DISTRIBUTED DATA PROCESSING USING SPARK IN RADIO ASTRONOMY

Panos Labropoulos
Bright Computing, Inc.

Sarod Yatawatta
ASTRON

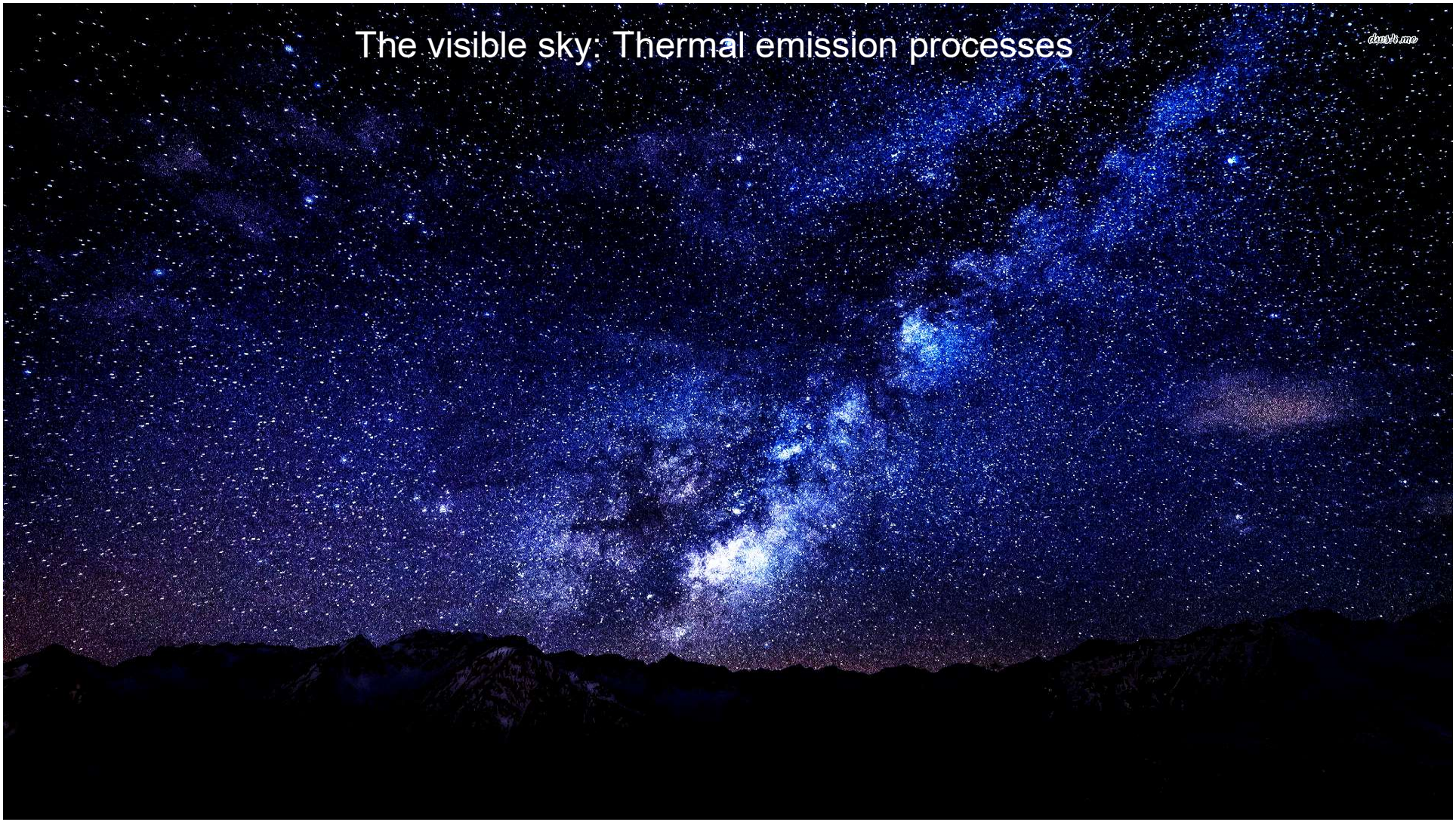
Who are we?

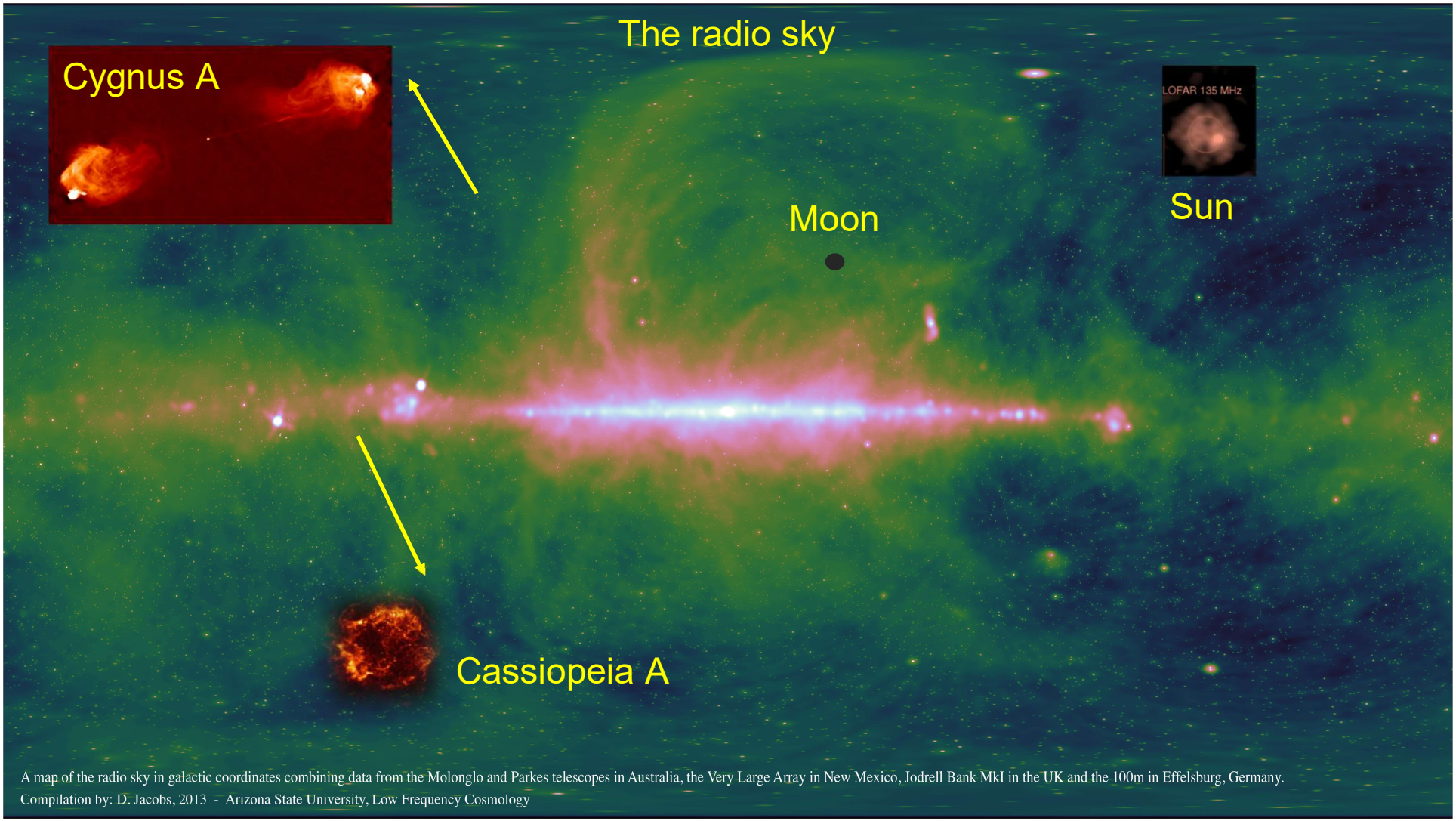
- Panos Labropoulos
  - Works at Bright Computing
  - HPC, accelerators, high-speed interconnects
  - PhD Radio Astronomy, University of Groningen

- Sarod Yatawatta
- Works at ASTRON
- Signal and Image processing
- PhD Electrical engineering, Drexel

Outline

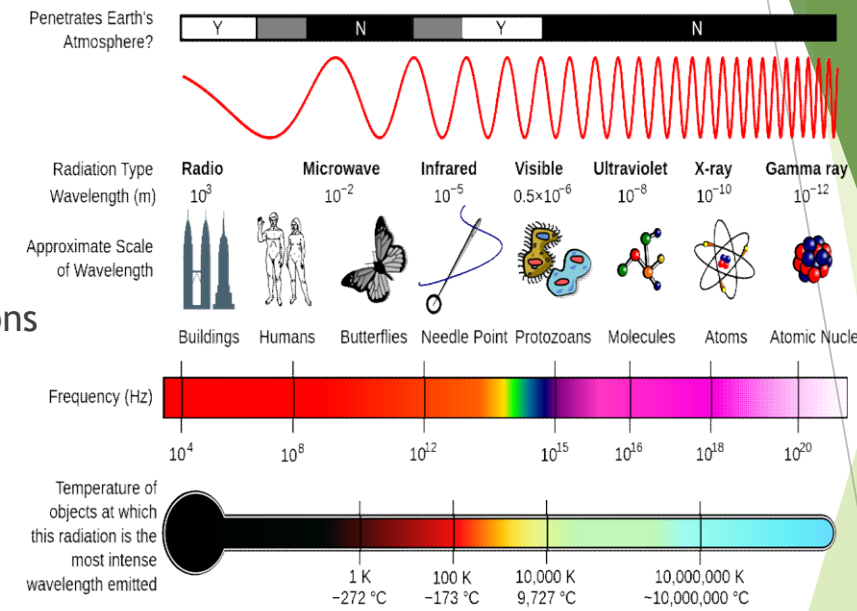
- Basics of radio interferometry
- Scientific motivation
- Why is Spark relevant?
- Two case studies: calibration and imaging


The visible sky: Thermal emission processes

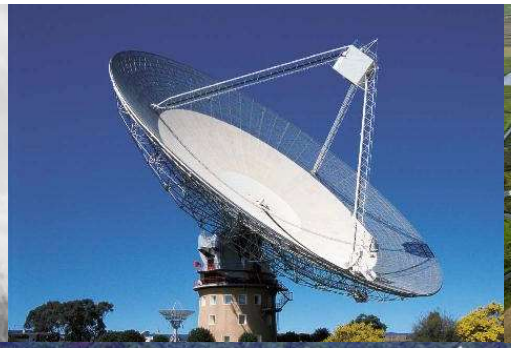

The radio sky
Cygnus A
LOFAR 135 MHz
Sun
Moon
Cassiopeia A
A map of the radio sky in galactic coordinates combining data from the Molonglo and Parkes telescopes in Australia, the Very Large Array in New Mexico, Jodrell Bank MkI in the UK and the 100m in Effelsburg, Germany.
Compilation by: D. Jacobs, 2013 - Arizona State University, Low Frequency Cosmology

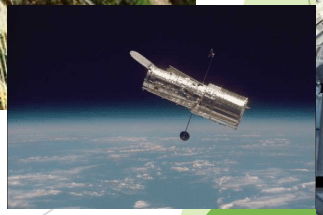
Why radio?

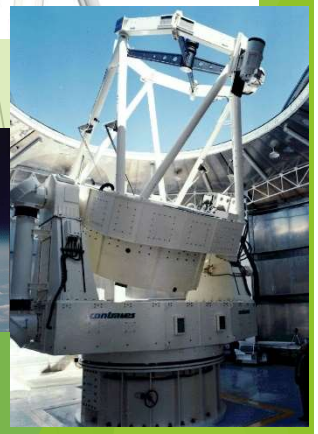
- Thermal Radiation
- Synchrotron Radiation
  - Relativistic e^- in magnetic fields
- Bremstrahlung
  - "Breaking Radiation" e^- /ion collisions
- Maser
  - Microwave Laser e^- oscillations in molecular clouds
- Atomic Transitions (emission spectra)
  - Hydrogen e^- spin flip



Tools required: antennas, receivers
(e.g. voltage amplifiers, ADCs, computers)







Similar principles to optical astronomy – different wavelengths

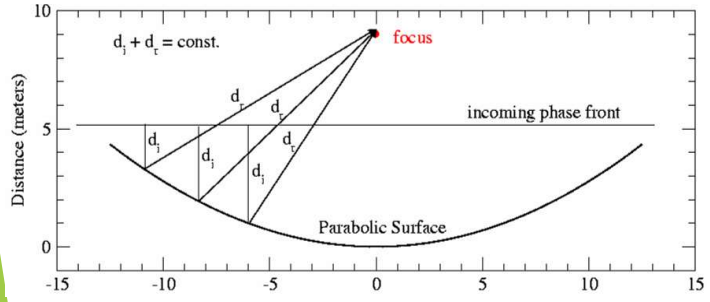
Basic characteristics of a telescope

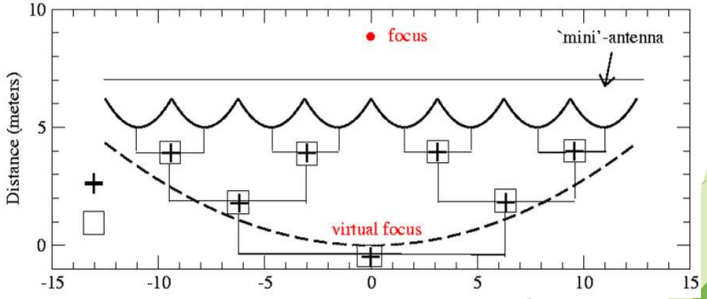
Angular resolution and sensitivity proportional to size

- Oldest telescope: human eye
  - Human pupil: 0.5 cm diameter, 0.05 deg. Resolution at λ=600 nm (yellow)
- Radio telescope: 100m diameter, 1.4 deg. resolution at λ=2m
  - 2.8 km diameter required for same resolution as human eye
  - 250 km dimeter required for same resolution as an optical telescope

$$\vartheta = 1.22\frac{\lambda}{D}$$

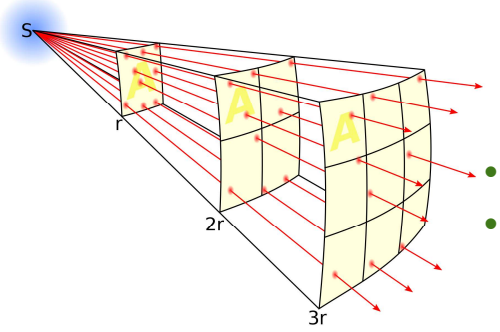
Constructing a 250km parabolic dish is simply not feasible





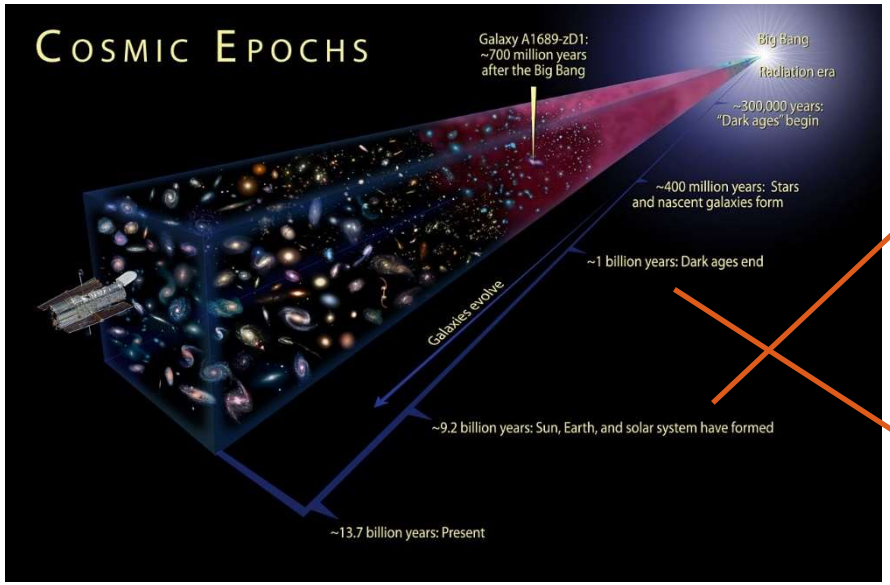
Use smaller antennas and add-up signals using a computer

How many antennas? How big?



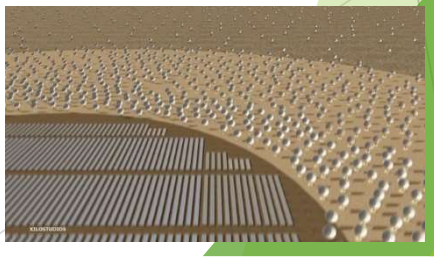
Intensity follows an inverse square law

- Source 3 times further away become 9 times fainter
- Size of telescope depends on distance at which we wan t to look objects at.



10.000 sq. m. VLA

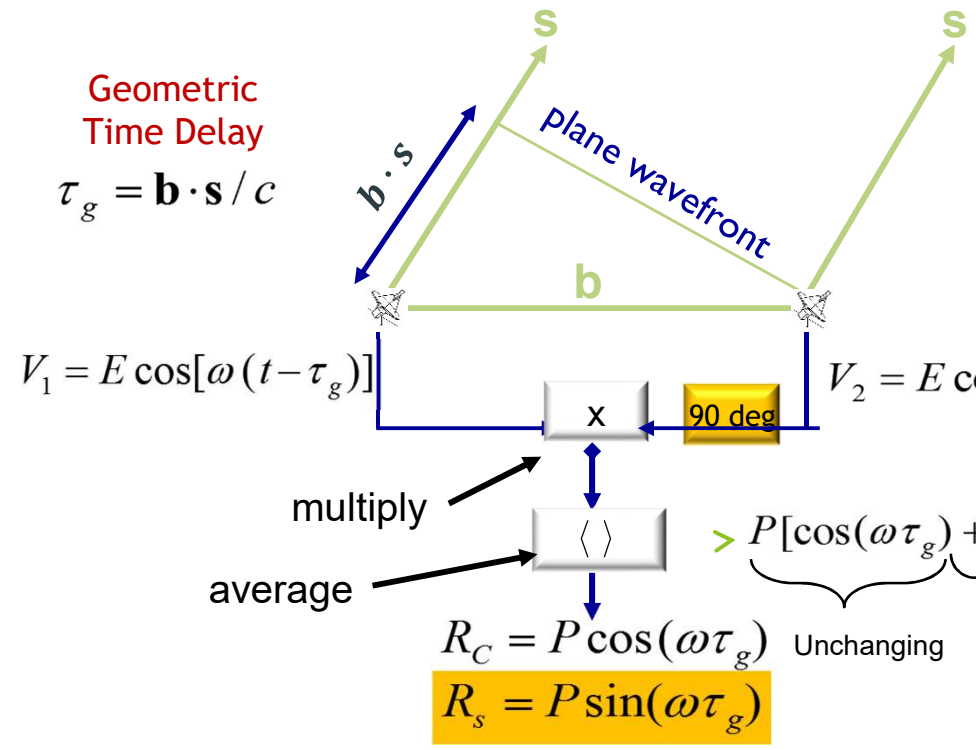
10x further -> 100x bigger
1 sq. Km



Measurement process

Geometric Time Delay

$$\tau_g = \mathbf{b}\cdot\mathbf{s}/c$$



The path lengths from sensors to multiplier are assumed equal!

$$> P[\cos(\omega\tau_g) + \cos(2\omega t - \omega\tau_g)] \quad P[\sin(\omega\tau_g) + \sin(2\omega t - \omega\tau_g)]$$

Unchanging — Rapidly varying,

$$R_C = P\cos(\omega\tau_g)$$

$$R_s = P\sin(\omega\tau_g)$$

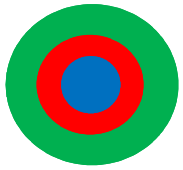
$$V_\upsilon(\mathbf{b}) = R_C - iR_S = \iint I_\nu(s)e^{-2\pi\lambda i\,\mathbf{b}\cdot\mathbf{s}}d\Omega$$

2-D FT of the sky's brightness distribution

Examples of 1-Dimensional Visibilities

- Simple pictures are easy to make illustrating 1-dimensional visibilities.

- Unresolved Doubles

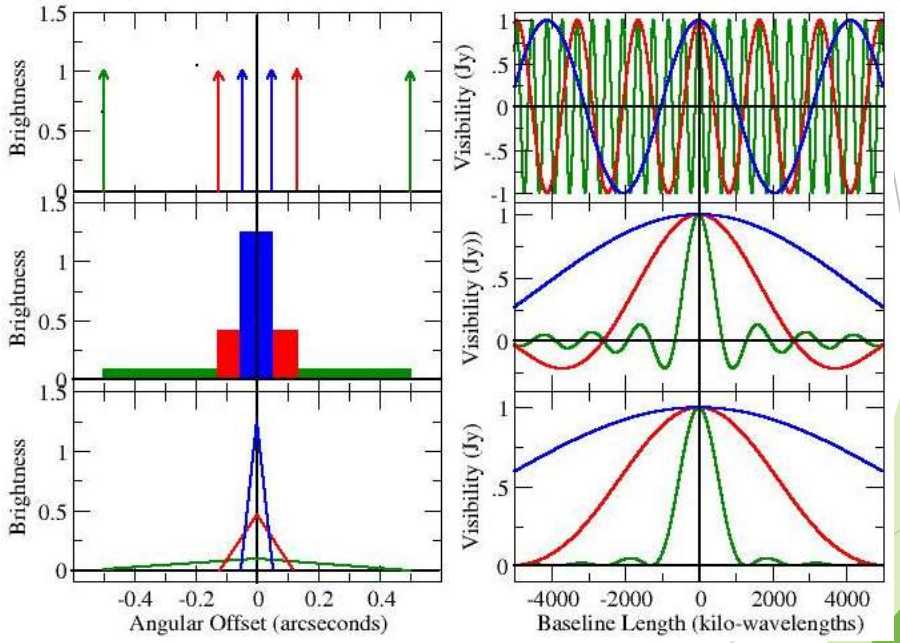


- Uniform

- Central Peaked

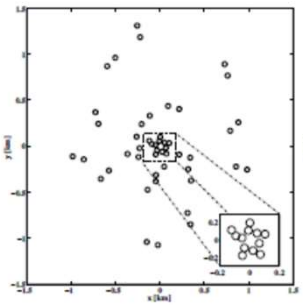
Brightness Distribution

Visibility Function

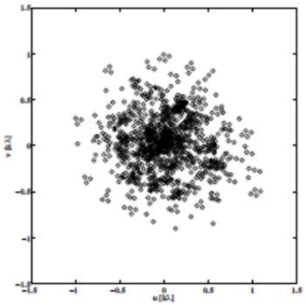


Earth rotation synthesis

- An interferometer with N antennas has N (N -1) / 2 “baseline” pairs
- Real interferometers are built on the surface of the earth – a rotating platform. From the observer’s perspective, sources move across the sky, but from the source’s perspective the antenna array is rotating in such a way that the baseline pair points form tracks over time
- u-v tracks have gaps: incomplete sampling of the 2-D transform / missing information

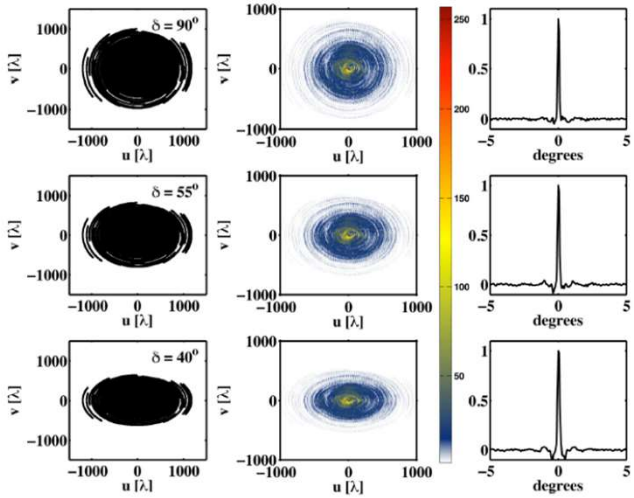


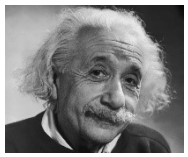
Antenna layout



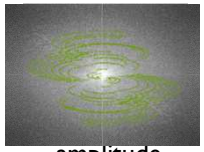
Baseline pairs layout

360 min

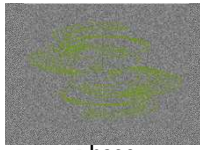




⇌

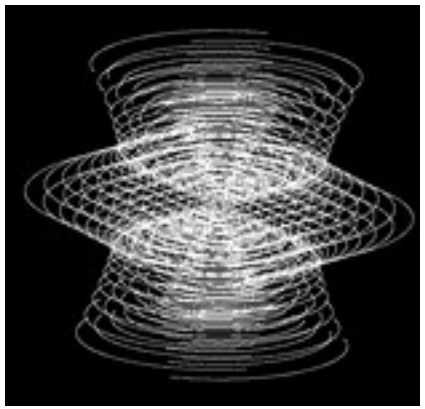


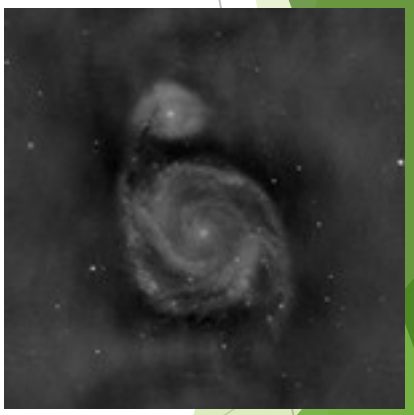
amplitude



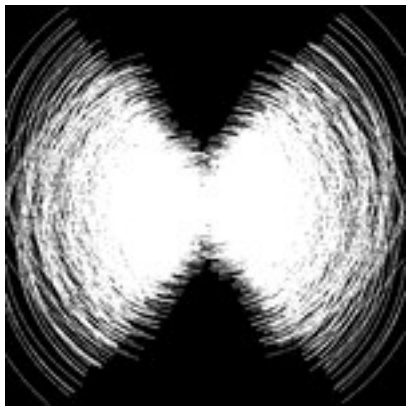
phase

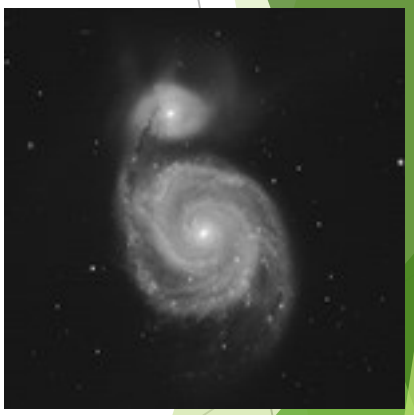
Example: synthesis image



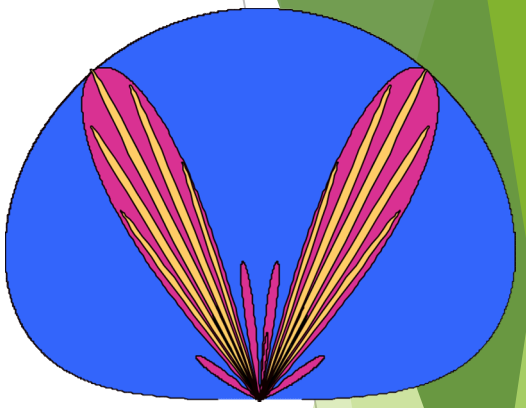


Example: synthesis image

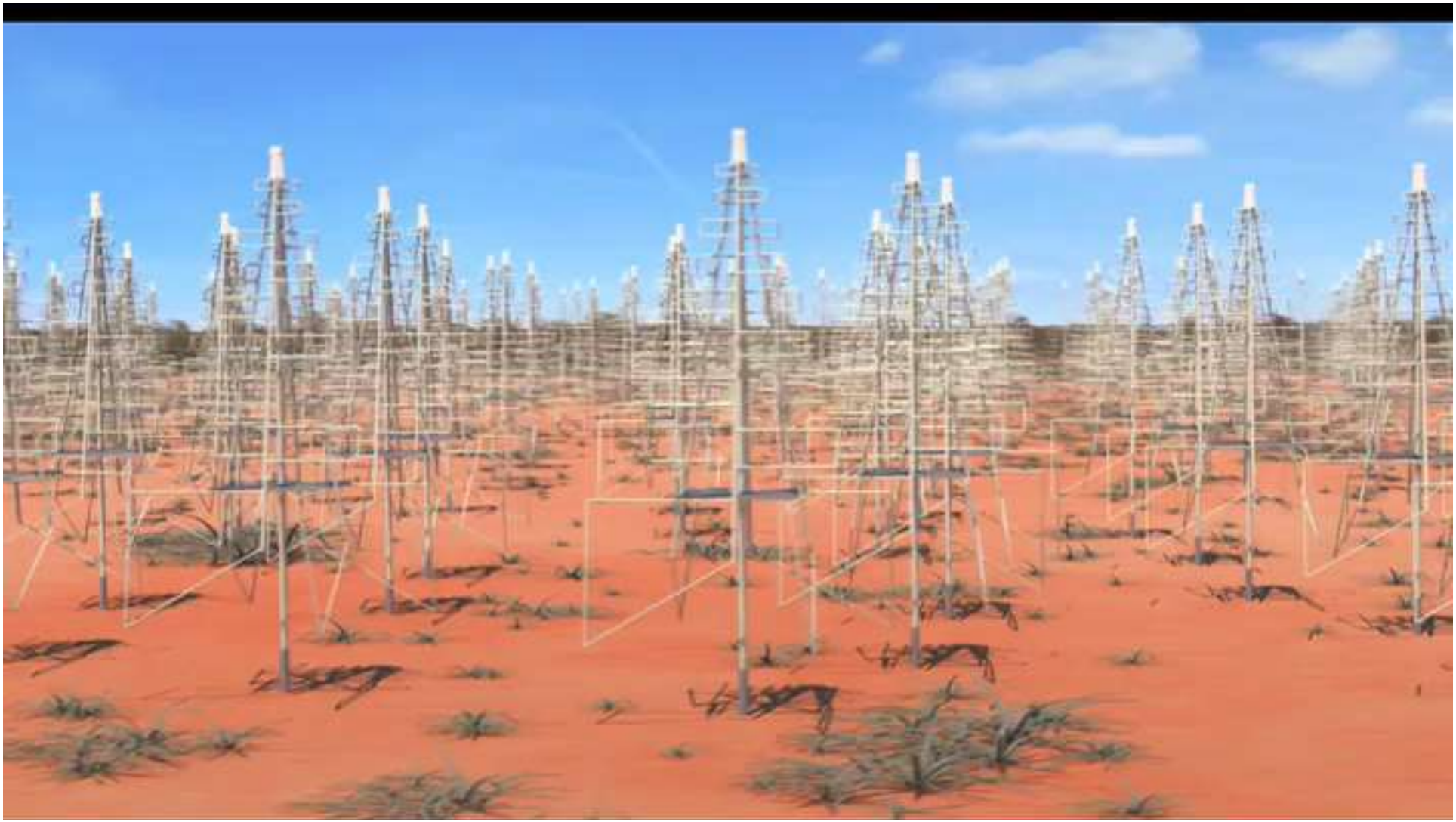




Kilometer Array



- 100x more sensitive, 1,000,000x faster imaging
- Up to 5 sq.km collecting area and 3500km baselines
- Sub-arcsecond resolution, large field-of-view
- 3 types of antennas, 2 locations, a single observatory
- No mechanical steering for low- and mid-frequency antennas, multiple beams
- Will address key questions of astronomy, cosmology and physics
- Ambitious IT project. Probably will be the first production exascale system
- Will be generating several exabytes of processed data per year
- Builds up on techniques developed in Europe
- Major HPC facility
- UK, RSA, AUS, NZ, Canada, China, NL, Germany, Italy, India, Sweden collaboration

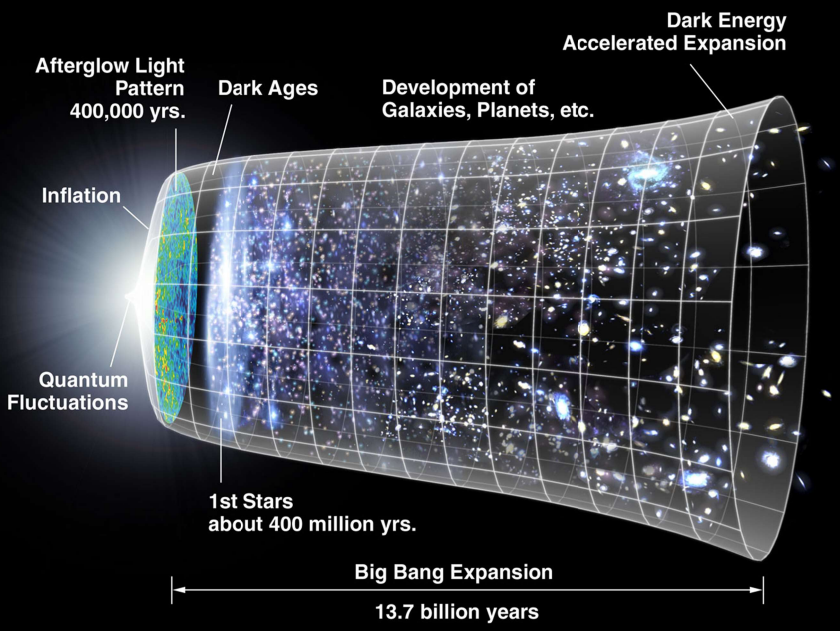


Five Key Science Areas for the SKA

Topic	Goals
Probing the Dark Ages	1. Map out structure formation using HI from the era of reionization (6 < z < 13) 2. Probe early star formation using high-z CO 3. Detect the first light-emitting sources
Gravity: Pulsars & Black Holes	1. Precision timing of pulsars to test theories of gravity approaching the strong-field limit (NS-NS, NS-BH binaries, incl Sgr A*) 2. Millisecond pulsar timing array for detecting long-wavelength gravitational waves
Cosmic Structure	1. Understand **dark energy** [e.g. eqn. of state; W(z)] 2. Understand structure formation and galaxy evolution 3. Map and understand dark matter
Cosmic Magnetism	Determine the structure and origins of cosmic magnetic fields (in galaxies and in the intergalactic medium) vs. redshift z
The Cradle of Life	1. Understand the formation of Earth-like planets 2. Understand the chemistry of organic molecules and their roles in planet formation and generation of life 3. Detect signals from ET

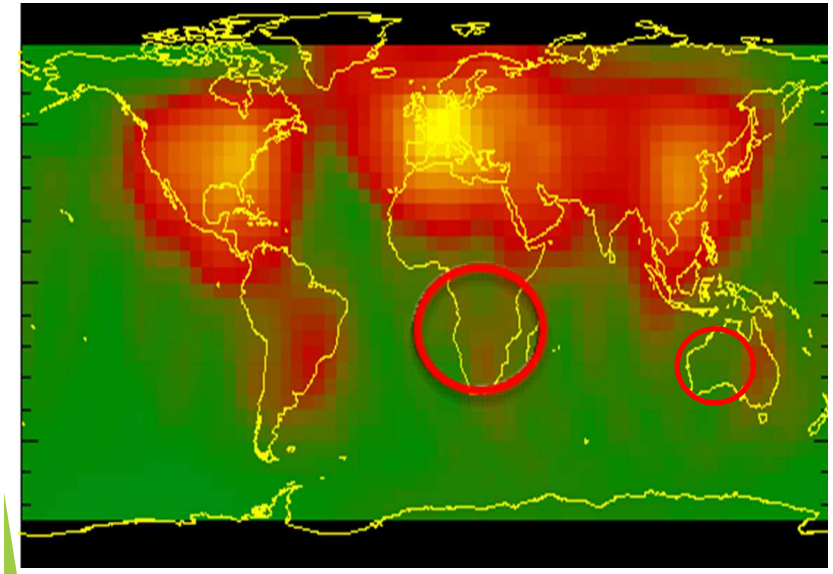
KSP ID	KSP Description	Frequency Range GHz	FoV	Sens-itivity	Survey Speed	Resn.	Base-line	Dyn. Range Driver	Poln. Driver
		0.1 0.3 1.0 3.0 10 30	deg^2	m^2/K	deg^{2}m^{4}K^{-2}	mas	Km		
1	**The Dark Ages**								
1a†	EoR				>~3x10^7		1	*	**
1b	First Metals		0.003	15,000		50	125		
1c	First Galaxies & BHs			20,000		10	4500	*	**
2	**Galaxy Evolution, Cosmology & Dark Energy**								
2a†	Dark Energy				6x10^9		5		
2b†	Galaxy Evolution			20,000	1x10^9		10		
2c	Local Cosmic Web				2x10^7		0.5		
3	**Cosmic Magnetism**								
3a†	Rotation Measure Sky				2x10^8		10-30		**
3b	Cosmic Web				1x10^8		5		**
4	**GR using Pulsars & Black Holes**								
	Search				1x10^8		< 1		
4a†	Gravitational Waves		-	>15,000		1	200		**
4b	BH Spin		1	10,000			-		**
4c†	Theories of Gravity			>15,000		1	200		**
5	**Cradle of Life**								
5a†	Protoplanetary Disks		<10^{-5}	10,000		2	1000		
5b	Prebiotic Molecules		0.5-1	10,000		100	60		
5c	SETI		1						
6	**Exploration of the Unknown**		Large	Large	Large				

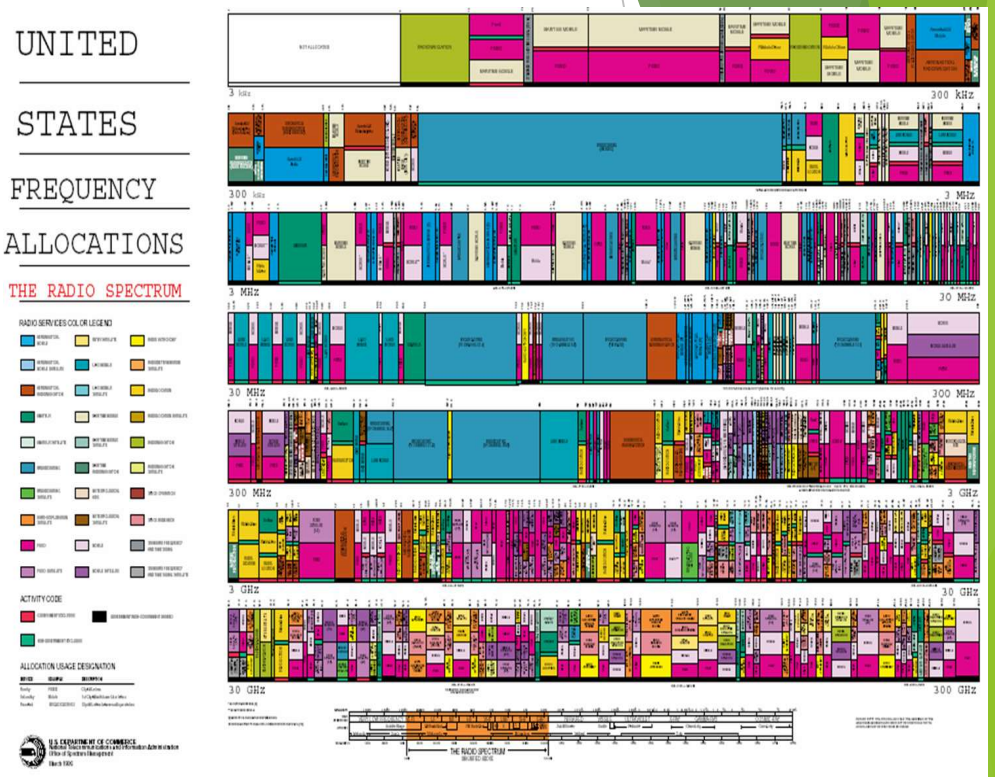
† Headline science, see Section 3.2



NASA/WMAP Science Team

Site selection: determined by man-made interference



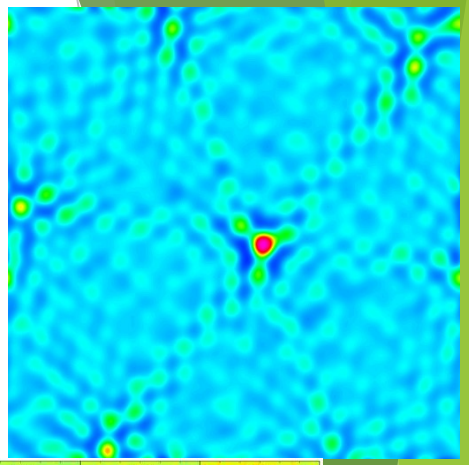


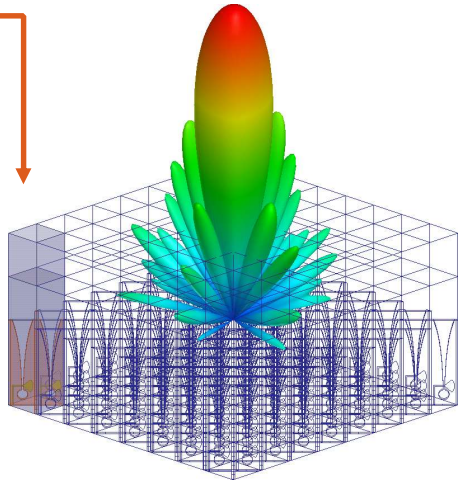
The majority of the radio spectrum has been allocated

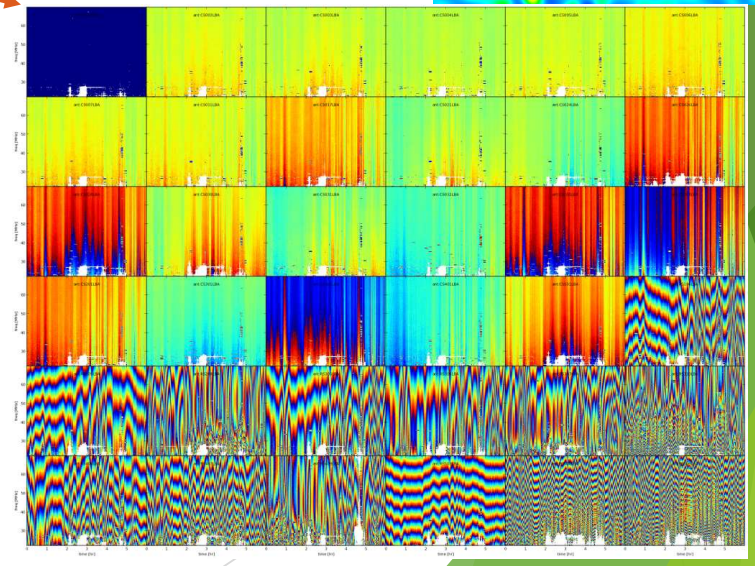
Only available radio-quit sites: south Africa and western Australia
(or the back side of the Moon)

Signal corruptions

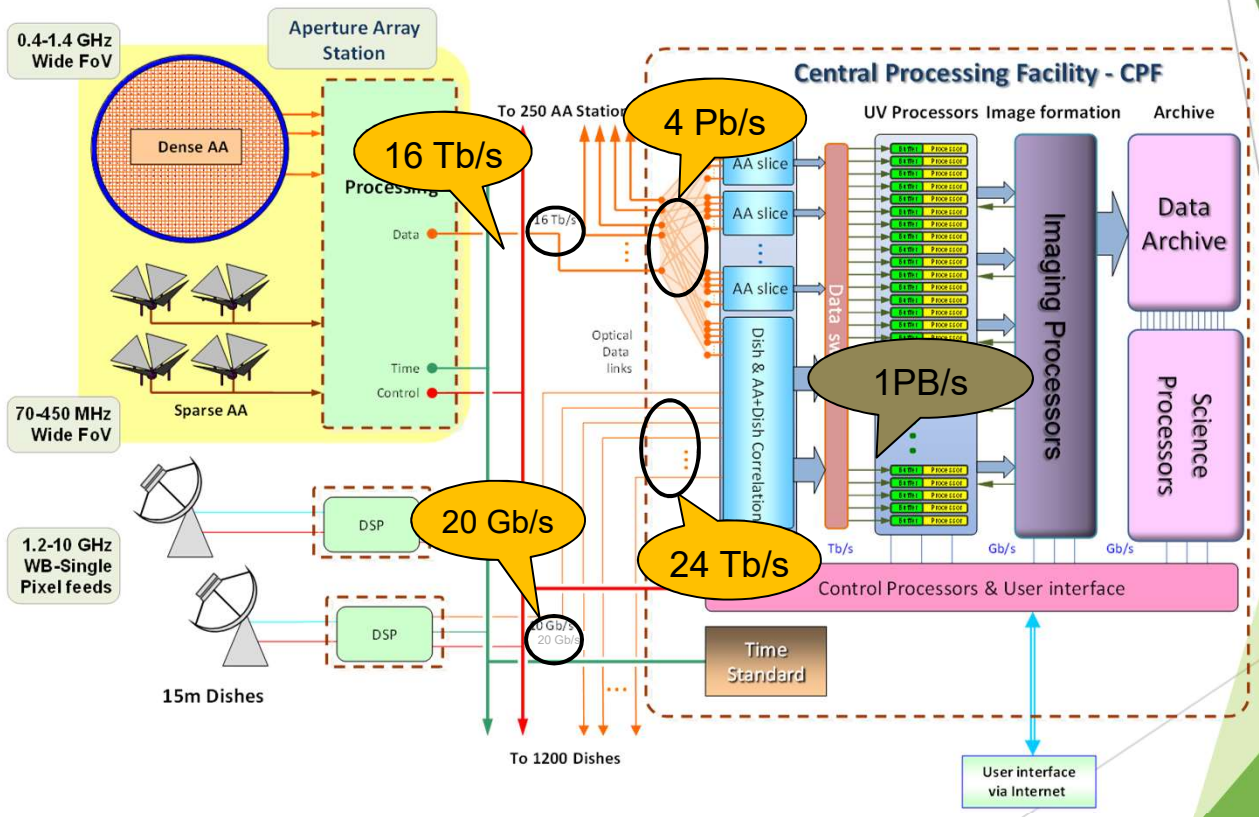
- Astronomical radio signals are very weak
- Calibration is the name of the game
- Several sources of corruption
  - Ionospheric Faraday Rotation/phase
  - Antenna/station beam patterns
  - Receiver gain and phase errors
- If not tackled, they will limited the sensitivity and therefore the science



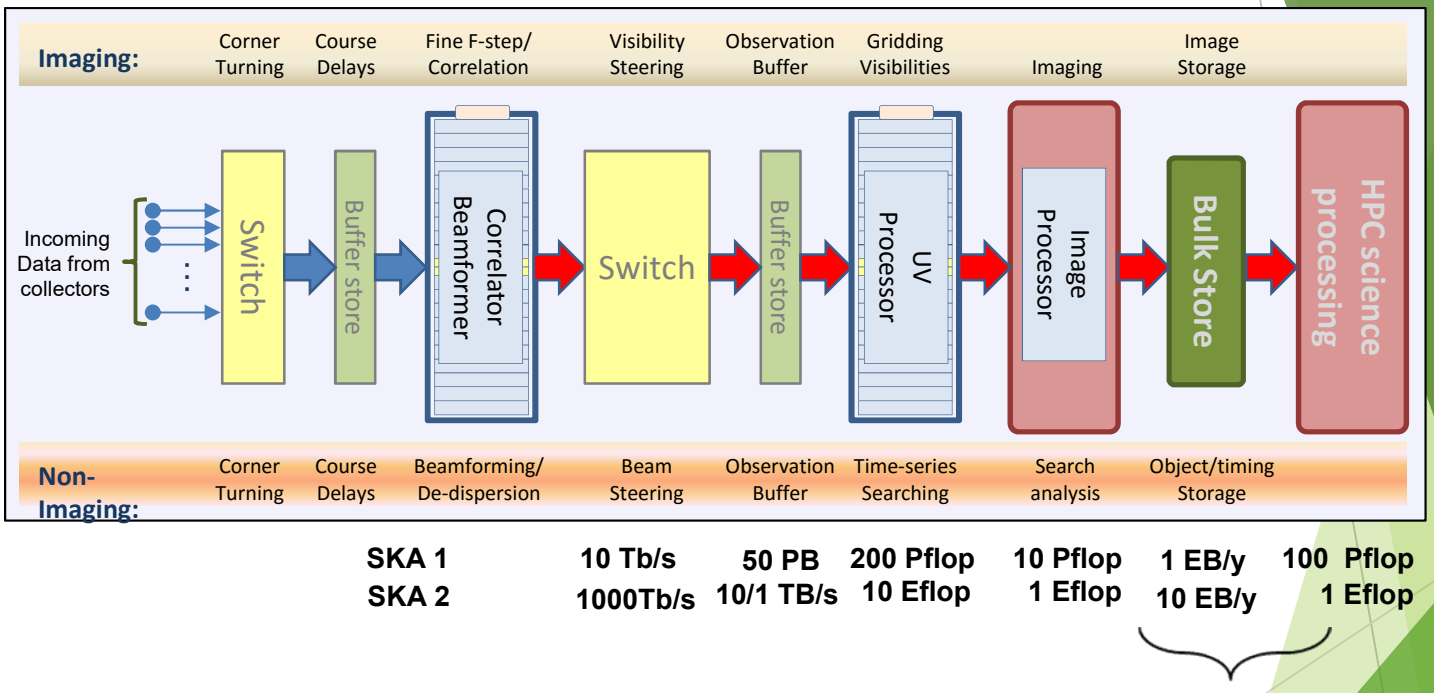




SKA data rates



Science data processor pipeline



Imaging and calibration determines problem size

30x more computationally intensive (deconvolution, gridding, prediction, FFTs)
90% of power consumption

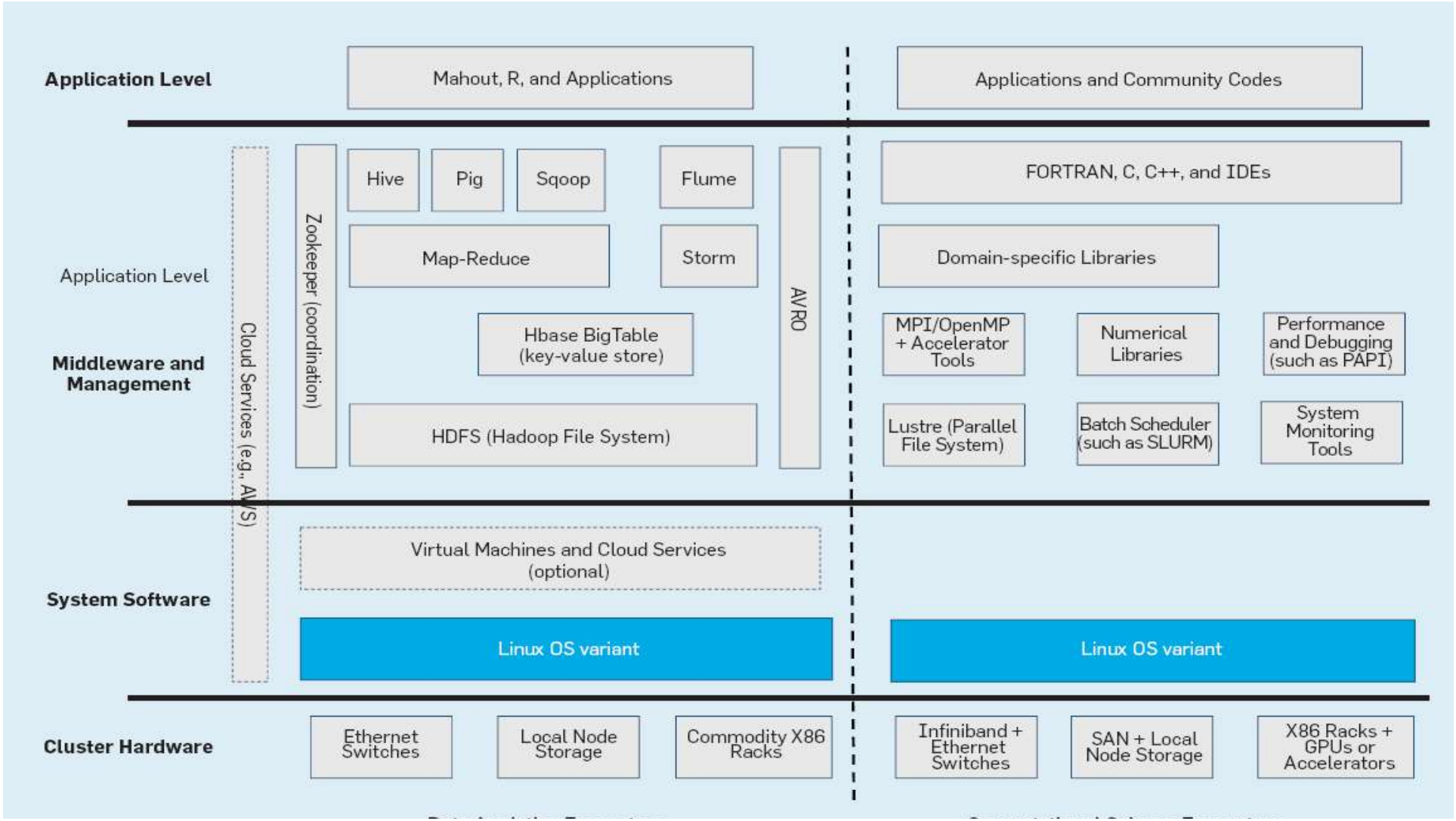
Current data reduction scheme

- Software developed in the 80s/90s
- Fortran libraries
- Minimal use of multi-cores/No use of accelerators
- Design for interactive processing of single dataset/Support for scripted batched processing
- ipython/custom REPL
- Will not work for SKA

```
#  simobserve :: mosaic simulation task:
project             =       'sim'        #  root prefix for output file names
skymodel            =         ''         #  model image to observe
complist            =         ''         #  componentlist to observe
setpointings        =       True
     integration    =       '10s'        #  integration (sampling) time
     direction      =         ''         #  "J2000 19h00m00 -40d00m00" or "" to center on model
     mapsize        =   ['', '']         #  angular size of map or "" to cover model
     maptype        =     'ALMA'         #  hexagonal, square, etc
     pointingspacing =         ''        #  spacing in between pointings or "0.25PB" or "" for 0.5 PB

obsmode             =       'int'        #  observation mode to simulate [int(interferometer)|sd(singledish)|""(none)]
     antennalist    = 'alma.out10.cfg'   #  interferometer antenna position file
     refdate        = '2012/05/21'       #  date of observation - not critical unless concatting simulations
     hourangle      =  'transit'         #  hour angle of observation center e.g. -3:00:00, or "transit"
     totaltime      =    '7200s'         #  total time of observation or number of repetitions
     caldirection   =         ''         #  pt source calibrator [experimental]
     calflux        =       '1Jy'

thermalnoise        = 'Sopapilla'        #  add thermal noise: [tsys-atm|tsys-manual|""]
leakage             =        0.0         #  cross polarization (interferometer only)
graphics            =     'both'         #  display graphics at each stage to [screen|file|both|none]
verbose             =      False
overwrite           =       True        #  overwrite files starting with $project
async               =      False        #  If true the taskname must be started using simobserve(...)
```


Application Level
Mahout, R, and Applications
Applications and Community Codes
Application Level
Middleware and Management
System Software
Cluster Hardware
Cloud Services (e.g., AWS)
Zookeeper (coordination)
Hive
Pig
Sqoop
Flume
Map-Reduce
Storm
Hbase BigTable (key-value store)
HDFS (Hadoop File System)
AVRO
FORTRAN, C, C++, and IDEs
Domain-specific Libraries
MPI/OpenMP + Accelerator Tools
Numerical Libraries
Performance and Debugging (such as PAPI)
Lustre (Parallel File System)
Batch Scheduler (such as SLURM)
System Monitoring Tools
Virtual Machines and Cloud Services (optional)
Linux OS variant
Linux OS variant
Ethernet Switches
Local Node Storage
Commodity X86 Racks
Infiniband + Ethernet Switches
SAN + Local Node Storage
X86 Racks + GPUs or Accelerators

What is Spark?

Fast and **Expressive** Cluster Computing System Compatible with Apache Hadoop

Up to **10x** faster on disk, **100x** in memory

2-5x less code

Efficient

- General execution graphs
- In-memory storage

Usable

- Rich APIs in Java, Scala, Python
- Interactive shell

Key Concepts

Write programs in terms of **transformations** on **distributed datasets**

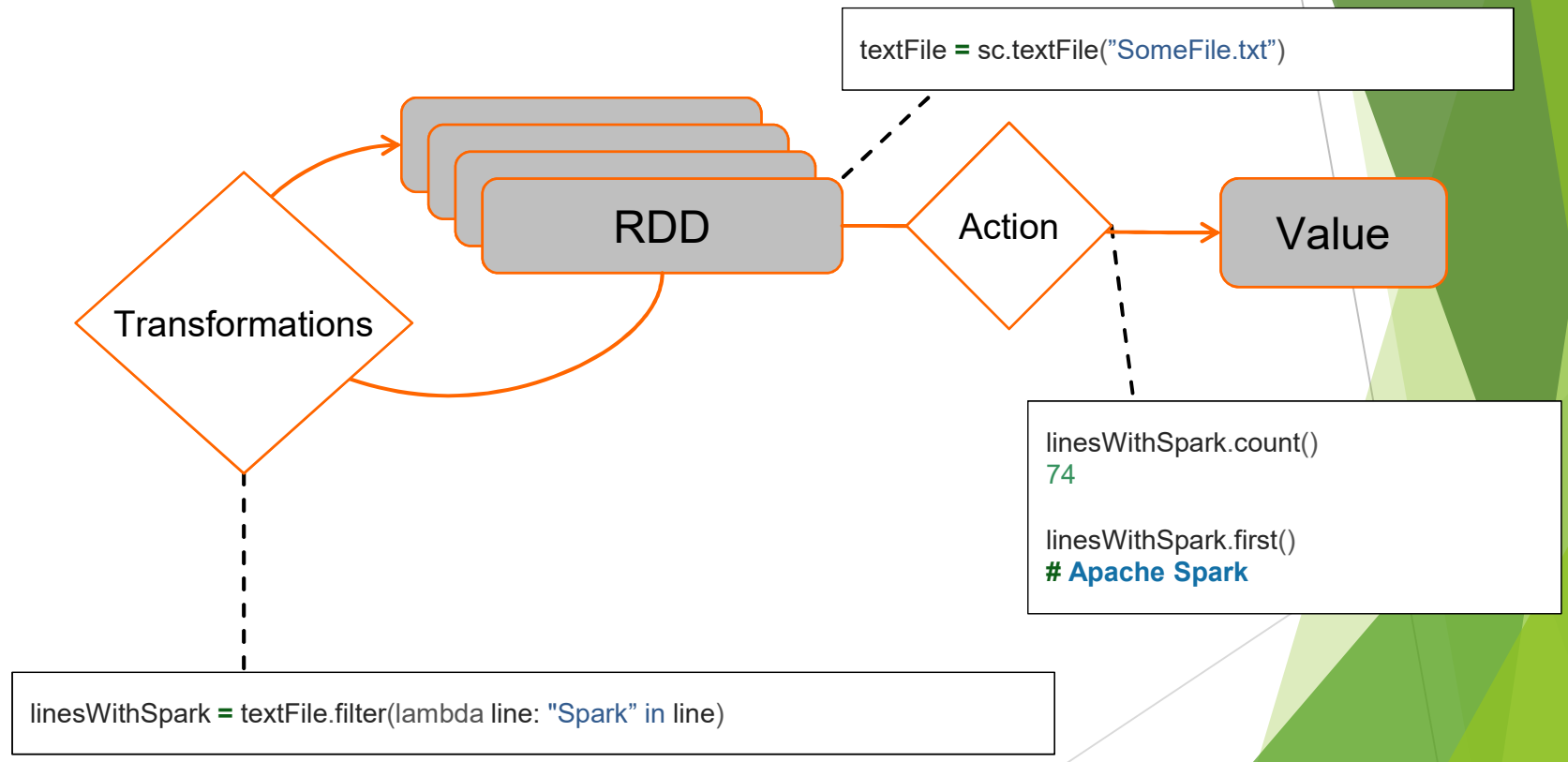
Resilient Distributed Datasets

- Collections of objects spread across a cluster, stored in RAM or on Disk
- Built through parallel transformations
- Automatically rebuilt on failure

Operations

- Transformations
  (e.g. map, filter, groupBy)
- Actions
  (e.g. count, collect, save)

Working With RDDs



Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

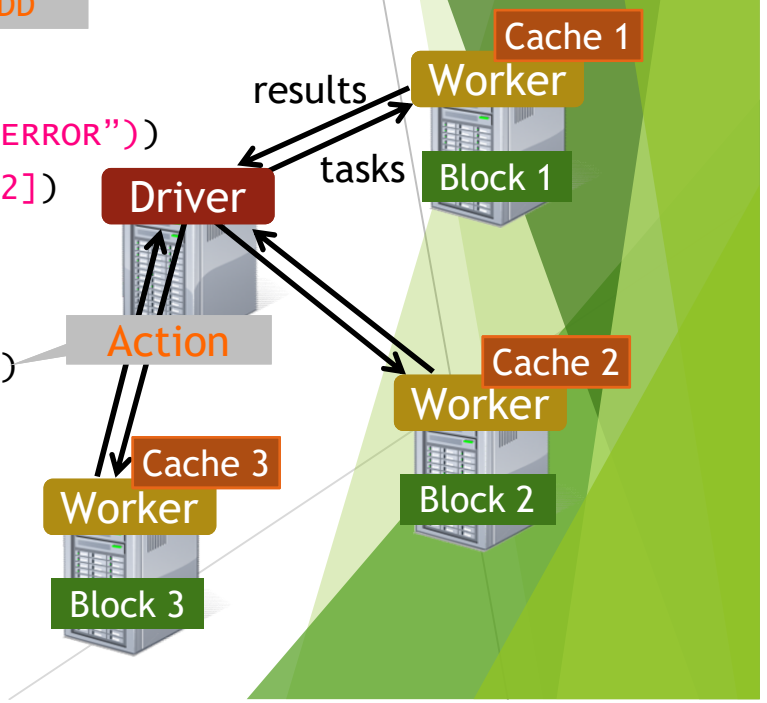
Transformed RDD

Base RDD

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()

messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
. . .
```

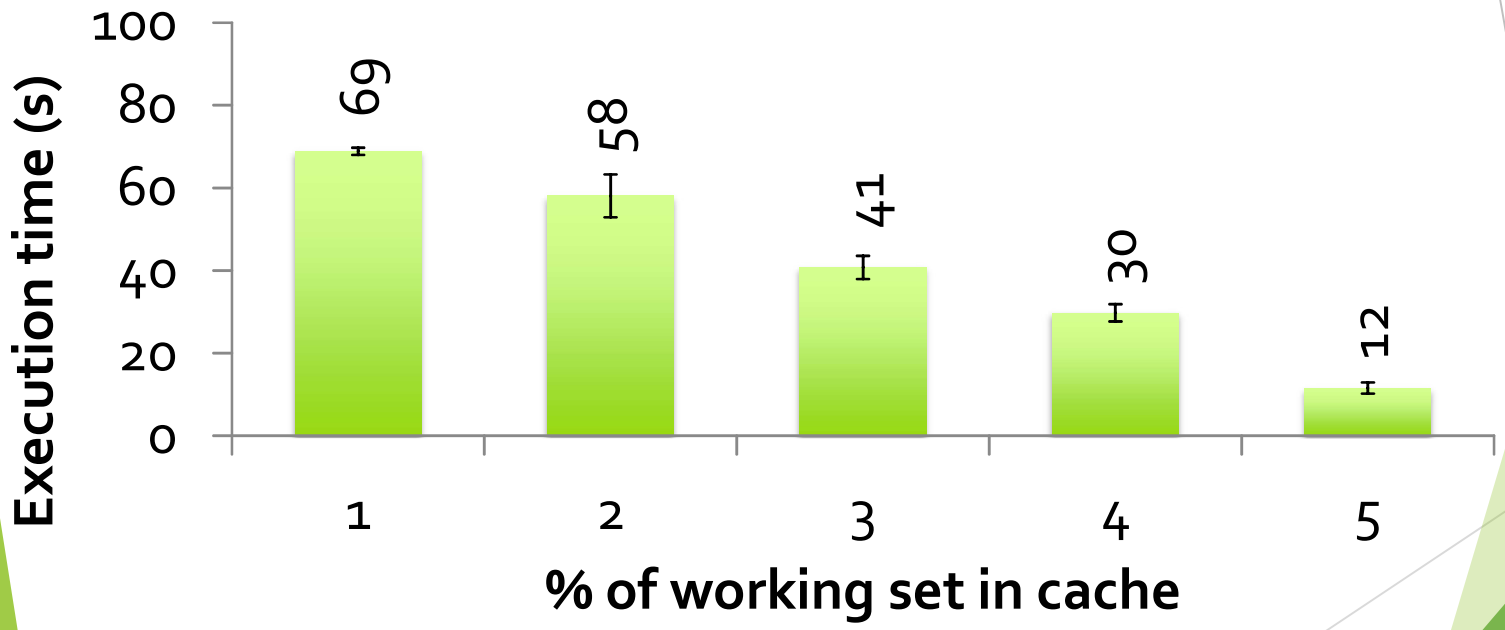
Action



Full-text search of Wikipedia

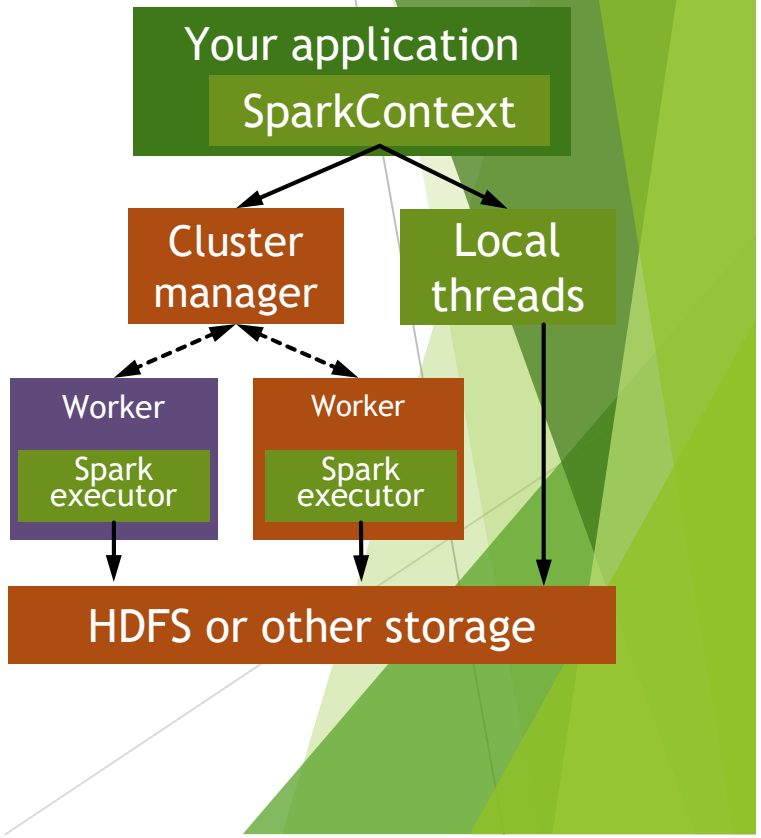
- 60GB on 20 EC2 machine
- 0.5 sec vs. 20s for on-disk

Scaling Down



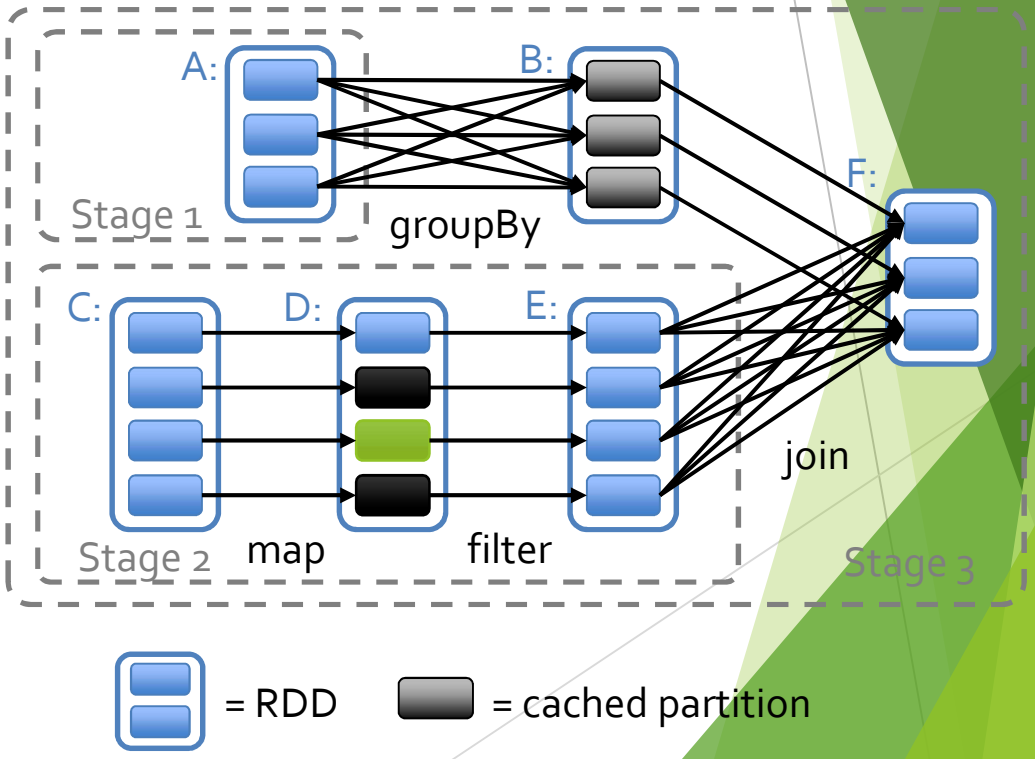
Software Components

- Spark runs as a library in your program (1 instance per app)
- Runs tasks locally or on cluster
  - Mesos, YARN or standalone mode
- Accesses storage systems via Hadoop InputFormat API
  - Can use HBase, HDFS, S3, …



Task Scheduler

- General task graphs
- Automatically pipelines functions
- Data locality aware
- Partitioning aware to avoid shuffles

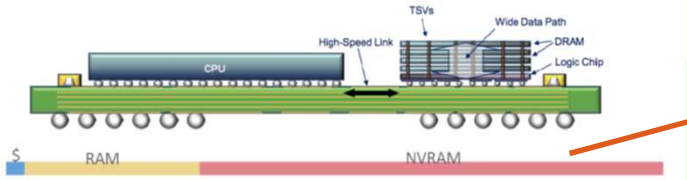


Advanced Features

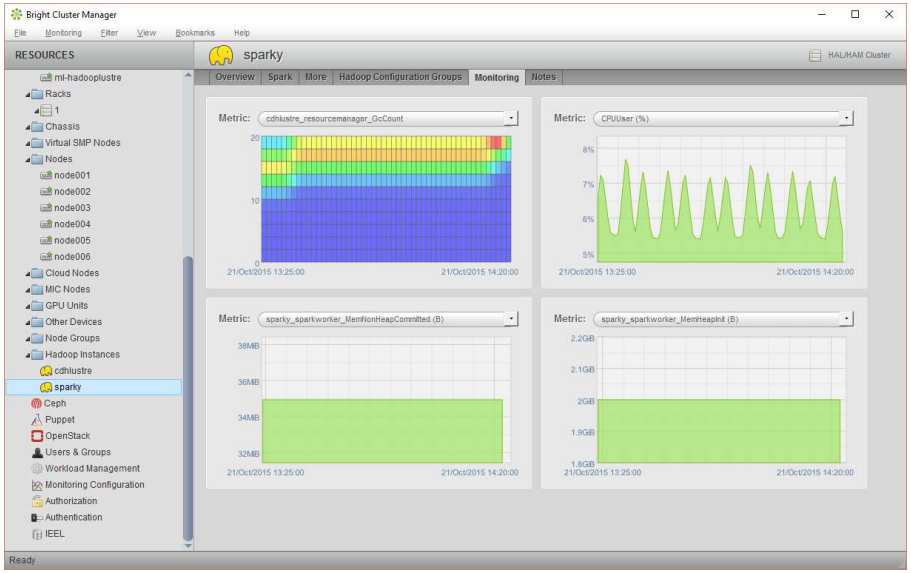
- Controllable partitioning
  - Speed up joins against a dataset
- Controllable storage formats
  - Keep data serialized for efficiency, replicate to multiple nodes, cache on disk
- Shared variables: broadcasts, accumulators
- See online docs for details!

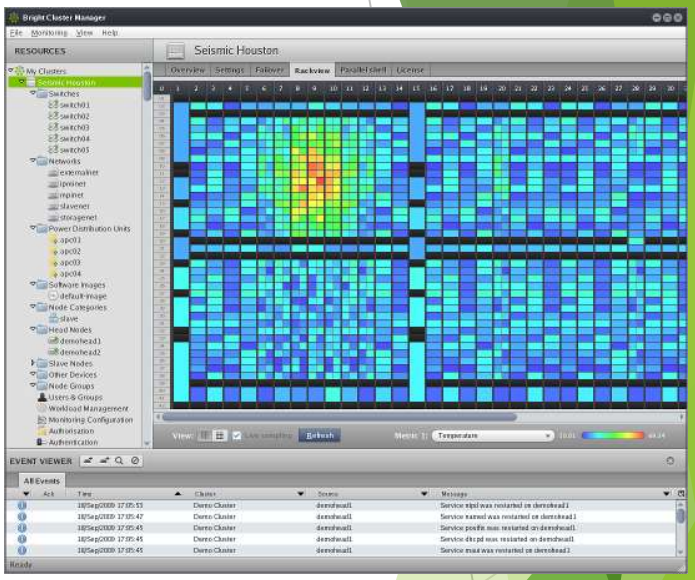
Why Spark

- Power consumption will be the most important differentiator –minimize movement
  - supercomputer in the middle of the desert
  - data locality – automatic work placement
  - In-memory processing
- Iterative algorithms / Pipelines that enable piggybacking
- New algorithms that have modest inter-process communication
  - Not all features of MPI are required -> Simple collectives: broadcast, aggregation
  - Can MPI handle the increased pararellism available on exascale systems?
- Fault tolerance
  - Current MTBF: 2 out of 600 nodes per week -> 300 nodes per week for SKA
  - MPI applications need to be designed for fault-tolerance/checkpointing
- Straggler mitigation
- Friendly APIs



Fault Tolerance





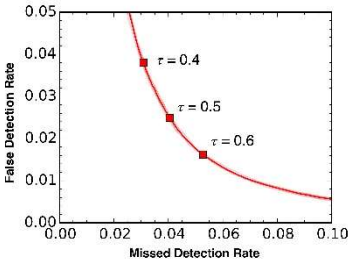
- Bright cluster manager
  - HDFS/Lustre/Ceph support
  - Early warning via e-mail, SMS etc and automate actions
  - Single plane of glass: monitor nodes, switches, PDUs, multiple clusters
  - Hadoop/Spark-specific metrics and healthchecks
  - Simple Python/REST/C++ APIs

How is data processed

- Datasets contain multiple sources of information
  - Ionosphere, lightning, cosmic rays, ET?, interference, compact and extended astronomical sources
  - Different algorithms for each
- Multidimensional data: spatial, time, frequency, polarization
- Pipeline should be able to chain different algorithms in order to extract as much information as possible with mini mal data movement

Types of processing:

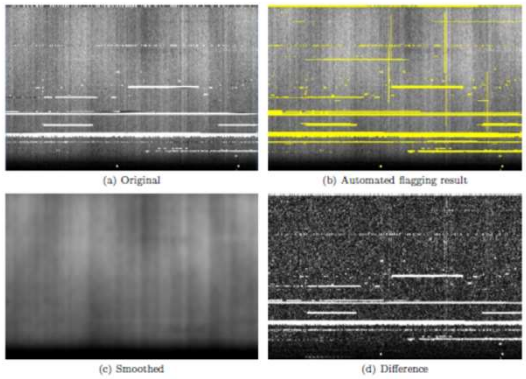
- Signal detection/extraction
- Object classification
- AGN spectral classification
- Time series
- Clustering
- LSS
- Class discovery
- RFI mitigation
- Transient detection
- Deconvolution

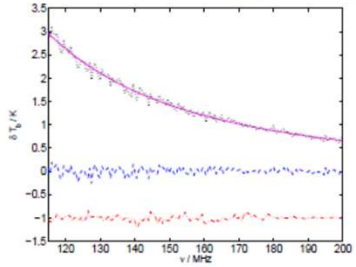


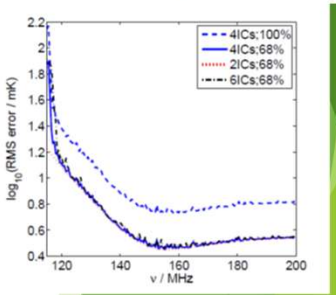
autoScan algorithm
Goldstein et al 2015

Types of algorithms:

- ANNs (this talk)
- Decision trees
- SVM
- Nearest neighbor
- Expectation maximization
- Non-linear optimization

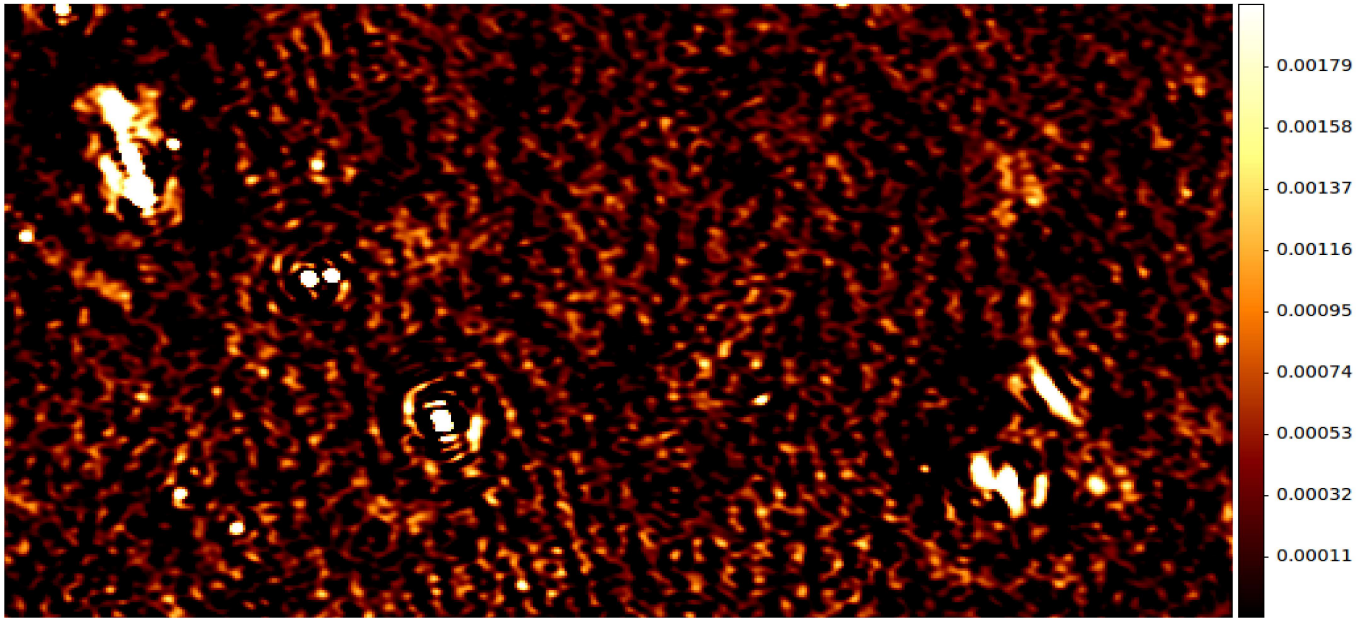




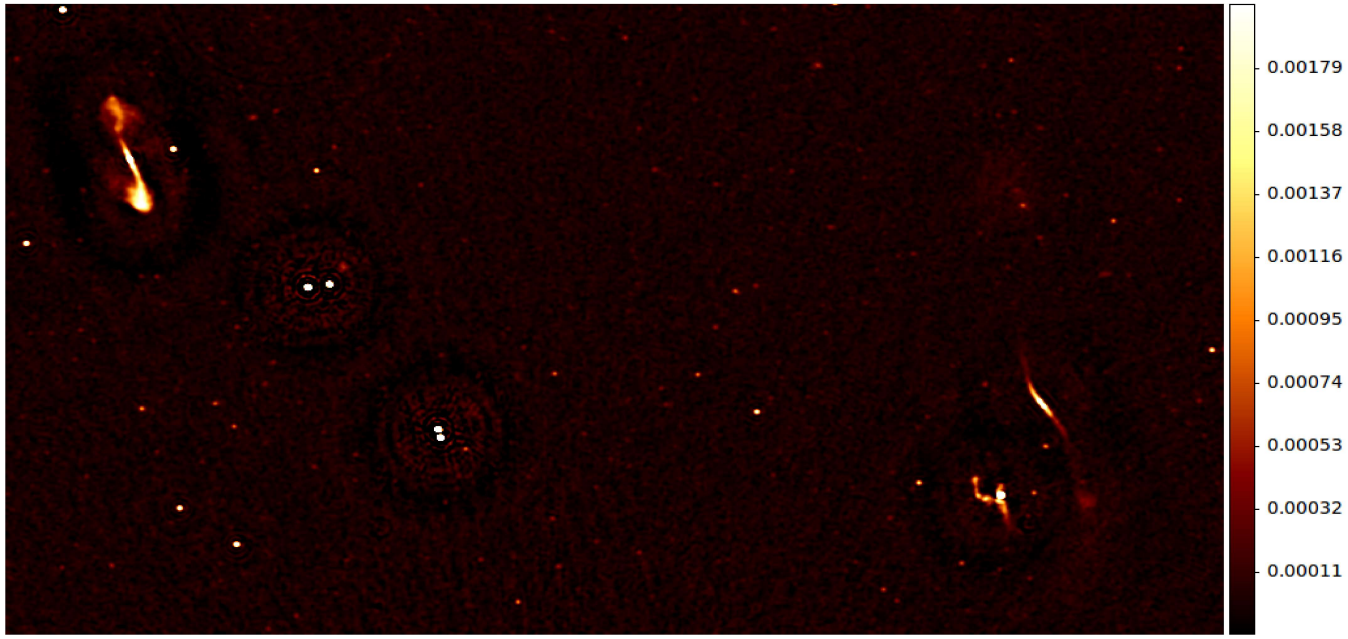


Chapman et al 2012

Case study: Calibration

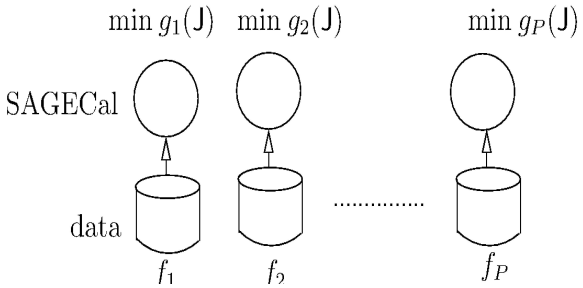


Case study: Calibration

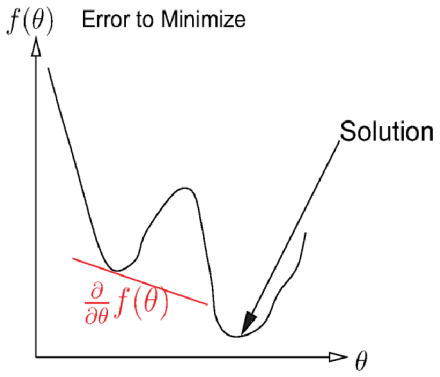


Case study: Calibration

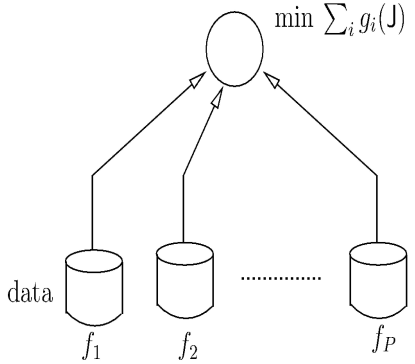
Data distributed over a network: all existing calibration algorithms work independently



$$\mathsf{J}_{f_i} = \arg\min_{\mathsf{J}} \; g_{f_i}(\mathsf{J})$$



We want a unified solution, collecting information from all data



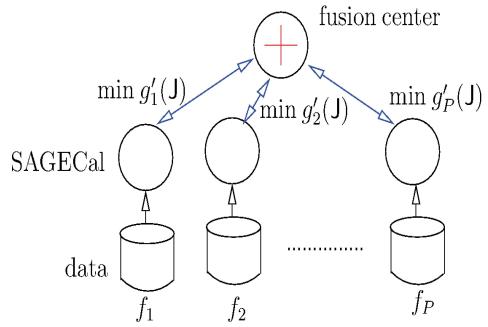
$$\{\mathsf{J}_{f_1}, \mathsf{J}_{f_2}, \ldots, \mathsf{Z}\} = \arg\min_{\mathsf{J}_{f_i},\ldots,\mathsf{Z}} \sum_i g_{f_i}(\mathsf{J}_{f_i})$$

$$\text{subject to } \; \mathsf{J}_{f_i} = \mathsf{B}_{f_i}\mathsf{Z}, \quad i \in [1, P]$$

Where $B_{f_i}Z$ is a smoothness constraint along time, space and freqency

Does not work in practice: data does not fit in memory, model not accurate enough to parametrize

Case study: Calibration

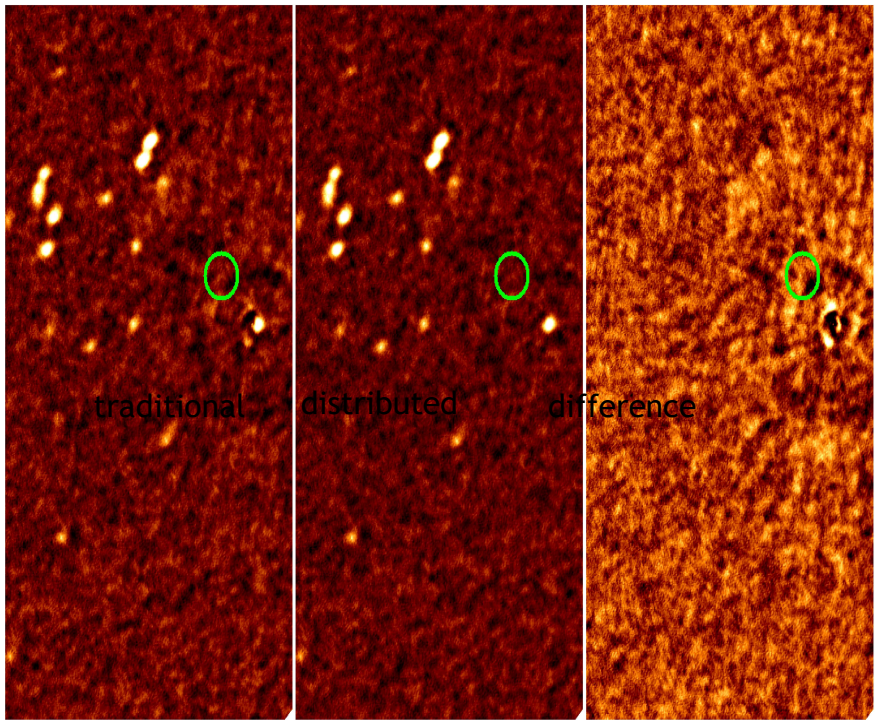


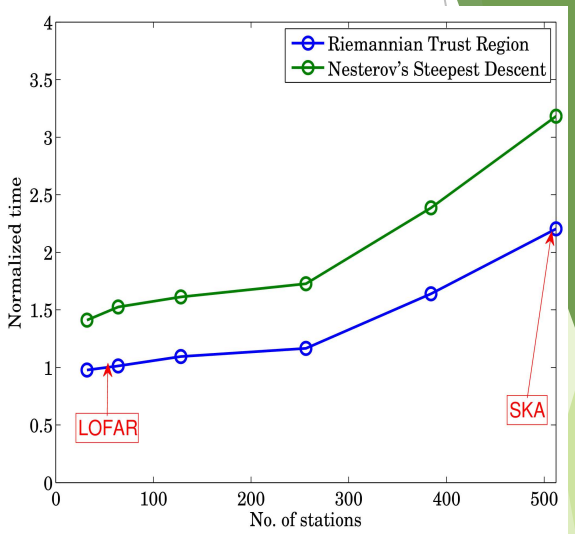
$$L(\mathbf{J}_{f_1},\ldots,\mathbf{Z},\mathbf{Y}_{f_1},\ldots)=\sum_i g_{f_i}(\mathbf{J}_{f_i})+\|\mathbf{Y}_{f_i}^H(\mathbf{J}_{f_i}-\mathbf{B}_{f_i}\mathbf{Z})\|+\frac{\rho}{2}\|\mathbf{J}_{f_i}-\mathbf{B}_{f_i}\mathbf{Z}\|^2$$

- Consensus optimization: exploit smoothness of systematic errors to find a unified solution
- Use of a fusion center is not essential, but easier
- Communication overhead is little, more important is robustness and fault tolerance (can throw away lost data).
- Large rho makes problem convex

1. INITIALIZE
   - Read data as Breeze BlockMatrix
   - P sub-problems, each with its own copy of Z,Y
2. REPEAT until convergence or N_max_iter
   1. Each worker solves:
      $$(\mathrm{J}_{f_i})^{n+1}=\arg\min_{\mathrm{J}} L_i\,(\mathrm{J},(\mathrm{Z})^n,(\mathrm{Y}_{f_i})^n)$$
   2. Globally calculate average:
      $$(\mathrm{Z})^{n+1}=\arg\min_{\mathrm{Z}}\sum_i L_i\,((\mathrm{J}_{f_i})^{n+1},\mathrm{Z},(\mathrm{Y}_{f_i})^n)$$
   3. Broadcast Z to P sub-problems
      Zb = sc.broadcast(Z)
   4. Locally calculate Lagrange multiplier
      $$(\mathrm{Y}_{f_i})^{n+1}=(\mathrm{Y}_{f_i})^n+\rho((\mathrm{J}_{f_i})^{n+1}-\mathrm{B}_{f_i}(\mathrm{Z})^{n+1})$$

Case study: Calibration





- Almost linear caling with number of antennas
- Broadcasted data is 10x less (4KNP) than actual data $(4TP\frac{N(N-1)}{2})$
- Can be further reduced with:
  - Use frequency multiplexing
  - Broadcast data only to neighbors

Case study: Imaging

1. INITIALIZE
   - a *residual* image set to the Fourier Transforr of the visibilities
   - var row: Int = _, var col: Int = _
   - (row , col, flux) MAX= tuple(l: Int, m: int)
   - a *Clean Component* list to empty

2. WHILE (img.getNoise < THERMAL_NOISE)
   1. (row , col, flux) = img.getMax
   2. resimg = resimg – g * PSF.imshift(l,m,size)
   3. CCList.++ = (row , col, flux)

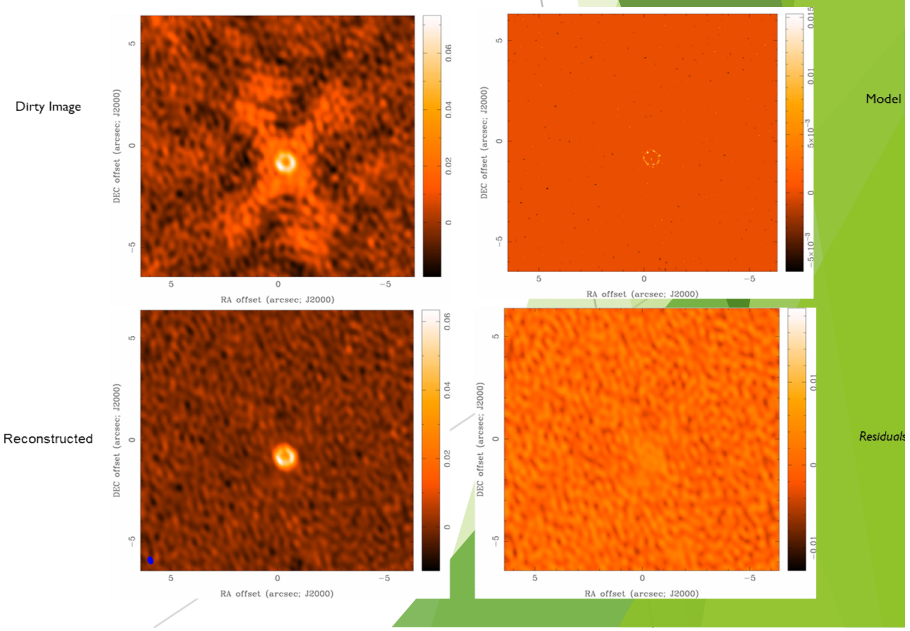
- linear scaling
- fits in memory
- easy to implement
- gridding/FFT can be offloaded to GPU via JNI
- Does not work well with extended sources, still useful for catalogs a nd calibration

Poor man's Compressed Sensing (matching pursuit)

Variations:

- Clarke: subtract multiple components in on iteration (~ StOMP)
- Cotton-Schwab: Work in measurement space instead of image space

In the absence of noise, CLEAN is equivalent to a least squares fit of sinusoids to visibilities



Imaging

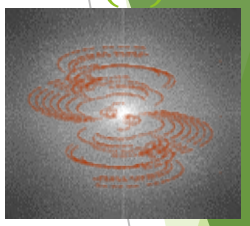
- The “dirty” image is the convolution of the true underlying image with the PSF, corrupted by Gaussian random noise

$$y = k * x + n$$

Convolution Theorem

$$F(y) = F(k)\,F(x)$$

$F(k)$



$$x = F^{-1}\left(\frac{1}{F(k)}\right) * y$$

This pseudo-inverse cannot be computed in the presence of corruptions

$$x = F^{-1}\left(\frac{1}{F(k)}\frac{|F(k)|^2}{|F(k)|^2 + SNR^{-1}}\right) * y$$

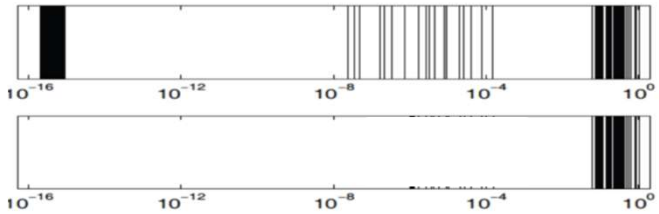
Tikhonov Regularization
High noise strong → regularization

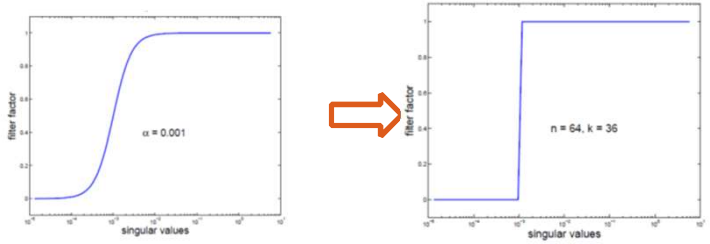
$$x = \mathrm{argmin}\|y - x * k\| + R(x) + \cdots$$

Imaging

The previous kernel can be transformed as a sum of 1D filters:

$$k^+ * y = S \quad (k^+) * y = \sum_j \sigma_j u_j * (v_j^T * y) \quad \sim \frac{MNPQ}{2MN(P+Q)} \text{ faster}$$



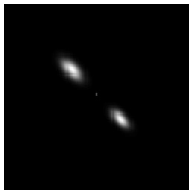
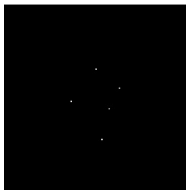
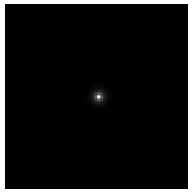
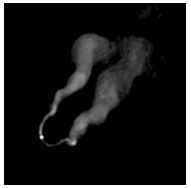


Goal: use the following feed-forward, multi-layer convolutional neural network and train it to minimize

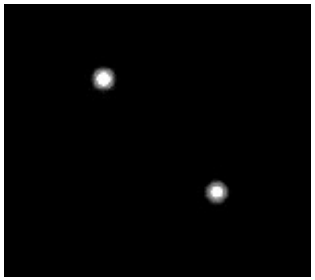
$$h_l = \begin{cases} \hat{y}, l = 0 \\ \tanh(W_l * h_{l-1} + b_{l-1}), l \in \{1,2\} \\ W_3 * h_2 \end{cases}$$

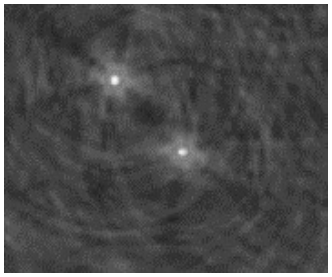
Initialize W from k^+ or from random, uniform distribution

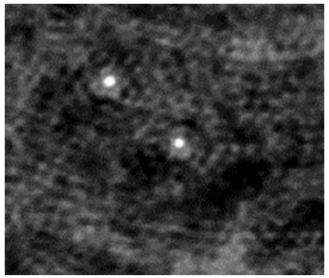
Astrophysical sources are complication, but most radio sources have simple shapes, and occasionally irregular:

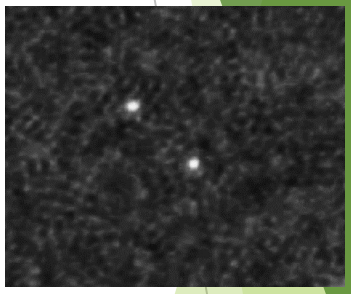





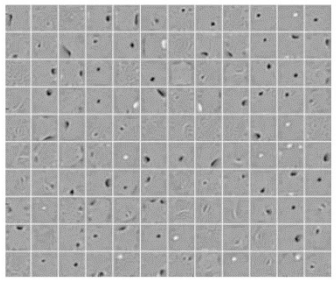
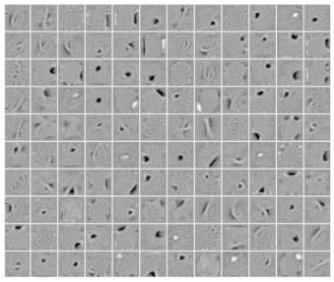
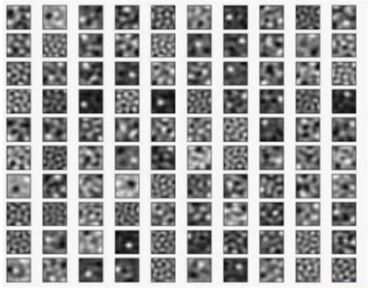
Use high SNR images from the VLA FIRST survey supplemented with simulated sources (elliptical Gaussians, points sources, narrow gaussians) to construct a training pair set of 10.000 64x64 images


uncorrupted


convolved


noise added


recovered

Challenges

- Spark is feature already but some useful components are not yet available

Issue Links

incorporates	SPARK-2352 [MLLIB] Add Artificial Neural Network (ANN) to Spark	IN PROGRESS
	SPARK-10409 Multilayer perceptron regression	OPEN
	SPARK-10408 Autoencoder	OPEN
	SPARK-10627 Regularization for artificial neural networks	OPEN
is related to	SPARK-2623 Stacked Auto Encoder (Deep Learning)	OPEN
	SPARK-4251 Add Restricted Boltzmann machine(RBM) algorithm to MLlib	OPEN
	SPARK-2352 [MLLIB] Add Artificial Neural Network (ANN) to Spark	IN PROGRESS
	~~SPARK-9273~~ Add Convolutional Neural network to Spark MLlib	RESOLVED
	~~SPARK-9471~~ Multilayer perceptron classifier	RESOLVED
	~~SPARK-2351~~ Add Artificial Neural Network (ANN) to Spark	CLOSED
	~~SPARK-4752~~ Classifier based on artificial neural network	CLOSED
	SPARK-4288 Add Sparse Autoencoder algorithm to MLlib	OPEN
relates to	SPARK-8449 HDF5 read/write support for Spark MLlib	OPEN

Spark / SPARK-1543
Add ADMM for solving Lasso (and elastic net) problem

- JNI
- Analytical implementations needed
- Hard to debug and argue about performance sometimes
- Application specific formats
- Accelerators such as GPUs are almost mandatory – JNI

Conclusions

- New generation interferometers will enable us
- The availability of HPC (exascale) resources is a prerequisite
- Energy cost will be the decisive factor when it comes to design
- New hardware architectures require new algorithmic approaches
- Is Spark the answer? To early to answer but maybe
  - Distributed algorithms not only address computational issues, but are more robust and accurate as well
  - Consensus calibration is the best way forward
  - Spark offers many of the tools need to construct radio astronomical pipelines