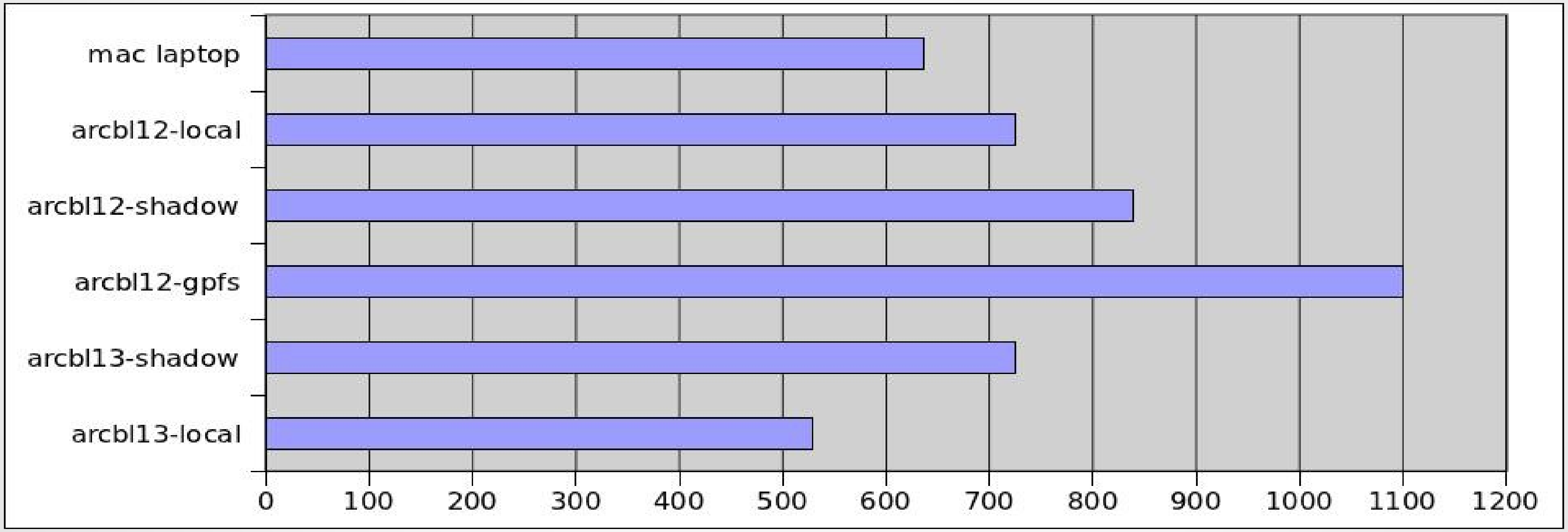
In this way the application, while reading input files from the original MS, writes its scratch files on local disk.

A benchmark (5 ant, 4000ch, 9GB)



Total time

For a common, mixed workload, no recipe fits all...



Recipes

- Some tasks use a lot of temporary files (clean, uvcontsub, split, gaincal, setxy). Those benefit from shadow MS, using local disk for scratch and network storage for input/output. Also the use of SSDs can be worthwhile.
- Some read huge amounts of inputs or produce a lot of outputs (flagdata, applycal). Only fast disks can help.
- The other ones are not I/O intensive so they mostly depend from CPU speed. Some of these (flagdata, clearcal, applycal, and partition) can be improved via parallelization.

Shortcomings

- Storage space, speed and latency

  @ 5 TiB/experiment, users need fast and trasparent access to archives of hundreds of TiB.

- Single-core performance:

  CPU, memory and bus speed are crucial in not-parallelizable algorithms