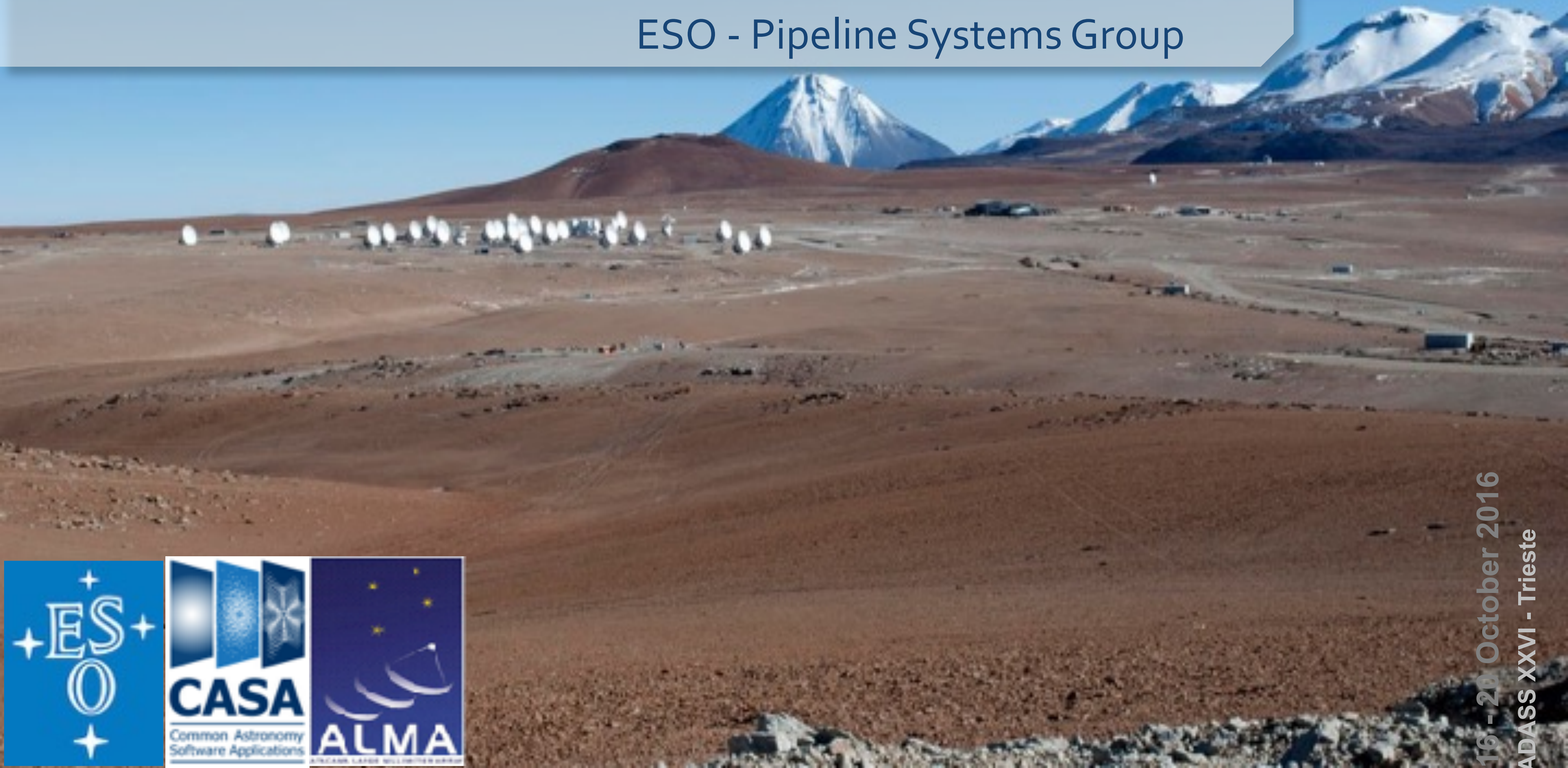
HPC Development for the ALMA Pipeline

Sandra Castro

ESO - Pipeline Systems Group



16 - 20 October 2016

ADASS XXVI - Trieste

CASA Development Team

On behalf of the CASA team:

J. Gonzalez, J. Taylor, S. Bhatnagar, M. Caillat, P. Ford, K. Golap, J. Jacobs, S. Loveland, D. Mehringer, G. Moellenbrock, D. Petry, M. Pokorny, U. Rao, D. Schiebel, V. Suoranta, T. Tsutsumi, K. Sugimoto, W. Kawasaki, M. Kuniyoshi, T. Nakazato, R. Miel

CASA Lead: J. Kern (NRAO)

Outline

- What is the ALMA pipeline?
- What is CASA?
- CASA Parallelisation concept and framework
- CASA Parallelisation Tiers
- Initial performance and benchmarking
- Using AWS - initial tests

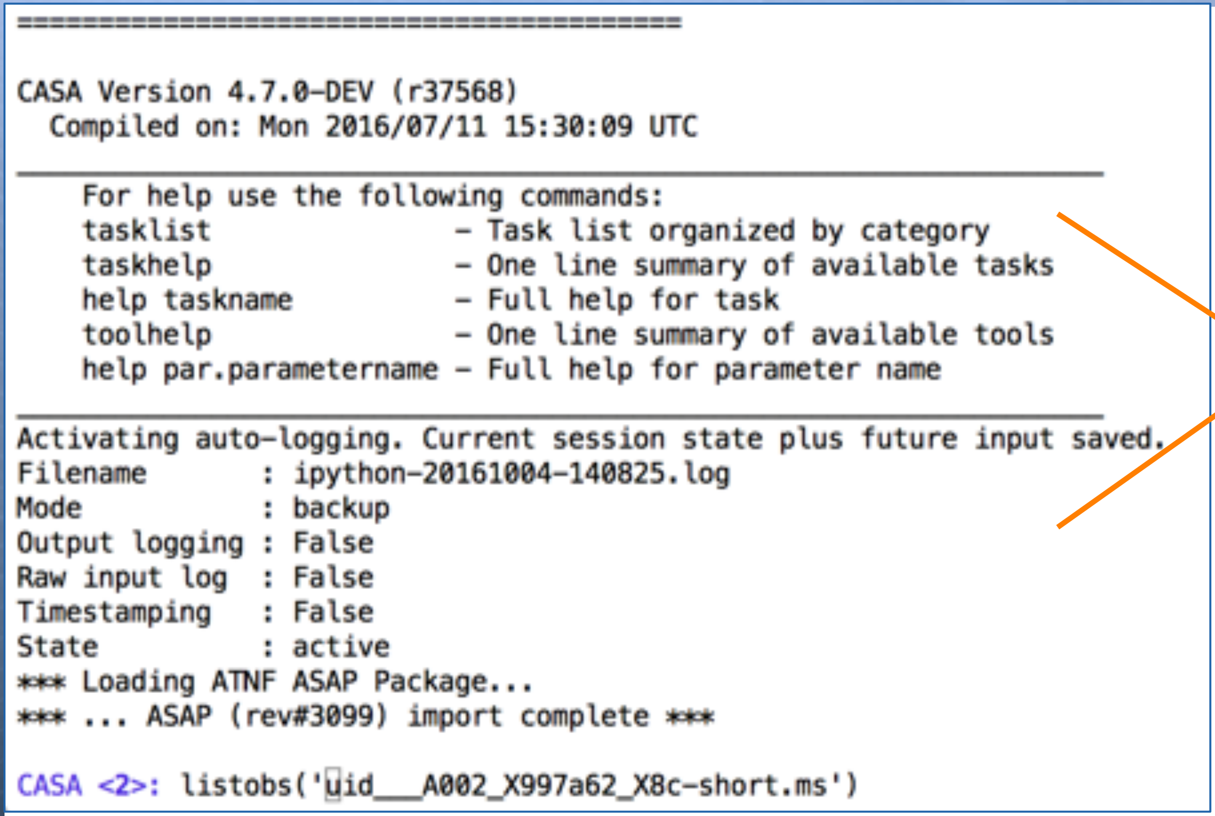
The ALMA Pipeline

- The ALMA Science Pipeline has been developed with the goal of performing automated data processing before delivery to the user.
- The pipeline is data-driven; data characteristics are handled by pipeline heuristics.
- The pipeline represents a standard path through calibration and imaging.
- Pipeline tasks uses **CASA tasks** and tools and can be executed in the same way as CASA tasks.

What is CASA?

- The *Common Astronomy Software Applications* (CASA) package, is developed with the primary goal of supporting the data post-processing needs of ALMA and EVLA.
- The CASA infrastructure consists of a set of C++ tools bundled together under an iPython interface as a set of data reduction tasks.
- It currently has ~ 1.7 million lines of code

CASA layout



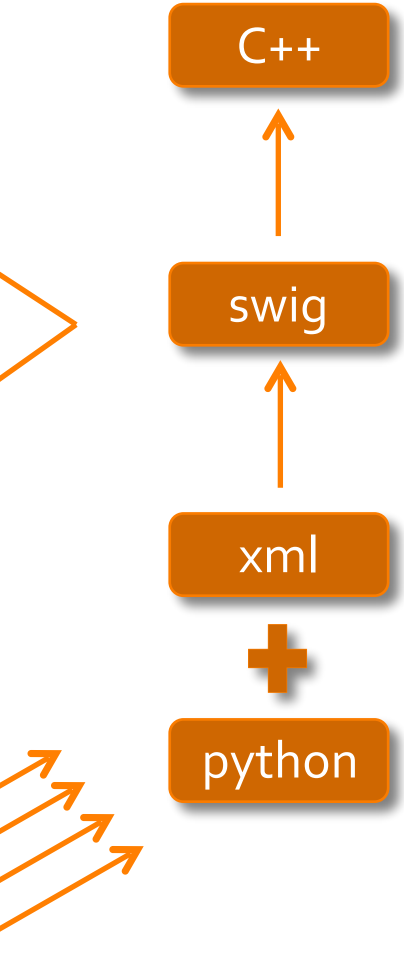
```
=========================================


CASA Version 4.7.0-DEV (r37568)
  Compiled on: Mon 2016/07/11 15:30:09 UTC

    For help use the following commands:
    tasklist               - Task list organized by category
    taskhelp               - One line summary of available tasks
    help taskname          - Full help for task
    toolhelp               - One line summary of available tools
    help par.parametername - Full help for parameter name

Activating auto-logging. Current session state plus future input saved.
Filename       : ipython-20161004-140825.log
Mode           : backup
Output logging : False
Raw input log  : False
Timestamping   : False
State          : active
*** Loading ATNF ASAP Package...
*** ... ASAP (rev#3099) import complete ***

CASA <2>: listobs('uid___A002_X997a62_X8c-short.ms')
```



parallelisation

CASA Parallelisation Concept

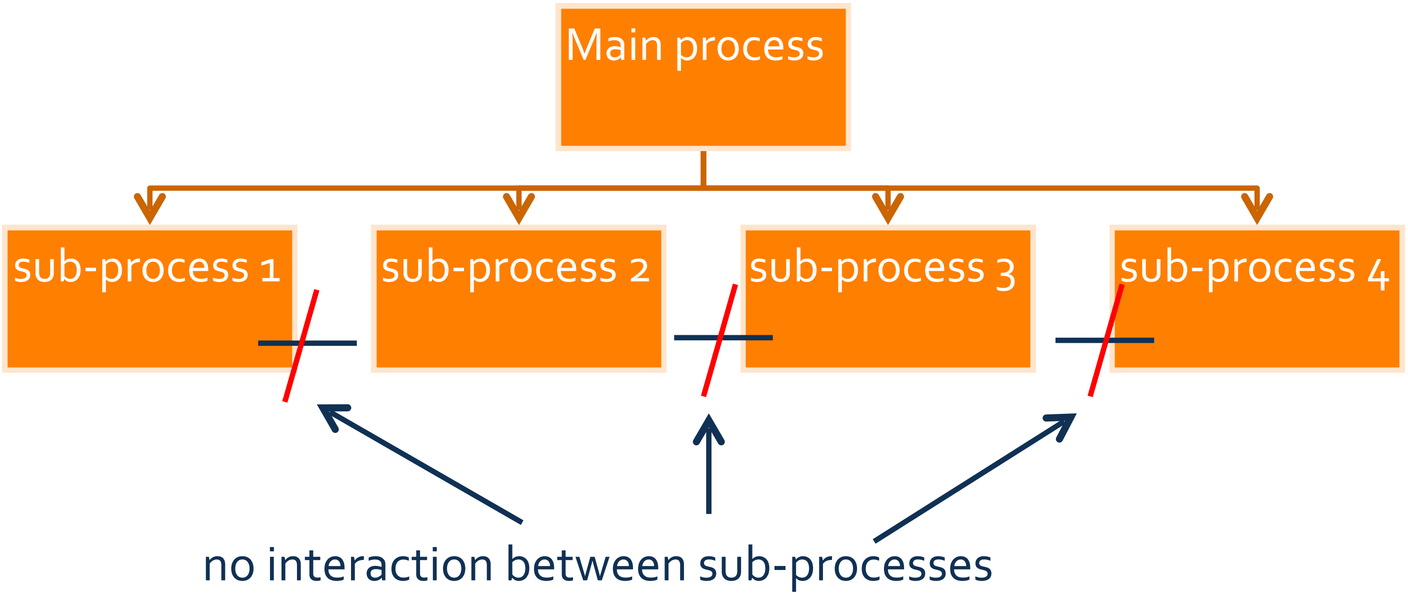
Trivial parallelisation

- Partition the MeasurementSet into sub-MSs (spw, scan axes)
- Run a CASA instance on each sub-MS in parallel

à partitioned data is called **Multi-MS or MMS**

à It is possible to create a Multi-MS at import time (importasdm)

Data Parallelisation Principle



Main process

sub-process 1

sub-process 2

sub-process 3

sub-process 4

no interaction between sub-processes

Theory: $\text{run-time parallel} = \dfrac{\text{run-time sequential}}{\#\ \text{sub-processes}}$

mpi4casa - J. Gonzalez (2014)

CASA parallelisation framework

- Uses the Message Passing Interface (MPI)
  - openMPI - MPI 3.0 standard
- Easy launching using custom **mpicasa** script
- Controls the number of processes at startup time
- Provides method to change maximum run-time memory used in each engine
- Although MPI allows a much richer inter-process communication we only use it for process control

(see also Gonzalez poster P6.11)

Implementation - Tiers

- Tier-0 Parallelisation
  - Parallel execution of not internally parallelised tasks

    → plotms, gaincal → see Multi-MS as a monolithic MS
- Tier-1 Parallelisation
  - Internal parallelisation within tasks

    → will work in parallel, on each Sub-MS separately

used in the pipeline

- Tier-2 Parallelisation (future)
  - Parallel execution of internally parallelised tasks

    → running several flagdata calls in parallel, each on an MMS

What is internally parallelised?

- Tasks that require traversing the entire data set and are I/O limited.

→ flagdata, applycal, time averaging, uvcontsub, split

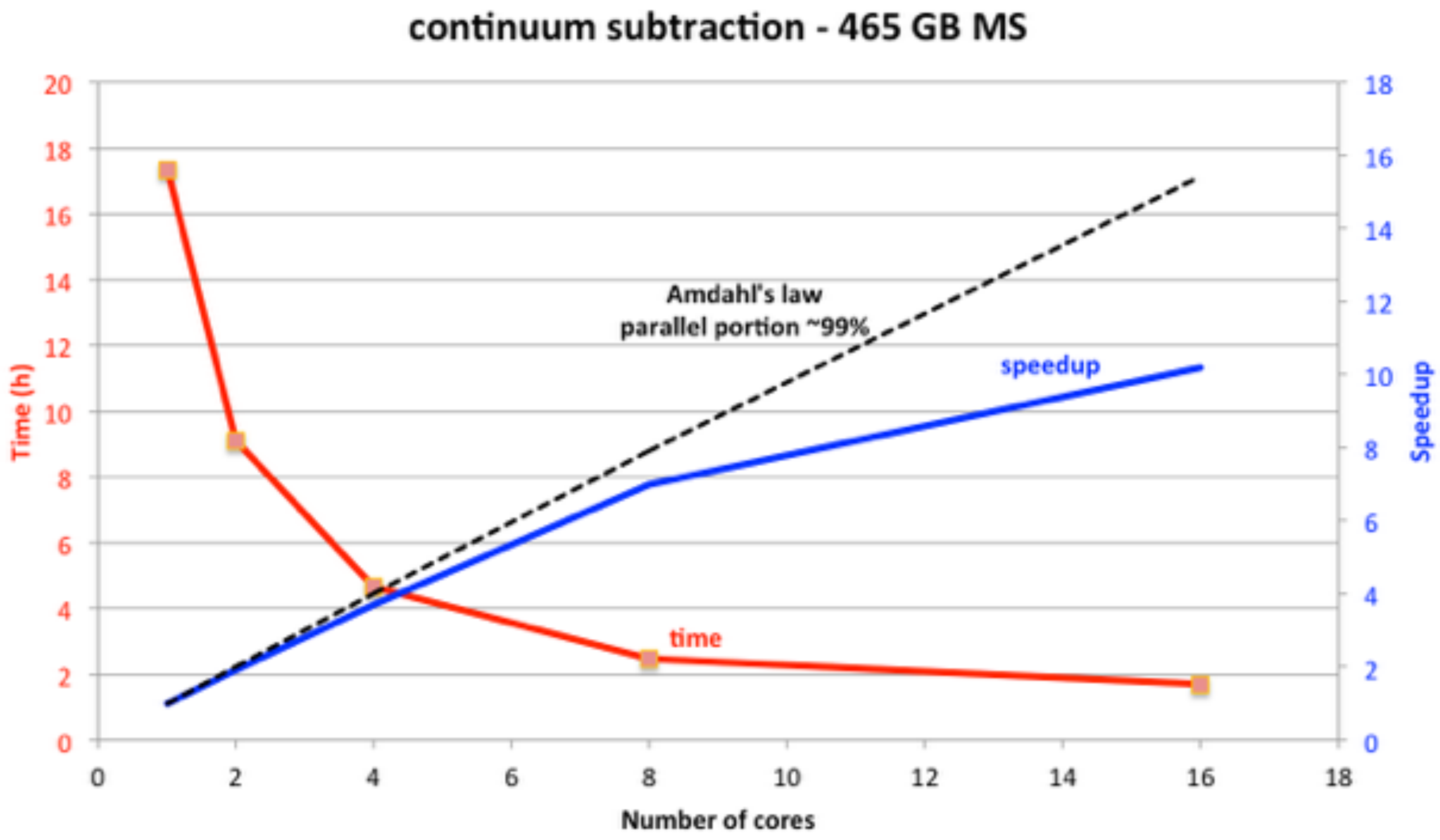
Benchmarking tests

- ➢ Used NRAO architecture
  - + Computer cluster: 50 node cluster: Dual 8-core 2.6 GHz Intel processors (16 cores total), 64 GB memory
  - + I/O Cluster (Lustre): parallel distributed file system. All clients see same coherent file system, which allows parallelisation across multiple nodes.
- ➢ 14 storage nodes, ~1PB total storage with 12+GB/s throughput. 40 Gbit Infiniband fabric connects computer to I/O cluster

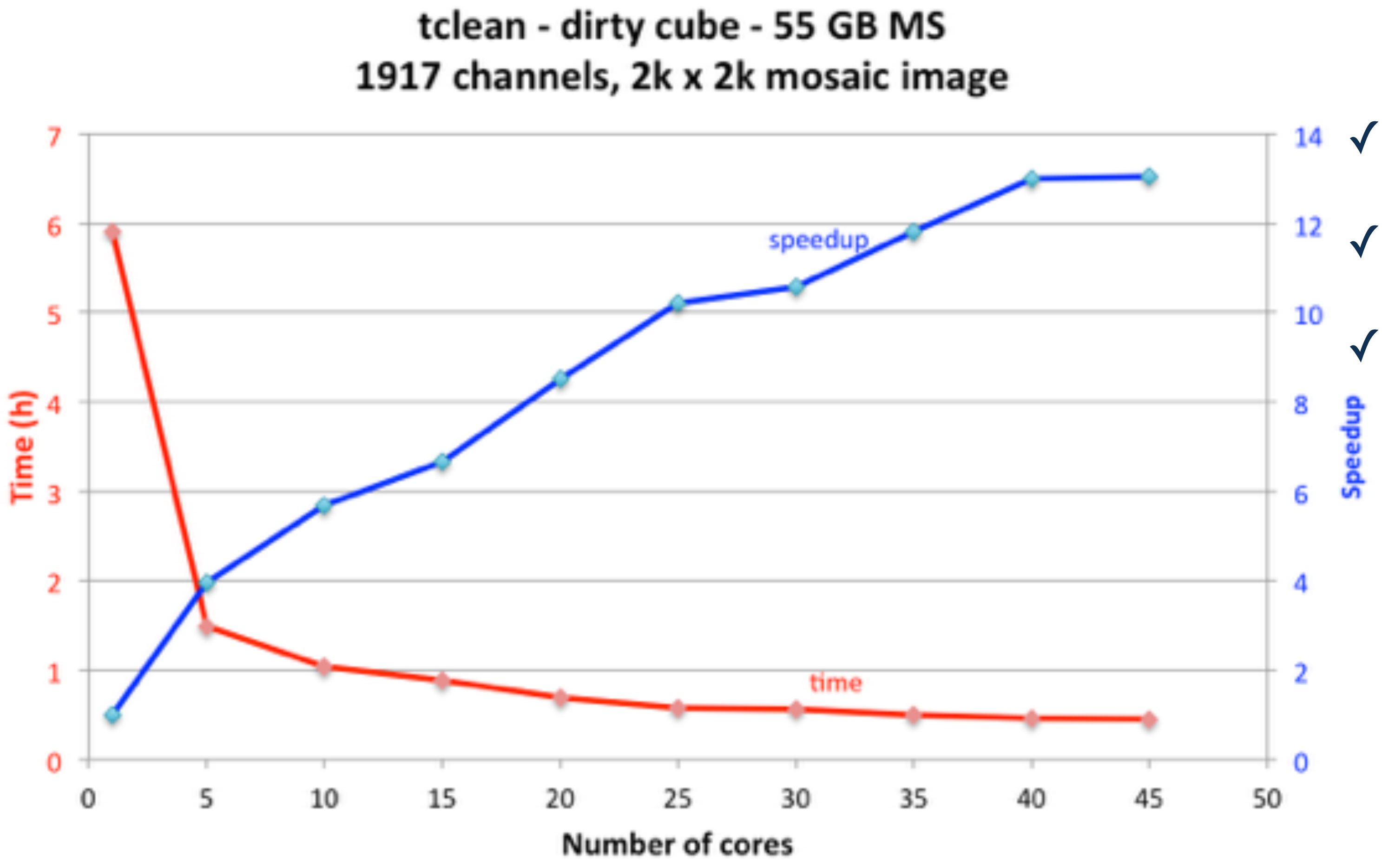
à Plots show **speedup ratio** of some internally parallelised tasks

$$S = T_S/T_P$$

Speedup and Amdahl's law



Imaging (tclean)- speedup

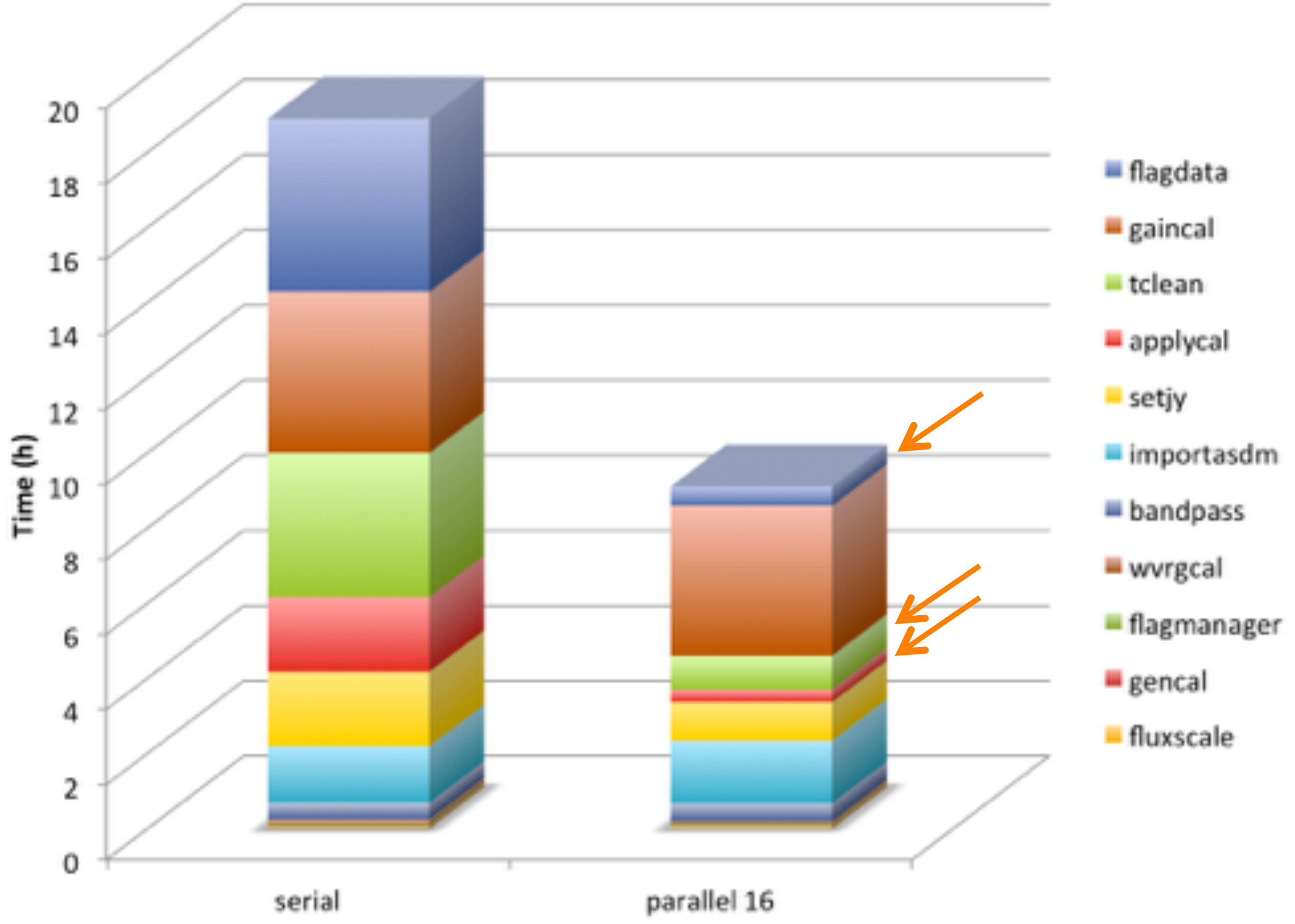


✓no iterations

✓ dirty cube

✓ 6h in serial

ALMA calibration pipeline



1 EB ~ 218 GB

majority of steps are sequential

16 cores on a single node

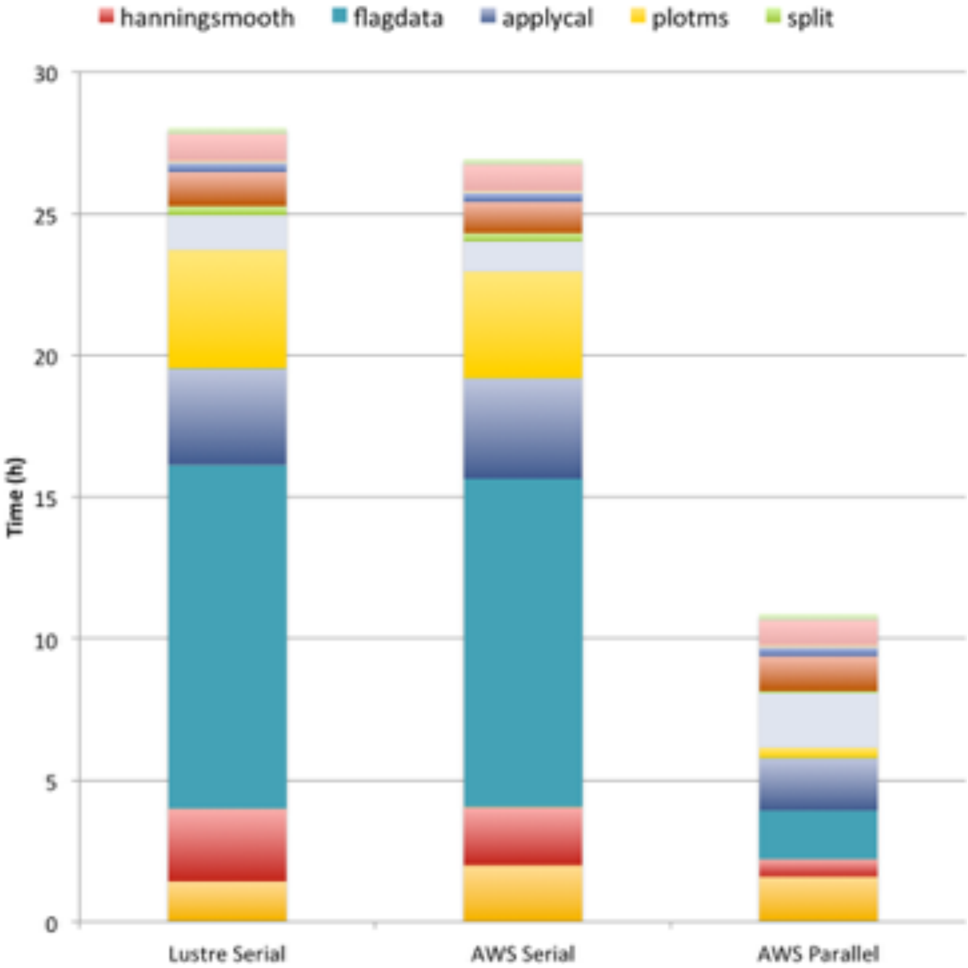
- parallel speedup

✓ flagdata ~9x
✓ applycal ~6x
✓ tclean ~4x

EVLA Pipeline Processing

Processed MS: 1018 GB

- Lustre AOC: 1PB Lustre system
- AWS serial: 1.5 TB provisioned storage (SSDs)
- AWS parallel: 8 engines



Thank you

- Download CASA and the ALMA pipeline

  http://casa.nrao.edu

- CASA newsletter

  https://science.nrao.edu/enews/casa_004/

- Using Amazon Machine Images (AMI) containing CASA 4.7.0

  https://casa.nrao.edu/casa_aws_introduction.shtml