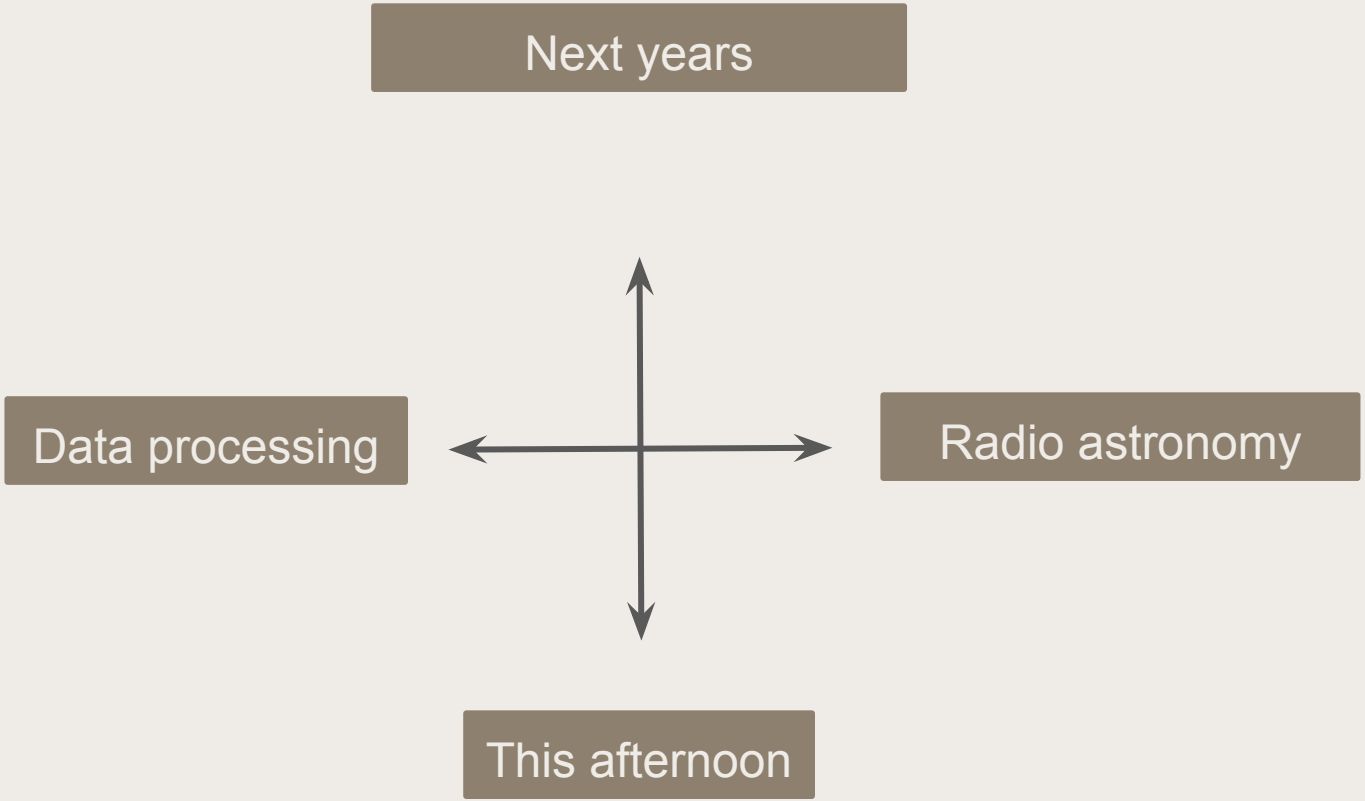
The SKA Telescope Data Deluge

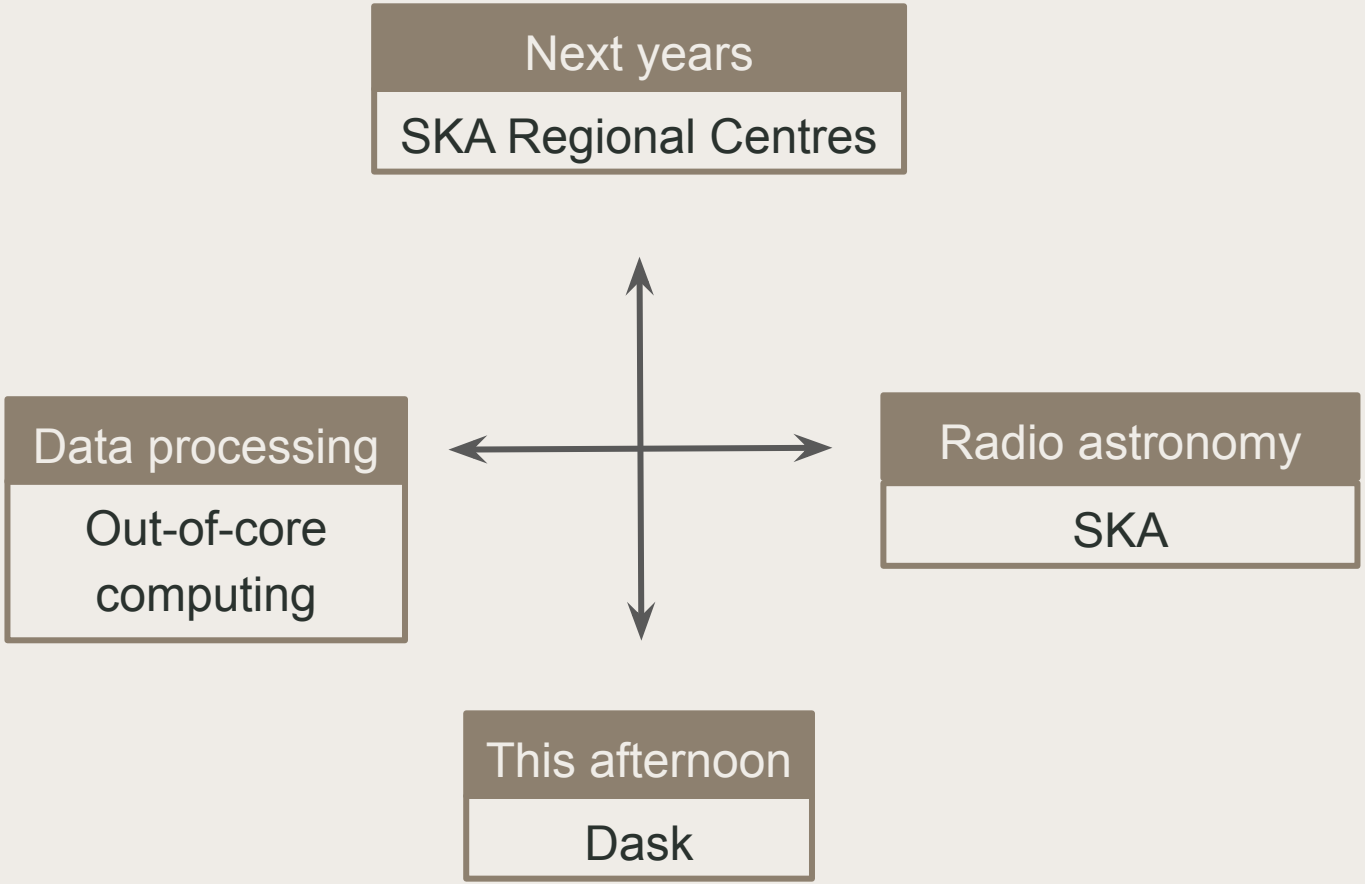
Javier Moldón
IAA-CSIC

SOMACHINE 2020 (Granada - 📡)
November 27, 2020

Outline


Next years
Data processing
Radio astronomy
This afternoon

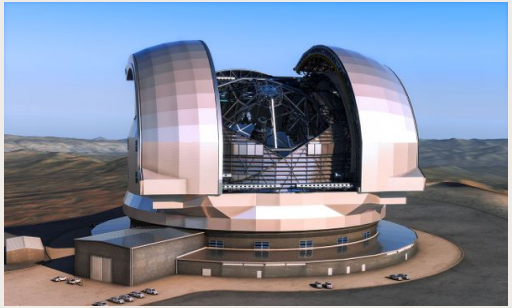
Outline


Next years
SKA Regional Centres
Data processing
Out-of-core computing
Radio astronomy
SKA
This afternoon
Dask

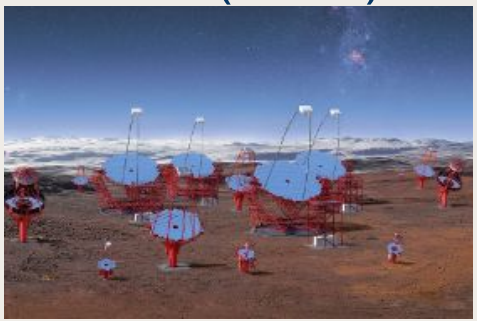
The SKA Telescope

ALMA

e-ELT/TMT (2025/27)

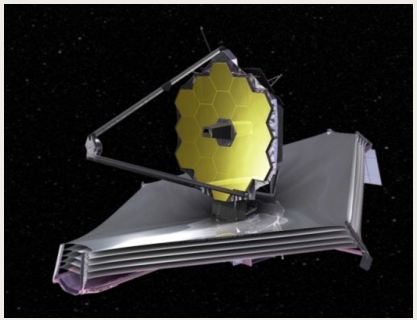


CTA (2025)



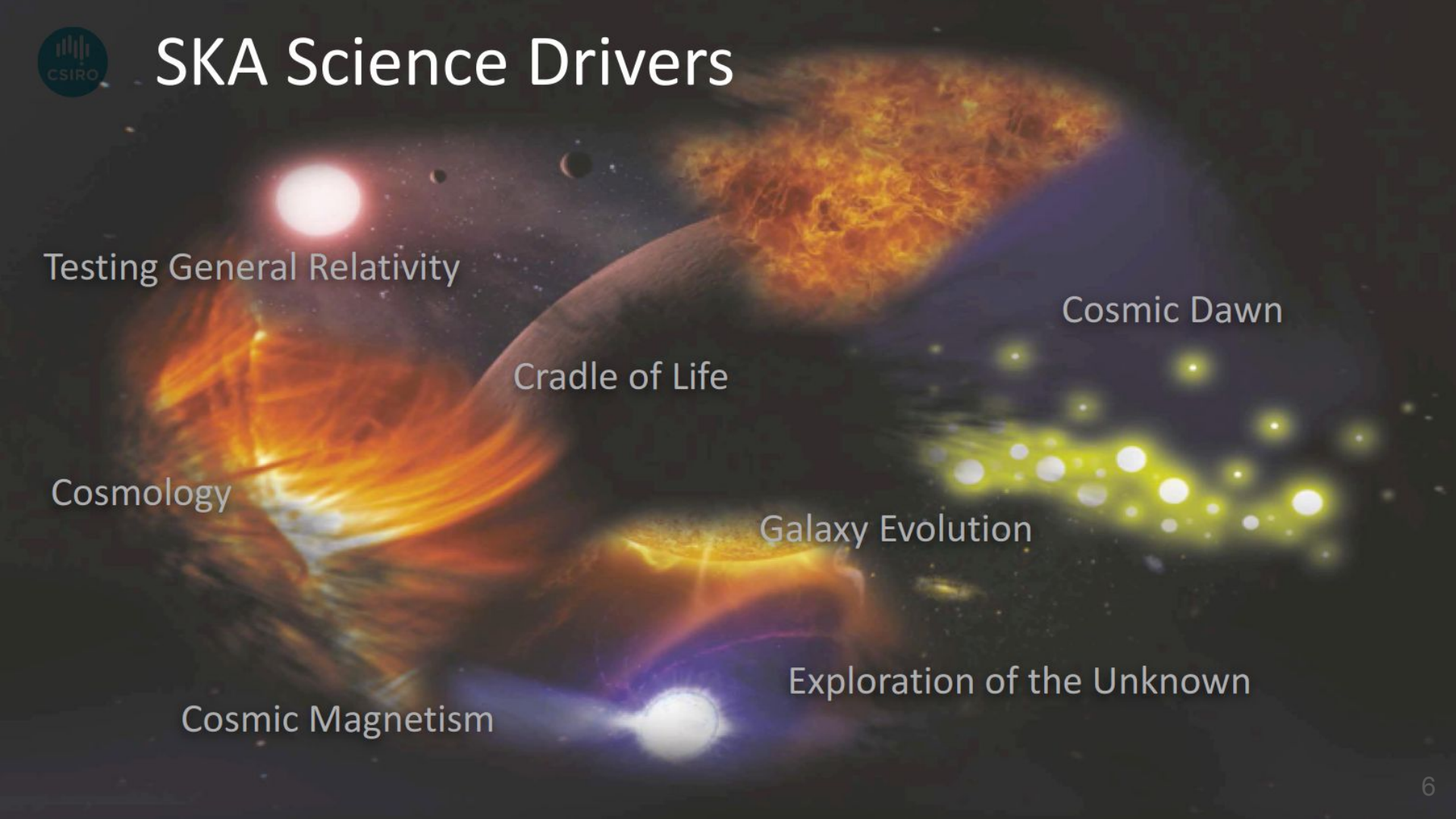
SKA (2028)

JWST (2021)



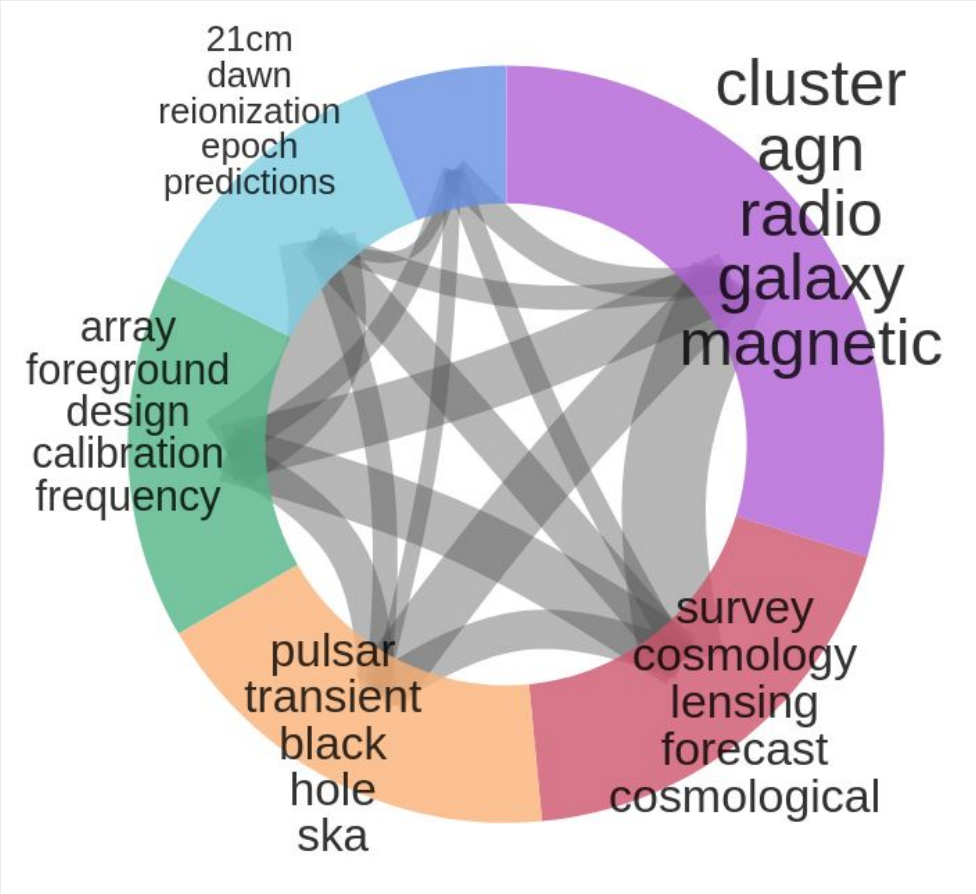
Athena (2021)

SKA Science Drivers



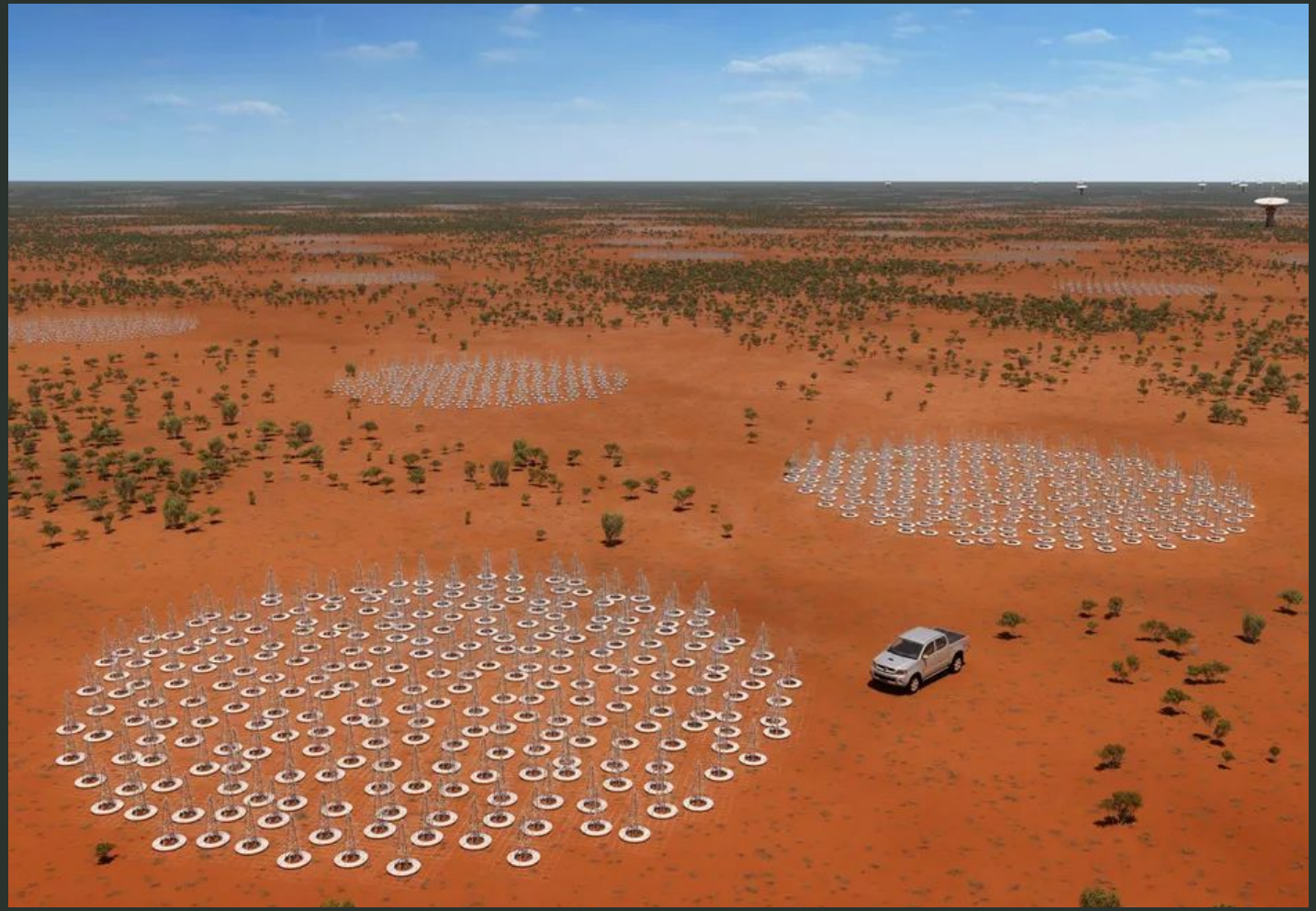
SKA papers

ADS: 270+ refereed papers with SKA in the title.
A lot of science **already** going on



SKA global footprint

SKA as an IGO: intergovernmental organization



Quick summary of the array

SKA1-mid

the SKA's mid-frequency instrument

Location: South Africa

Frequency range:
350 MHz
to
15.3 GHz
with a goal of 24 GHz

197 dishes
(including 64 MeerKAT dishes)

Maximum baseline:
150km

SKA1-low

the SKA's low-frequency instrument

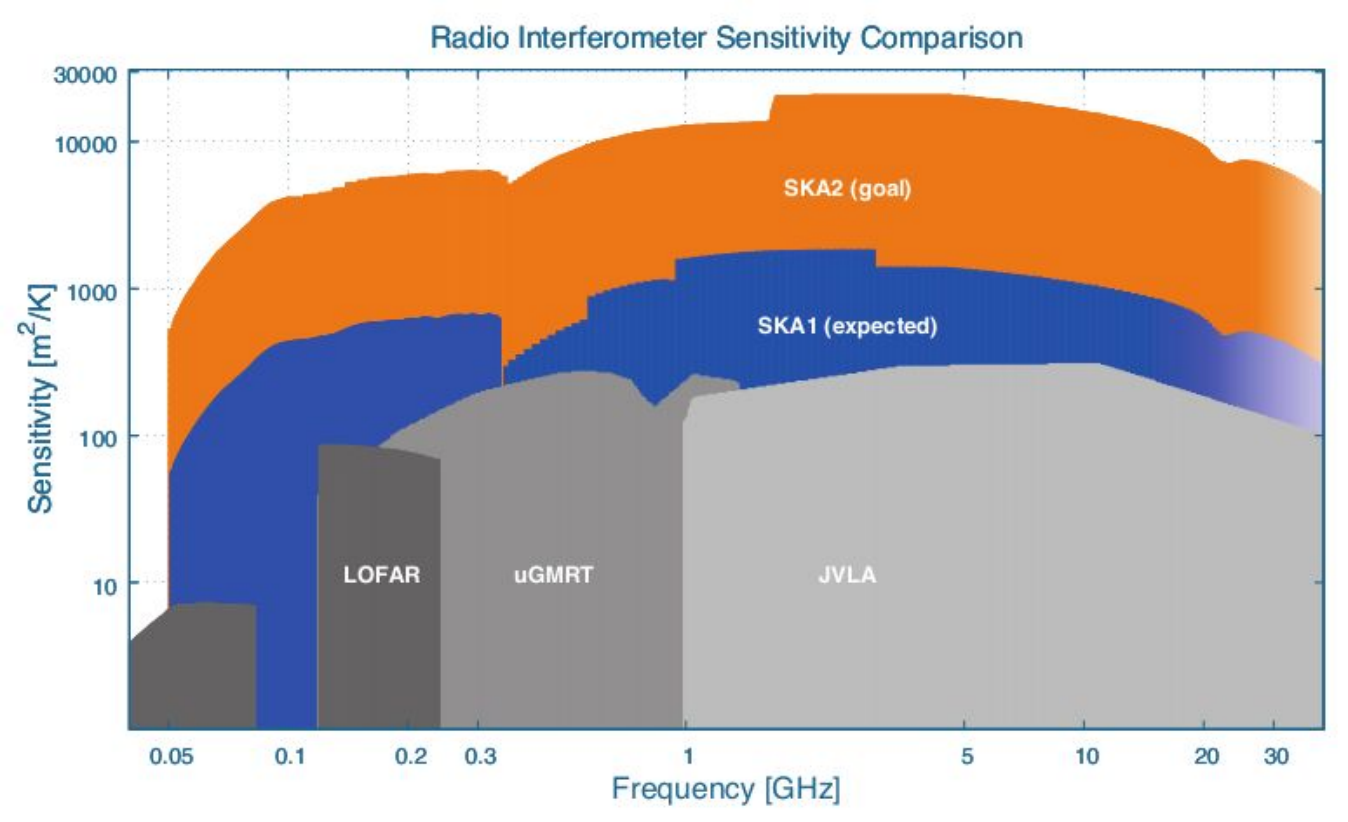
Location: Australia

Frequency range:
50 MHz
to
350 MHz

~131,000
antennas spread between
512 stations

Maximum baseline:
~65km

Sensitivity



Survey speed

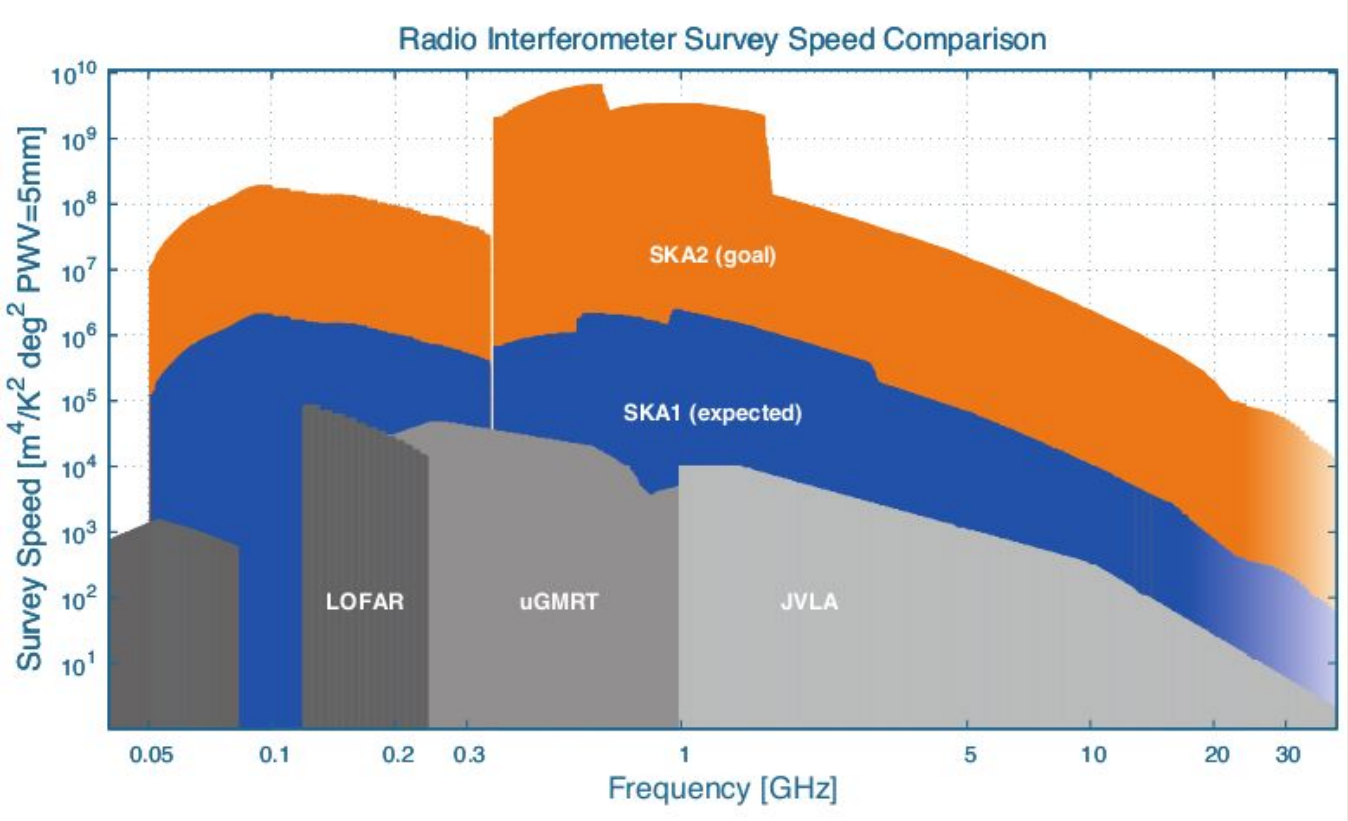
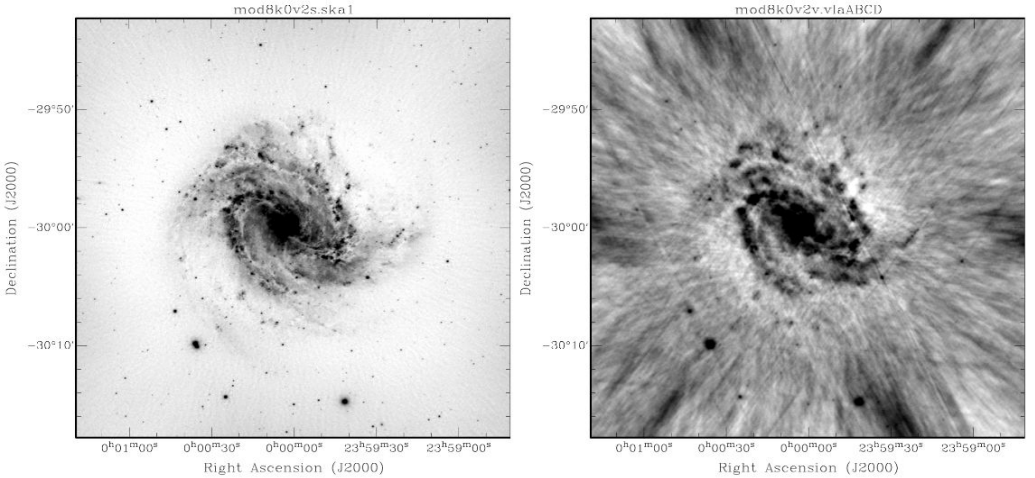


Image fidelity

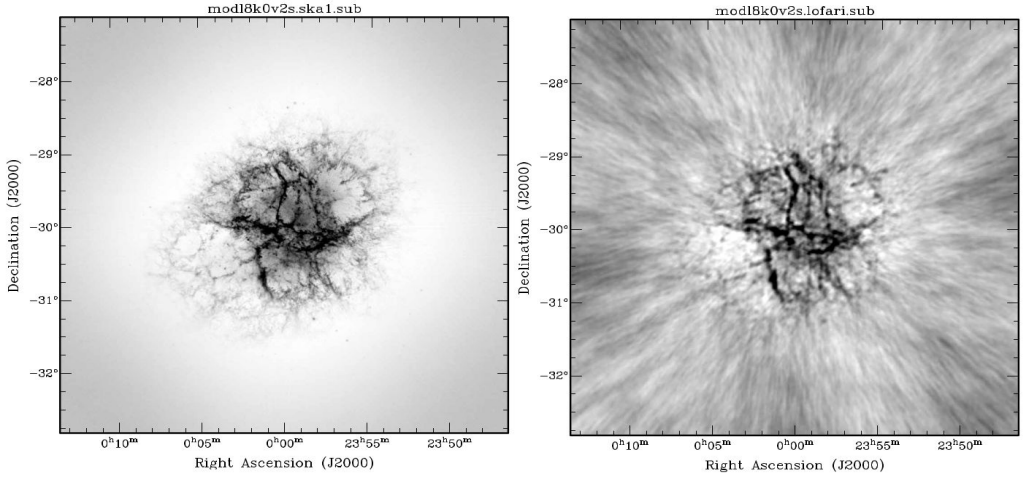
Between 10 and 100 times the image fidelity of current facilities



SKA1-Mid

vs

VLA A+B+C+D snapshots combined

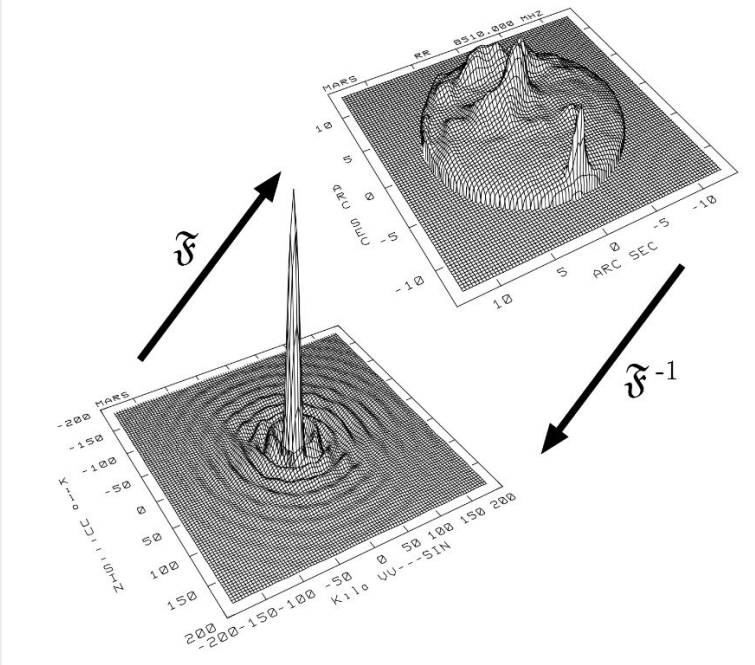


SKA1-Low

vs

LOFAR “dirty” snapshot

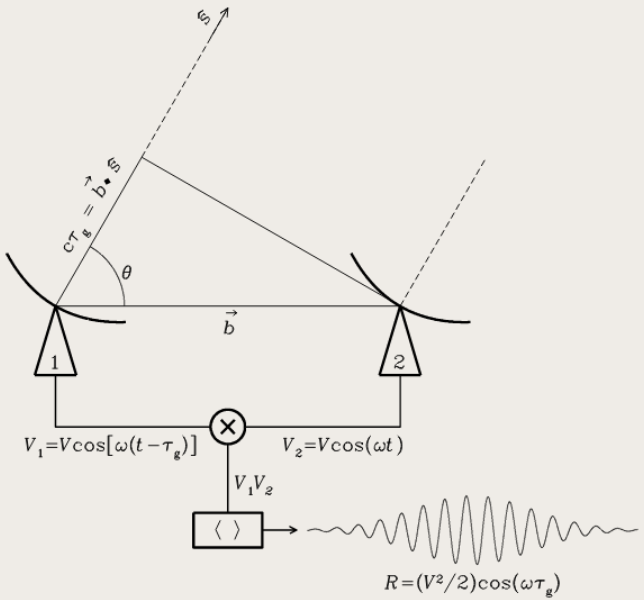
Radio astronomers don’t take images...

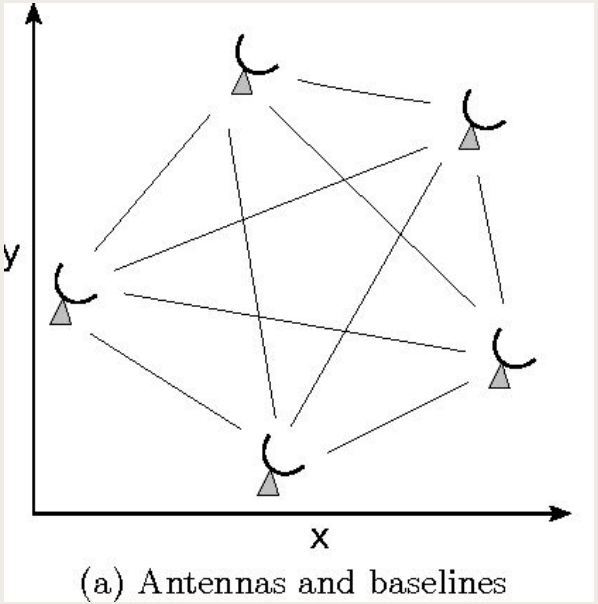


… we synthesize images!

Data acquisition is done per baseline

For N antennas you have **N(N-1)/2** baselines

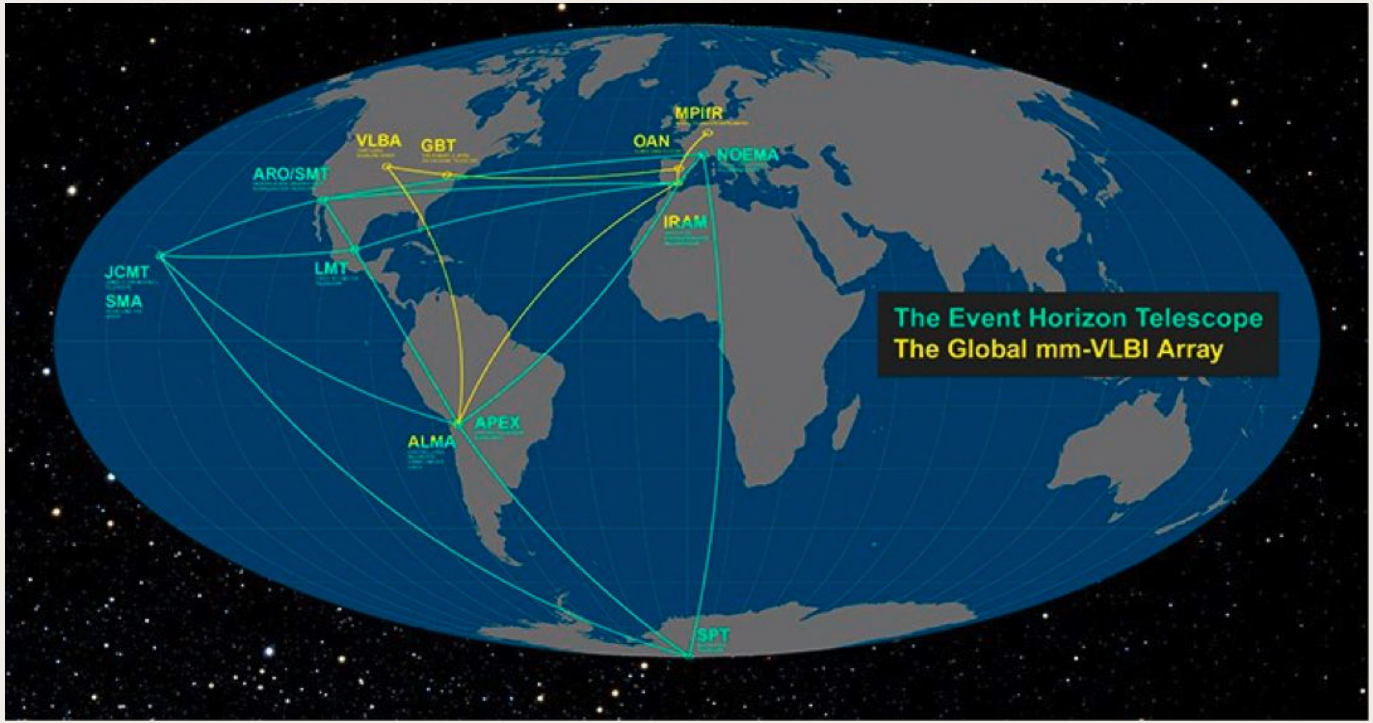




(a) Antennas and baselines

Event Horizon Telescope

7 antennas
21 baselines



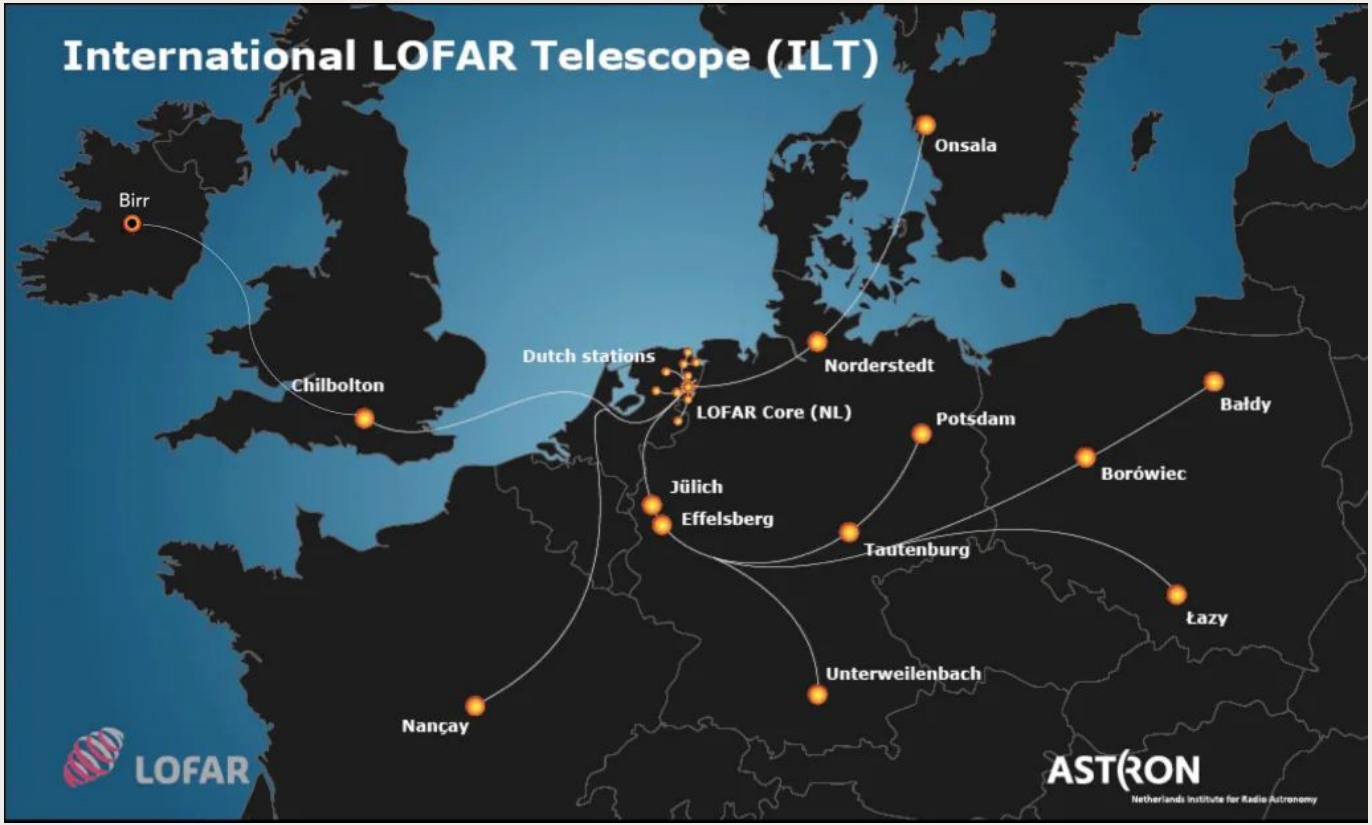
VLA

27 antennas
351 baselines

LOFAR

51 operational stations:
24 CS
14 RS
14 international

1275 baselines



MeerKAT

SKA precursor in South Africa

64 antennas
2016 baselines

Data structure and scale

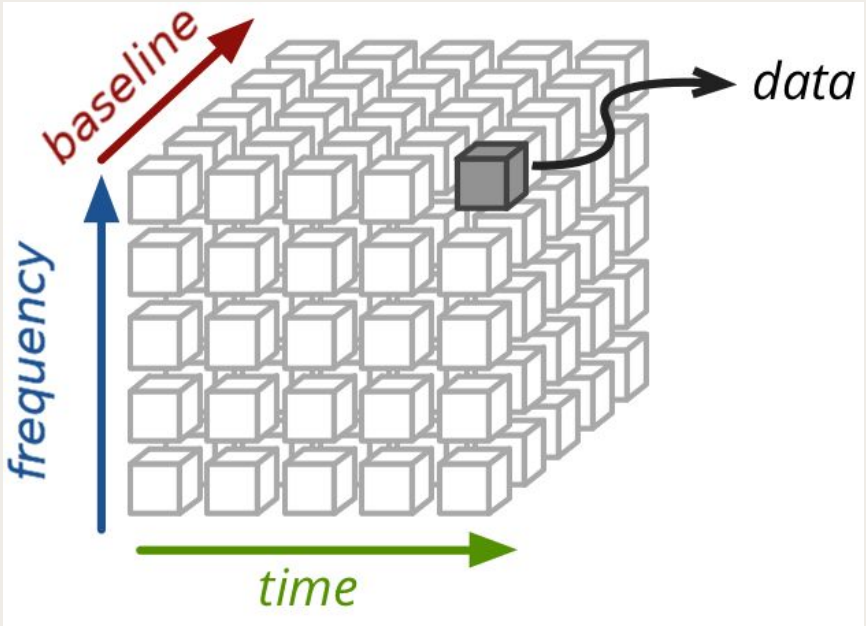
Baselines $\propto N^2$

You cannot integrate (at this stage)

Baselines

Visibility

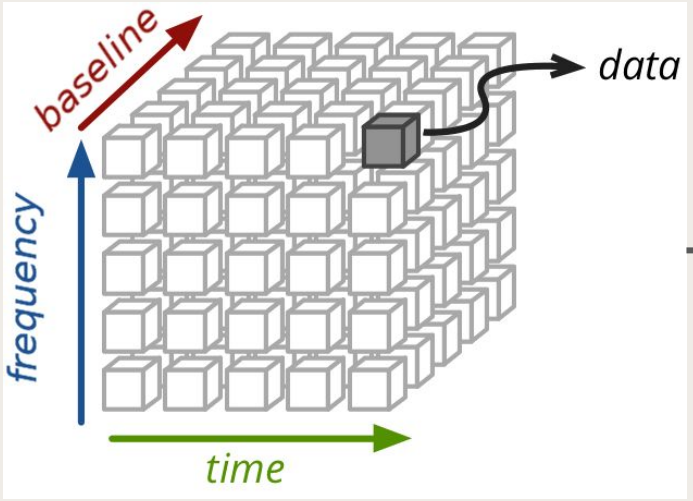
Channels

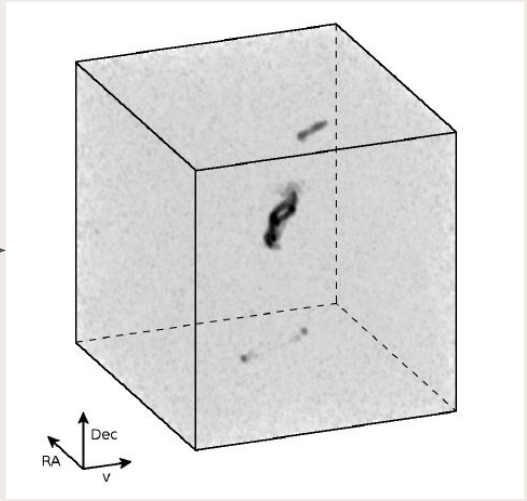


Integration time

Adapted from https://pythonhosted.org/cubes/introduction.html

From data to images

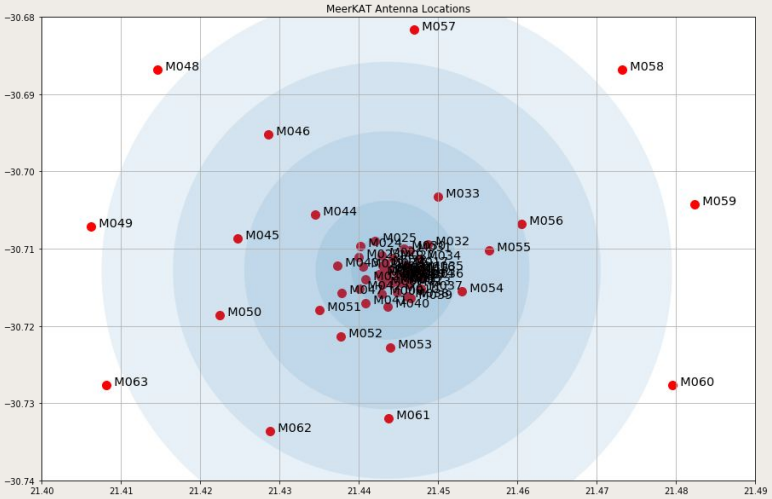




SKA
SQUARE KILOMETRE ARRAY
SDP
SCIENCE DATA PROCESSOR

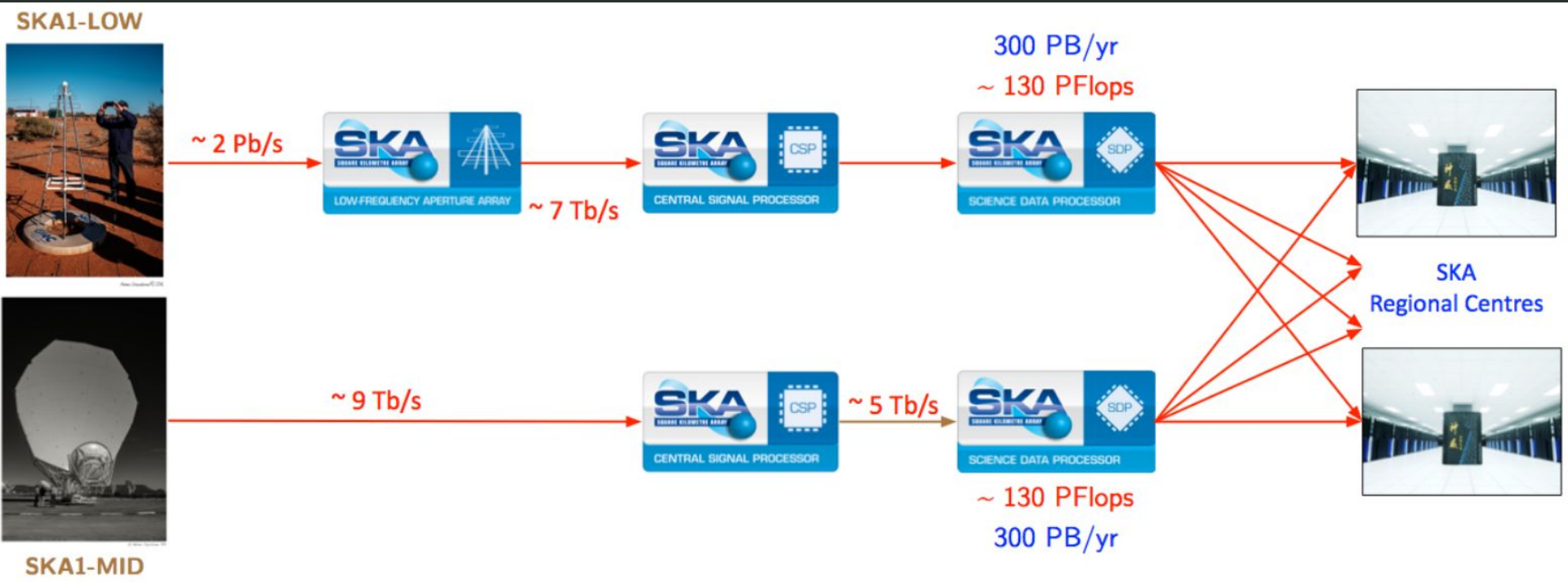
MeerKAT (SKA precursor)

64 antennas
2016 baselines



	Visibilities (TB per hour)		
Visibility integration period	**2 s**	**4 s**	**8 s**
4K mode	0.55	0.28	0.14
32K mode	4.40	2.20	1.10

SKA Data flow



SKA1-LOW

~ 2 Pb/s

LOW-FREQUENCY APERTURE ARRAY

~ 7 Tb/s

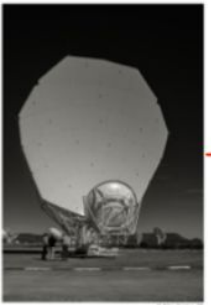
CENTRAL SIGNAL PROCESSOR

300 PB/yr

~ 130 PFlops

SCIENCE DATA PROCESSOR

SKA
Regional Centres



~ 9 Tb/s

CENTRAL SIGNAL PROCESSOR

~ 5 Tb/s

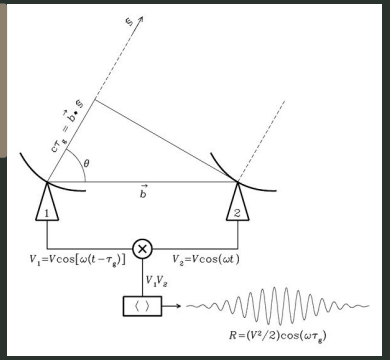
SCIENCE DATA PROCESSOR

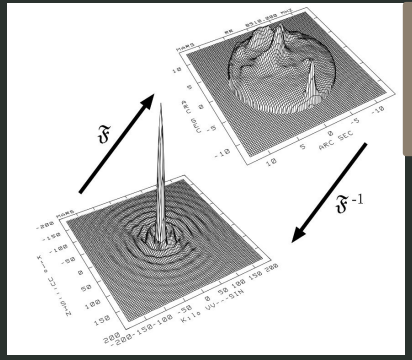
~ 130 PFlops

300 PB/yr

SKA1-MID

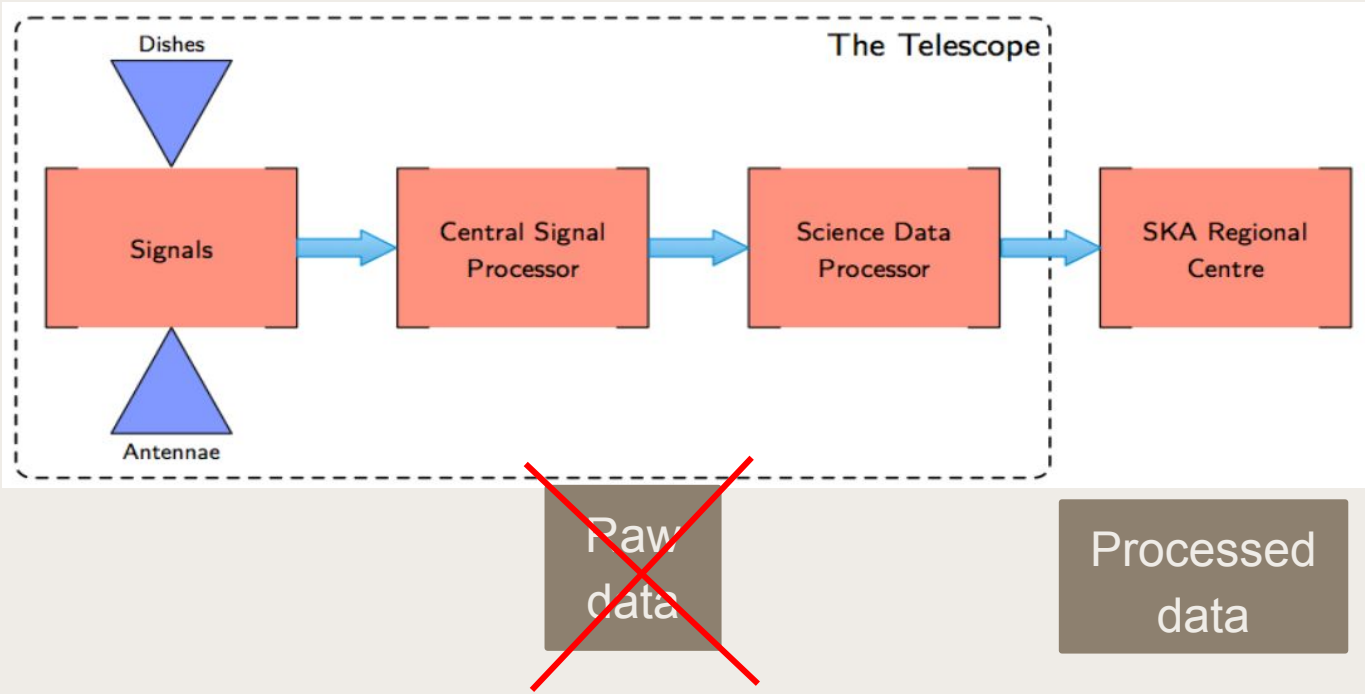
Raw data





Processed data

Data flow

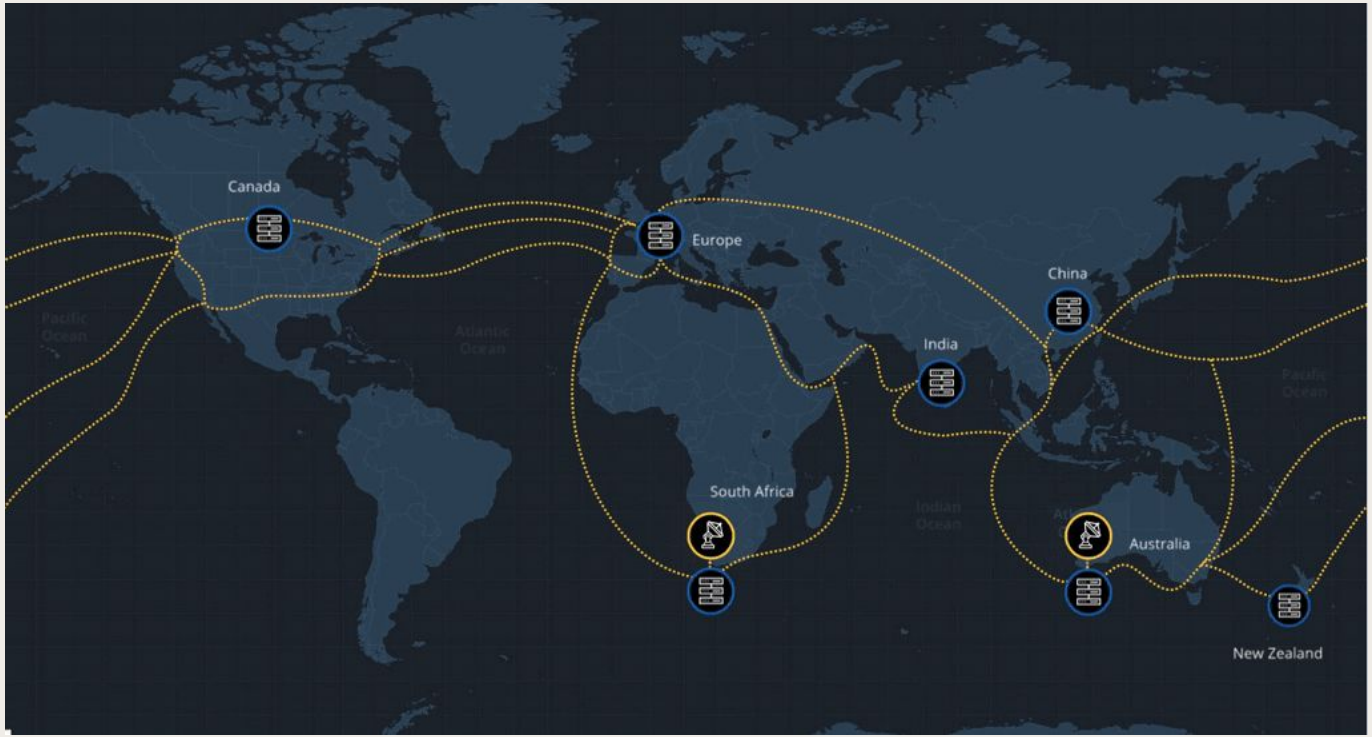

The Telescope
Dishes
Signals
Antennae
Central Signal Processor
Science Data Processor
SKA Regional Centre
Raw data
Processed data

Raw data will not be stored!

Stages of Grief of the SKA **raw data**	
1. Denial	That is wrong, redesign the SKA from scratch!
2. Anger	I will never use the SKA!
3. Bargaining	Send me the raw data, I will store it
4. Depression	Radio astronomy is doomed...
5. Acceptance and hope	OK, I will use the SKA Regional Centres

The SKA Regional Centres (SRC)

SKA Regional Centre Network

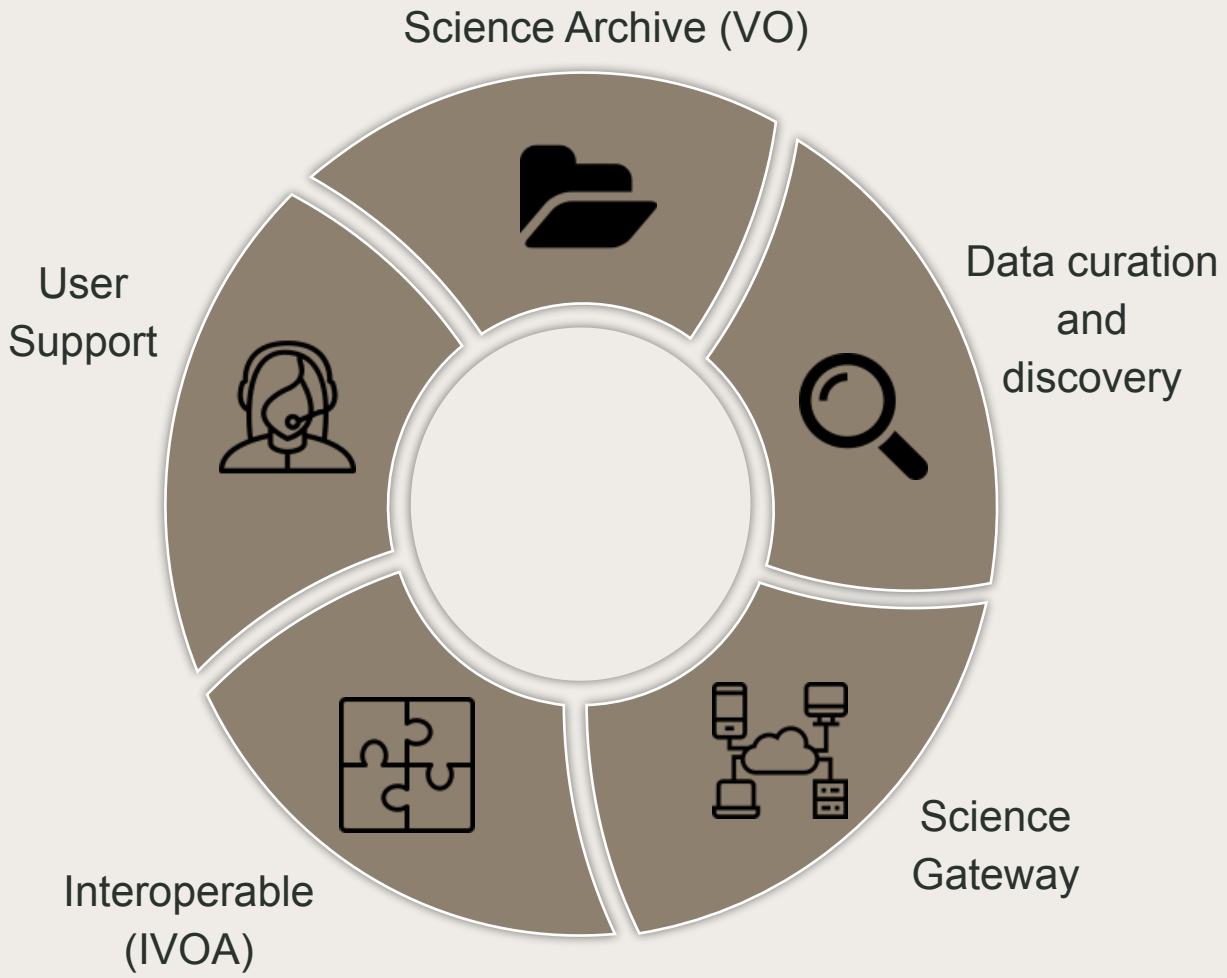


SKA Regional Centre

Beyond a computing cluster

Hardware	Software	Services	User support
A cloud-computing service Storage	VO Archive Collaborative analysis tools		Training: radioastronomy, software, Open Science

SKA Regional Centre



SRCs: The core of the SKA Science

We are developing an SRC Prototype at IAA-CSIC

Science Gateway

Enabling all the science on federated nodes, but transparent to the user

Transversal, wavelength agnostic

Following best practices: Open Science and FAIR

Support preparatory scientific activities with precursors and pathfinders

SKA Regional Centres

Following **Open Science** and **FAIR** principles

Why will radio astronomers **trust** the products?

Define Scientific Workflows

Detailed provenance

Well defined data lineage

Automate processing

Automate fault detection

FAIR Science

Progress in science relies on reproducibility

Deep Learning models are usually a **Black Box**

dtoolAI: Reproducibility for Deep Learning

Describes guidelines on how to make the training and use of DL models FAIR. (Matthew & Tjelvar (2020))

Transparency and reproducibility in artificial intelligence

On the importance of transparency in AI, for instance in medicine. (Nature (2020), 586, E14–E16)

Some challenges

Main SKA challenges

1. Image reconstruction

2. Data transfer and storage

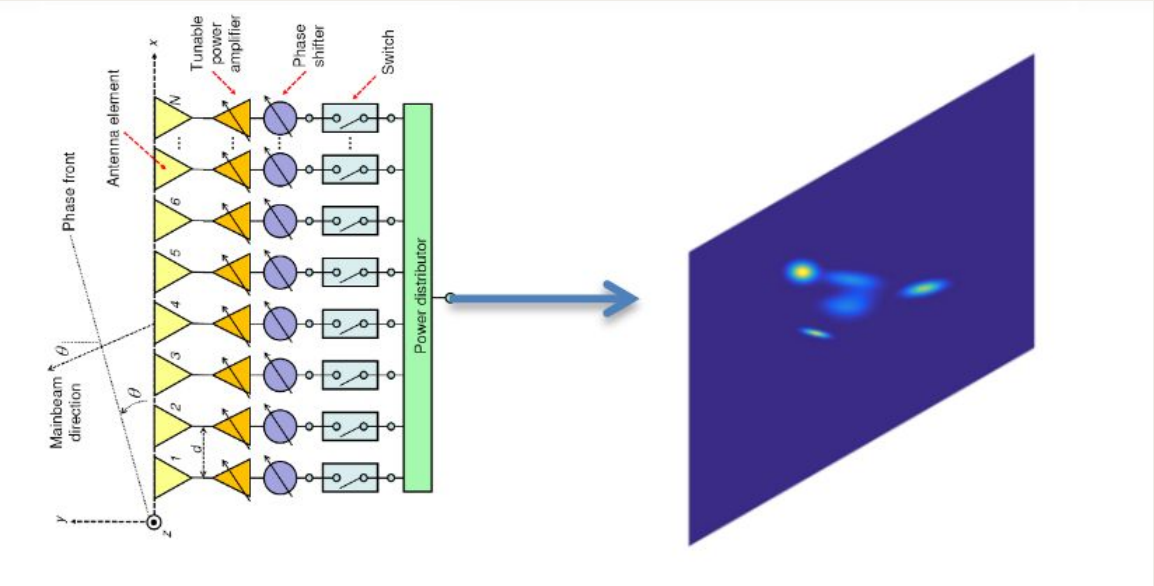
3. Analytics

Voloshynovskiy et al.

Main SKA challenges

1. Image reconstruction

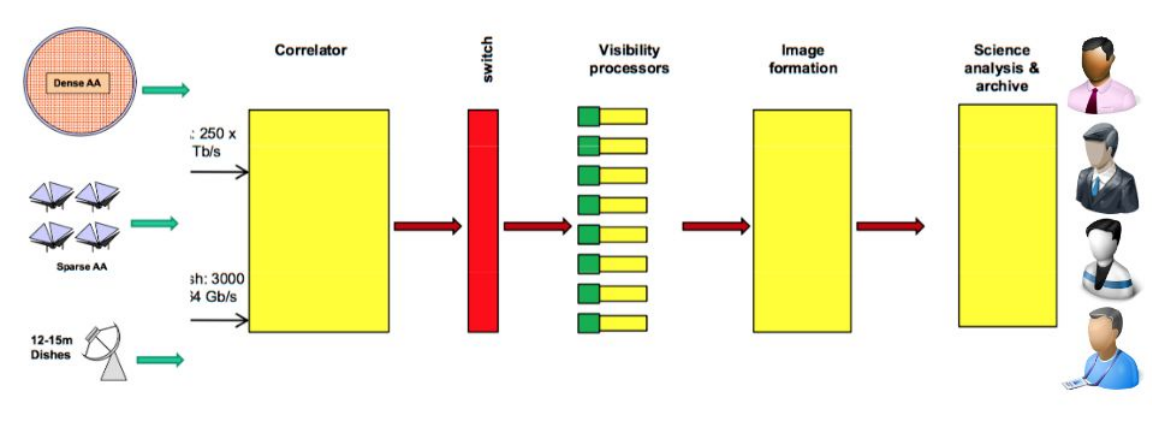
Huge amount of parallel real-time computation for correlation, calibration, reconstruction



Main SKA challenges

2. Data transfer and storage

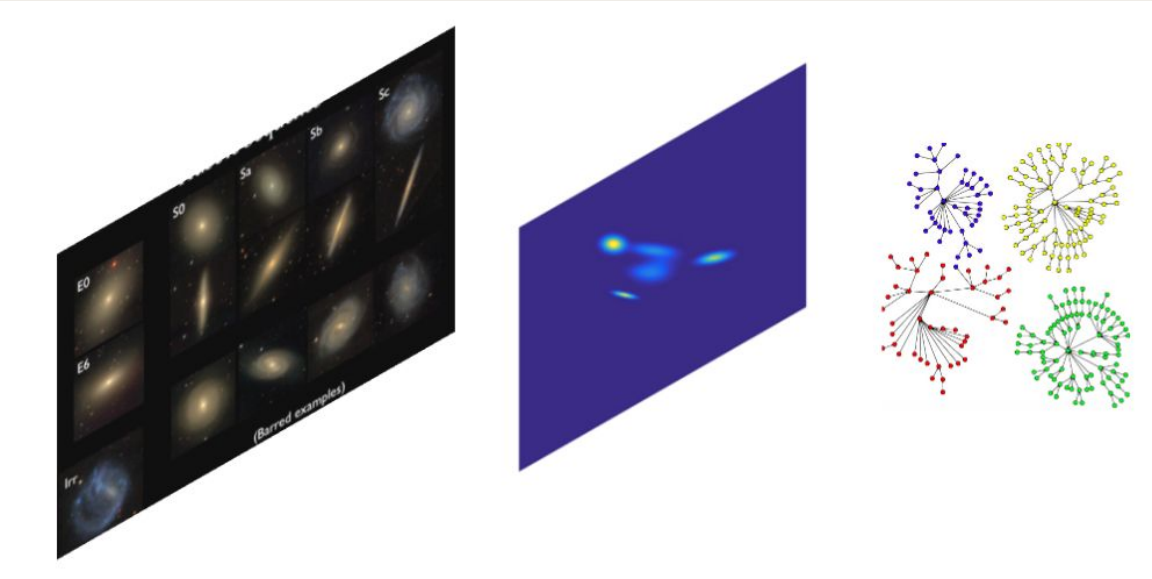
Data transfer from correlators to reconstruction servers, data centers, SDP and end users



Main SKA challenges

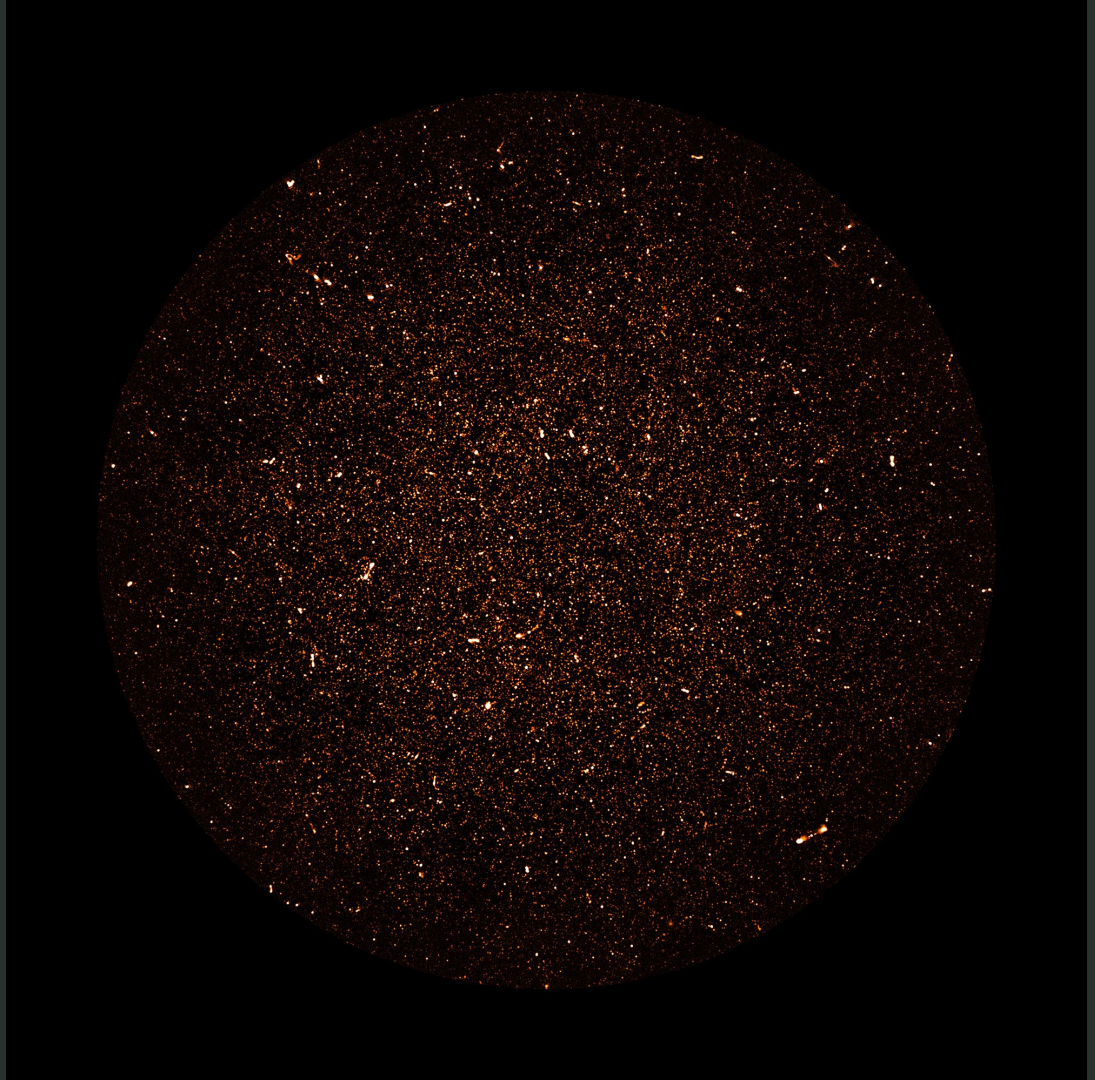
3. Analytics

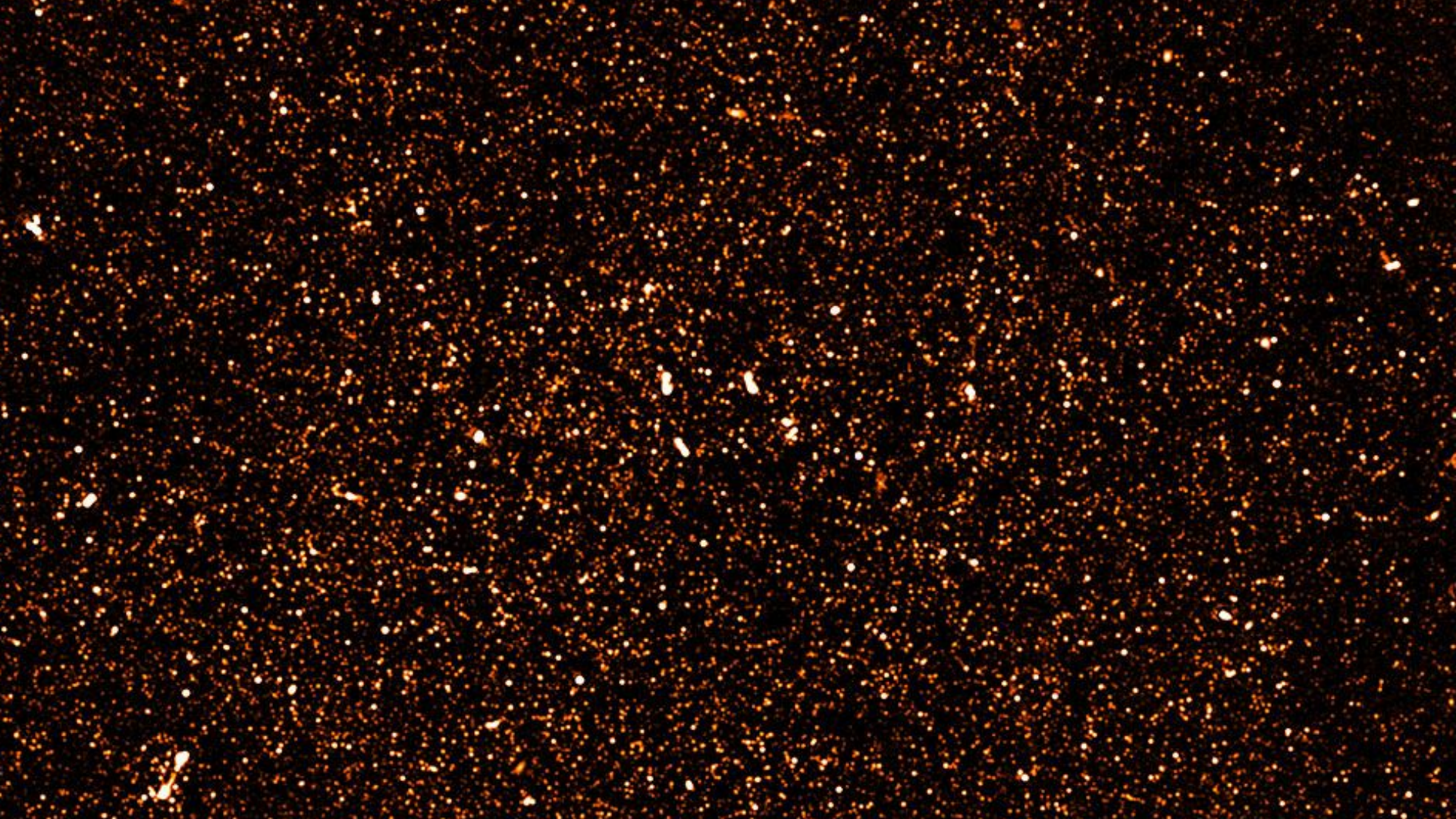
Automatic processing of produced data (recognition, mining, search, tracking,...)

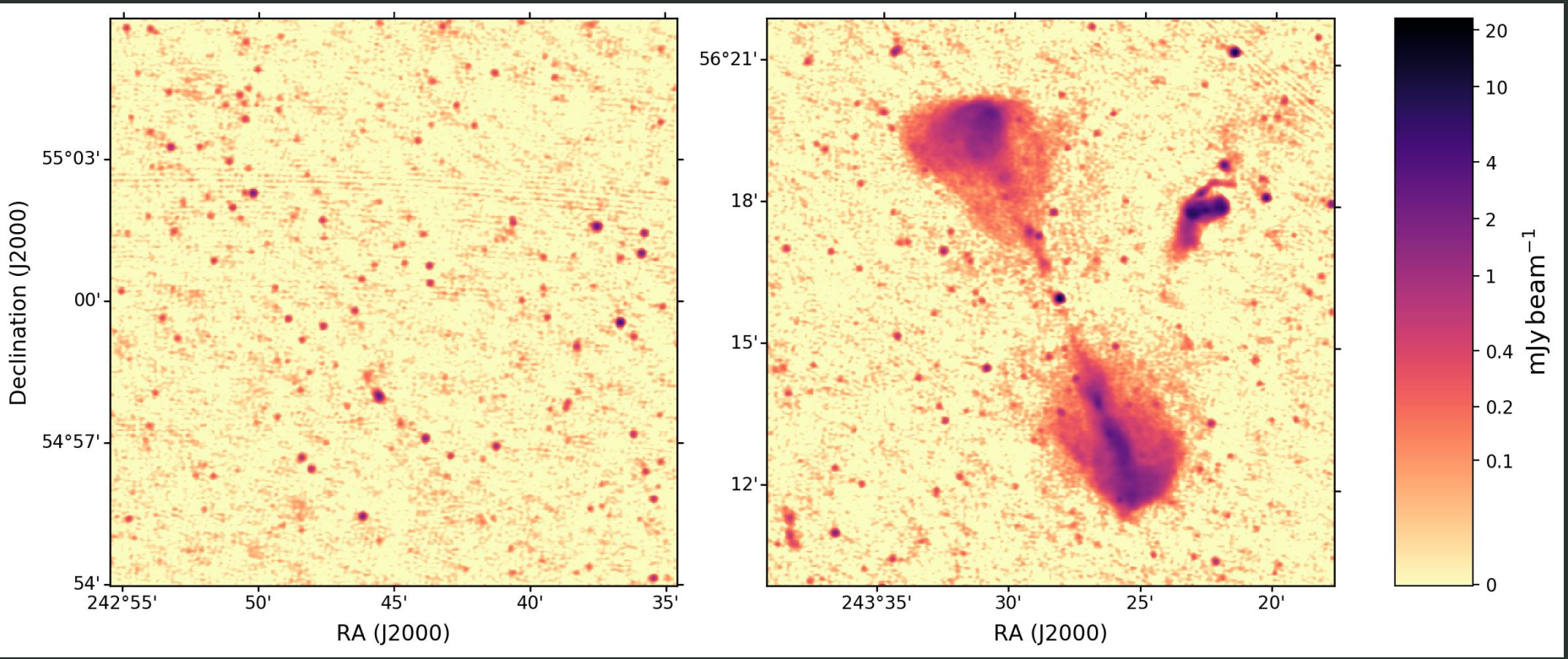


MeerKAT Deep2 image

Rms: 0.55 μJy/b

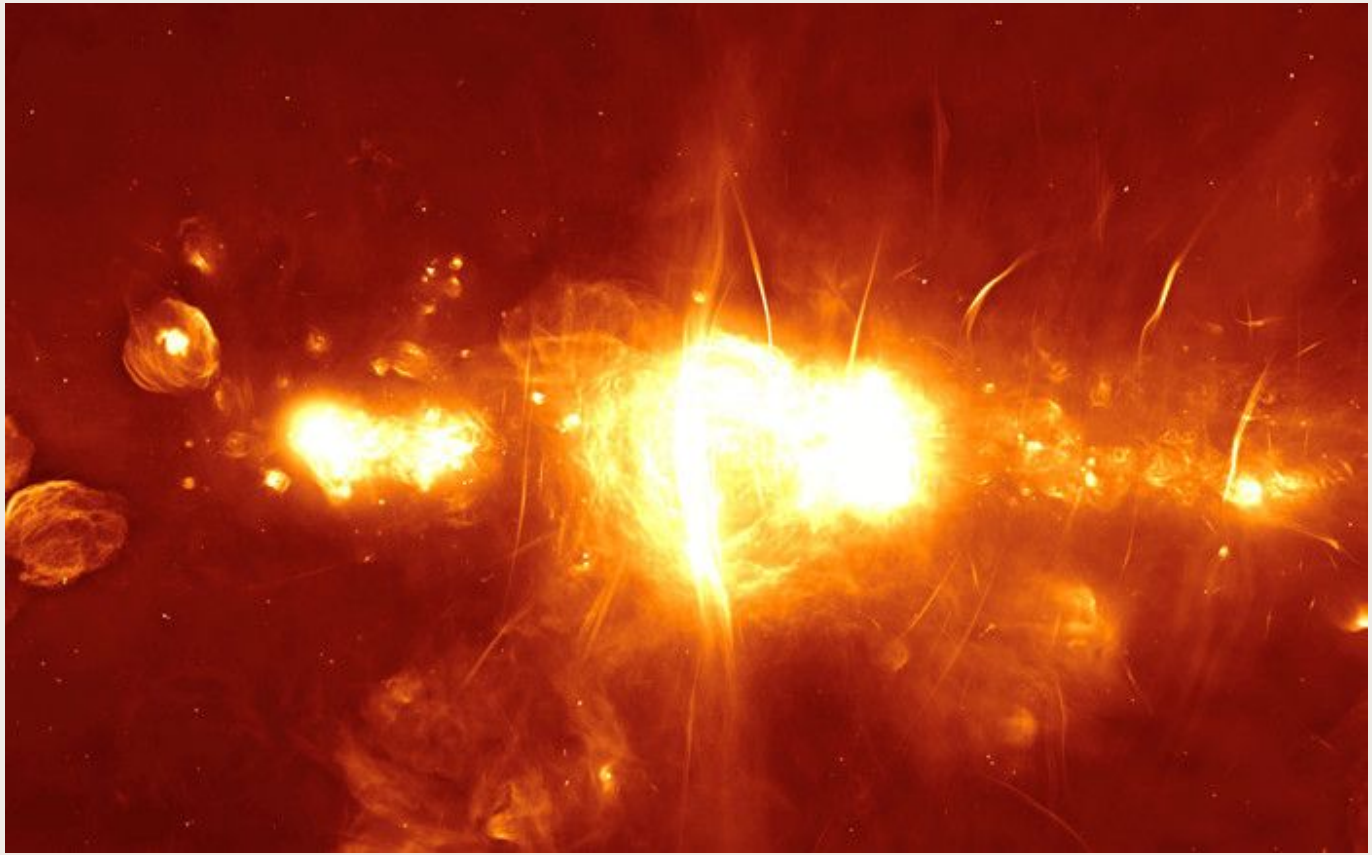




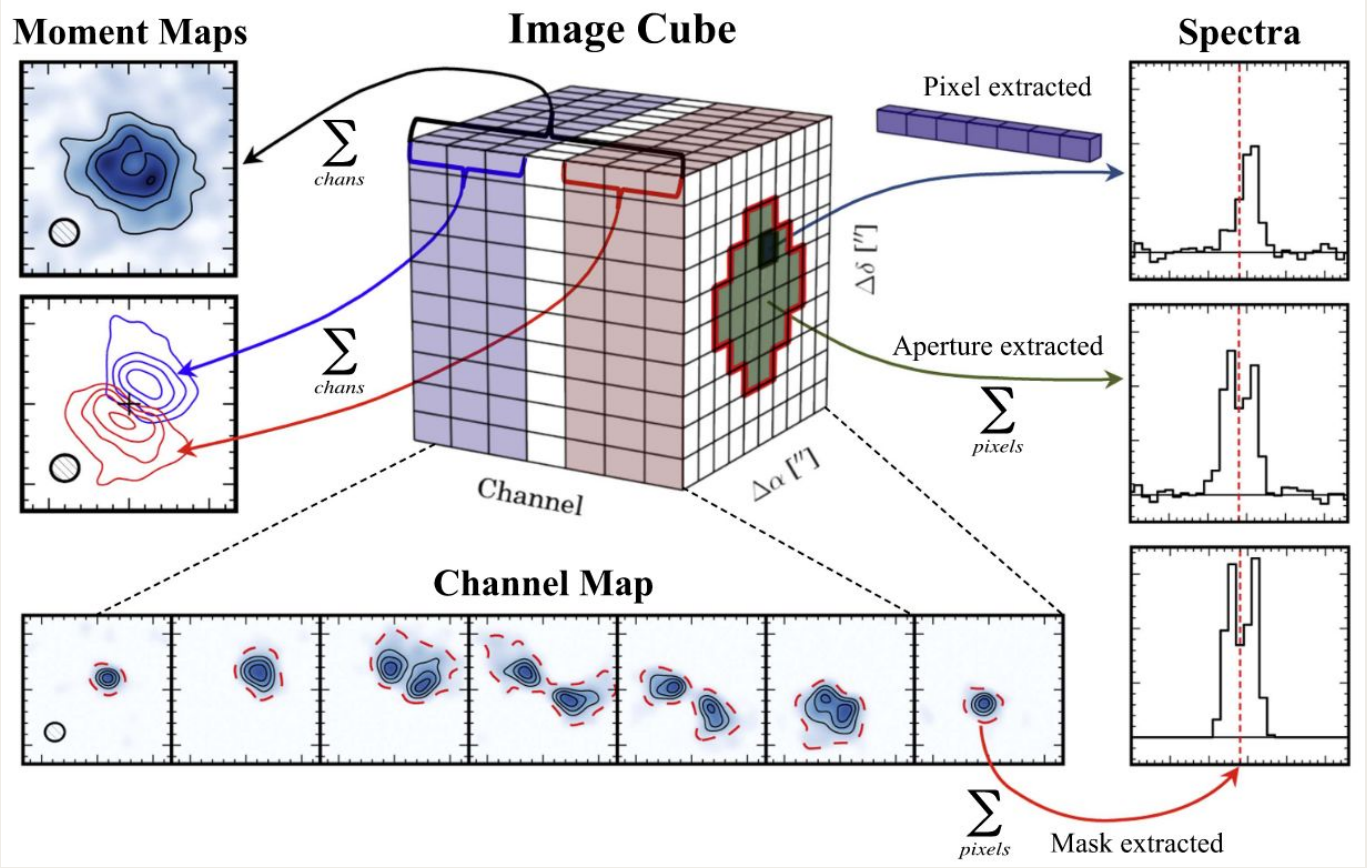


LOFAR, Sabater et al.

Complexity



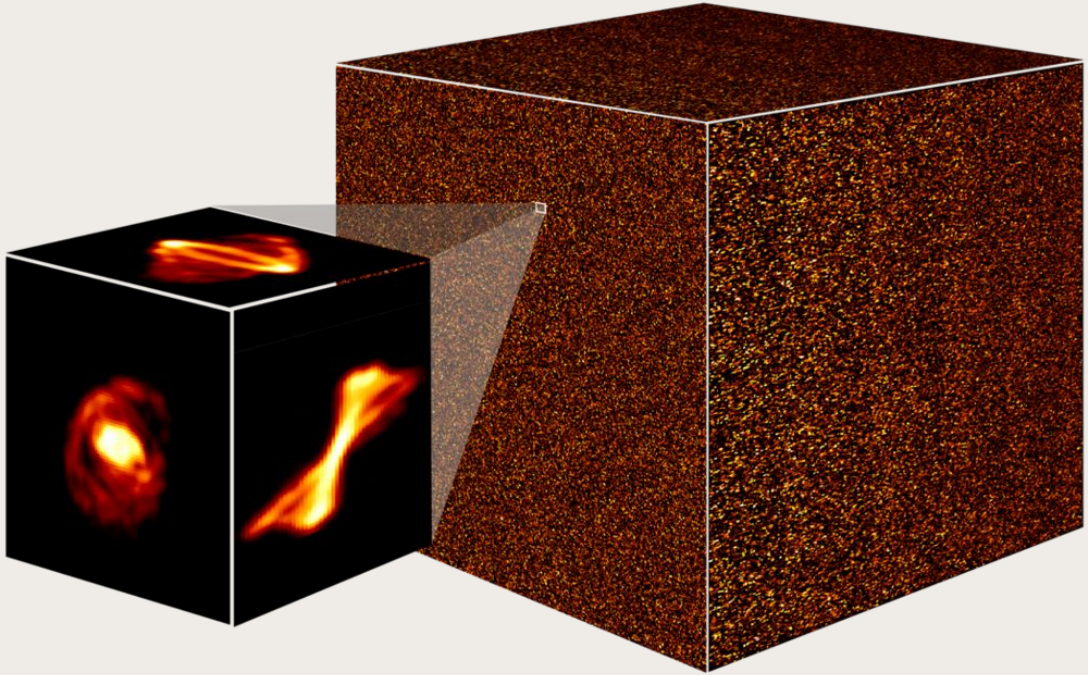
Multi-dimensional data cubes



Loomis et al 2018

Simulated SKA data cube

See Anna Bonaldi's talk



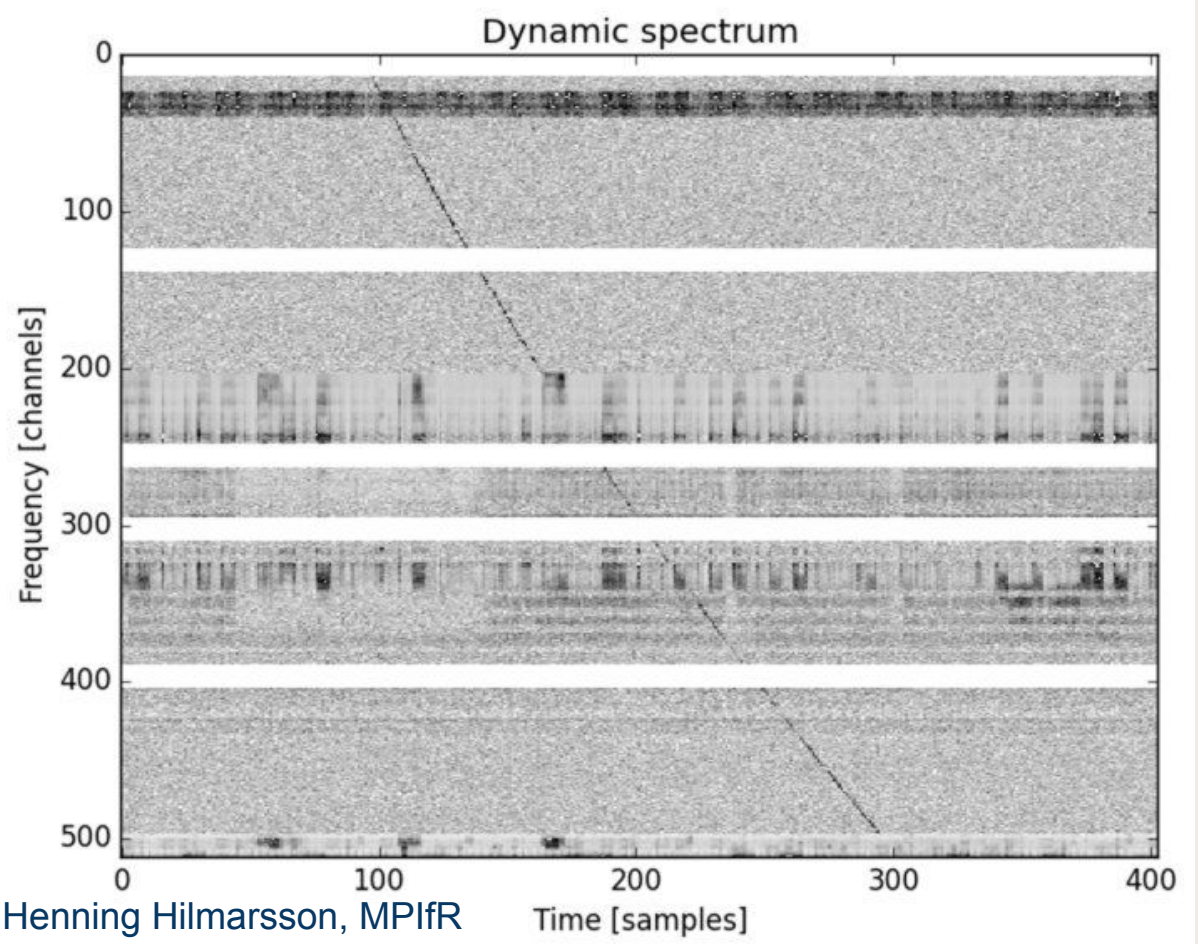
Beamformed data

Exploring millisecond science:

Pulsars

Single events (FRB)

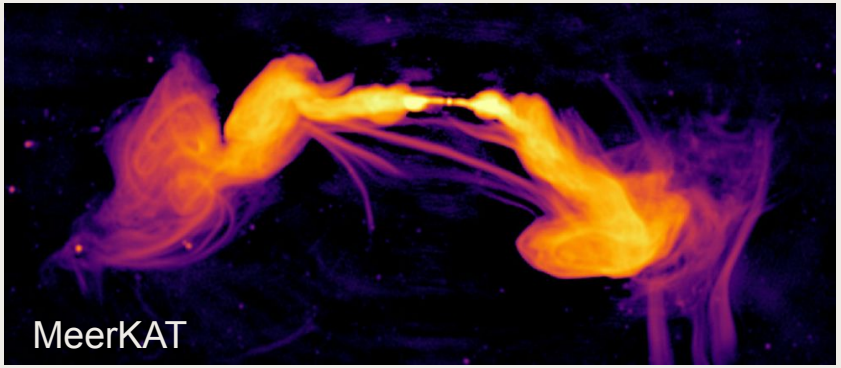
Real-time decisions

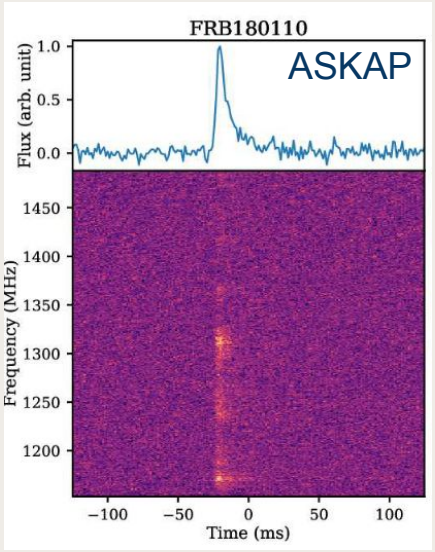


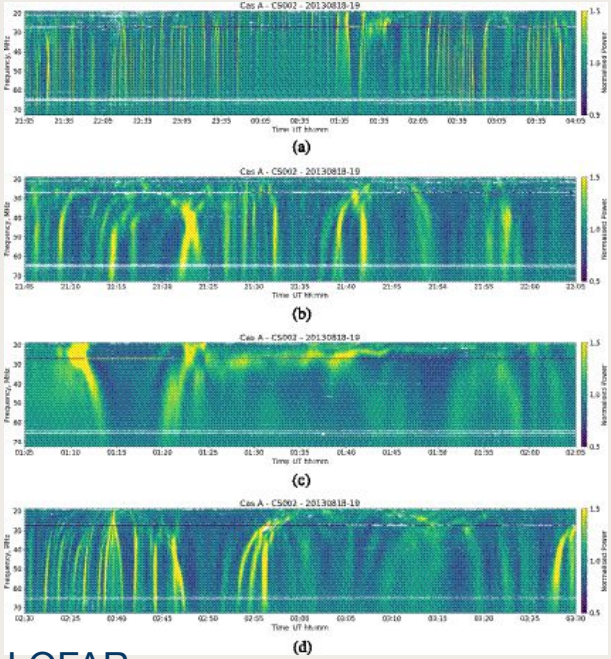
Henning Hilmarsson, MPIfR

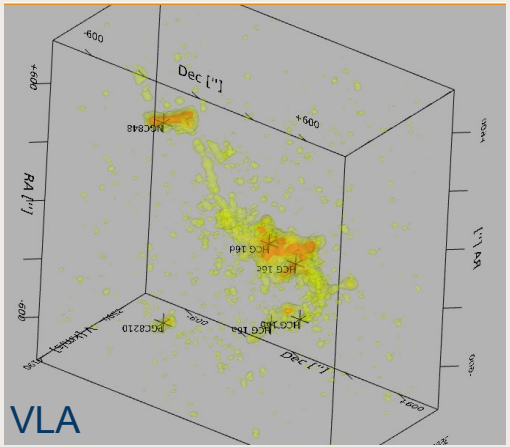
Data products will come in great variety

- Images
- Gridded data
- Beam-formed
- 2D-3D-4D cubes
- RM cubes
- Light curves
- catalogs



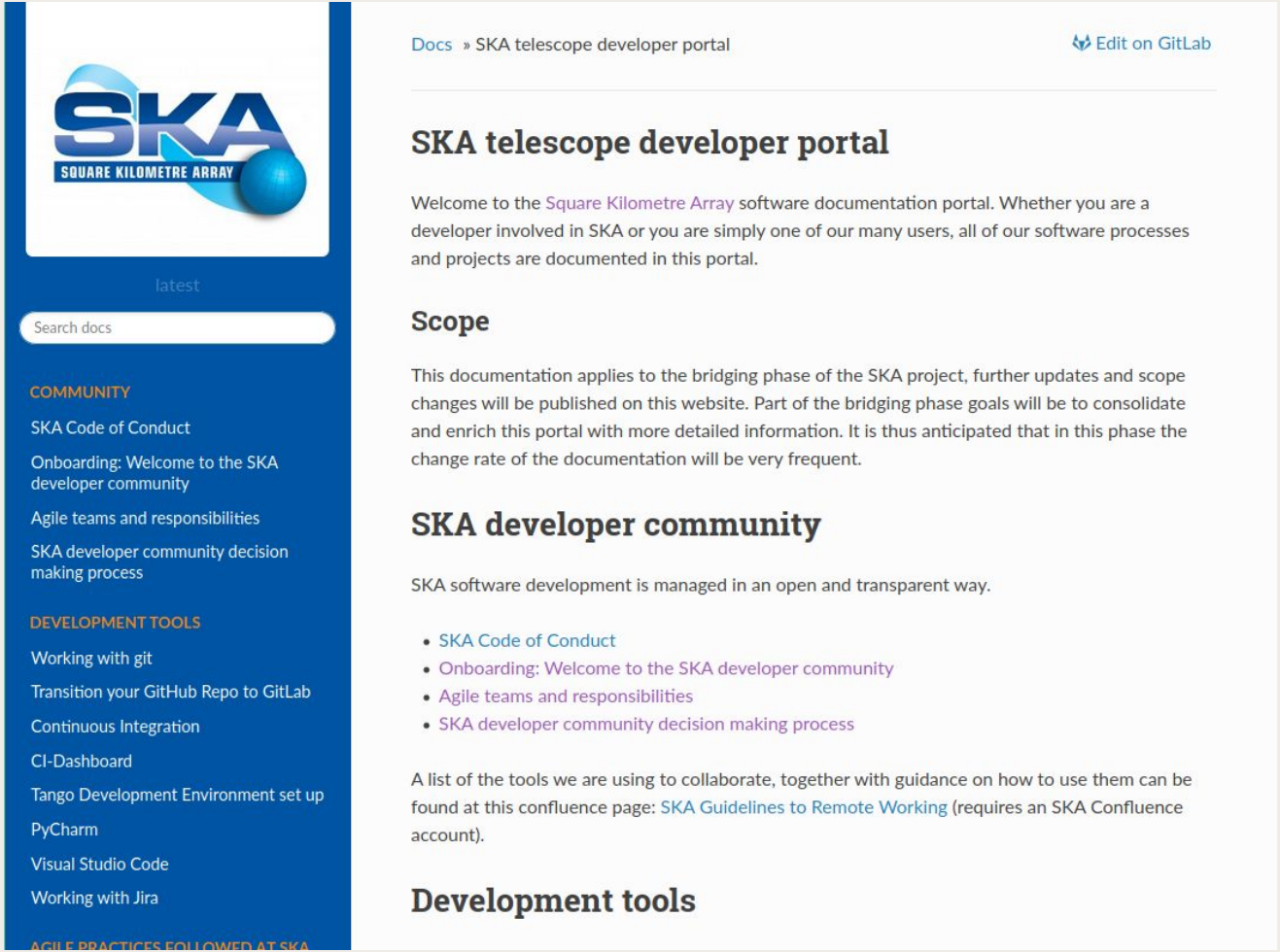






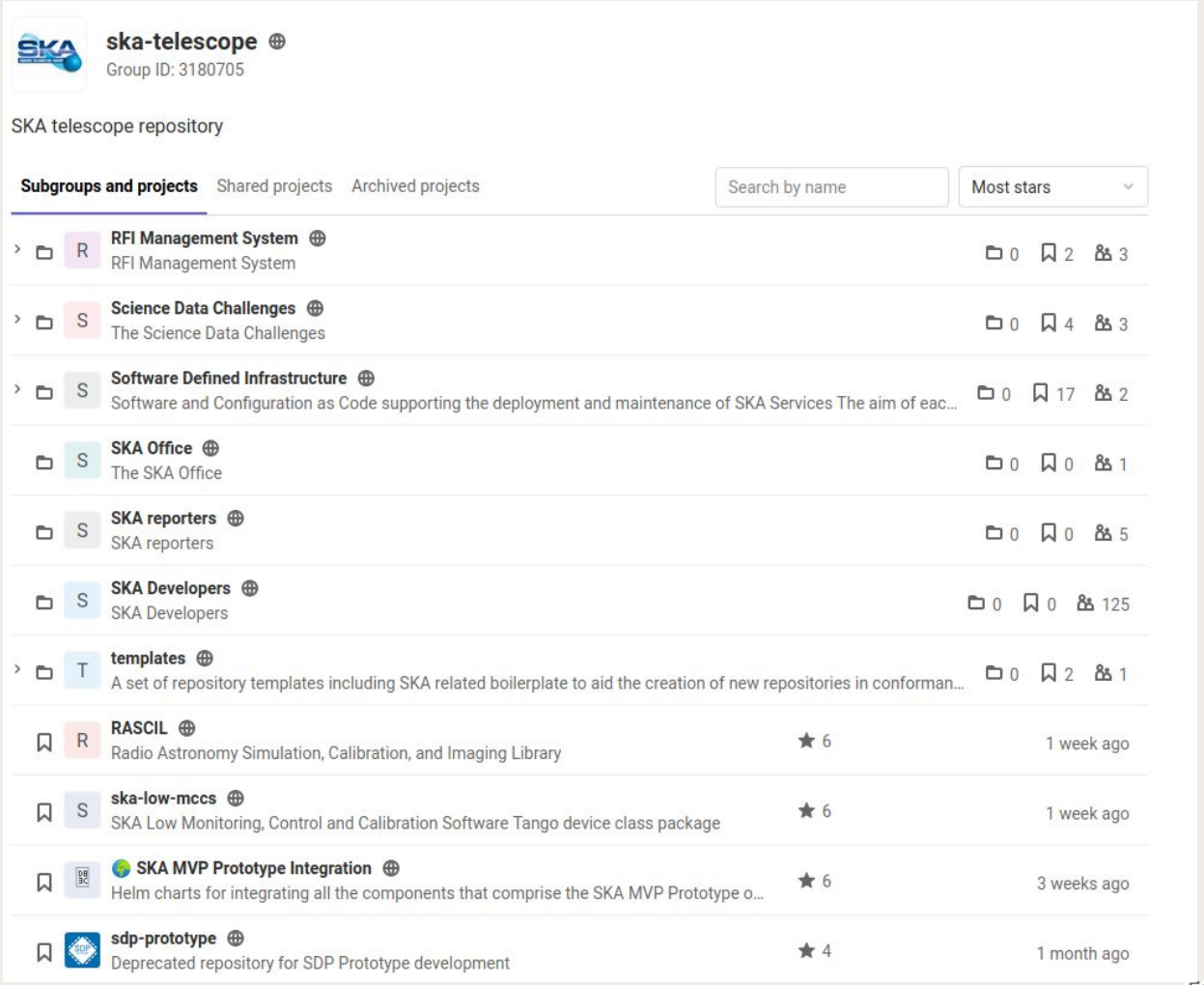
SKA ongoing development

All software development is open, and will impact many fields



SKA ongoing development

Source code in Gitlab



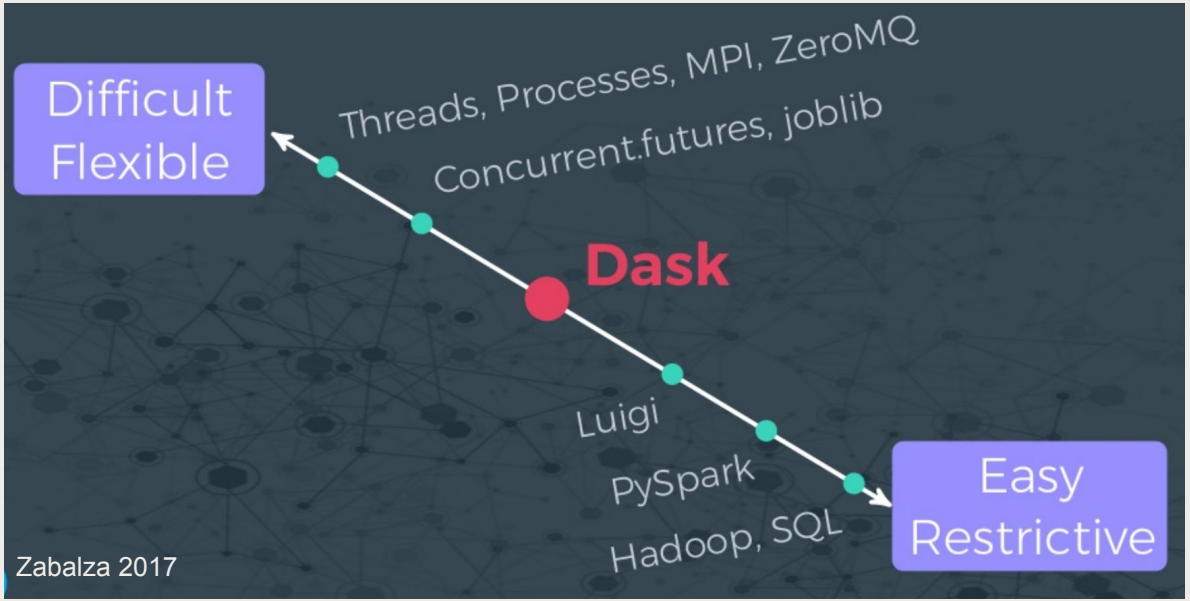
Dask

Out-of-core

Out-of-core computing

Processing data that is too large to fit into a computer's main memory.

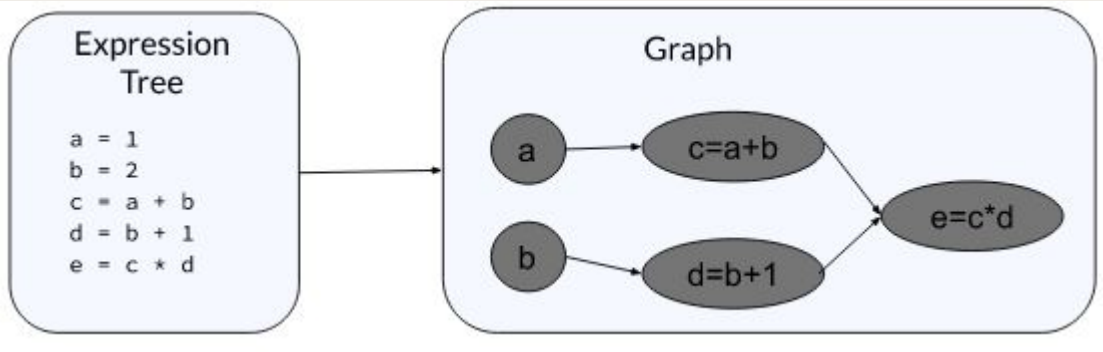
In simple terms: it is to chunk your data in a clever way, so parallel computation happens efficiently



Dataflow programming

How it works?

- User creates a **graph** of the computation
- Graph is submitted to a **scheduler**

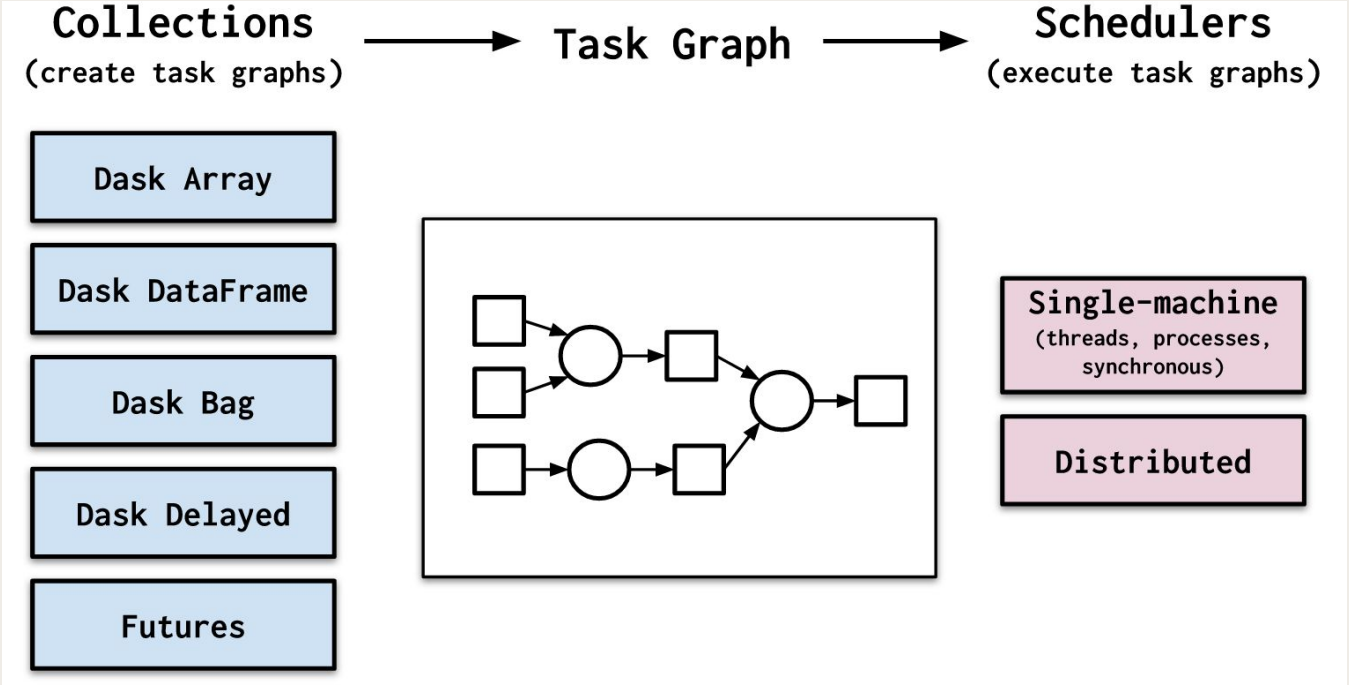


DAG

Dask

3 schedulers:

multi-threaded
multi-processing
distributed



Demo time

Dask Benefits

Python

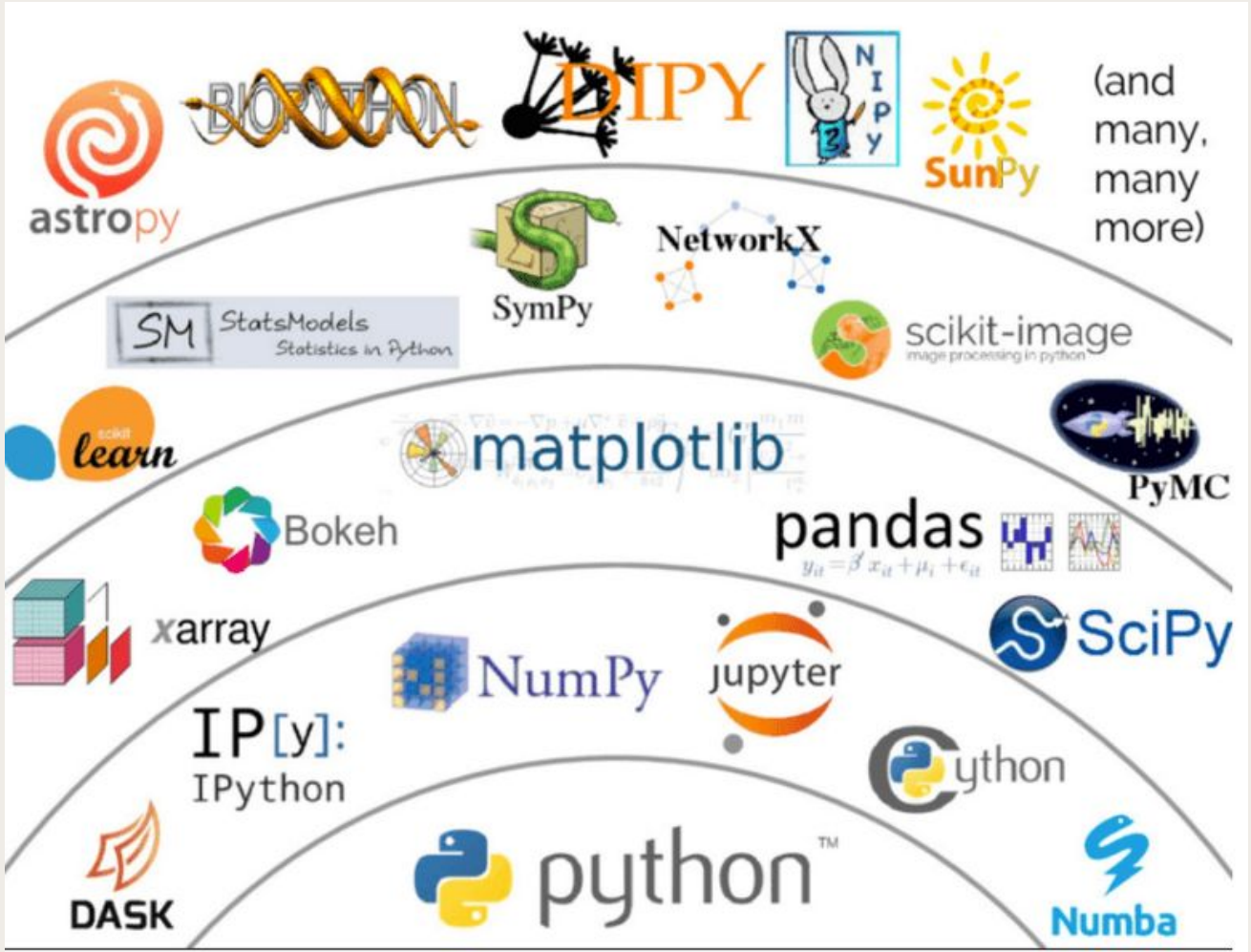
Dask integrates with PyData ecosystem:
numpy, pandas, scipy, datashader, etc.

Integration

Integrated in other higher-order systems.

Think as for numpy, basic and simple, as the foundation of more complex systems

Python Ecosystem



https://coiled.io/blog/pydata-dask/

Dask Benefits

Integrations:

Numpy
Pandas
Scikit-learn

PyTorch
Keras, Tensorflow

Familiar user interface

distributed computing in pure Python with access to the PyData stack

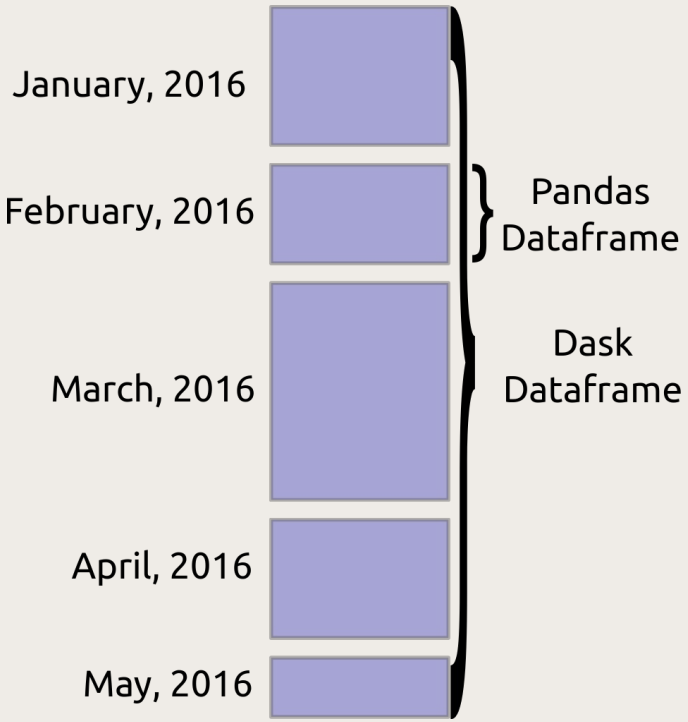
```
# Arrays implement the Numpy API
import dask.array as da
x = da.random.random(size=(10000, 10000),
                     chunks=(1000, 1000))
x + x.T - x.mean(axis=0)
```

```
# Dataframes implement the Pandas API
import dask.dataframe as dd
df = dd.read_csv('s3://.../2018-*-*.csv')
df.groupby(df.account_id).balance.sum()
```

```
# Dask-ML implements the Scikit-Learn API
from dask_ml.linear_model \
  import LogisticRegression
lr = LogisticRegression()
lr.fit(train, test)
```

Seamless integration with Pandas

Not all features are implemented, but most of the syntax works fine




```
[16]: df.loc['2000-01-05'].compute()
```

[16]:

timestamp	id	name	x	y
2000-01-05 00:00:00	1024	Xavier	0.88	0.59
2000-01-05 00:00:01	1009	Tim	0.54	0.89
2000-01-05 00:00:02	976	Xavier	-0.05	-0.97
2000-01-05 00:00:03	1019	Sarah	0.68	0.77
2000-01-05 00:00:04	1029	Quinn	-0.93	0.05
...	...	...	...	...
2000-01-05 23:59:55	997	Sarah	0.11	-0.81
2000-01-05 23:59:56	1005	Sarah	0.62	0.50
2000-01-05 23:59:57	1012	Oliver	-0.81	0.88
2000-01-05 23:59:58	1009	Wendy	-0.15	-0.95
2000-01-05 23:59:59	1028	Ingrid	-0.28	-0.03

86400 rows × 4 columns

Machine Learning dask_ml

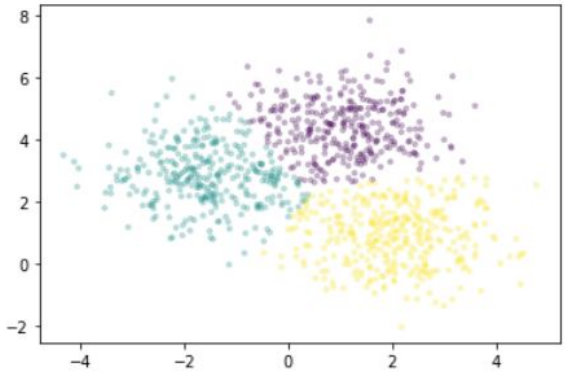
API following Scikit-learn

```
[10]: from dask_ml.linear_model import LinearRegression

      lr = LinearRegression(solver='lbfgs', max_iter=10)
      lr_model = lr.fit(X_train, y_train)
```

```
In [12]: km = dask_ml.cluster.KMeans(n_clusters=3, init_max_iter=2, oversampling_factor=10)
         km.fit(X)

Out[12]: KMeans(init_max_iter=2, n_clusters=3, oversampling_factor=10)
```

We'll plot a sample of points, colored by the cluster each falls into.

```
In [13]: fig, ax = plt.subplots()
         ax.scatter(X[::10000, 0], X[::10000, 1], marker='.', c=km.labels_[::10000],
                    cmap='viridis', alpha=0.25);
```



Also **image** processing

Supporting functions from SciPy ndimage

dask-image

- Support focuses on Dask Arrays.
- Provides support for loading image files.
- Implements commonly used N-D filters.
- Includes a few N-D Fourier filters.
- Provides some functions for working with N-D label images.
- Supports a few N-D morphological operators.

Dask Benefits

Scales from laptops to clusters

Dask is convenient on a laptop.
Scales to a cluster of 100s of machines (1000s cores)

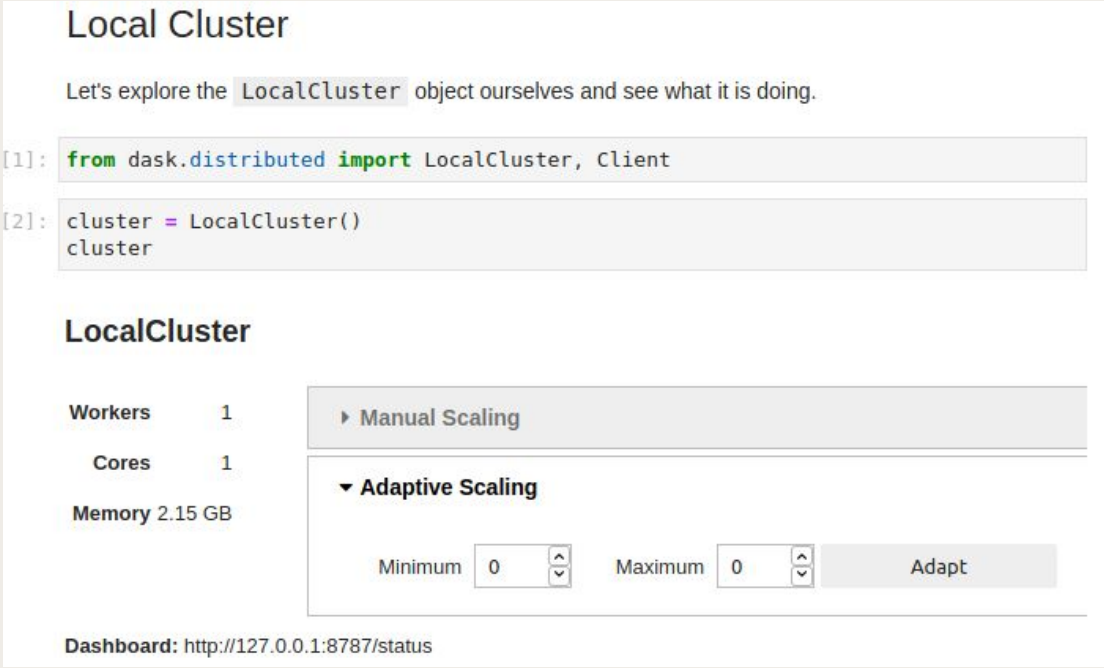
Responsive

Interactive computing in mind, it provides rapid feedback and diagnostics to aid humans

Trivial installation

It installs trivially with conda or pip

Go cluster!


Local Cluster
Let's explore the LocalCluster object ourselves and see what it is doing.
[1]:
from dask.distributed import LocalCluster, Client
[2]:
cluster = LocalCluster()
cluster
LocalCluster
Workers 1
Cores 1
Memory 2.15 GB
Manual Scaling
Adaptive Scaling
Minimum 0
Maximum 0
Adapt
Dashboard: http://127.0.0.1:8787/status

Go cluster!

Remote clusters via SSH

A common way to distribute your work onto multiple machines is via SSH. Dask has a cluster manager which will handle creating SSH connections for you called `SSHCluster`.

```
[8]: from dask.distributed import SSHCluster
```

Scalable clusters

We currently have cluster managers for Kubernetes, Hadoop/Yarn, cloud platforms and batch systems including PBS, SLURM and SGE.

Go GPU!

Dask is strongly supported by NVIDIA

Multi-GPU with Dask-cuDF

CUDA Dataframe: Dask is used for Multi-GPU cuDF

BlazingsSQL

GPU SQL engine built on cuDF

UCX-Py

Remote direct memory access between nodes (similar to MPI)

Applications to radio data

Most use **dask-ms** interface to Measurement Set

Tricolour

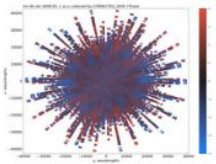
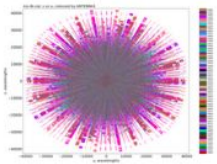
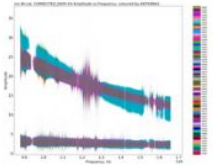
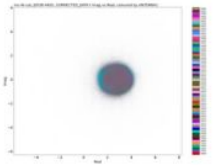
Flags Radio Frequency Interference (RFI)
Processing in excess of 400GiB / hr

QuartiCal

Parallel calibration with dask and numba

ShadeMS

Fast plotting interferometric data using Datashader

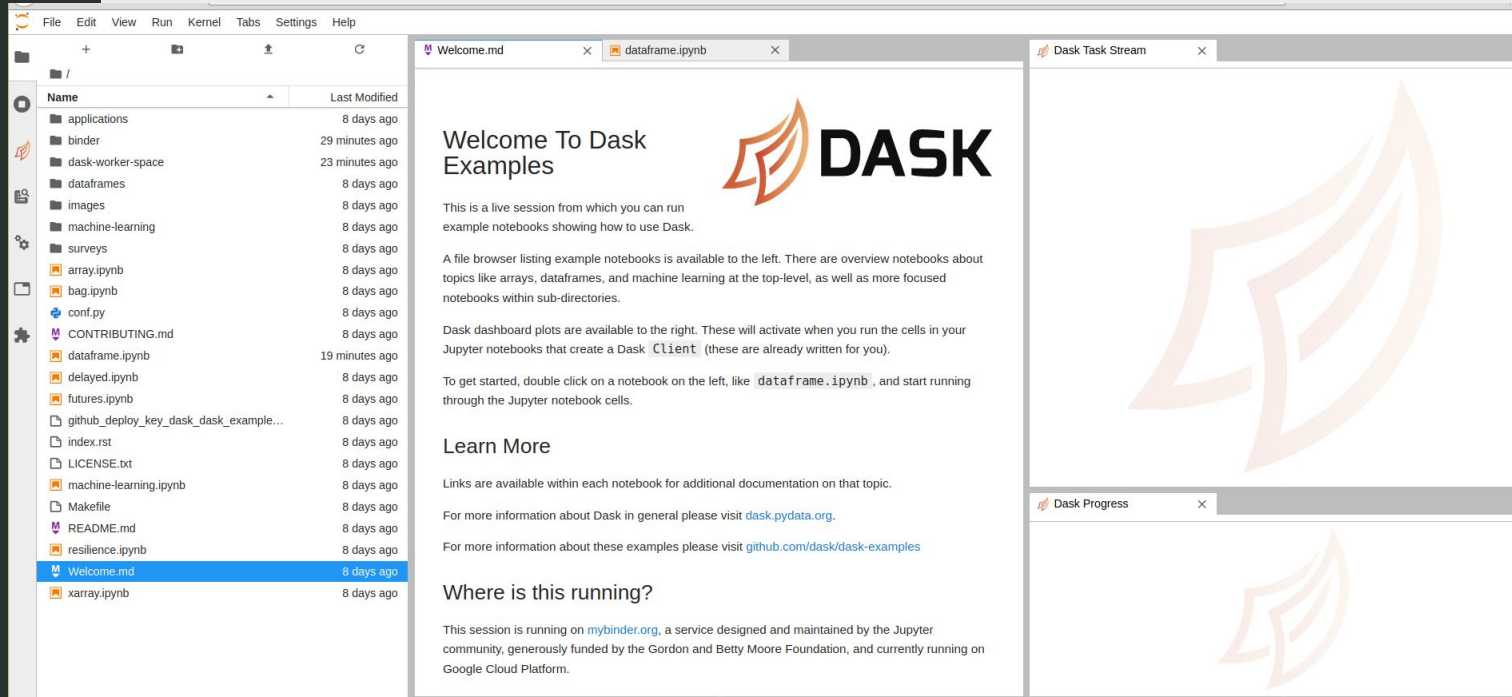
points plotted	5 billion	10 billion	5 billion	5 billion
runtime	250s	170s	250s	140s
				

Lots of tutorials and examples

Dask is only 2 years old!

Check Out Dask Tutorials

Explore Dask tutorials on **Github**, see Dask code examples on **dask.org** and **Binder**.



Conclusions

Use Dask. It is fun!

Revolution of Big Data is coming. Change of paradigm

Many interesting problems to apply Machine Learning

SKA precursors and pathfinders. Doing great now!

Be prepared… the SKA is coming