ICRAR International Centre for Radio Astronomy Research

SKA SQUARE KILOMETRE ARRAY SDP SCIENCE DATA PROCESSOR

Extracting science from massive data sets:

Experience from the Murchison Widefield Array (MWA) and the Very Large Array (VLA)

Andreas Wicenec

&

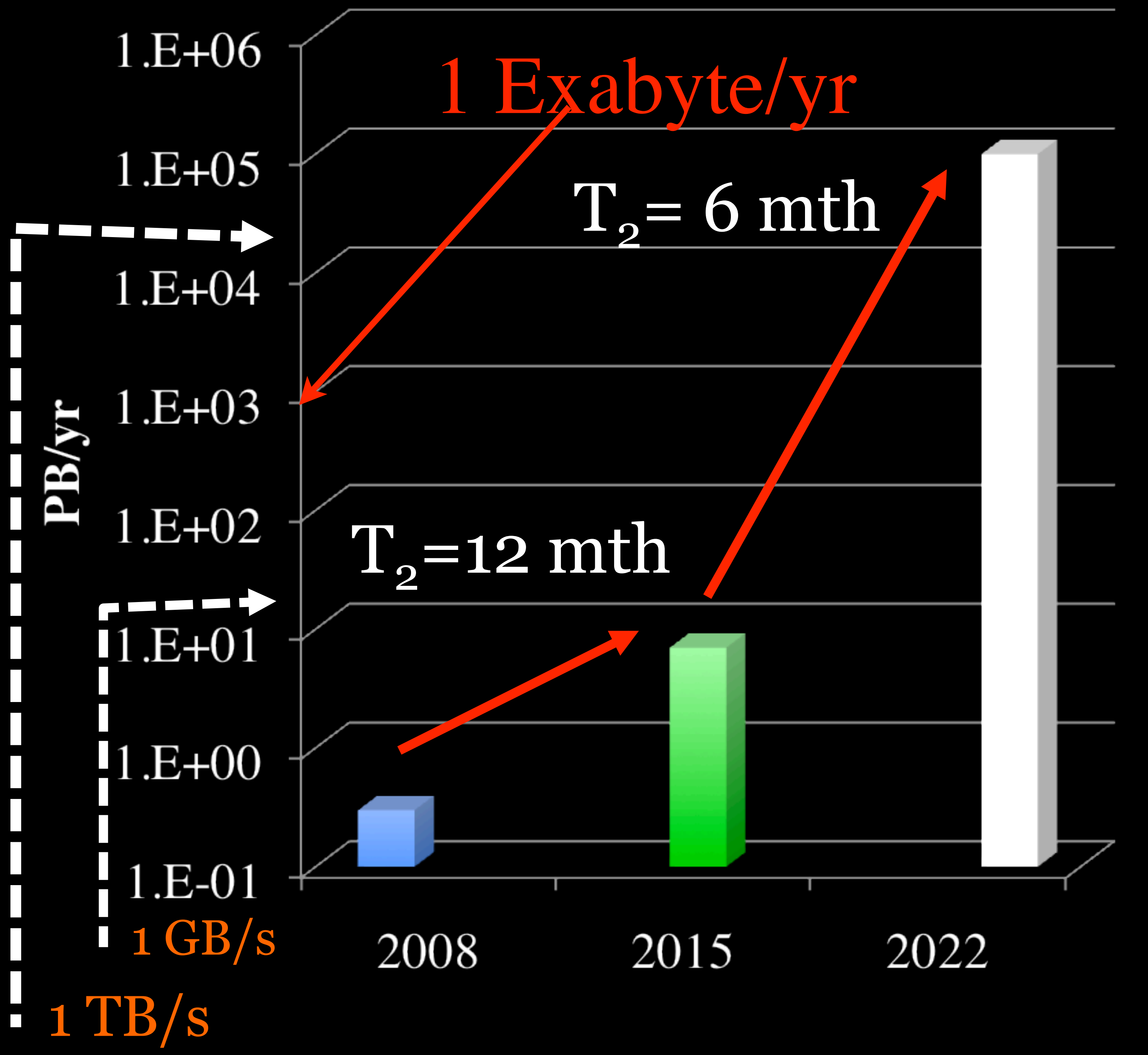
the Data Intensive Astronomy team at ICRAR

&

others

Context

The deluge continues



Questions

★Who should build:

- Dishes?
- Receivers?
- Beam formers?
- Correlators??
- Operational Software???
- Astronomy software????

Lessons Learnt

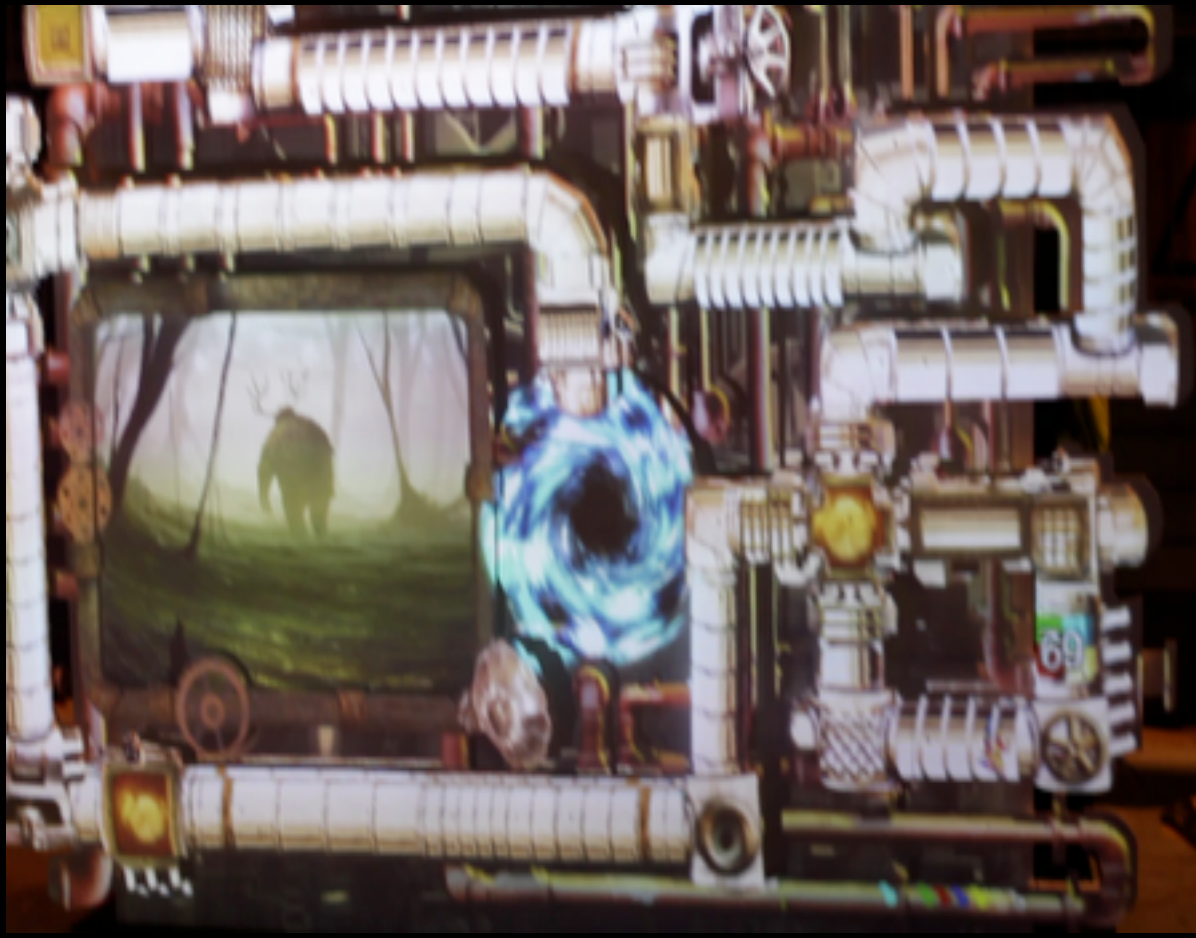
★ Don't let Astronomers write software if your requirements include:

- performance,
- parallelism,
- optimal use of network,
- optimal use of storage,
- optimal use of computers

... there are exceptions

Pipelines

- Everybody is talking about them
- New ones are ‘invented’ for almost every single project.
- Very often based on hacking CASA tasks, Miriad tasks, AIPS tasks and homegrown modules pulled together into a unmaintainable monster, that only a few people understand.
- Other scientists are adding more modules or replacing existing ones with ‘better’ ones.



Separation of Concerns

Separation of concerns

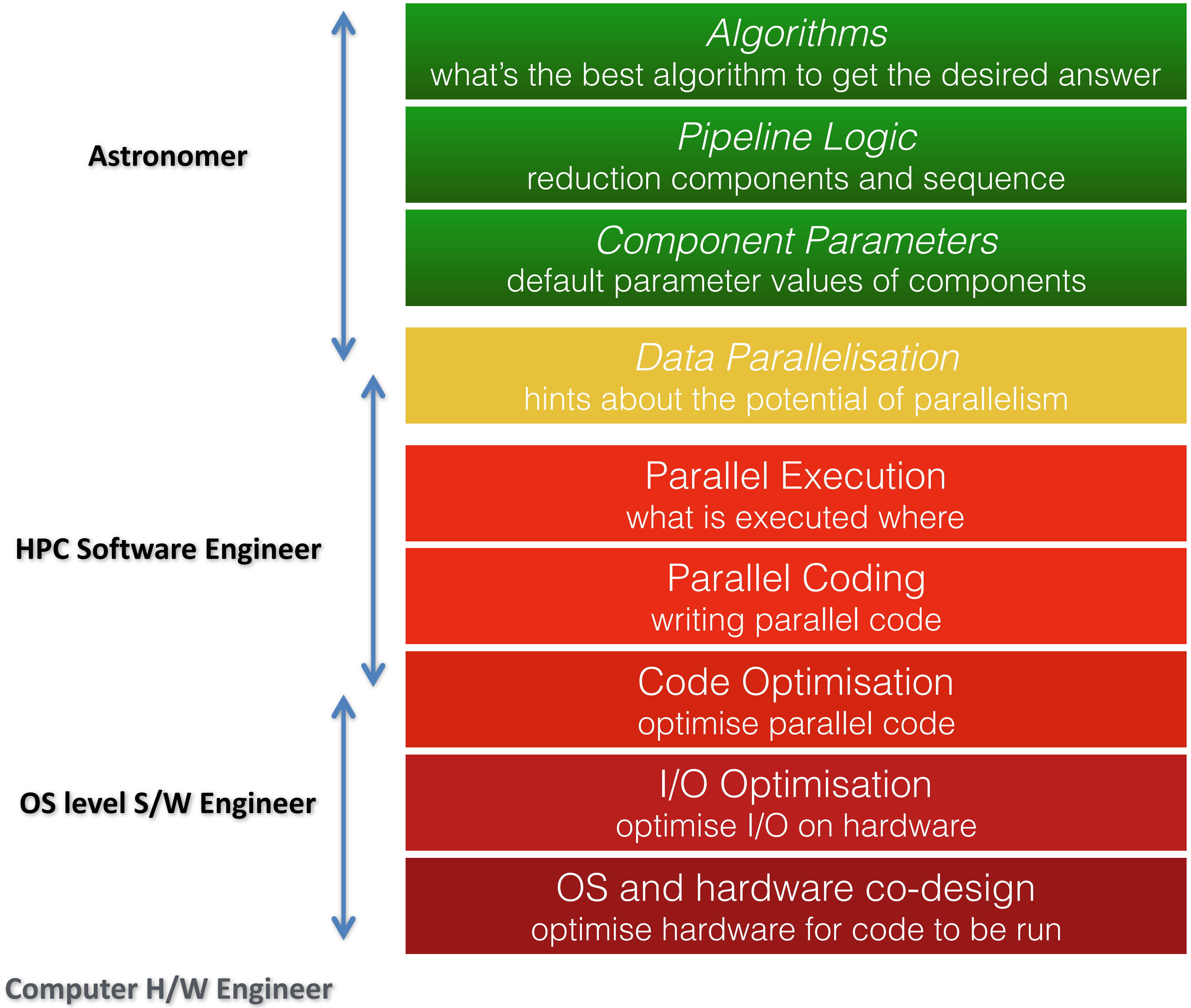
★ Let astronomers think about and do astronomy:

- Astronomical algorithms
- Pipeline logic
- Novel ways of extracting science
- New science
- Interpretation of extracted information
- Training of AI methods
- ….

Separation of concerns

★ Let software engineers think about and write software:

- Optimised code using the most appropriate language
- Novel ways of using latest hardware
- Using modern I/O techniques.
- Using advanced DB technologies
- Parallel code (even only a few software engineers can do this well!)
- HPC coding (even less people can do this well!)
- ….



Astronomer

Algorithms

what's the best algorithm to get the desired answer

Pipeline Logic

reduction components and sequence

Component Parameters

default parameter values of components

Data Parallelisation

hints about the potential of parallelism

HPC Software Engineer

Parallel Execution

what is executed where

Parallel Coding

writing parallel code

Code Optimisation

optimise parallel code

OS level S/W Engineer

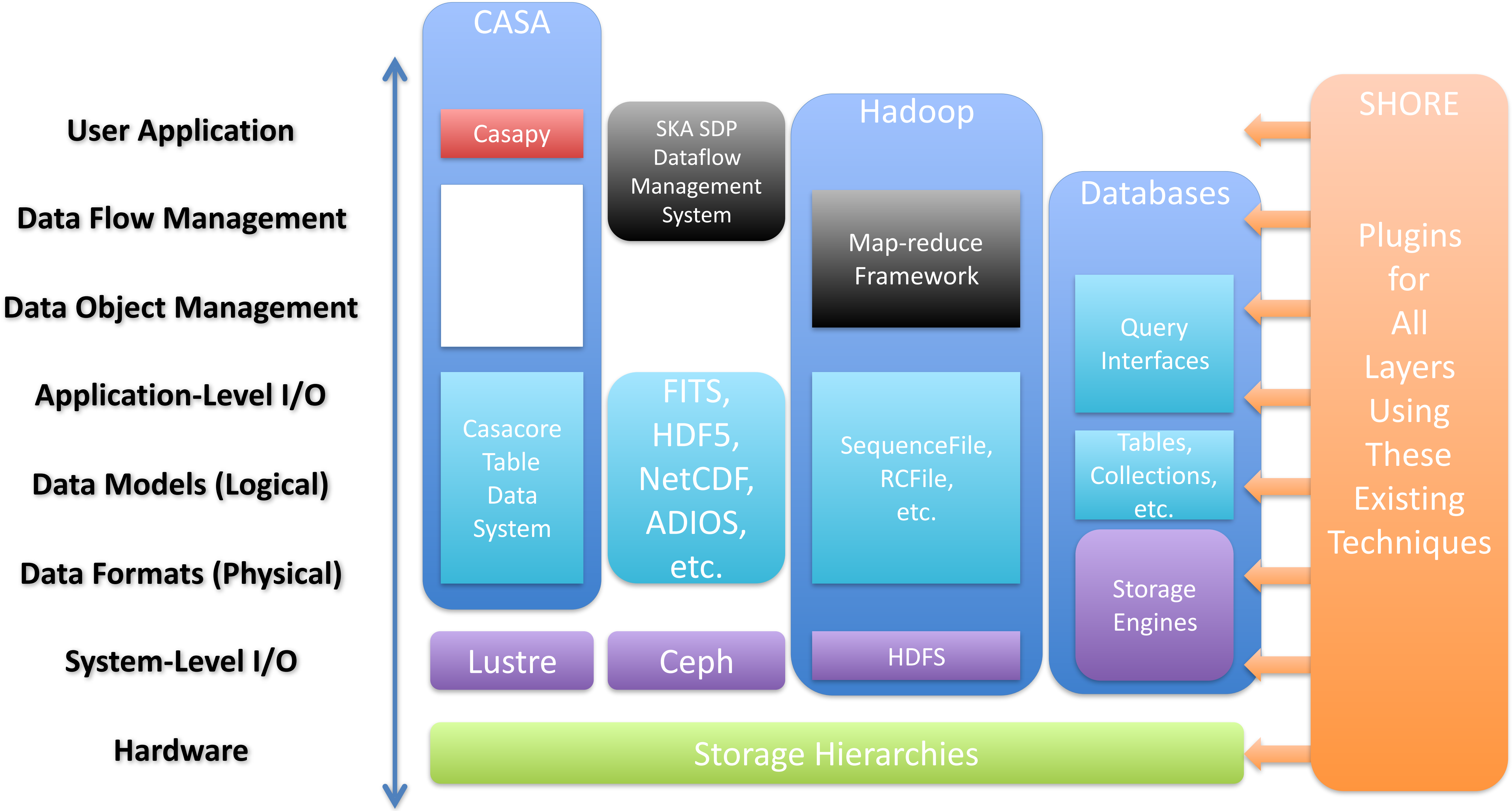
I/O Optimisation

optimise I/O on hardware

OS and hardware co-design

optimise hardware for code to be run

Computer H/W Engineer



User Application

Data Flow Management

Data Object Management

Application-Level I/O

Data Models (Logical)

Data Formats (Physical)

System-Level I/O

Hardware

CASA

Casapy

Casacore Table Data System

SKA SDP Dataflow Management System

FITS, HDF5, NetCDF, ADIOS, etc.

Hadoop

Map-reduce Framework

SequenceFile, RCFile, etc.

HDFS

Databases

Query Interfaces

Tables, Collections, etc.

Storage Engines

Lustre

Ceph

Storage Hierarchies

SHORE

Plugins for All Layers Using These Existing Techniques

courtesy Ruonan Wang

SKA and Precursors require even more attention...

We are at the limit!

★ SKA and ASKAP

- are producing very high data volumes at very high rates
- are a challenge for currently available compute infrastructures (at least at affordable costs)

★ that means

- just throwing more hardware at the problem won't do the trick anymore.
- we need to use existing hardware more efficiently.
- at least the SKA requires significant innovation in order to approach the science potential of the arrays.

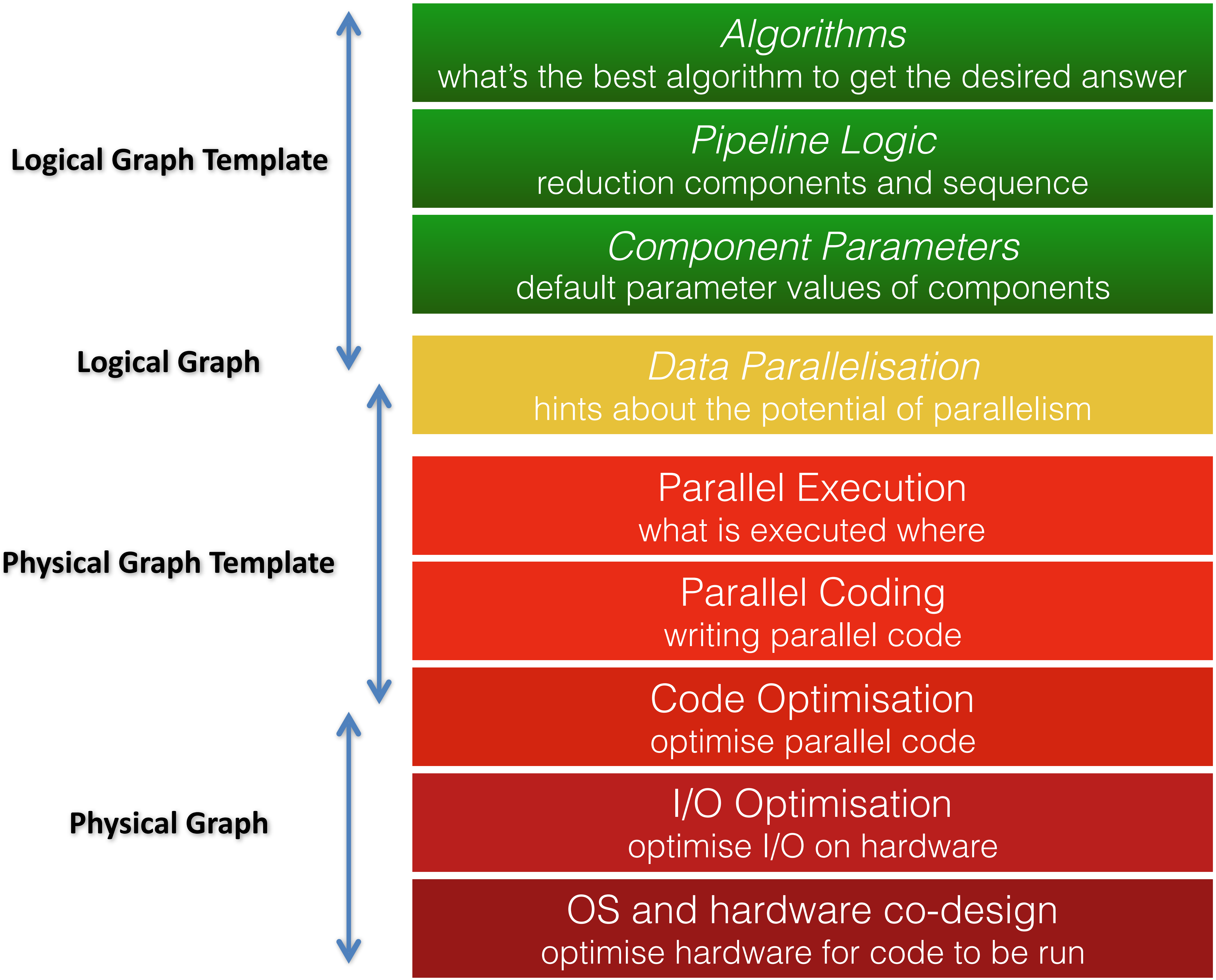
… A TINY BIT OF INNOVATION

Data Triggered Processing enabled by

Daliuge!

Data activated 流 flow graph engine

...think about deluge!



Logical Graph Template

Logical Graph

Physical Graph Template

Physical Graph

Algorithms
what's the best algorithm to get the desired answer

Pipeline Logic
reduction components and sequence

Component Parameters
default parameter values of components

Data Parallelisation
hints about the potential of parallelism

Parallel Execution
what is executed where

Parallel Coding
writing parallel code

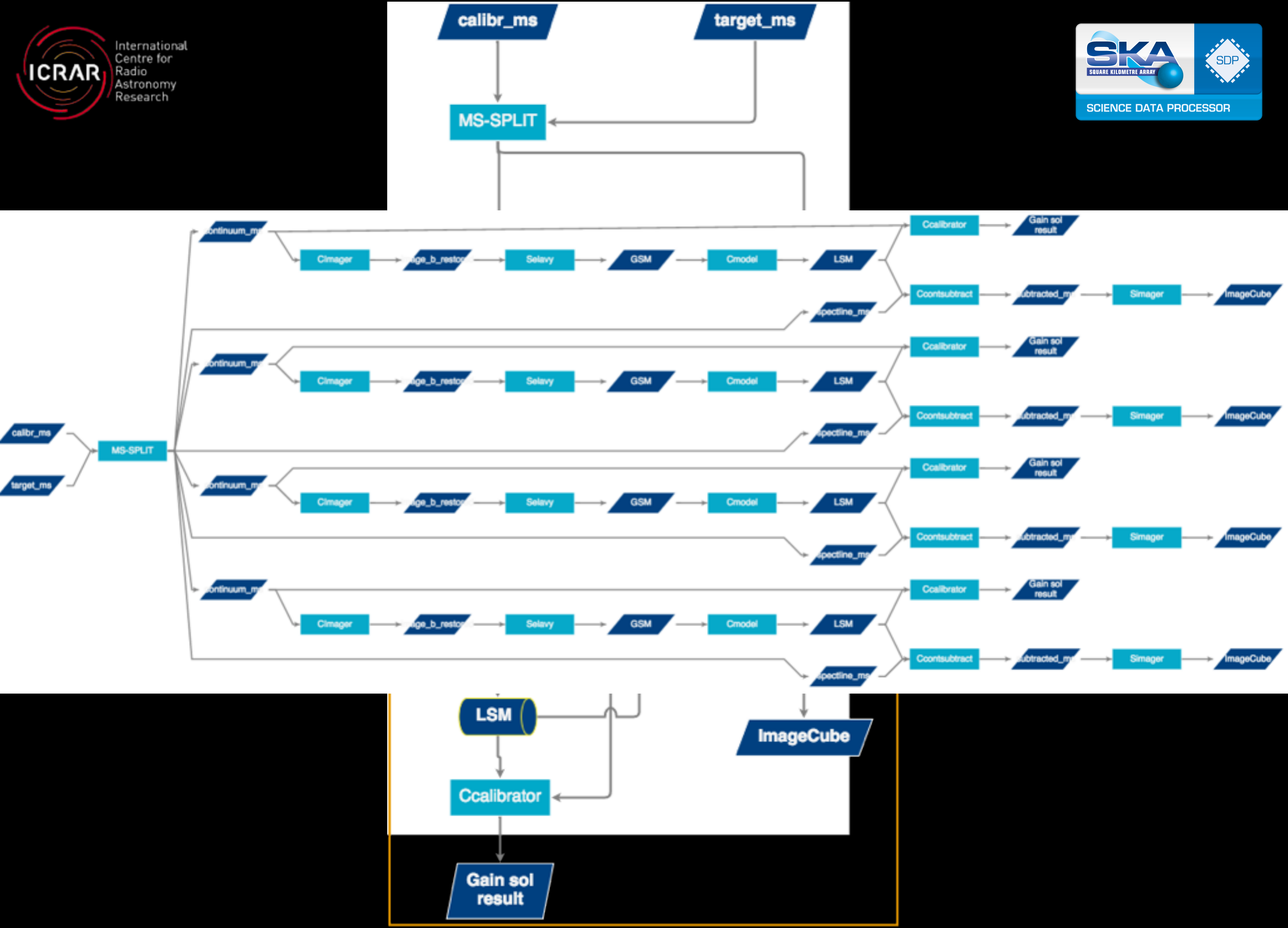
Code Optimisation
optimise parallel code

I/O Optimisation
optimise I/O on hardware

OS and hardware co-design
optimise hardware for code to be run

International Centre for Radio Astronomy Research

SCIENCE DATA PROCESSOR



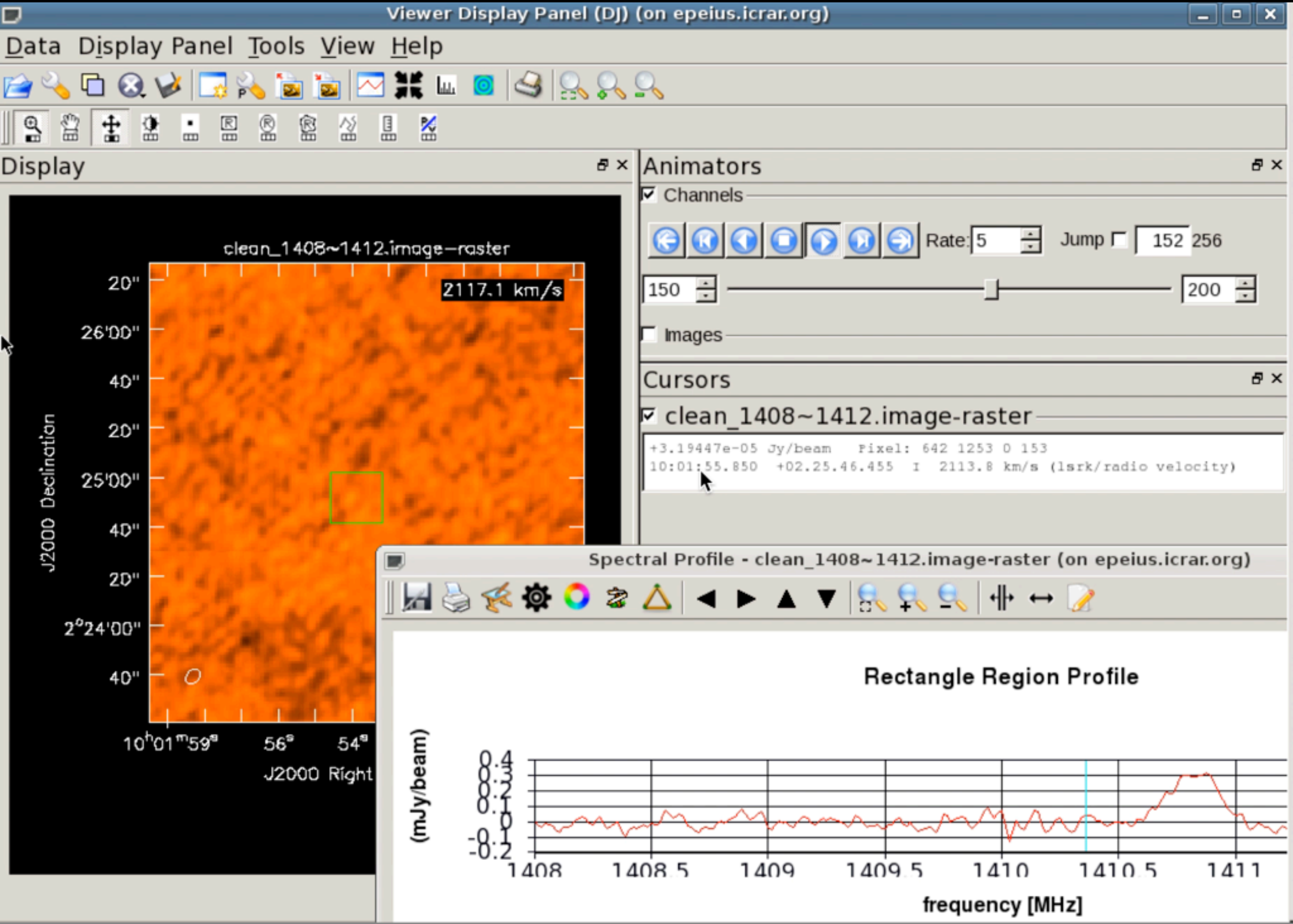
Real World Examples

CHILES

- Dailuge has been verified using CHILES data on AWS, in-house cluster, Magnus and Galaxy.
- The current code version of the code creates 40+ Node managers all running on separate heterogeneous AWS instances; with a single Data Island Manager controlling them.
- The graphs contain 7,000 ~ 18,000 Drops
- The graph generator knows the AWS instance types and can deploy more CPU/IO intensive tasks to more powerful nodes.
- The CasaPy tasks are all run from within Docker containers controlled by the Dailuge

...and ASKAP?

- CHILES is a small version of the DINGO survey (mainly larger field of view).
- If we can deal with CHILES, DINGO is not too far off.
- We are currently wrapping ASKAPsoft into Daliuge *Drops*.
- *Drops* are software objects and the enabling core elements of Daliuge.
- The various nodes on the graphs are all implemented as *Drops* in Daliuge.



Viewer Display Panel (DJ) (on epeius.icrar.org)

Animators

☑ Channels

Rate: 5 Jump □ 152 256

150 200

□ Images

Cursors

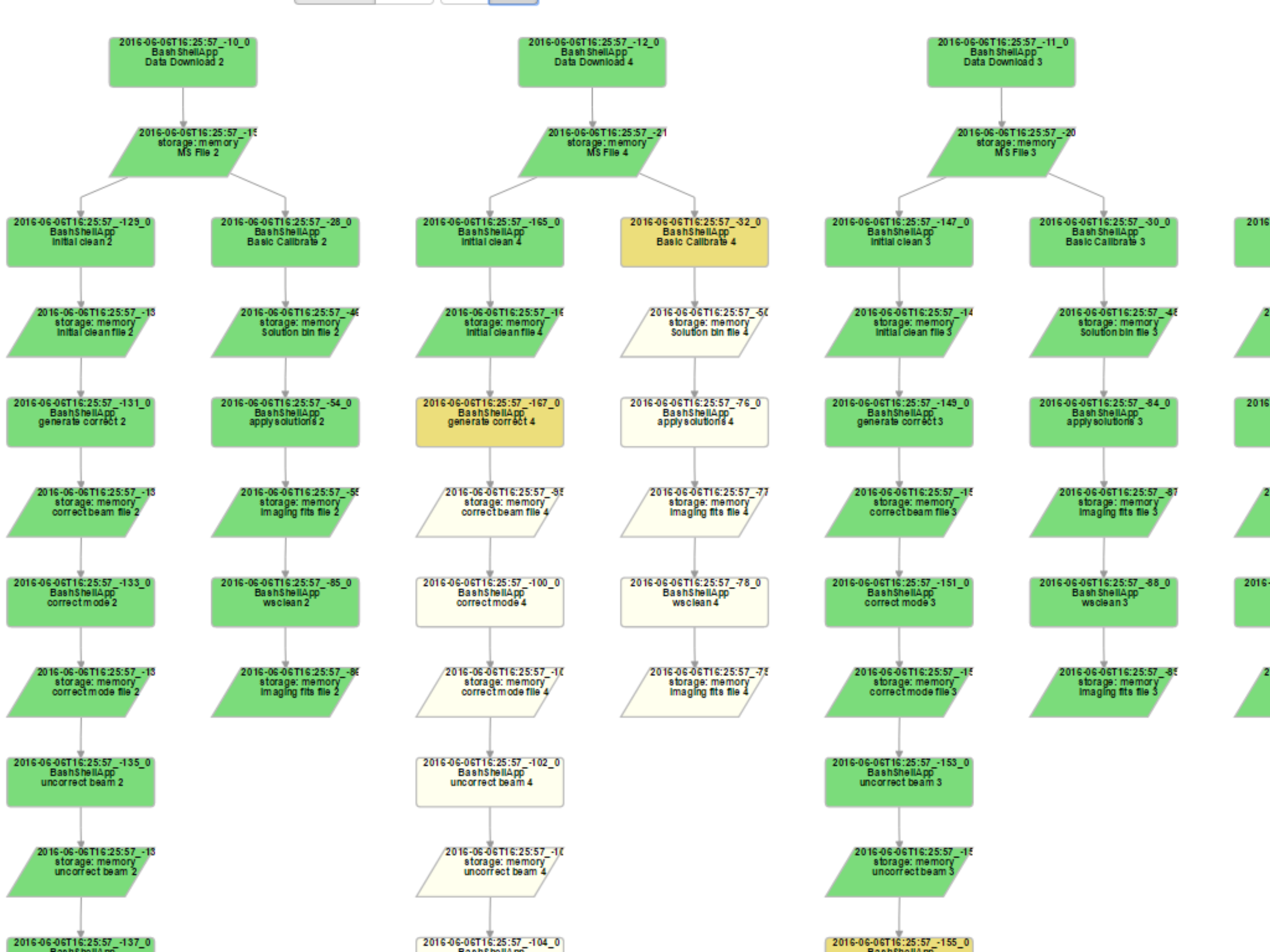
☑ clean_1408~1412.image-raster

```
+3.19447e-05 Jy/beam   Pixel: 642 1253 0 153
10:01:55.850  +02.25.46.455  I  2113.8 km/s (lsrk/radio velocity)
```

Spectral Profile - clean_1408~1412.image-raster (on epeius.icrar.org)

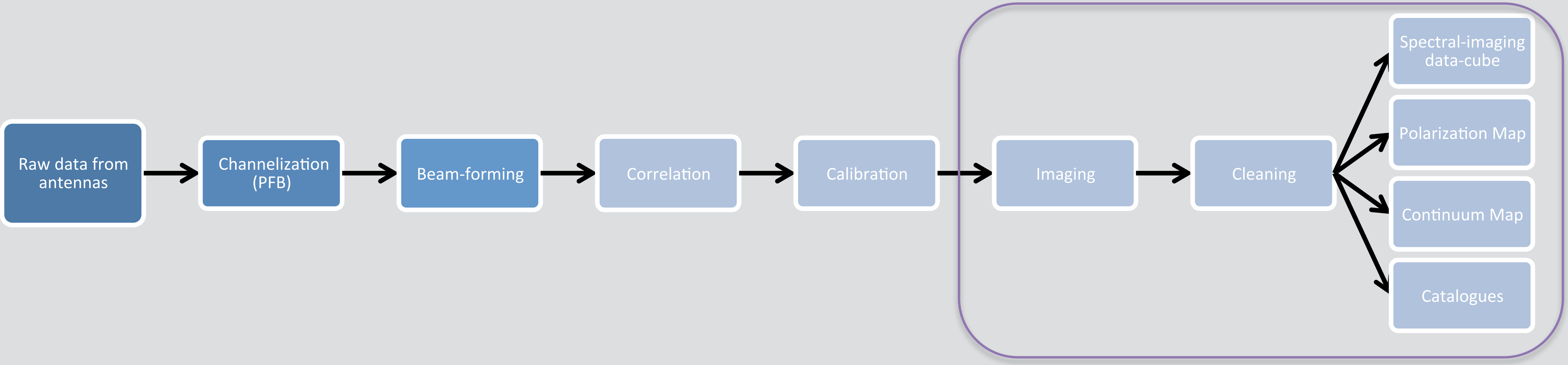
…and more

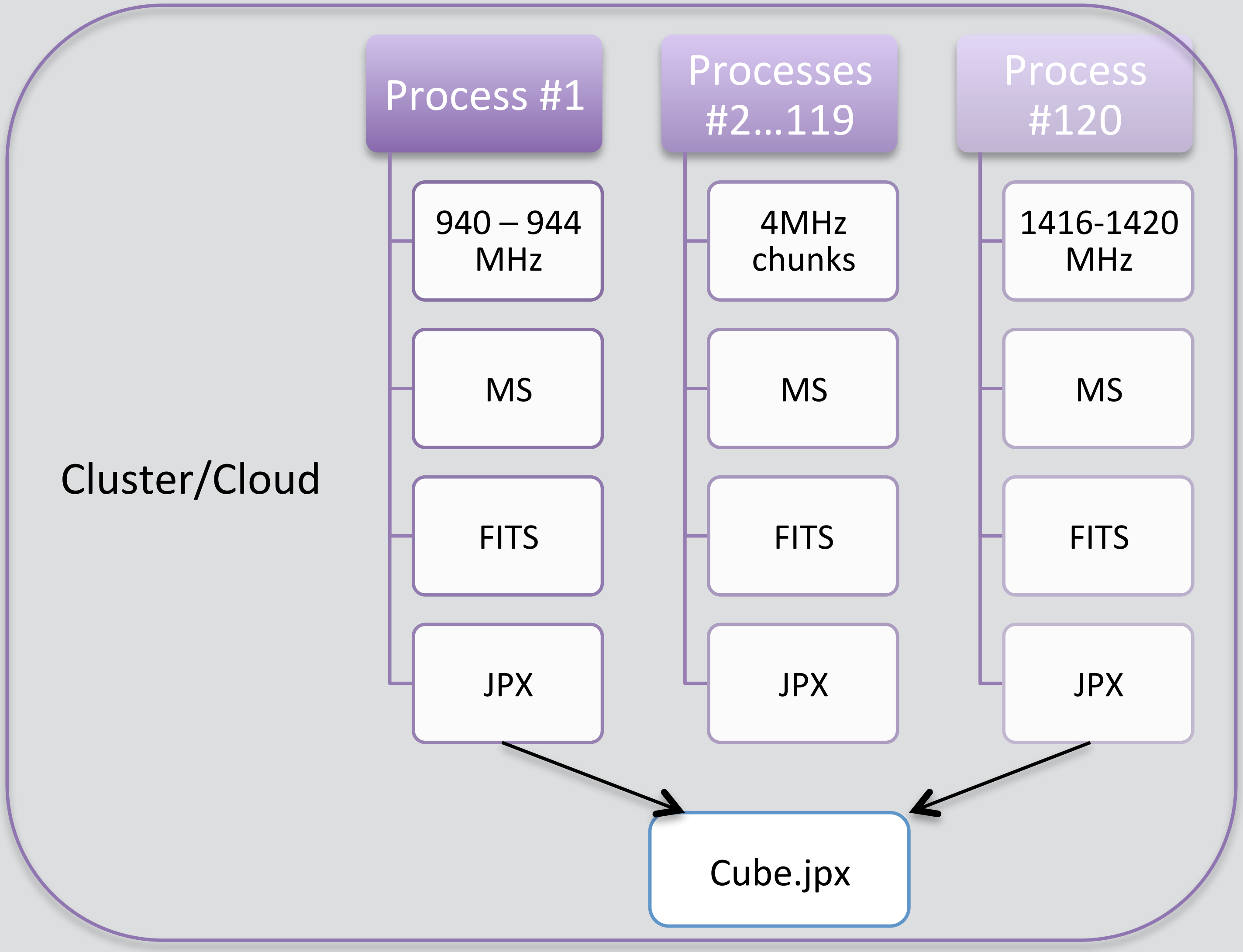
- we have ported the MWA GLEAM pipeline to Daliuge.
- ASTRON is working to port and run the LOFAR pipeline.
- Fudan University wants to run a movie encoding and analysis pipeline.
- we are also integrating OSKAR2 and can simulate and reduce MWA and ASKAP data.
- code is available on SKA SDP github.
- documented and fully tested code (continuous integration with loads of test code)
- Graph translation and scheduling is a really hard problem…


2016-06-06T16:25:57_-10_0
BashShellApp
Data Download 2
2016-06-06T16:25:57_-15
storage: memory
MS File 2
2016-06-06T16:25:57_-129_0
BashShellApp
Initial clean 2
2016-06-06T16:25:57_-13
storage: memory
initial clean file 2
2016-06-06T16:25:57_-131_0
BashShellApp
generate correct 2
2016-06-06T16:25:57_-13
storage: memory
correct beam file 2
2016-06-06T16:25:57_-133_0
BashShellApp
correct mode 2
2016-06-06T16:25:57_-13
storage: memory
correct mode file 2
2016-06-06T16:25:57_-135_0
BashShellApp
uncorrect beam 2
2016-06-06T16:25:57_-13
storage: memory
uncorrect beam 2
2016-06-06T16:25:57_-137_0
BashShellApp
2016-06-06T16:25:57_-28_0
BashShellApp
Basic Calibrate 2
2016-06-06T16:25:57_-49
storage: memory
Solution bin file 2
2016-06-06T16:25:57_-54_0
BashShellApp
apply solutions 2
2016-06-06T16:25:57_-55
storage: memory
imaging fits file 2
2016-06-06T16:25:57_-85_0
BashShellApp
wsclean 2
2016-06-06T16:25:57_-86
storage: memory
imaging fits file 2
2016-06-06T16:25:57_-12_0
BashShellApp
Data Download 4
2016-06-06T16:25:57_-21
storage: memory
MS File 4
2016-06-06T16:25:57_-165_0
BashShellApp
Initial clean 4
2016-06-06T16:25:57_-16
storage: memory
initial clean file 4
2016-06-06T16:25:57_-167_0
BashShellApp
generate correct 4
2016-06-06T16:25:57_-98
storage: memory
correct beam file 4
2016-06-06T16:25:57_-100_0
BashShellApp
correct mode 4
2016-06-06T16:25:57_-10
storage: memory
correct mode file 4
2016-06-06T16:25:57_-102_0
BashShellApp
uncorrect beam 4
2016-06-06T16:25:57_-10
storage: memory
uncorrect beam 4
2016-06-06T16:25:57_-104_0
BashShellApp
2016-06-06T16:25:57_-32_0
BashShellApp
Basic Calibrate 4
2016-06-06T16:25:57_-50
storage: memory
Solution bin file 4
2016-06-06T16:25:57_-76_0
BashShellApp
apply solutions 4
2016-06-06T16:25:57_-77
storage: memory
imaging fits file 4
2016-06-06T16:25:57_-78_0
BashShellApp
wsclean 4
2016-06-06T16:25:57_-75
storage: memory
imaging fits file 4
2016-06-06T16:25:57_-11_0
BashShellApp
Data Download 3
2016-06-06T16:25:57_-20
storage: memory
MS File 3
2016-06-06T16:25:57_-147_0
BashShellApp
Initial clean 3
2016-06-06T16:25:57_-14
storage: memory
initial clean file 3
2016-06-06T16:25:57_-149_0
BashShellApp
generate correct 3
2016-06-06T16:25:57_-15
storage: memory
correct beam file 3
2016-06-06T16:25:57_-151_0
BashShellApp
correct mode 3
2016-06-06T16:25:57_-15
storage: memory
correct mode file 3
2016-06-06T16:25:57_-153_0
BashShellApp
uncorrect beam 3
2016-06-06T16:25:57_-15
storage: memory
uncorrect beam 3
2016-06-06T16:25:57_-155_0
BashShellApp
2016-06-06T16:25:57_-30_0
BashShellApp
Basic Calibrate 3
2016-06-06T16:25:57_-48
storage: memory
Solution bin file 3
2016-06-06T16:25:57_-84_0
BashShellApp
apply solutions 3
2016-06-06T16:25:57_-87
storage: memory
imaging fits file 3
2016-06-06T16:25:57_-88_0
BashShellApp
wsclean 3
2016-06-06T16:25:57_-85
storage: memory
imaging fits file 3
2016
2016
2016

Visualisation of TB and PB data cubes

JPEG2000 and JPIP

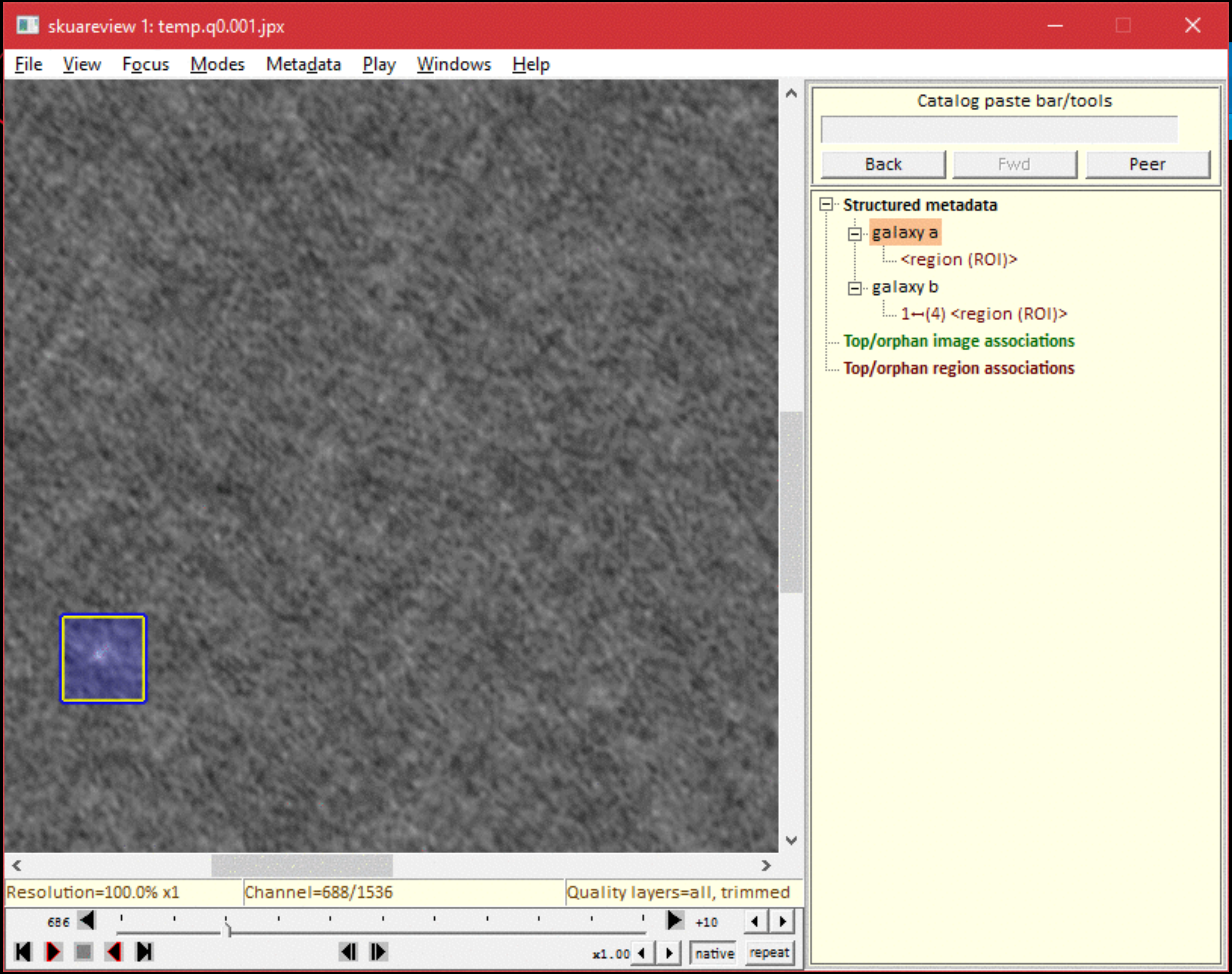




courtesy Slava Kitaeff and JT Malarecki

Advanced Features

- Commercial standard backed by many companies and OSs already.
- Highly optimised implementations.
- Multi-dimensional encoding.
- Distributed client-server infrastructure for interactive low-bandwidth adaptive visualisation.
- Multi-component transforms built-in.
- Very rich and flexible metadata (keep all of FITS, plus a lot more…)
- Region of interests built-in with support for quality variation, i.e. rather than overlays, catalogues can be built-in.



skuareview 1: temp.q0.001.jpx

Structured metadata
- galaxy a
  - <region (ROI)>
- galaxy b
  - 1↔(4) <region (ROI)>
- Top/orphan image associations
- Top/orphan region associations

Resolution=100.0% x1

Channel=688/1536

Quality layers=all, trimmed

Next Steps

- Large and very large scale deployments:
  - Magnus@Pawsey ✔
  - Tianhe2 (almost there)
- More real-world use cases (logical graphs of *your* pipeline).
- Profiling of existing code (e.g. ASKAPsoft).
- Work on better parallelisation.
- Work on optimisation of key algorithms.
- Collaboration with other organisations and companies.

Conclusions

Questions

★ Who should build:

- Dishes?
- Receivers?
- Beam formers?
- Correlators??
- Operational Software???
- Astronomy software????

ON body Execute with Luigi Rotate the graph