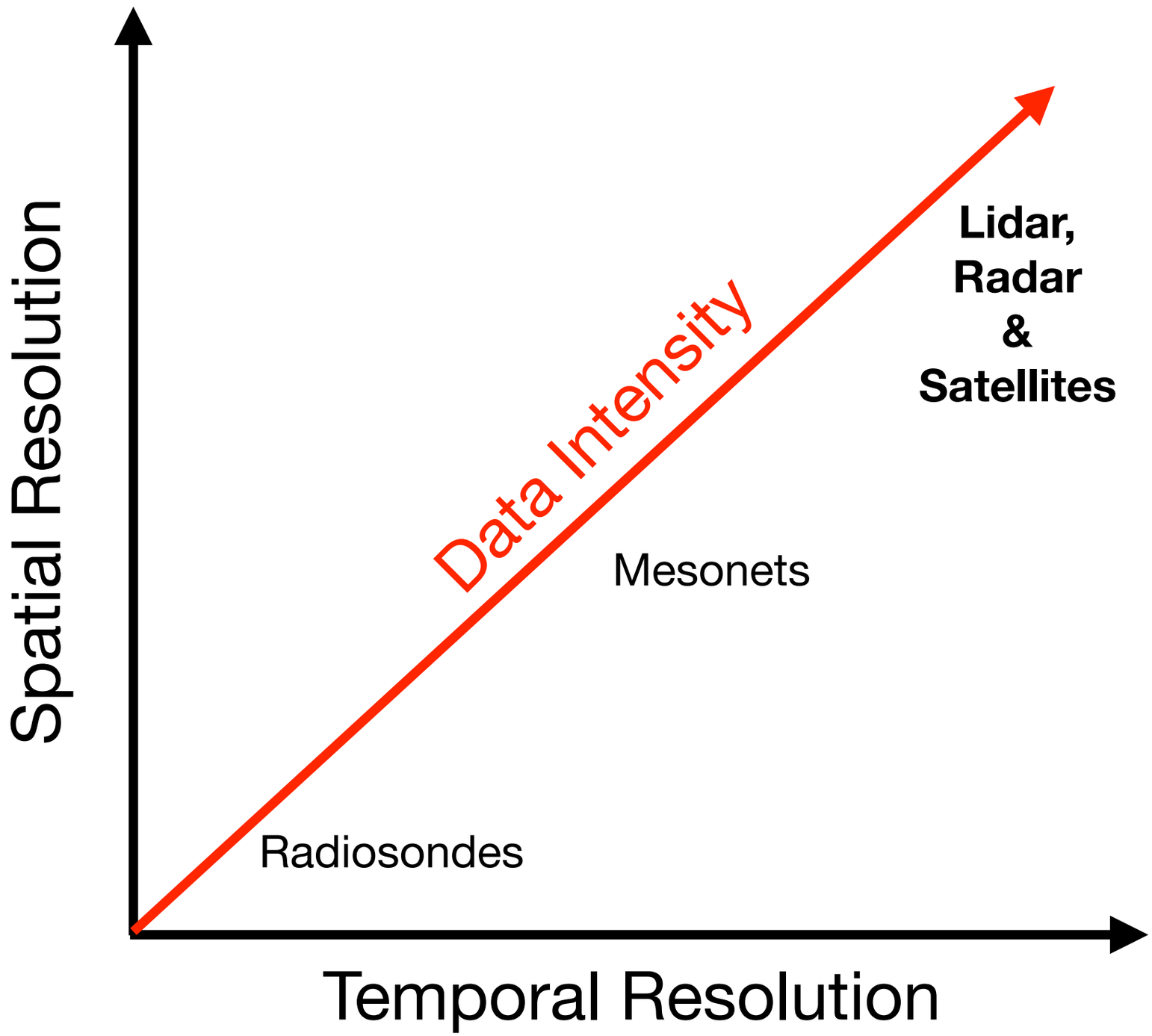
Kick-off Workshop April 11 - 12, 2017

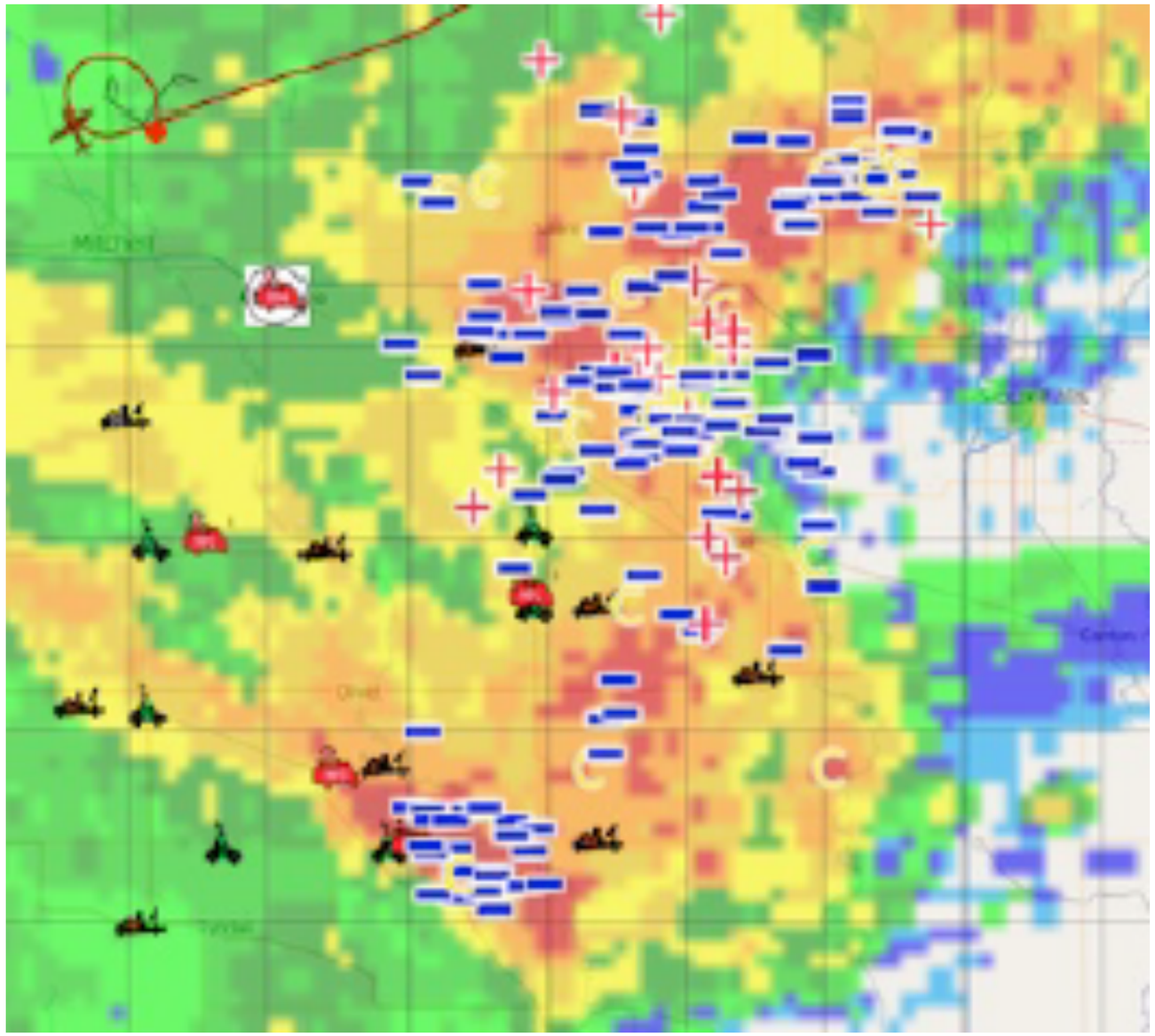
Lidar-Radar Open Software Environment (LROSE) Overview and Workshop Goals

Michael M. Bell
Colorado State University

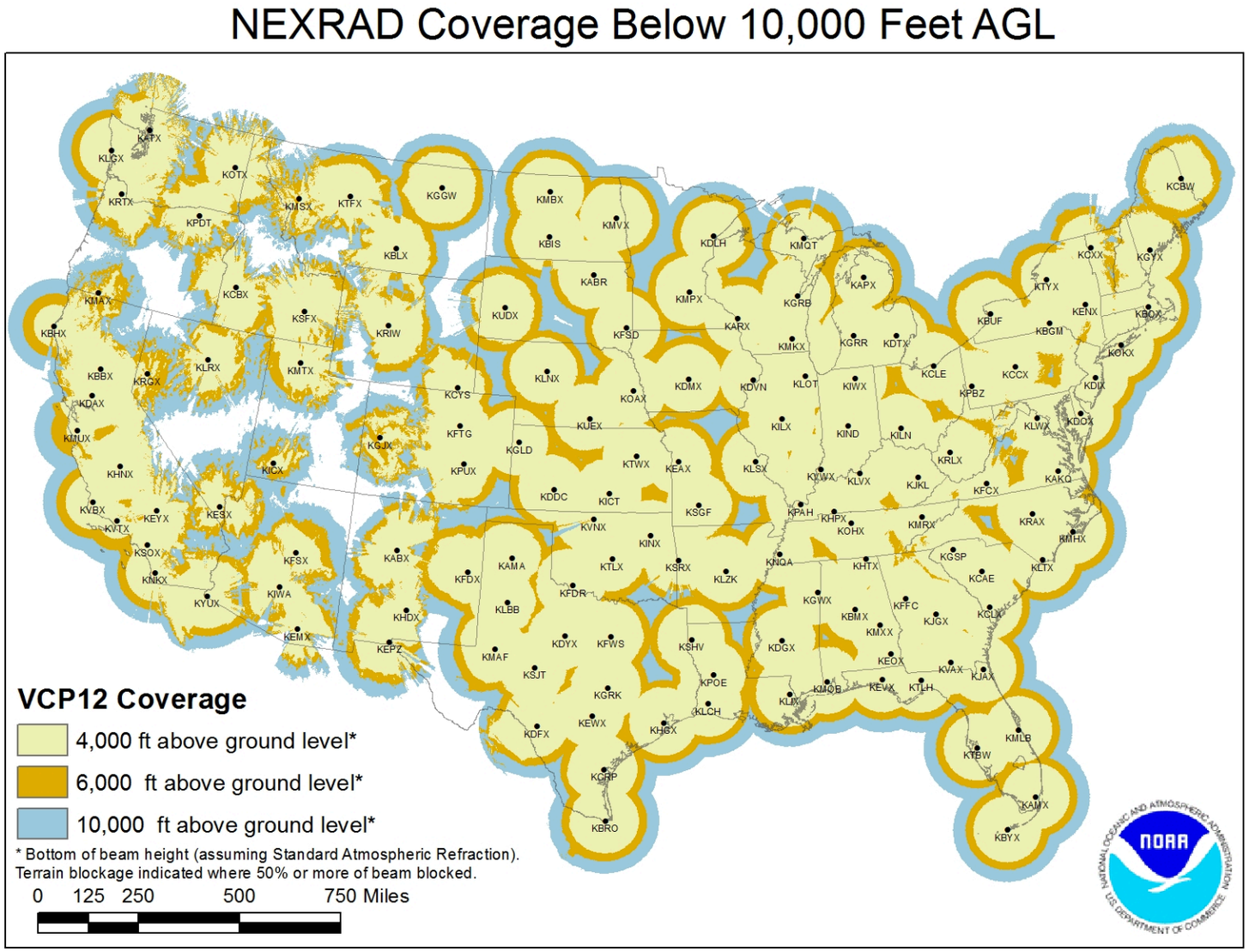
Wen-Chau Lee and Mike Dixon
NCAR

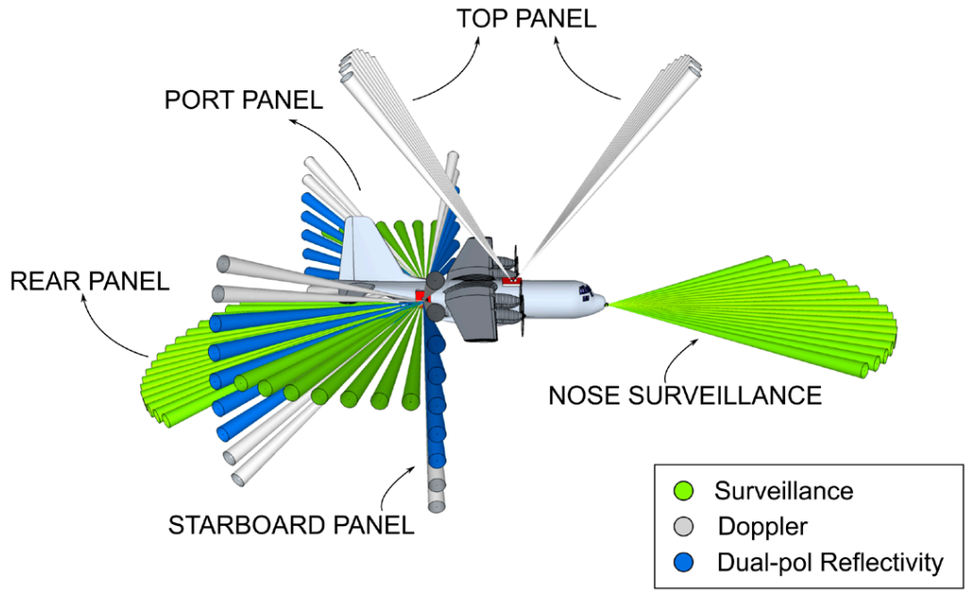

Spatial Resolution
Temporal Resolution
Data Intensity
Lidar,
Radar
&
Satellites
Mesonets
Radiosondes

- 5 minutes of NSF sponsored PECAN radar data on 6 July 2015 from 12 radars contains over 200 sweeps and ~**50 million** individual radar gates in a 100 x 100 km domain
- Polarimetric radars produce Doppler velocity, radar reflectivity, differential reflectivity, differential phase, and correlation coefficient (**5x = 2.5x10^{8} obs / 5 minutes**)
- Research radars record time series of each radar pulse, ranging from 1000 - 10,000 Hz. Assimilate just 2nd order moments of the H & V spectra (**4x ~ 10^{9} obs / 5 minutes)**
- Squall line maturation and decay occurs over 9 hour IOP (**100x = 10^{11} obs = 0.1 Tobs**)



Exascale challenges: Massive data flow, storage, quality control, compression, and assimilation



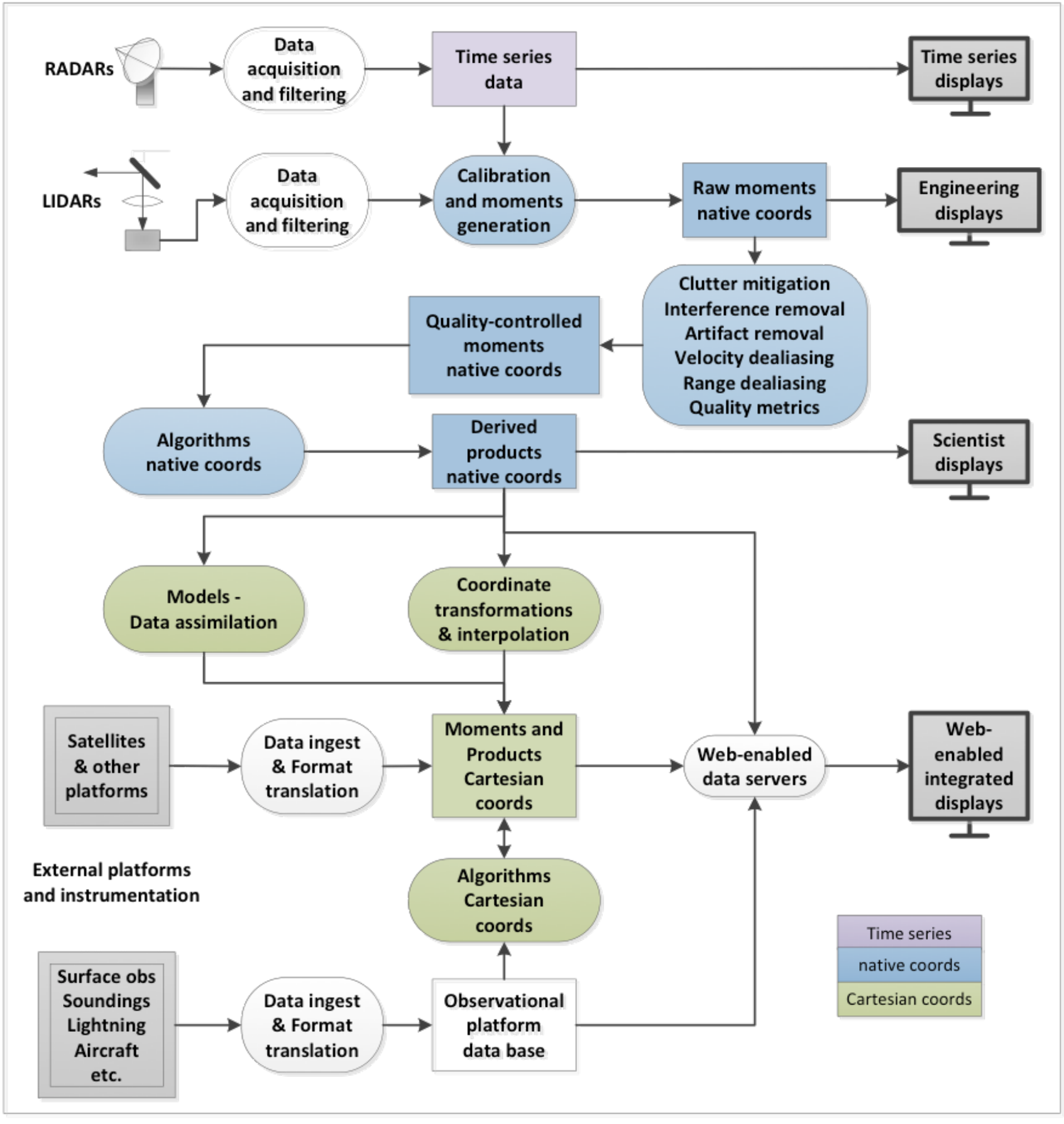


APAR

LROSE: Lidar-Radar Open Software Environment

- LROSE is a project to develop common software for the LIDAR, RADAR and Profiler community
- Joint 4-year project between Colorado State University and NCAR Earth Observing Laboratory recently funded by NSF SI2-SSI (Software Infrastructure for Sustained Innovation)
- It is based on collaborative, open source development, with algorithms and analysis tools developed and supported by the community.
- Data stored in portable data formats such as CfRadial, based on UNIDATA NetCDF, following the Climate and Forecasting (CF) conventions to facilitate data assimilation by models

https://nsf-lrose.github.io


RADARs
Data acquisition and filtering
Time series data
Time series displays
LIDARs
Data acquisition and filtering
Calibration and moments generation
Raw moments native coords
Engineering displays
Clutter mitigation
Interference removal
Artifact removal
Velocity dealiasing
Range dealiasing
Quality metrics
Quality-controlled moments native coords
Algorithms native coords
Derived products native coords
Scientist displays
Models - Data assimilation
Coordinate transformations & interpolation
Satellites & other platforms
Data ingest & Format translation
Moments and Products Cartesian coords
Web-enabled data servers
Web-enabled integrated displays
External platforms and instrumentation
Algorithms Cartesian coords
Surface obs
Soundings
Lightning
Aircraft
etc.
Data ingest & Format translation
Observational platform data base
Time series
native coords
Cartesian coords

Radar Software Needs	Personal Research Needs	Community Needs	Total Score
NCAR-maintained centralized repository for radar software (esp. including wind synthesis) with data sets for software testing	55	41	96
Standardized software packages and toolkits (multi-platform, modular, menu-driven, easy for community to add to, ease of conversion among new and old radar data formats)	30	53	83
Training (workshops/online tutorials)	48	15	63
Ability to integrate radar and non-radar data sets	25	32	57
Open source tools and software	30	16	46
3D/4D Visualization Software (with publication quality output)	24	21	45
Next generation wind synthesis software to replace the legacy (REORDER/CEDRIC) algorithms, while maintaining current functionality	15	27	42
Common radar data format standard and a common metadata standard (e.g. CfRadial)	19	15	34
64-bit compatible real-time display software tool	19	11	30
Improved radar data quality control (solo) (Oye et al., 1995)	12	20	32
Automated quality control software	14	13	27
Detailed documentation for data products, tools, and code	18	7	25
Improved dual-polarization processing	10	12	22
Accessible variational Doppler radar assimilation and thermodynamic retrieval	7	4	11
Totals	326	287	613

LROSE Overview

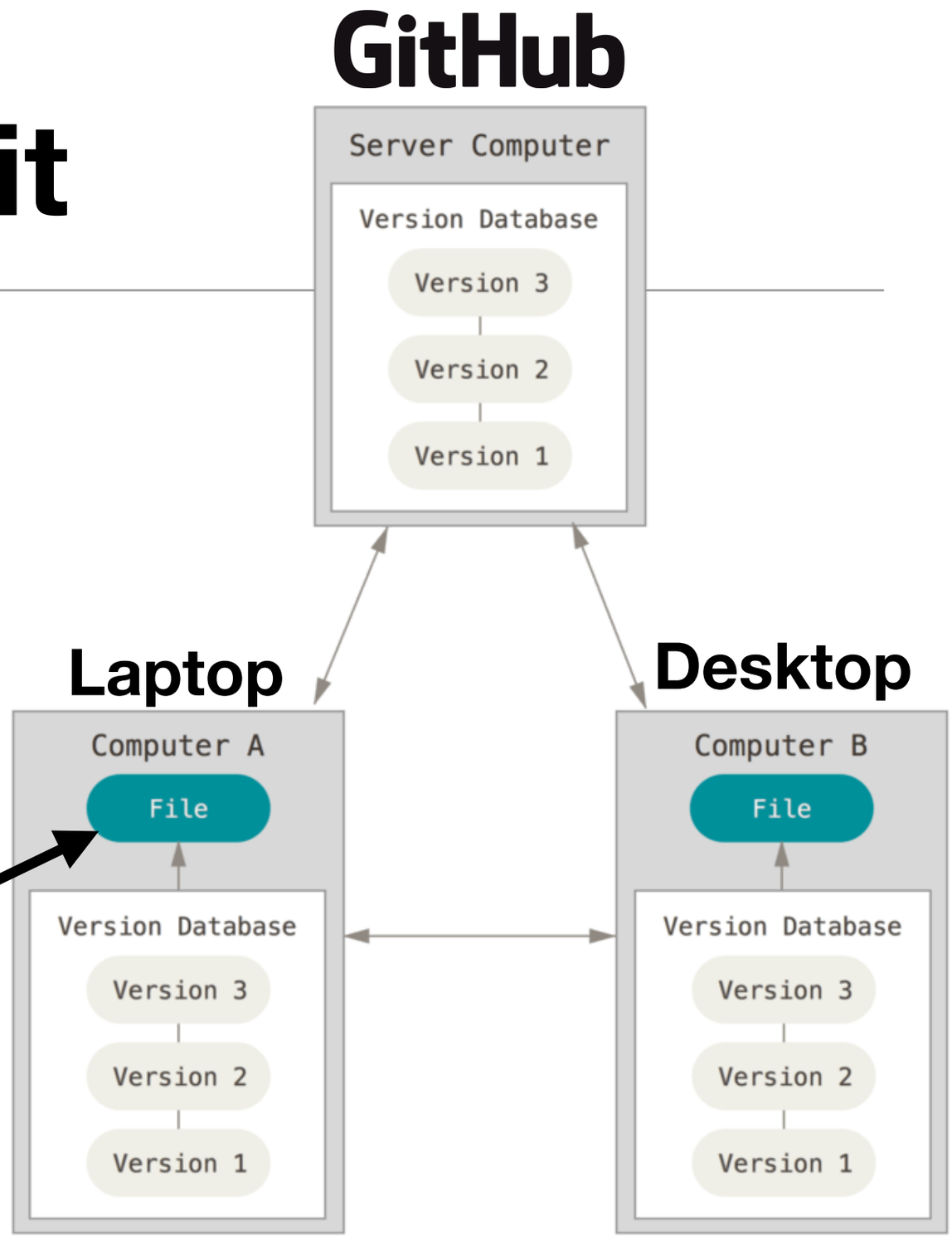
- GitHub open source collaboration (github.org)
  - Free personal plan for .edu users allows for private repositories
- LROSE radar virtual toolbox
  - Low setup time, just open the box using Docker or on the cloud to get the latest tools (see Heistermann et al. 2015)
- LROSE one part of larger National Strategic Computing Initiative (NSCI)
  - Integration with SI2 Big Weather Web for data assimilation
  - Cloud computing (e.g. full NEXRAD database on Amazon S3)

Brief Primer on Git

```
$ ls
my_awesome_script.jl
my_awesome_script_20170110.jl
my_awesome_script_bad_version.jl
my_awesome_script_new_version.jl
my_awesome_script_v1.jl
my_awesome_script_v2.1.jl
my_awesome_script_v2.jl
```

my_awesome_script.jl

Files are stored in a database that keeps track of your changes and is distributed across multiple devices and the cloud



mmbell / lrose-core
forked from NCAR/lrose-core

Watch 0 | Star 0 | Fork 3

Code | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs | Settings

Core C/C++ code for LROSE. https://www.eol.ucar.edu/content/lida... Edit

587 commits | 1 branch | 3 releases | 2 contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

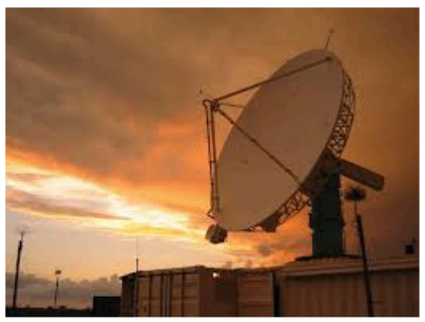
This branch is even with NCAR:master. Pull request Compare

mike-dixon apps/Radx/RadxPartRain: working on melting layer Latest commit 05378fc a day ago

build	exporting LROSE_INSTALL_DIR	2 months ago
codebase	apps/Radx/RadxPartRain: working on melting layer	a day ago
docs	Editing make docs	12 days ago
release_notes	adding-from-cvs	5 months ago
.gitignore	Merge branch 'master' of https://github.com/NCAR/lrose-core	5 days ago
LICENSE.txt	adding	5 months ago
README.md	Updating README	5 months ago

README.md

lrose-core



L🌹 - The Lidar Radar Open Software Environment

LROSE is a co-operative project between:

- Dept. of Atmospheric Science at Colorado State University (CSU) and the
- The Earth Observing Lab at the National Center for Atmospheric Research (NCAR).

LROSE is funded by the National Science Foundation.

‘Fork’ a repo to contribute to LROSE development

‘Clone’ a repo to use it yourself locally

View your code and revision history

Use a pretty README file to document your project or algorithm

Infrastructure

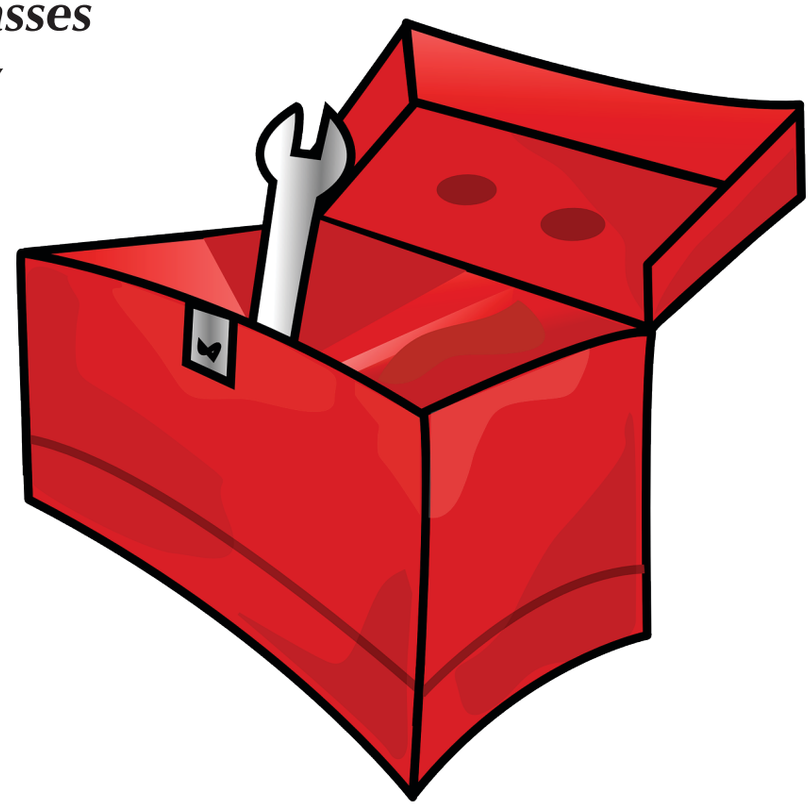
NetCDF data support layer
Data servers for display applications
Higher-level language bindings for C++ library classes
Higher-level language native data handling library
Portability layer
Open source software distribution mechanism

Displays

ASCOPE for spectral radar I/Q time series
BSCAN for vertically pointing data
Low-level viewer and editor for polar data
Platform-independent viewer for data integration
Display and editor tool for profiler data

Core Suite Algorithms

Well-tested, published algorithms
Common tasks and widely used tools



Community Algorithms

Specialized software toolsets
Configurable modules for different instruments and science goals
Partner-maintained packages (DOE PyArt, BALTRAD, and others)

Time period	Proposed tasks		
	Infrastructure	**Displays**	**Algorithms**
Year 1	Develop and test initial virtual machine image. Set up software distribution mechanism (GitHub). Enhance portability and compilation on target platforms.	Develop (or enhance) the most widely required displays (e.g. those that integrate radar/lidar data with other sources)	Develop/enhance tier 1 algorithms – i.e. those most in demand (refer to Tables 1 and 4)
Year 2	Develop and test advanced virtual images. Begin development of bindings for higher level languages.	Enhance existing displays. Begin development on BSCAN and profiler editing display tools.	Develop/enhance tier 2 algorithms – i.e. those next most in demand
Year 3	Enhance virtual images. Begin development of native APIs for higher level languages.	Enhance existing displays. Begin ASCOPE spectral display development.	Develop/enhance tier 3 algorithms – i.e. those next most in demand
Year 4	Enhance virtual images.	Enhance existing displays.	Develop/enhance tier 4 algorithms – i.e. those next most in demand

Workshop Goals

1. Discuss current state of LROSE software and partner tools, and develop future plans for maximum synergy and community benefit.

2. Discuss software, science, and technical priorities, building on the 2012 NSF Radar workshop. Begin to define reproducible workflows for different users, platforms, and disciplines.

3. Refine the mechanisms by which community input and code are included in the project.

4. Solicit and discuss new transformative ideas that would be enabled by improved lidar and radar software tools.

nsf-lrose.github.io

Logistics

- ReadyTalk enabled for all talks and breakouts. Please talk into the microphone or telephone, and repeat questions.

- 3 breakout groups in each of three sessions. Need volunteers for moderation and rapporteurs. Keep the discussion lively and open for all (especially students).

- Plenary discussion to summarize and assimilate breakouts, with aim towards some group consensus.

- White paper from workshop discussion and recommendations will be distributed to wider community for feedback.

nsf-lrose.github.io

Breakout #1 “Building common infrastructure”

- 1) “Working Together”: LROSE, PyART, BALTRAD, and more

- “Defining Workflows”: Common tasks for common goals and reproducibility

- 2) Quality Control and Data Assimilation

- 3) Polarimetric Radar Applications

nsf-lrose.github.io

Breakout #2 “Science and technical needs”

- 1) “Tech talk”: LROSE internals and design principles
- “Science Priorities”: Meeting the needs of scientists across disciplines
- 2) Airborne Doppler Radar
- 3) Ground-based Radar

nsf-lrose.github.io

Breakout #3 “Enabling Science”

- “Building Community”
  - 1) Mechanisms for including externally developed or maintained code, including testing and verification
  - 2) Documentation, community wiki, and support
- 3) “Big Ideas”: Leveraging LROSE for high-risk high-reward science

Lidar Radar Open Software Environment
LROSE

Existing Software Status

First User's Group Workshop
NCAR, Boulder, Colorado
2017/04/11 – 2017/04/12

Mike Dixon
Earth Observing Laboratory (EOL)
National Center for Atmospheric Research (NCAR)
Boulder, Colorado

LROSE

LROSE is an NSF-funded project to enhance the software tools available to the community of users of radars, profilers and lidars.

- Funded via NSF SI2 software infrastructure program.
- Open source, community data formats, permissive free licensing.
- Portable code, high-level languages where appropriate.
- Virtualization – supports containers and virtual hosts, as well as running natively.
- Makes maximum use of existing software, refactoring and modifying as appropriate.
- Provides common applications for research, teaching and real-time operations.

LROSE

LROSE CORE development
CSU
NCAR

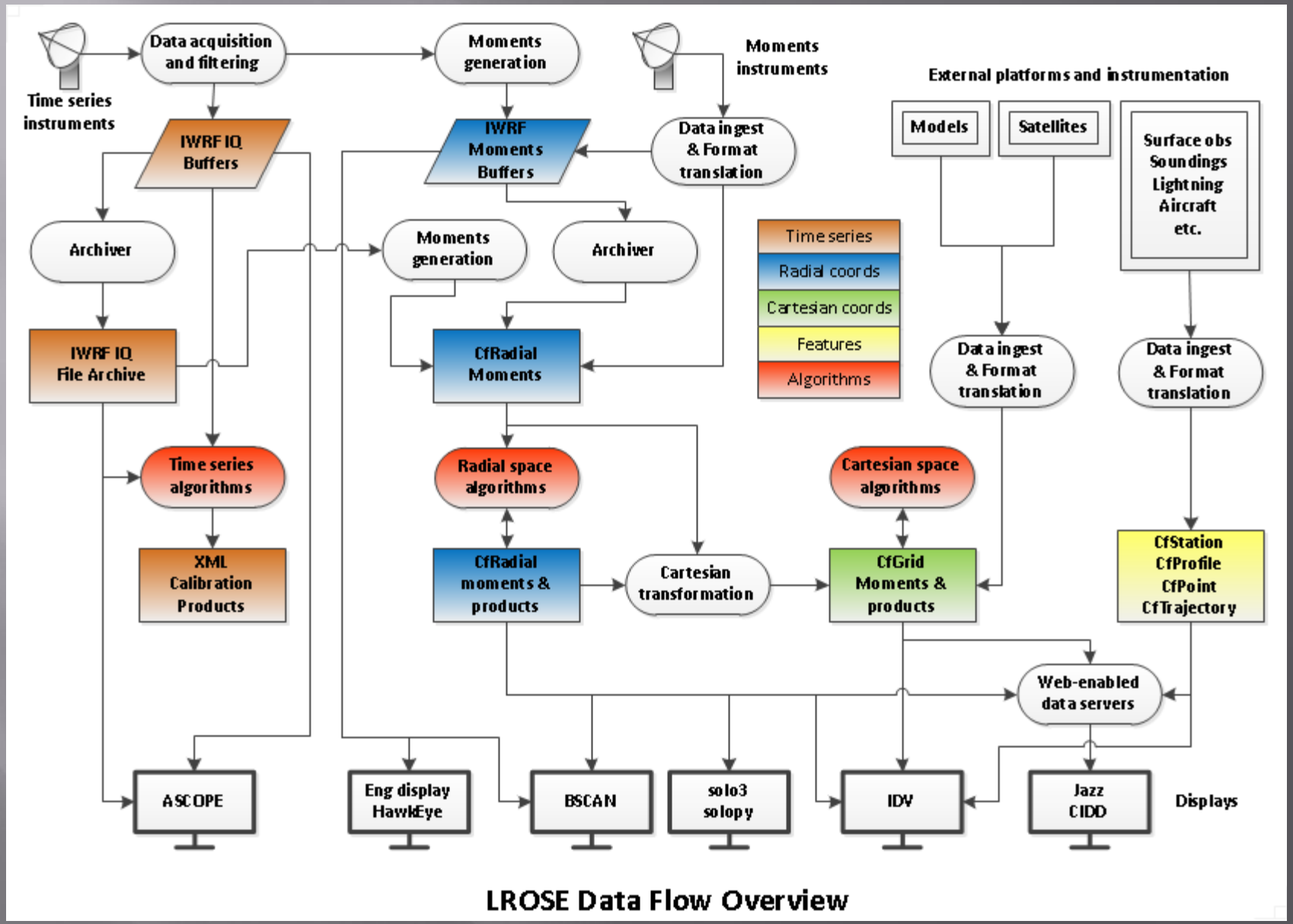
Collaboration

Universities
Research Centers (e.g. CSWR)
PyArt toolbox (DOE/ARM)
BALTRAD toolbox (Sweden)
wradlib toolbox (Germany)
BOM (Australia)
NCAS (UK)
Environment Canada
NCMS (UAE)
WMO data exchange
Commercial sector
etc.

NSF's prime interest aligns with the university and research community.

The LROSE team recognizes that collaboration with other institutions will enhance our capabilities and reduce redundancy of effort.

PyArt, wradlib and BALTRAD are mature, but still under active development.



LROSE Data Flow Overview

Application Development Status

The status of the LROSE applications varies, with some being mature, some actively under development, and some that do not yet exist.

Mature applications will be shown in blue: **RadxQpe**

Applications under active development will be shown in red: **AScope**

Applications which do not yet exist will be shown in gray: **TitanPerfectForecast**

Calibration, Digital receiver, Time series generation Clutter filtering, Moments computation

Solar scans and cross-polar power monitoring for Z and ZDR calibration

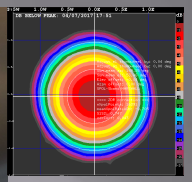
RadxSunMon

SunCal

AltCpCompute

ZdrCalSimHv

RadxClutMon



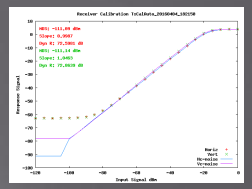
Vertical scanning for ZDR calibration

VertCompute

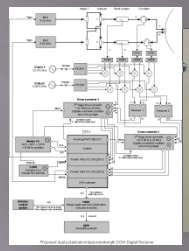
Engineering ATE calibration

RadarConst

TsCalAuto



Digital receiver, pulse analysis
Time series generation



dowdrx

spoldrx

hcrdrx

TsSmartSave

Time series archive

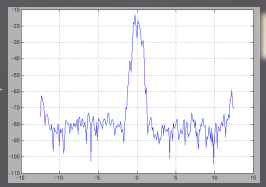
CMD

Iq2Dsr

Dsr2Radx

Moments archive

Clutter filtering
Moments generation



AScope

Time series analysis

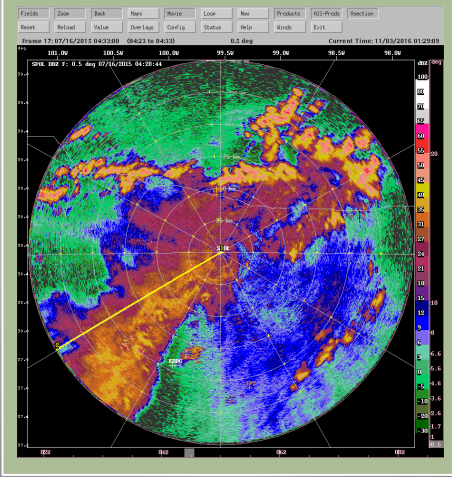
Polar space format conversion

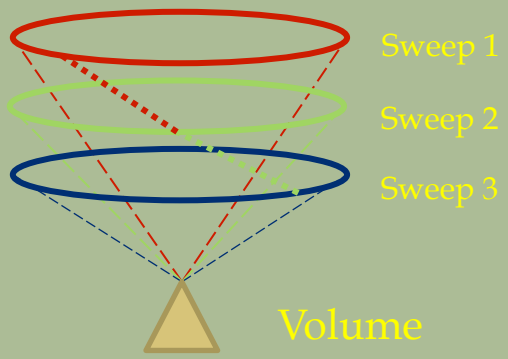
RadxConvert

Type	Notes	Type	Notes
D3R	Read-only	NEXRAD Level 2	Read-write
DOE	Read-only	NEXRAD Level 1,3	Read-only
DORADE	Read-write	NOXP	Read-only
EEC-Edge	Read-only (work in progress)	NSSL-MRD	Read-only
FORAY	Read-write	ODIM-H5	Read-write (work in progress)
Gamic	Read-only	RAPIC	Read-only
Gematronik Rainbow	Read-only (writeable with python script)	SIGMET Raw (Vaisala)	Read-only
HSRL (LIDAR)	Read-only	TDWR	Read-only
HRD (Hurricane Research Division)	Read-only	TWOLF	Read-only
Leosphere (LIDAR)	Read-only	UF	Read-write
CfRadial-1	Read-write	CfRadial-2 (WMO)	Read-write (work in progress)

RADAR and LIDAR volumes are made up of gates, rays and sweeps

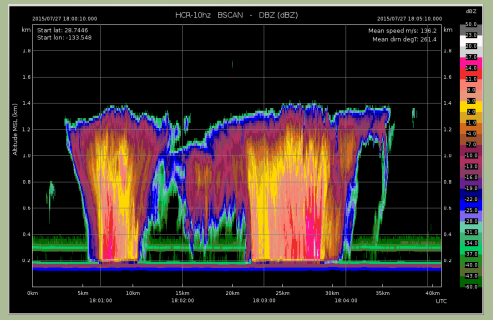
PPI mode – surveillance / sector – constant elevation sweeps

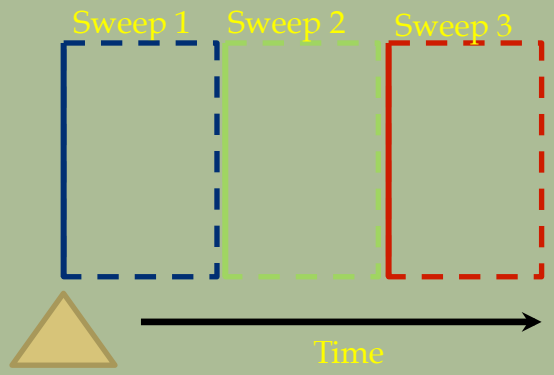




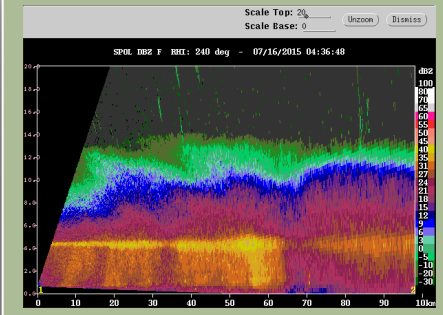
A collection of GATES forms a RAY.
A collection of RAYS forms a SWEEP.
A collection of SWEEPS forms a VOLUME.

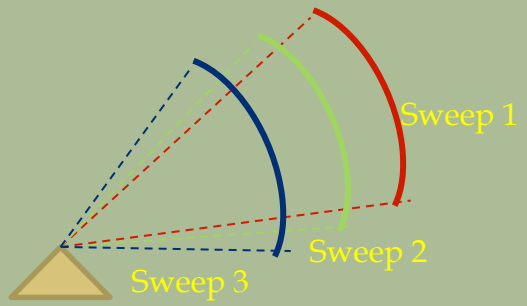
Vertical pointing mode
Sweeps are time-delimited





RHI mode – constant azimuth sweeps



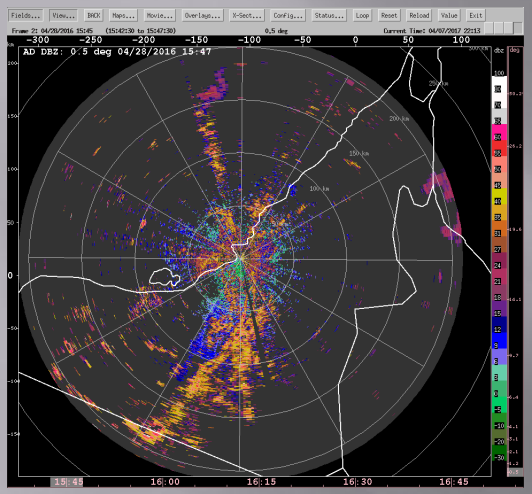


Data manipulation and filtering

Application Name	Description
Ts2NetCDF	Convert NEXRAD level 1 and IWRF time series to NetCDF
TsPrint	Print NEXRAD level 1 and IWRF time series
RadxPrint	Print contents of Radx-supported file
RadxConvert	Convert file formats, data representation, compression. Select on elevation angles, sweep numbers. Override selected metadata.
RadxCheck	Check for validity of Radx (CfRadial) file
RadxMergeVols	Merge Radx volumes into a single file
RadxMergeFields	Merge selected fields from different Radx files into a single file
RadxDwellCombine	Combine dwells from Radx volumes into longer dwells (for example for vertical pointing radars)
RadxDiffFields	Checks for differences between fields in Radx files
RadxTimeMedian	Compute time-base median values for selected fields in RadxFile
RadxPersistentClutter	Determine the points for which clutter is persistent over a large number of volumes – helps to identify residual clutter locations
RadxFilter	Compute filtered fields in range (e.g.median filter over n gates)
RadxCov2Mom	Compute moments from co-variances in Radx files
RadxMon	Monitor incoming Radx data in real-time data stream
FixRadxValues	Make selected modifications to data values in Radx file
FixCfradialPaths	Convert file names to conform to CfRadial directory structure and naming convention

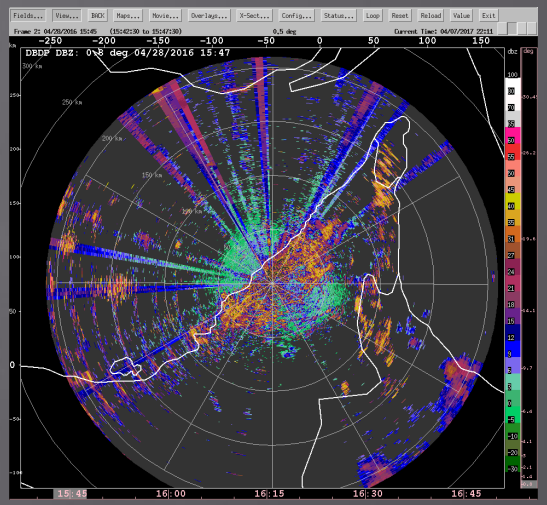
Data Quality Tools

RadxQc

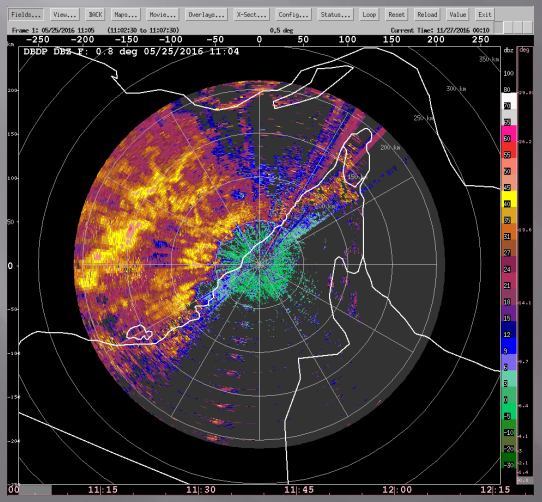


Anomalous propagation

Tools are under development for mitigation of common data quality problems

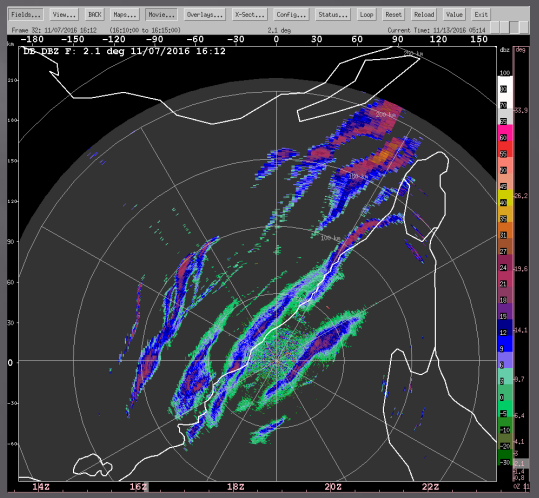


Radio-LAN interference at C-band



Sea clutter

Work is ongoing since the quality issues vary by location and radar type



Military chaff

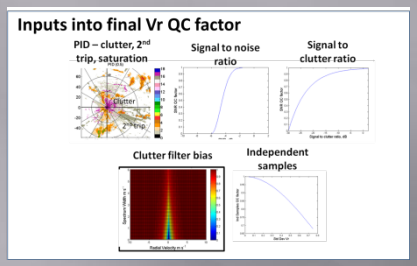
Examples are from the UAE NCMS Abu Dhabi and Dubai radars

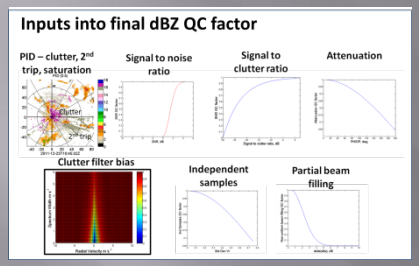
Real-time data QC For Model Data Assimilation in collaboration with USWRP-funded NCAR STEP program

RadxModelQc

Thanks to Scott Ellis for this material

For now consider Z and Vr only
Compute QC Factors between 0 and 1





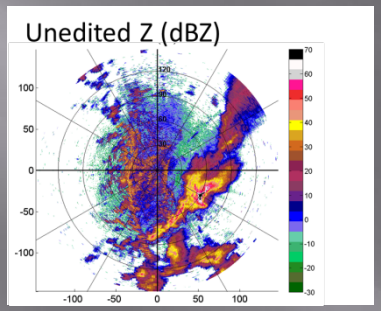
QC factors designed for numerous error sources

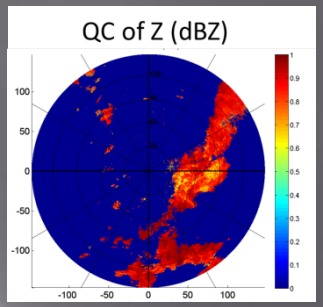
- PID (clutter, 2nd trip, saturation); SNR; SCR; clutter filter bias; independent samples; attenuation; partial beam filling
- Vary from 0 to 1
- Designed using radar theory, studies in the literature

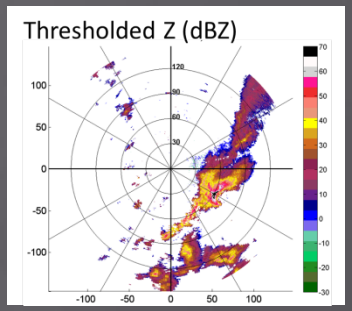
Final Vr and Z QC factors are computed by multiplying all of the error source QC factors together

A threshold of 0.5 is then applied

Create thresholded fields using points with QC factor > 0.5
Example below is for Reflectivity Z

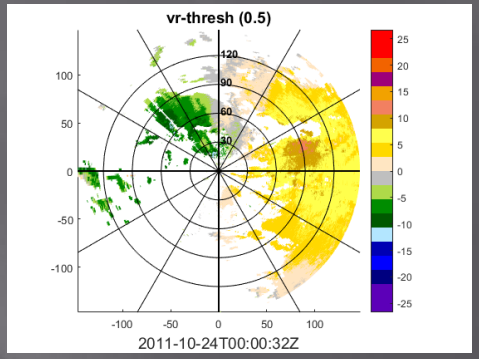


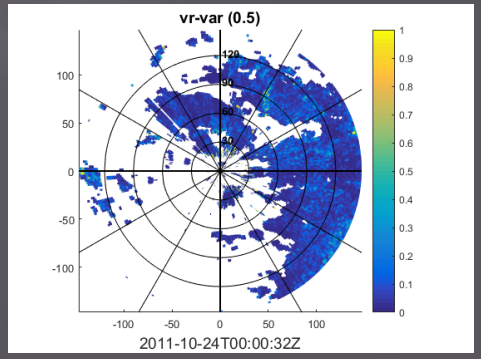




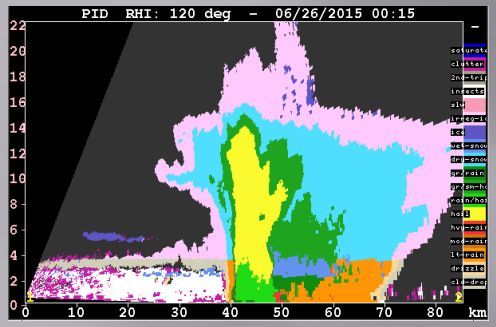
Estimation of variance

- Estimate variance from the data itself: direct measurement, measurement variance and small scale natural variations, takes into account all sources of measurement variance
- Theoretical estimate of variance: radar characteristics (dwell time, wavelength), echo characteristics (SNR, Spectrum Width)



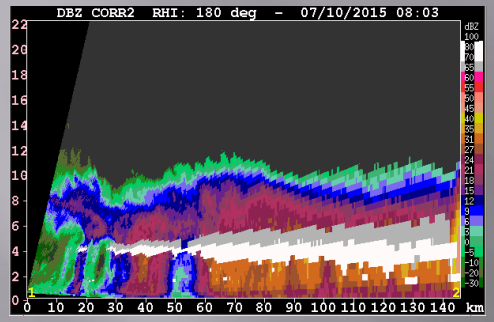


Microphysics, Precipitation Rate and Accumulation

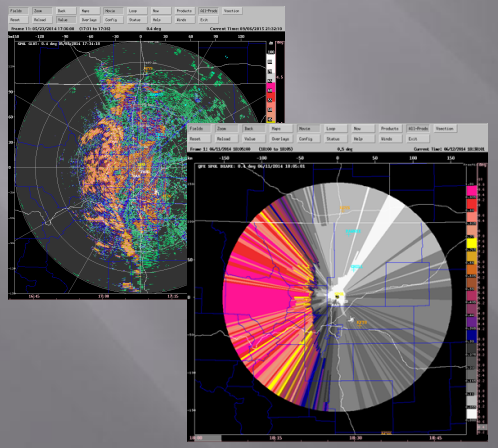


NCAR PID
NCAR Hybrid
Precip Rate

RadxPartRain

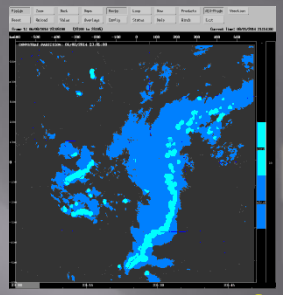


Melting layer
identification



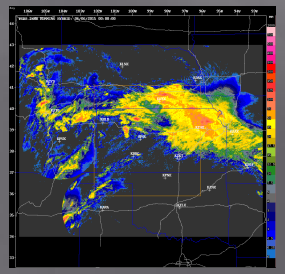
Beam blockage
identification

RadxBeamBlock



ConvStrat
stratiform_filter

Convective-stratiform
partitioning

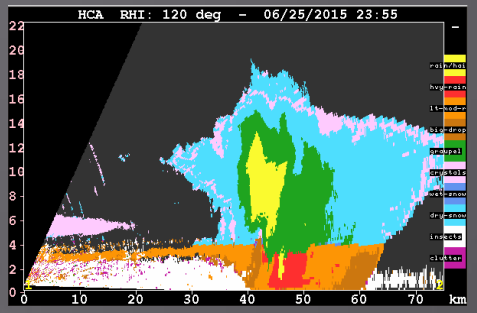


Precip
accumulation

PrecipAccum

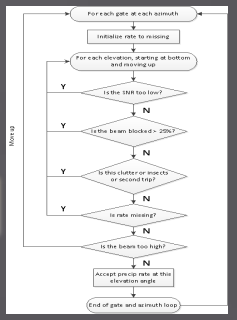
NEXRAD HCA

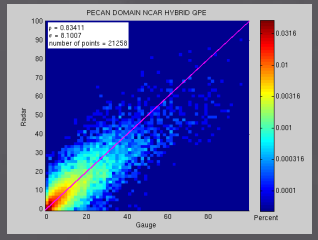
RadxHca



Quantative precip
estimation

RadxQpe





Precip
verification

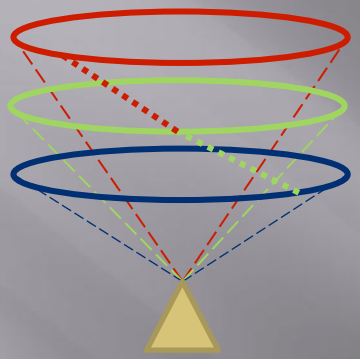
QpeVerify

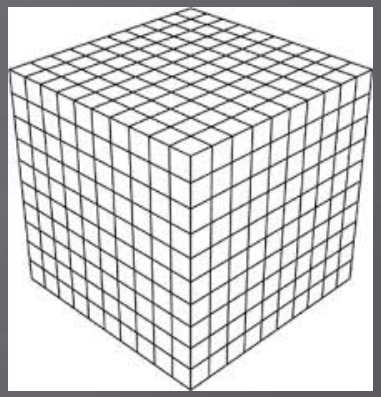
Tools for hydrometeor type, precip rate estimation and QPE

Coordinate transformation

Radx2Grid

Radx2Cart





Radial/polar coordinates

Rectangular/Cartesian coordinates

Radx2Grid can produce 3 types of regular grid:

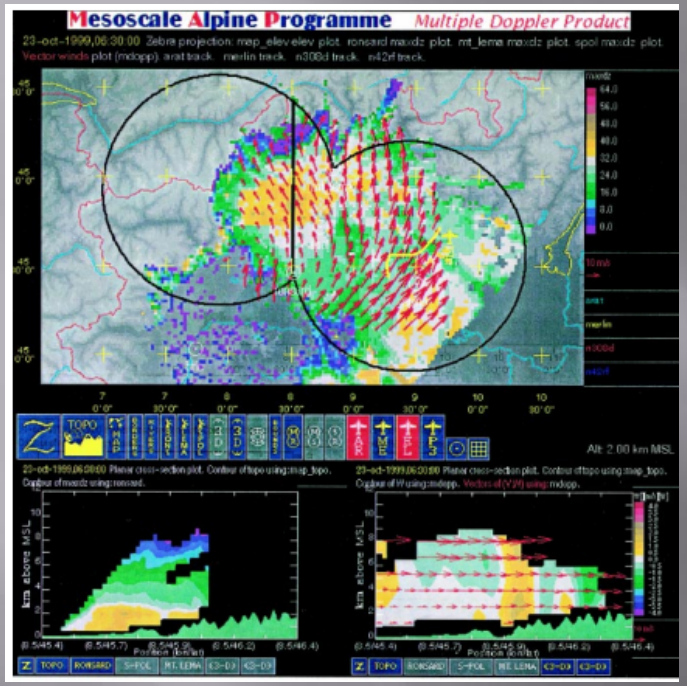
- 3-D Cartesian grid (z, y, x)
- Cartesian PPIs (elev, y, x)
- Regular polar grid (elev, az, range)

Some users find Radx2Grid daunting to use because of the many options in the parameter file.
Perhaps we need to add a simpler version (Radx2Cart) which only performs the most common transformation.

sprint Legacy app that interpolates from the data values immediately surrounding the output grid point.

reorder Legacy app that uses a weighted interpolation of all grid points within a certain distance of the output grid point.

Doppler wind retrieval



Example of multi-Doppler wind synthesis
(Chong et al., BAMS, Vol. 81, No. 12, December 2000)

cedric Legacy multi-Doppler wind retrieval app in FORTRAN, widely used in the research community, often in conjunction with sprint or reorder.

RadarWind Multi-Doppler wind retrieval app developed by Bell, Lee and others for the analysis of Doppler airborne tail radars e.g. eldora and NOAA tail radars

SAMURAI 3-D variational multi-Doppler wind retrieval app , Bell et al, 2012

VDRAS 4-D variational multi-Doppler wind retrieval app , Crook and Sun, 2002.

Doppler wind retrieval, and velocity dealiasing, seem to be ideal candidates for LROSE-community collaboration to develop new easy-to-use methods.

RadxEvad Single Doppler wind retrieval application using the Enhanced VAD method.
Thomas Metejka and Ramesh C. Srivastava, 1991: 'An Improved Version of the Extended Velocity-Azimuth Display Analysis of Single-Doppler Radar Data' , Journal of Atmospheric and Oceanic Technology, Vol 8, No 4, August 1991

GVTD Single Doppler hurricane/tornado wind retrieval
Lee et al. 1994, 1999; Jou et al. 2008

JamesDealias 4D velocity dealiasing
James and Houze, JTech, 2001

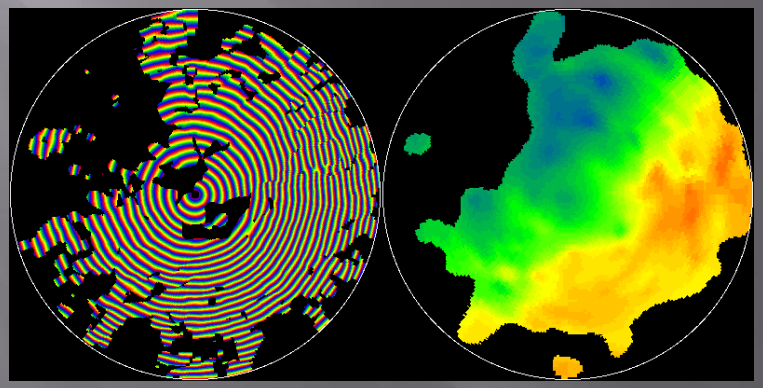
PyArtDealias Velocity dealiasing based on PyArt algorithm

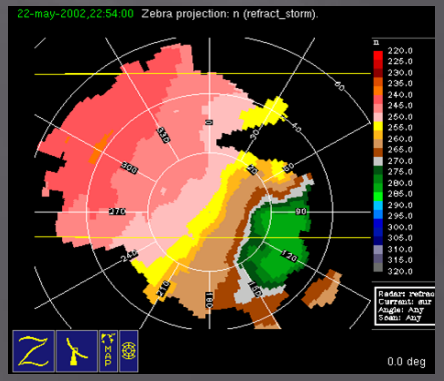
Nowcasting and Storm Analysis

App Name	Description	Reference
Titan	Storm tracking – convective.	TITAN (Dixon and Wiener, 1993)
ctrec	Echo tracking – stratiform. Produces U/V vectors.	CTREC (Tuttle and Foote, 1990)
OpticalFlow	Echo tracking for extrapolation. Produces U/V vectors.	Bab-Hadiashar, A, D. Suter, and R. Jarvis, 1996, 2-D Motion extraction using an image interpolation technique. SPIE Vol. 2564, 271-281.
ScaleSep	Scale separation of precipitation elements for nowcasting purposes.	Alan W. Seed, Clive E. Pierce, and Katie Norman, 2013. Formulation and evaluation of a scale decomposition-based stochastic precipitation nowcast scheme. WATER RESOURCES RESEARCH, VOL. 49, 6624–6641, doi: 10.1002/wrcr.20536, 2013
TitanVectors2Mdv	Produces a field of U/V vectors from Titan storm tracks.	
VerifyTracks	Verifies Titan track forecasts.	TITAN (Dixon and Wiener, 1993)
StormInitLocation	Location of Titan storm initiation events	
StormInitClimatology	Climatology of Titan storm initiation events	
Tracks2Ascii	Titan storm properties in ASCII table	
Tstorms2Xml	Titan storm properties in XML	
TrackGridStats	Titan storm properties over a grid	

Refractivity

Application Name	Refractivity apps	Reference
PhaseDiff	Compares radar volumes to measure and output phase differences, which are used to determine suitable time ranges to use in RefractCalib	Fabry, F., 2004: Meteorological value of ground target measurements by radar. J. Atmos. Oceanic Technol., 21, 560–573
RefractCalib	Reads radar I&Q data over an interval of time and uses that to identify targets, and to produce a static background calibration field.	
Refract	Reads radar data and uses the calibration field from RefractCalib and from the difference between the data and the calibration, estimates refractivity.	
CalcMoisture	Calculates water vapor pressure and dew point temperature based on the refractivity and the mean temperature and pressure values from a group of weather stations.	





Other algorithms

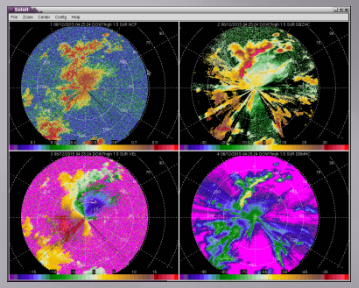
Purpose	Reference	Prototype?
Quality metrics / error assessment	Bell et al. 2013	Yes
Attenuation correction in precipitation	Gu et al. 2011	Yes
Compositing data from multiple instruments	Henja and Michelson, 2012	Yes
Vertical profile of reflectivity – VPR	Kirstetter et al. 2013	No
Vertically integrated liquid – VIL	Greene and Clarke, 1972	Yes
Thermodynamic retrieval	Roux et al. 1993	Yes
Wind shear detection	Albo and Kessinger 1996	No
Meso-cyclone detection	Stumpf et al. 1996	No
Wind profiler moments estimation	NIMA (Cornman et al. 1998)	No
Wind profiler clutter rejection	ICRA (Merritt 1995)	No

Reading other data types

LROSE applications are available for reading non-radar data types, including:

- model output
- satellite
- surface observations, soundings
- lightning
- aircraft tracks from experiments

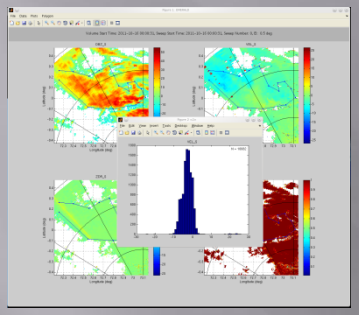
Displays



soloii

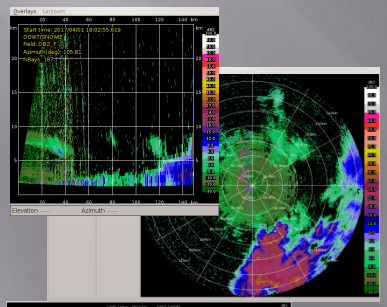
solo3

Legacy C/C++
polar display
in wide use

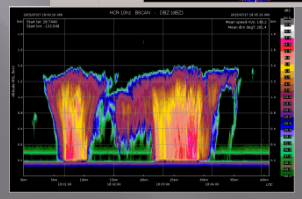


Emerald

MatLab based CfRadial
viewer and analysis tool



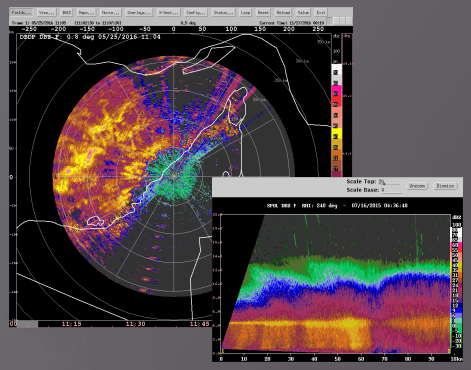
HawkEye



Engineering and real-time
display, used on SPOL,
HCR and HSRL
Suitable for vertical-pointing
and staring radars and lidars **

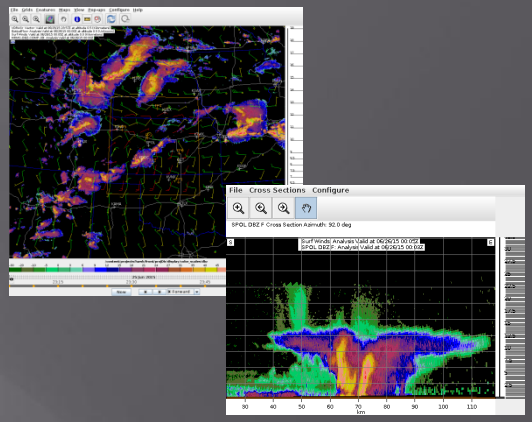
CIDD

Legacy C++
integrating display
in wide use **



Jazz

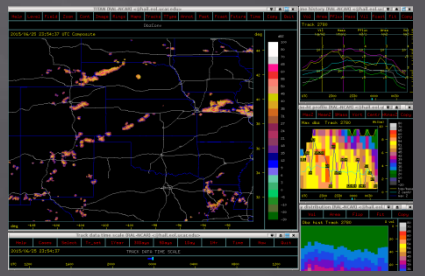
Java-based
integrating display
under active
development **



Rview

TimeHist

Legacy Titan display
optimized for storm
track analysis



** - capable of background image generation for web catalog

LROSE GitHub repositories

Description	GitHub URL
Main LROSE GitHub site	https://github.com/nsf-lrose
LROSE core	https://github.com/NCAR/lrose-core
LROSE NetCDF support libs	https://github.com/NCAR/lrose-netcdf
Color scales and maps for LROSE displays	https://github.com/NCAR/lrose-displays
Matlab display for CfRadial data	https://github.com/NCAR/lrose-emerald
Java-based Jazz display	https://github.com/NCAR/lrose-jazz
Code for legacy soloii application	https://github.com/NCAR/lrose-soloii
Code for legacy solo3 application	https://github.com/NCAR/lrose-solo3

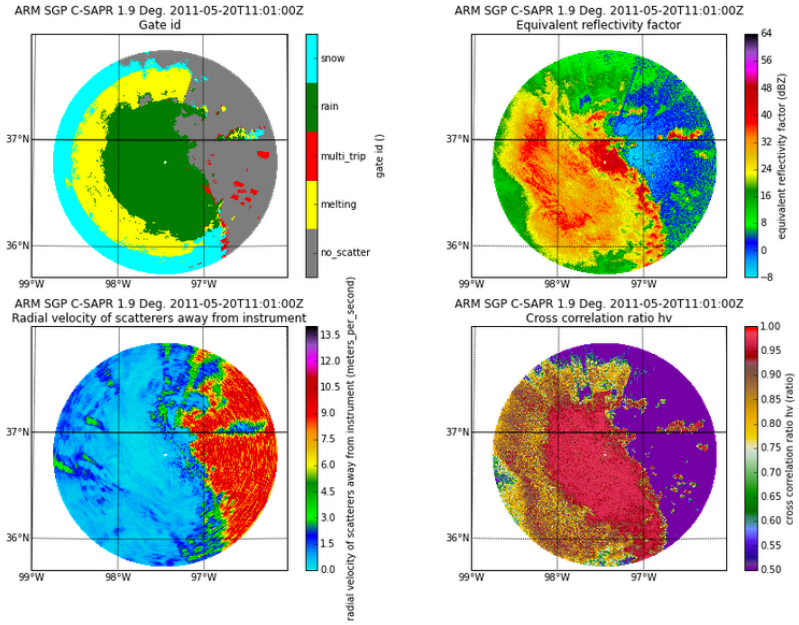
These are the current repositories for LROSE.
The location of these may change as the project matures.

THANK YOU

OPEN THE SOURCE.. FREE THE MIND…

Argonne NATIONAL LABORATORY

THE PYTHON ARM RADAR TOOLKIT: A COMMUNITY BASED ARCHITECTURE FOR INTERACTING WITH WEATHER RADAR DATA



SCOTT COLLIS, JONATHAN HELMUS*, CORY WEBER, ZACHARY SHERMAN AND MARK PICEL.
Argonne National Laboratory
*Now at Continuum Analytics

KIRK NORTH, ANDERSON GAMMA, NICK GUY, JOSEPH HARDIN, TIMOTHY LANG, JORDI FIGUERAS VENTURA, KAI MÜEHLBAUER, JULIA SIGNELL, STEVE NESBITT, ERIC BRUNING, MARTIN RUGNA, TULIPA SILVIA, BEN ROOT, CODY PIERSALL AND JOE VANANDEL

April11th
LROSE Kick off workshop, NCAR, Boulder.

HISTORY AS PRELUDE

Py-ART was born of frustration…

- In 2007 I started my postdoctoral appointment at BoM working on Tropical storm dynamics.
- Two key frustrations led to OSRA (Open Source Radar Applications, the precursor to Py-ART) : Stupid number of data formats and the cumbersome API (or lack thereof) of existing objective analysis software (Gridding).

PY-ART: INSPIRED BY THE SCIENTIFIC PYTHON COMMUNITY.

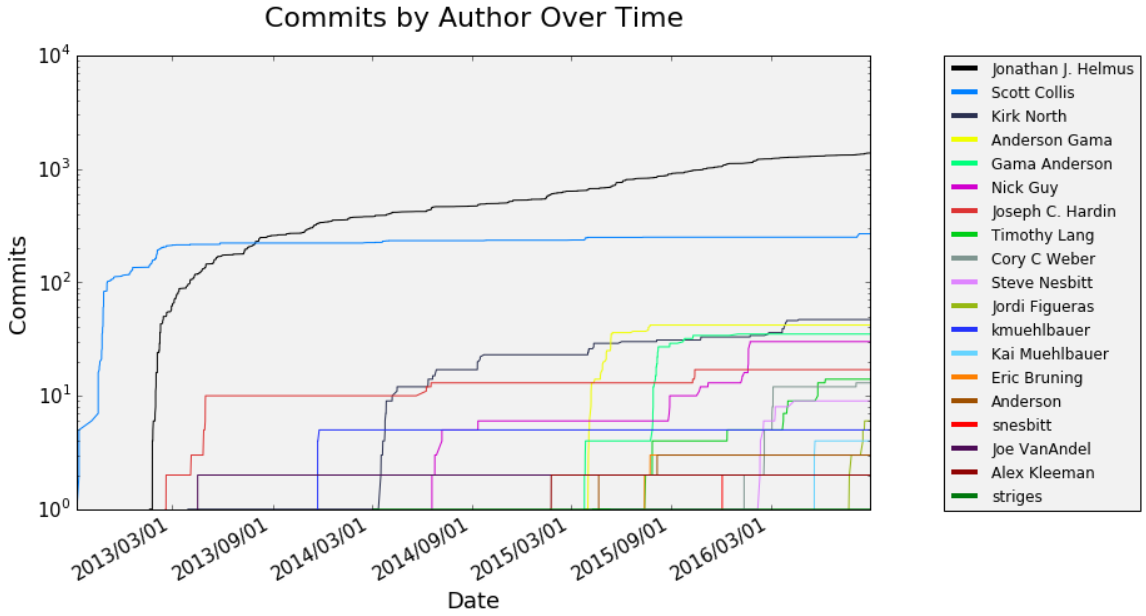
We've got nothing on cat videos…

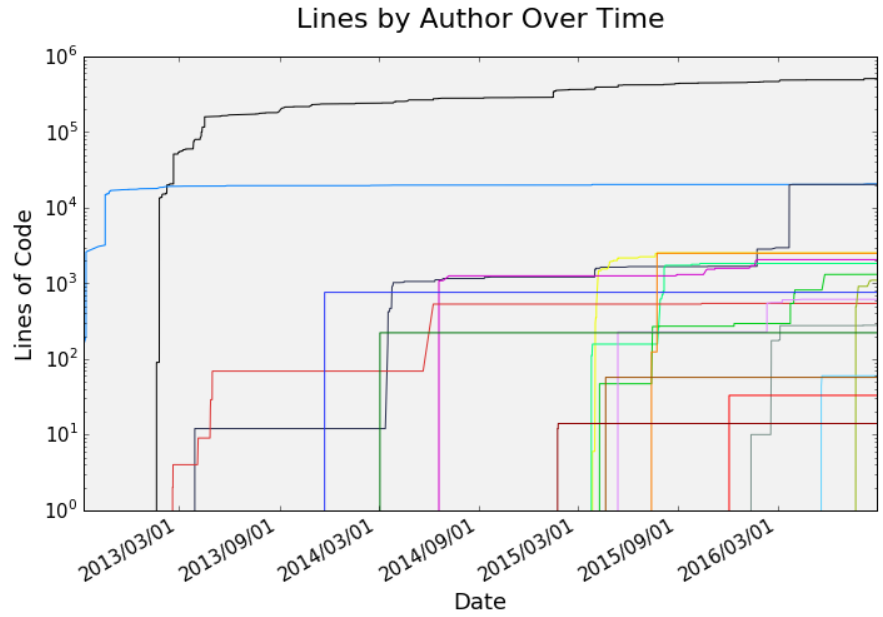
- There is a very large community of developers who work across funding lines on "Core scientific functionality" within the Python ecosystem.
- Key to this is community codebases which are bigger than one person or funding stream.
- As we were looking at licensing and ideas for Py-ART it became clear that using a permissive license and getting the code into GitHub and well tested was essential to building a codebase that lasts the test of time.

SO WHAT EXACTLY IS PY-ART?

Primarily it is a way of expressing gated data in the Python programming language.. Practically it is more.

- Py-ART is a data-model driven open architecture for working with radar data in the Python programming language.
- It is a community codebase which uses GitHub to allow users to interact with the code including submitting “Pull Requests”.
- Funded by ARM covering around 0.3FTE.
- Designed mainly for analysis and visualization and not operational software, although it could be and is used in operations by some groups.
- Py-ART has can read a large number of formats and the data model is based on the CF-Radial standard.
- 100’s of users, 1000’s of downloads, 15 contributors, 3 funded by ARM.





PY-ART IS THE BRIDGE BETWEEN RADAR AND THE SCIPY STACK

100's of millions of dollars get spent on building and optimizing core mathematical code. We need to leverage this.

- Companies like Continuum Analytics and Enthought are large companies who spend 10's of Millions of dollars each giving away software. Why?
- They know that $$ invested in the Scipy stack pays dividends. The same code we use in Py-ART is used to do short term market prediction.
- Cython has become the key tool for building high performance code by making a bridge between C and Python.
- Many promising early projects like Numba and Dask.
- Py-ART works great with IPyParallel for mapping problems to clusters.. My record is 1024 cores!

GORDON AND BETTY MOORE FOUNDATION

CONTINUUM ANALYTICS

ENTHOUGHT SCIENTIFIC COMPUTING SOLUTIONS

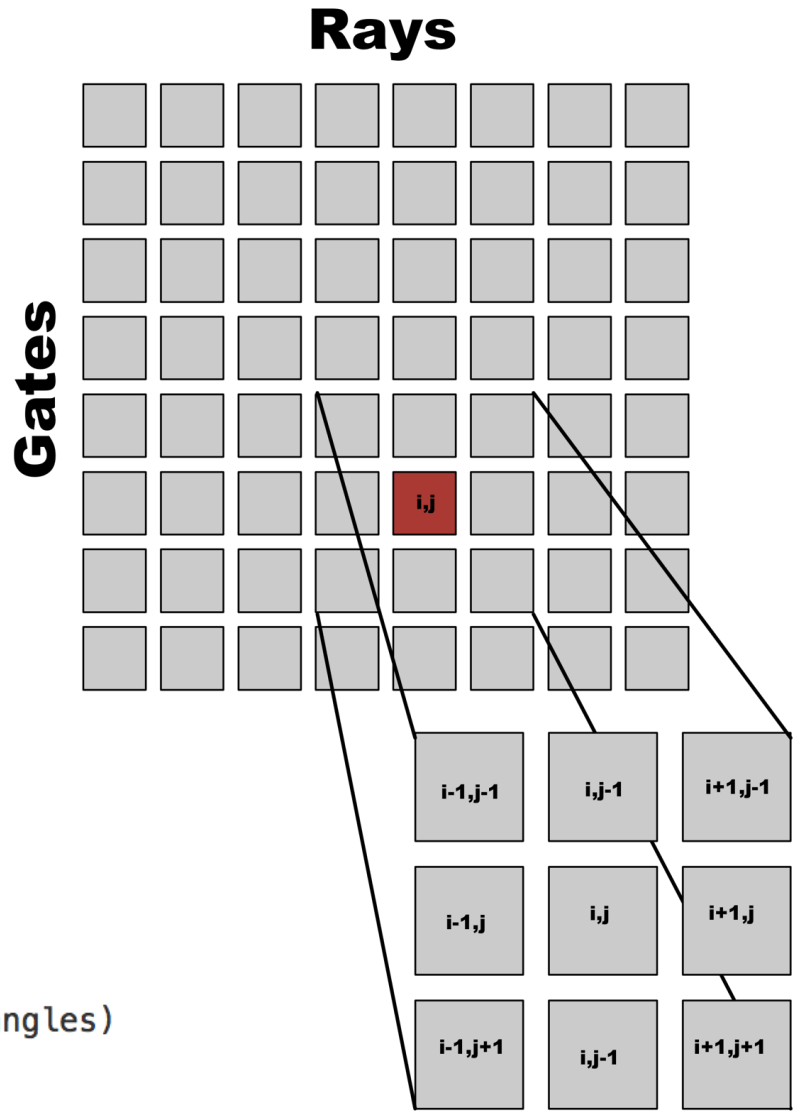
Cython

DEMONSTRATION

FUN EXAMPLES (1)

Texture

- We had early issues with our magnetron radars using Normalized Coherent Power (aka SQI) for filtering multi-trip echoes.
- Another determinant is the texture of radial velocity. As nth trips do not have the same saved phase as the local oscillator in areas that are no first trip returns Vr randomly varies between –nyq,+nyq
- BUT, folds can give false positives if you simply apply ray-wise std.
- Use statistics on the unit circle.. And scipy's general filter convolution.

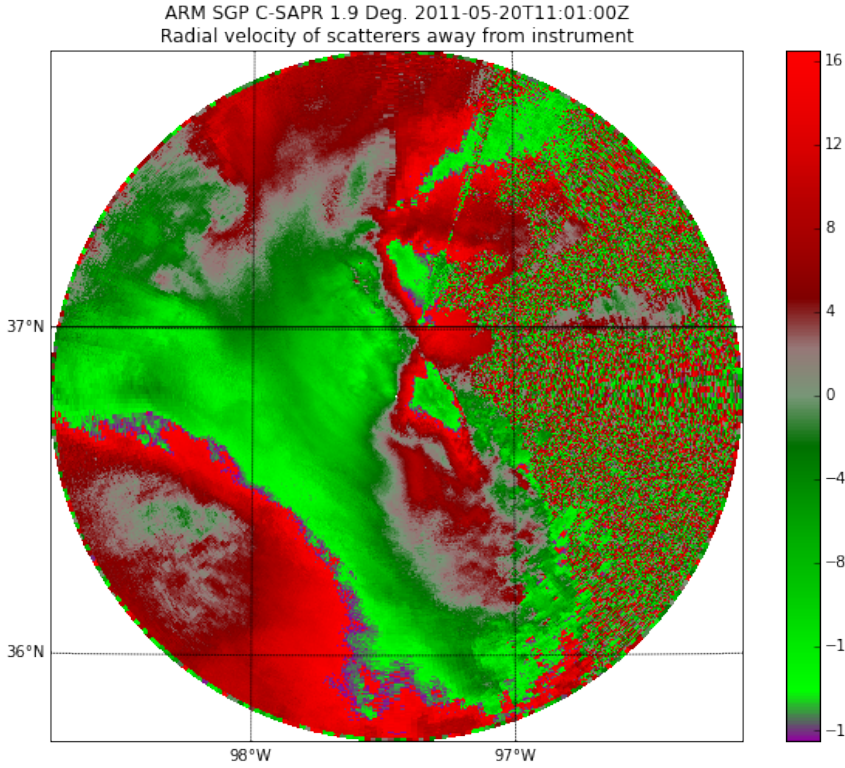



```
angles = np.asanyarray(angles)
x = np.cos(angles)
y = np.sin(angles)
norm = np.sqrt(x.mean()**2 + y.mean()**2)
return np.sqrt(-2 * np.log(norm))
```

FUN EXAMPLES (1)

Texture

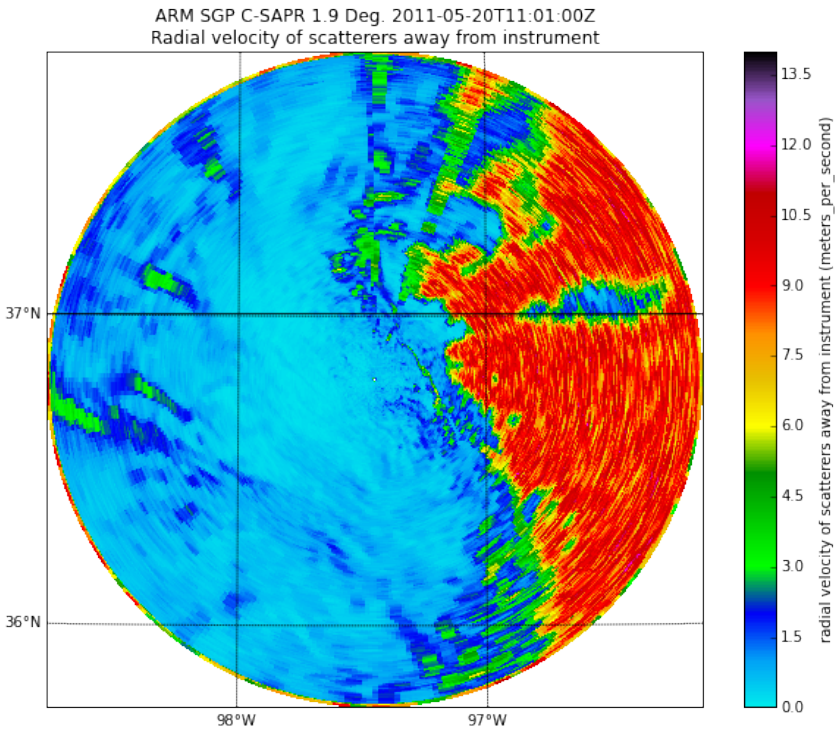
- We had early issues with our magnetron radars using Normalized Coherent Power (aka SQI) for filtering multi-trip echoes.
- Another determinant is the texture of radial velocity. As nth trips do not have the same saved phase as the local oscillator in areas that are no first trip returns Vr randomly varies between –nyq,+nyq
- BUT, folds can give false positives if you simply apply ray-wise std.
- Use statistics on the unit circle.. And scipy's general filter convolution.



FUN EXAMPLES (1)

Texture

- We had early issues with our magnetron radars using Normalized Coherent Power (aka SQI) for filtering multi-trip echoes.
- Another determinant is the texture of radial velocity. As nth trips do not have the same saved phase as the local oscillator in areas that are no first trip returns Vr randomly varies between –nyq,+nyq
- BUT, folds can give false positives if you simply apply ray-wise std.
- Use statistics on the unit circle.. And scipy's general filter convolution.



FUN EXAMPLES (2)

Image shifting for advective interpolation

- For radars than are non-synchronized or we need to determine and correct for advection of radial velocity patterns.
- We have implemented a image shift detection technique to get X/Y advection between volumes using cross correlation (same is in image stabilization).
- We have implemented an image shifter using NDImage
- Todo: Combine forward and backward projected images, “Advective averaging”

$$G(t+\Delta t, z, y, x) = (1 - \frac{t+\Delta t - t_1}{t_2 - t_1}) G_1(t_1, z, y + v\Delta t, x + u\Delta t) + \frac{t+\Delta t - t_1}{t_2 - t_1} G_2(t_2, z, y - v\Delta t, x - u\Delta t)$$

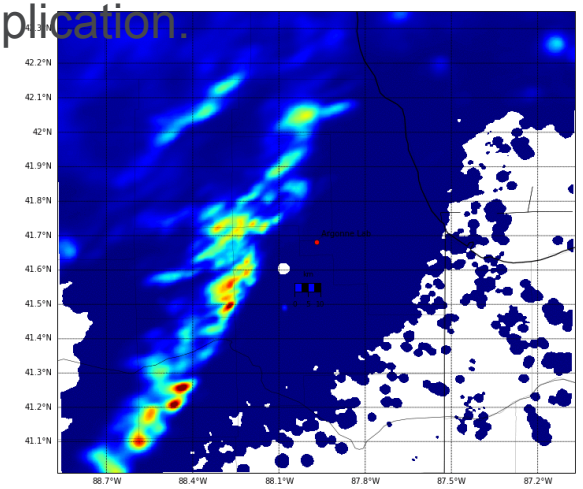
$$\boldsymbol{G}_{t1} = \mathcal{F}\{R_{t1}\}, \boldsymbol{G}_{t2} = \mathcal{F}\{R_{t2}\}$$

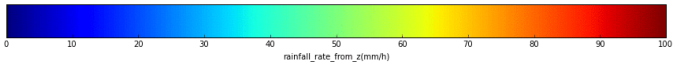
$$C = \frac{\boldsymbol{G}_{t1} \circ \boldsymbol{G}_{t2}^*}{|\boldsymbol{G}_{t1} \circ \boldsymbol{G}_{t2}^*|}$$

$$r = \mathcal{F}^{-1}\{C\}$$

$$\Delta x, \Delta y = \text{argmax}\{r\}$$

where F is the Fourier transform, $*$ is the complex conjugate and $\circ$ represents element
wise multiplication.

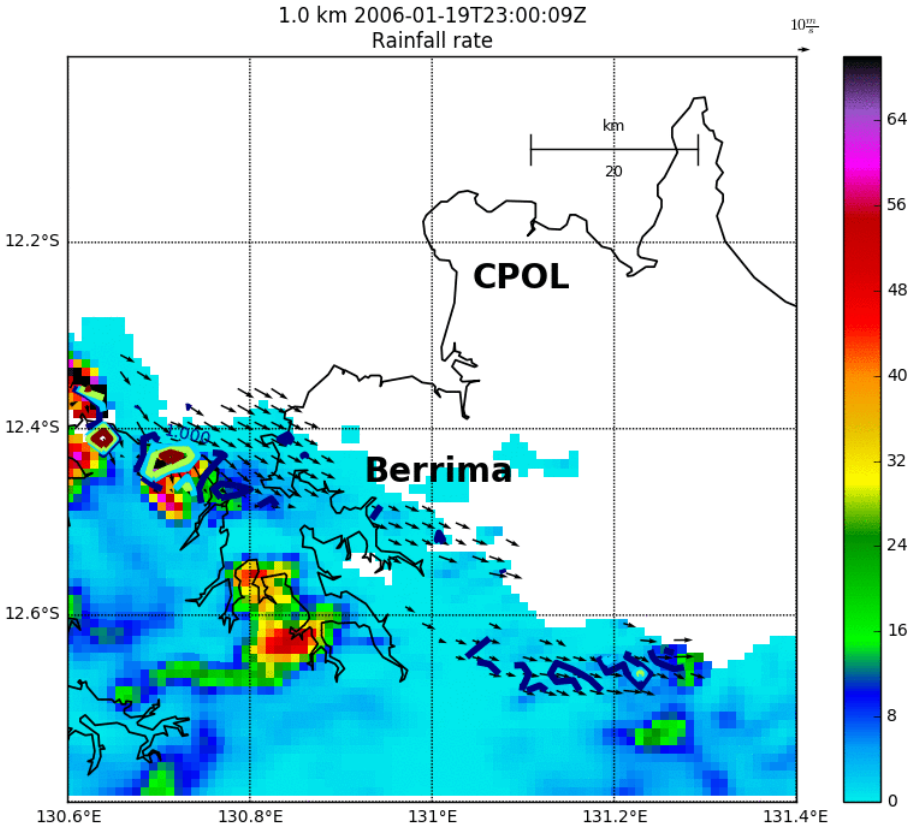




WORKS WITH PY-ART

Py-ART has enabled a rich ecosystem of dependent apps

- ARTView
  https://github.com/nguy/artview
- SingleDop
  https://github.com/nasa/SingleDop
- PyTDA
  https://github.com/nasa/PyTDA
- DualPol
  https://github.com/nasa/DualPol
- CSU Radar Tools
  https://github.com/CSU-Radarmet/CSU_RadarTools
- Multidop
  https://github.com/tjlang/MultiDop



THE PY-ART ROADMAP

Assuring we are good stewards of public funds and meet the needs of our stakeholders

Survey Completed: ✓

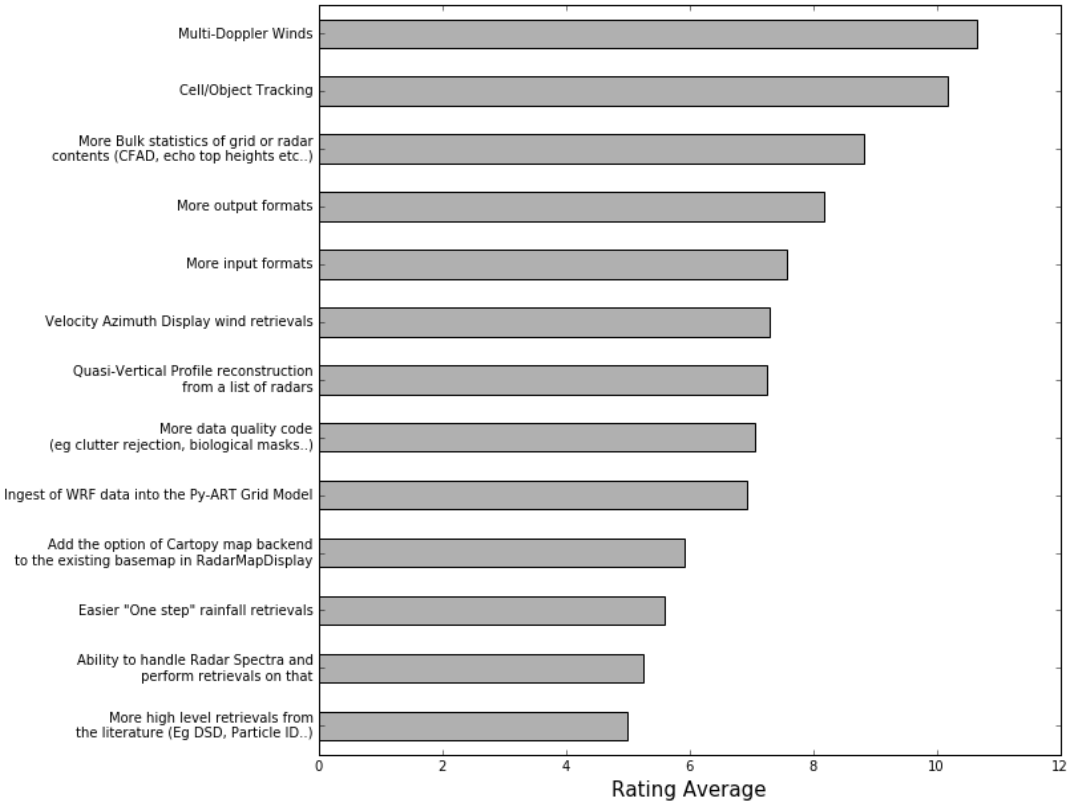
Roadmap Drafted: ✓

Initial Review: ✓

Stakeholder Review: ?

Public Review: ✗

DoE Review: ✗



SOME ADVICE (?) FOR LROSE

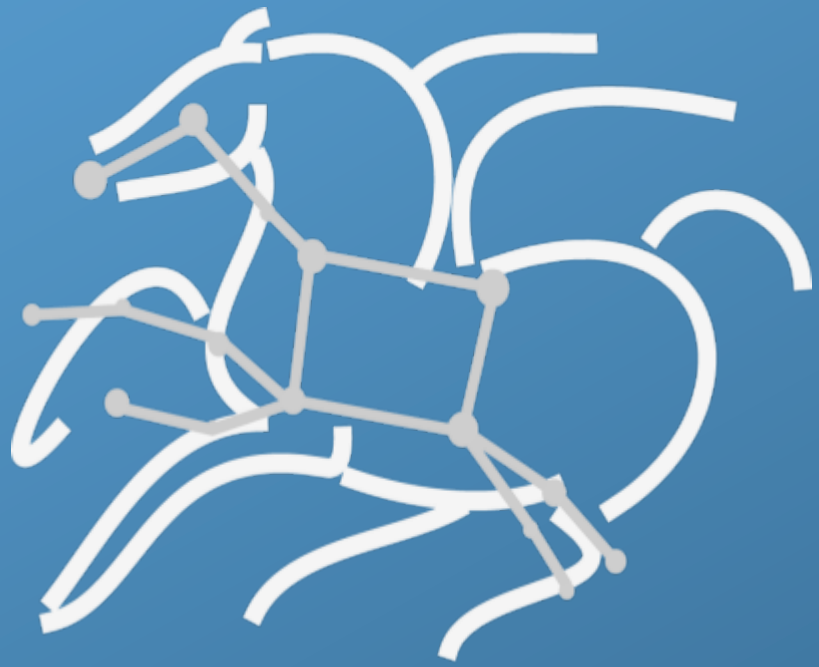
Benefit from our pain.

- Always think about how your package will survive when funding runs out.. Disaster planning
- Identify staff that you never want to leave, and get them **to document everything they do.**
- Do less, better.
- All the stuff the geeks geek out on, Continuous integration, coverage tests, document generation and self documenting code **is actually really important to make sure you do not spend the rest of your days maintaining code.**
- Be mindful of every dependency you take on. Consider conditional functionality.

THANK YOU!

Argonne NATIONAL LABORATORY

Compute Pipelines using Pegasus Workflows: An Introduction

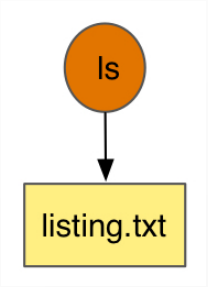


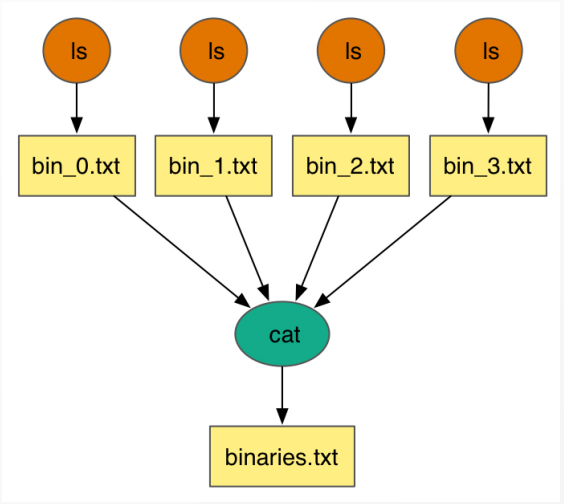
Karan Vahi
vahi@isi.edu

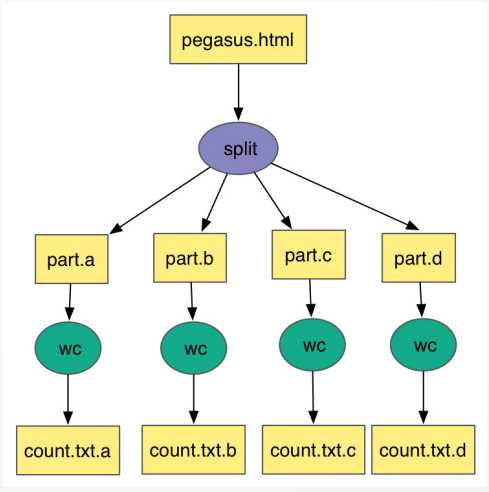
USC Viterbi
School of Engineering
Information Sciences Institute

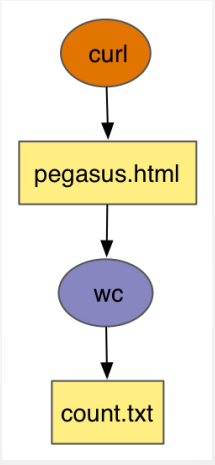
https://pegasus.isi.edu

Compute Pipelines – Building Blocks









Challenges scientists face while developing pipelines

Portability
How can you run a pipeline on Amazon EC2 one day, and a PBS cluster the next?

Data Management
How do you ship in the small/large amounts data required by your pipeline?
Different protocols: Can I use SRM? How about GridFTP? HTTP and Squid proxies?
Can I use Cloud based storage like S3 on EC2?

Debug and Monitor Computations.
Users need automated tools to go through the log files
Need to correlate data across lots of log files
Need to know what host a job ran on and how it was invoked

Restructure Pipelines for Improved Performance
Short running tasks?
Data placement?

Why Pegasus?

Automates complex, multi-stage processing pipelines

Enables parallel, distributed computations

Portable: Describe once; execute multiple times

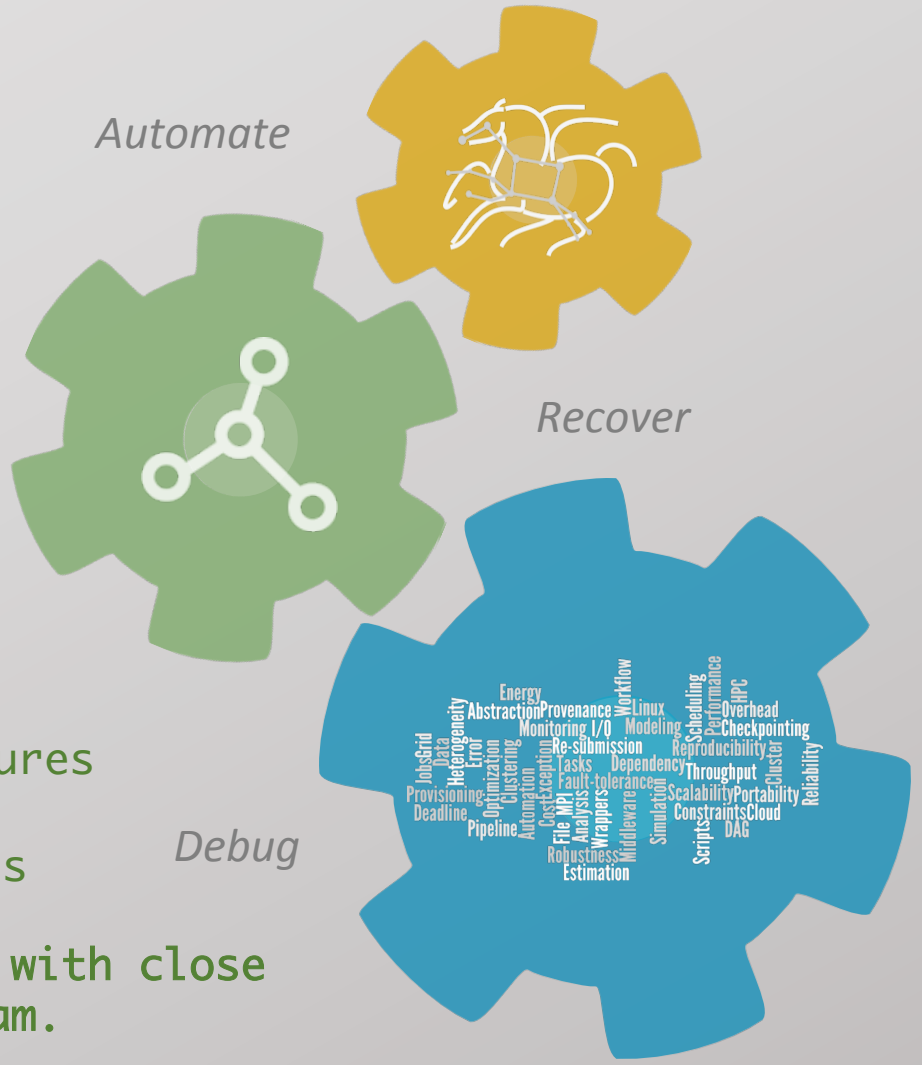
Automatically executes data transfers

Reusable, aids reproducibility

Records how data was produced (provenance)

Provides to tools to handle and debug failures

Keeps track of data and files



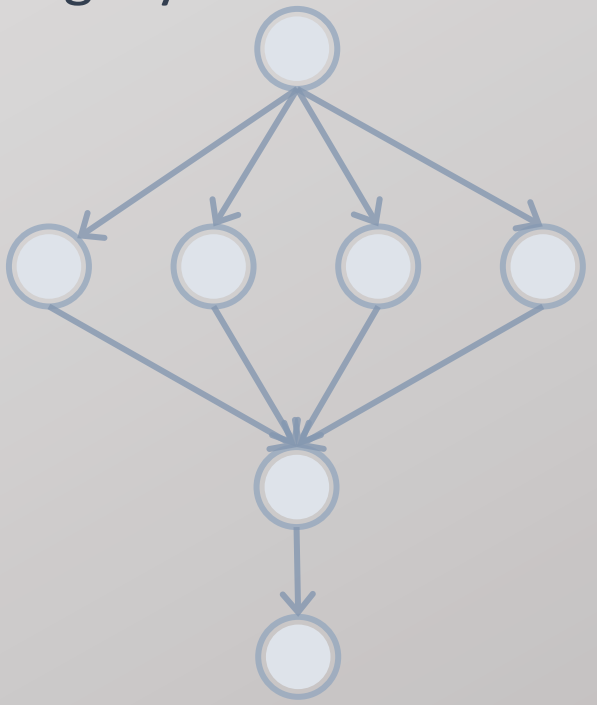
HTCondor
High Throughput Computing

NSF funded project since 2001, with close Collaboration with HTCondor team.

Key Pegasus Concepts

Pegasus WMS == Pegasus planner (mapper) + DAGMan workflow engine + HTCondor scheduler/broker + Pegasus Monitoring layer

- Pegasus maps workflows to infrastructure
- DAGMan manages dependencies and reliability
- HTCondor is used as a broker to interface with different schedulers
- Monitoring layer parses condor logs and puts them in a relational database



Workflows are DAGs (or hierarchical DAGs)

- Nodes: jobs, edges: dependencies
- No while loops, no conditional branches

Planning occurs ahead of execution

- (Except hierarchical workflows)

Planning converts an abstract workflow into an executable workflow

- Planner is like a workflow compiler. Compiles user workflows to target execution environment.

Taking a closer look into a Pegasus workflow…

abstract workflow

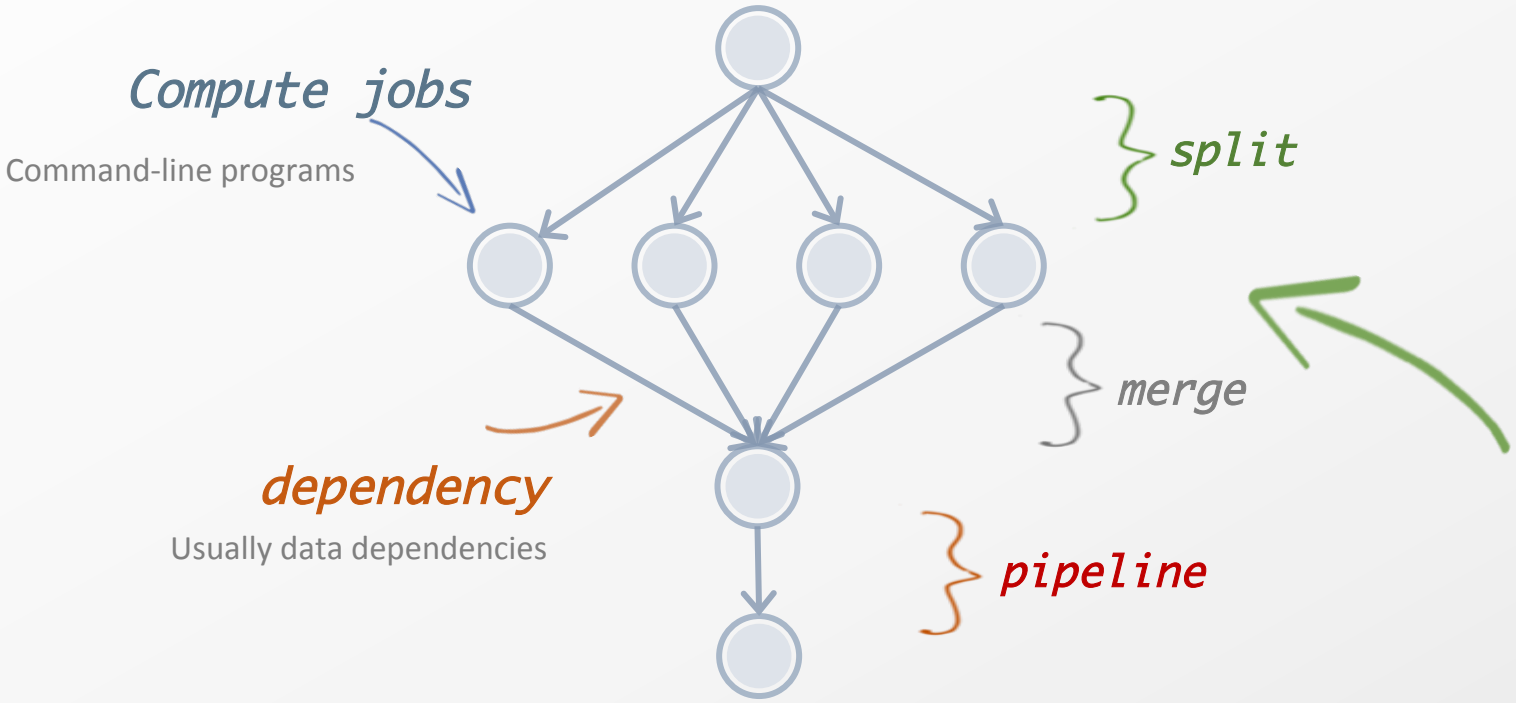
executable workflow

optimizations

storage constraints

DAG

directed-acyclic graphs



DAX

DAG in XML

Pegasus input workflow description

Looks very similar to how users describe their pipelines

Only identifies computation steps, devoid of resource descriptions and data locations

File aware For each node you specify the input and output files by use of logical identifiers

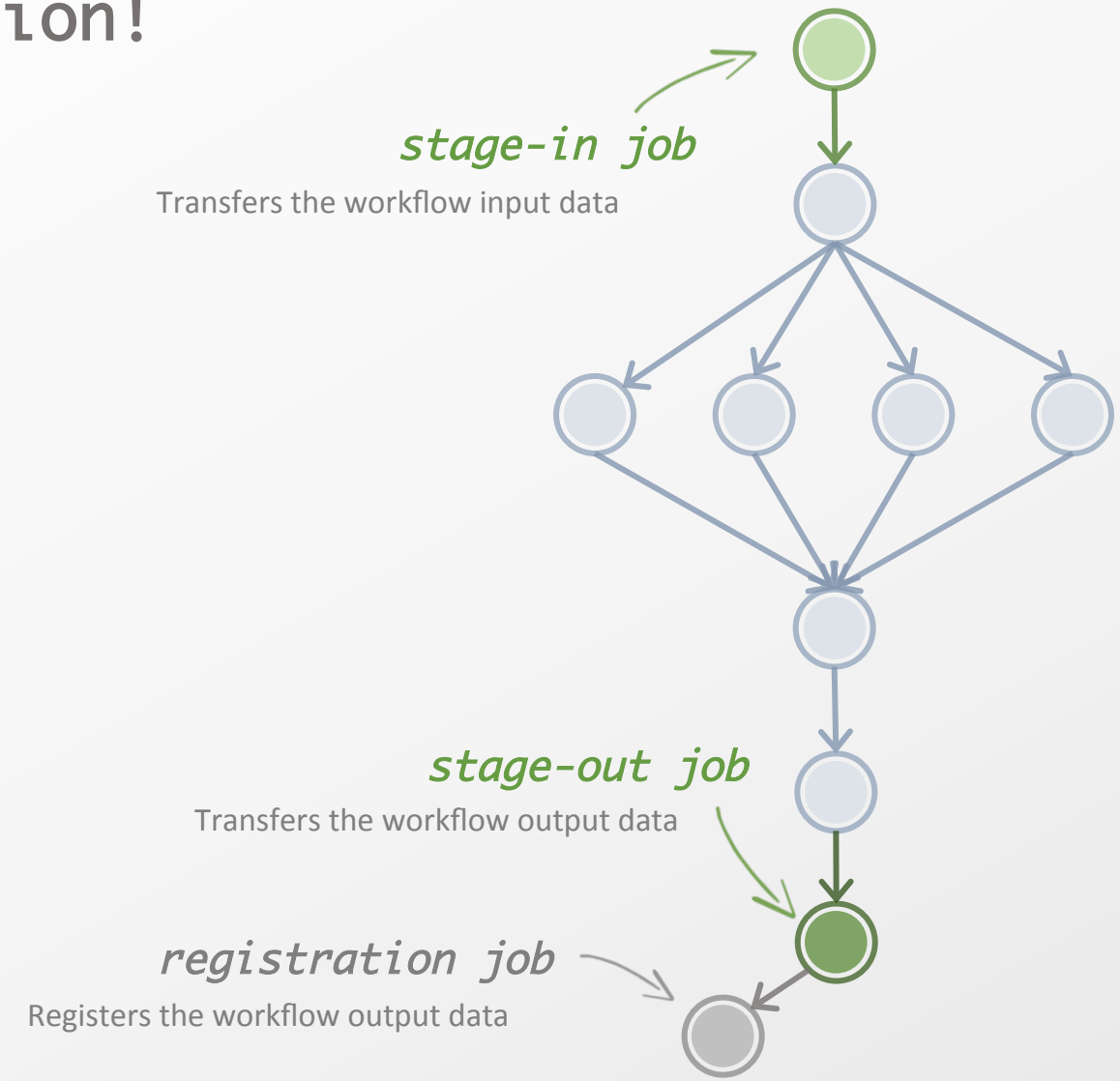
From the abstraction to execution!

abstract workflow

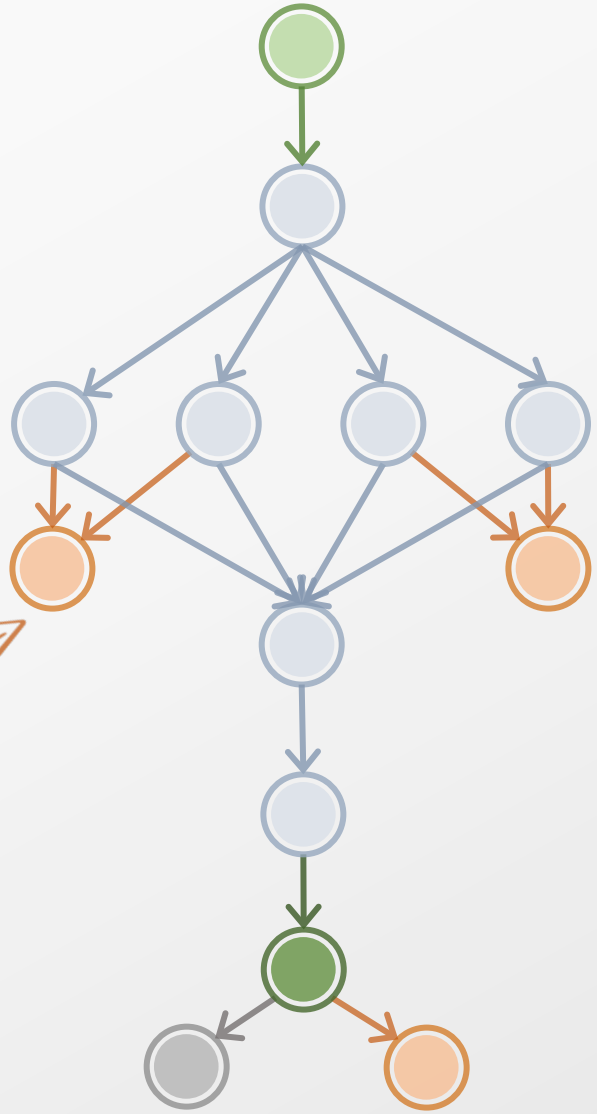
executable workflow

optimizations

storage constraints



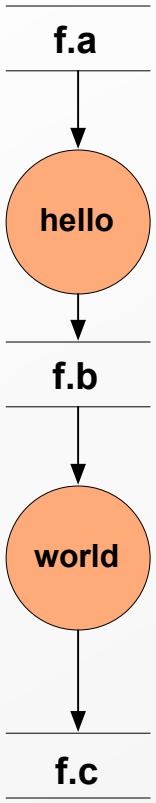
Optimizing storage usage…



cleanup job

Removes unused data

Pegasus also provides tools to generate the abstract workflow



Java

perl

python

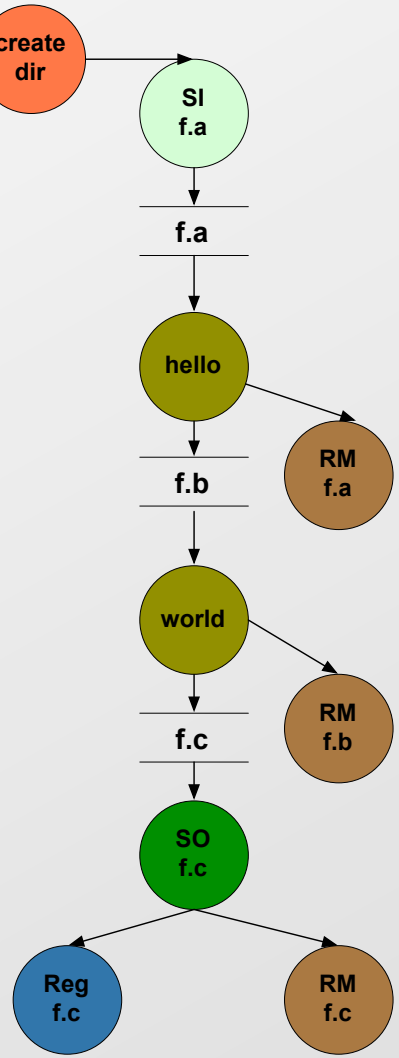
```python
#!/usr/bin/env python

from Pegasus.DAX3 import *
import sys
import os

# Create a abstract dag
dax = ADAG("hello_world")

# Add the hello job
hello = Job(namespace="hello_world",
            name="hello", version="1.0")
b = File("f.b")
hello.uses(a, link=Link.INPUT)
hello.uses(b, link=Link.OUTPUT)
dax.addJob(hello)

# Add the world job (depends on the hello job)
world = Job(namespace="hello_world",
            name="world", version="1.0")
c = File("f.c")
world.uses(b, link=Link.INPUT)
world.uses(c, link=Link.OUTPUT)
dax.addJob(world)

# Add control-flow dependencies
dax.addDependency(Dependency(parent=hello,
                             child=world))

# Write the DAX to stdout
dax.writeXML(sys.stdout)
```

DAG in XML

DAX

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!-- generator: python -->
<adag xmlns="http://pegasus.isi.edu/schema/DAX"
            version="3.4" name="hello_world">

    <!-- describe the jobs making
        up the hello world pipeline -->
    <job id="ID0000001" namespace="hello_world"
                    name="hello" version="1.0">

        <uses name="f.b" link="output"/>
        <uses name="f.a" link="input"/>
    </job>

    <job id="ID0000002" namespace="hello_world"
                    name="world" version="1.0">

        <uses name="f.b" link="input"/>
        <uses name="f.c" link="output"/>
    </job>

    <!-- describe the edges in the DAG -->
    <child ref="ID0000002">
        <parent ref="ID0000001"/>
    </child>
</adag>
```

Plan



So, what information does Pegasus need?

Transformation Catalog

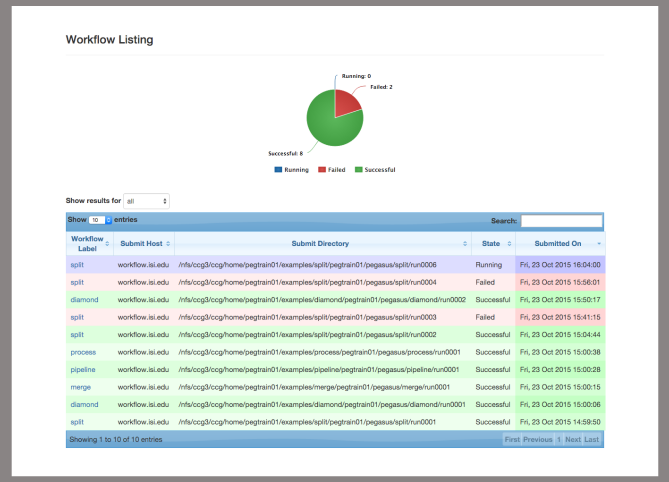
describes all of the executables (called “transformations”) used by the workflow

Site Catalog

describes the sites where the workflow jobs are to be executed

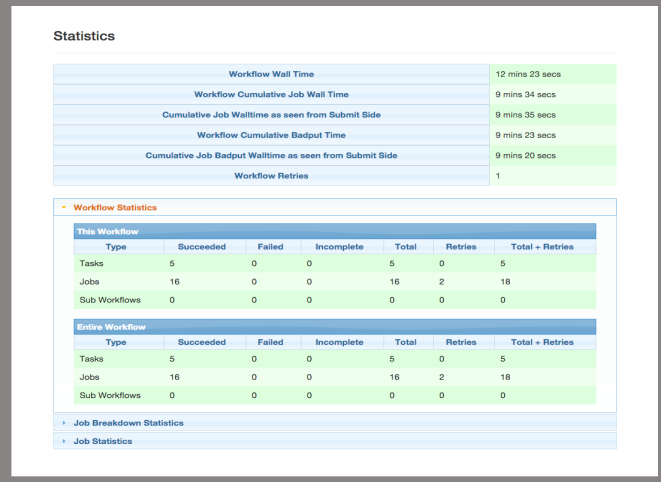
Replica Catalog

describes all of the input data stored on external servers

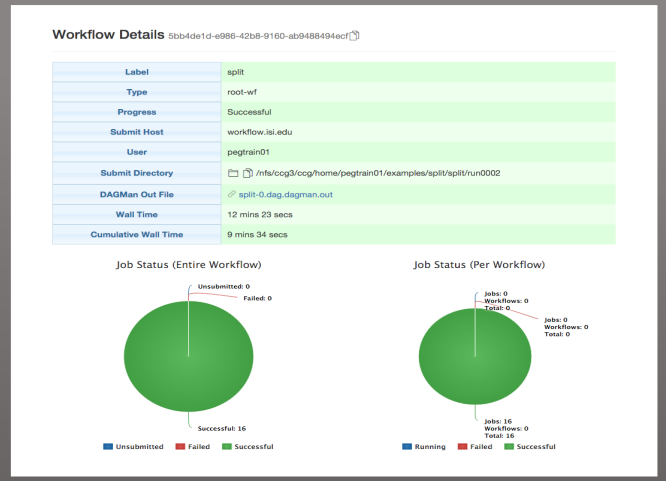


Pegasus dashboard

web interface for monitoring and debugging workflows



Real-time monitoring of workflow executions. It shows the status of the workflows and jobs, job characteristics, statistics and performance metrics. Provenance data is stored into a relational database.



Real-time Monitoring

Reporting

Debugging

Troubleshooting

RESTful API

But, if you prefer the command-line…

```
$ pegasus-status pegasus/examples/split/run0001
STAT IN_STATE JOB
Run 00:39 split-0 (/home/pegasus/examples/split/run0001)
Idle 00:03  └─split_ID0000001
Summary: 2 Condor jobs total (I:1 R:1)

UNRDY READY PRE IN_Q POST DONE FAIL %DONE STATE   DAGNAME
 14     0    0    1    0   2    0    11.8 Running *split-0.dag
```

```
$ pegasus-analyzer pegasus/examples/split/run0001
pegasus-analyzer: initializing...

**************************Summary***************************

Total jobs : 7 (100.00%)
# jobs succeeded : 7 (100.00%)
# jobs failed : 0 (0.00%)
# jobs unsubmitted : 0 (0.00%)
```

```
$ pegasus-statistics -s all pegasus/examples/split/run0001
------------------------------------------------------------------------------
Type           Succeeded Failed Incomplete Total Retries Total+Retries
Tasks              5       0        0        5     0         5
Jobs              17       0        0       17     0        17
Sub-Workflows      0       0        0        0     0         0
------------------------------------------------------------------------------

Workflow wall time : 2 mins, 6 secs
Workflow cumulative job wall time : 38 secs
Cumulative job wall time as seen from submit side : 42 secs
Workflow cumulative job badput wall time :
Cumulative job badput wall time as seen from submit side :
```

…Pegasus provides a set of concise and powerful tools

And if a job fails?

Job Failure Detection

detects non-zero exit code
output parsing for success or failure message
exceeded timeout
do not produced expected output files

Job Retry

helps with transient failures
set number of retries per job and run

Checkpoint Files

job generates checkpoint files
staging of checkpoint files is
automatic on restarts

Rescue DAGs

workflow can be restarted from checkpoint file
recover from failures with minimal loss

pegasus-kickstart

() login02.osgconnect.net — Konsole

File Edit View Bookmarks Settings Help

```
<?xml version="1.0" encoding="UTF-8"?>
<invocation xmlns="http://pegasus.isi.edu/schema/invocation" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://pegasus.isi.edu/schema/invocation http://pegasus.isi.edu/schema/iv-2.3.xsd" version="2.3" start="2016-11-28T14:27:48.909-06:00" duration="11200.691" transformation="job-wrapper.sh" derivation="ID0013214" resource="condorpool" wf-label="particleshower" wf-stamp="2016-11-22T21:14:13-06:00" interface="eth0" hostaddr="131.225.208.240" hostname="fnpc4593.fnal.gov" pid="1725084" uid="12740" user="osg" gid="9652" group="osg" umask="0022">
  <mainjob start="2016-11-28T14:27:49.007-06:00" duration="11200.593" pid="1725089">
    <usage utime="10921.591" stime="30.304" maxrss="395820" minflt="128741" majflt="18" nswap="0" inblock="85776" outblock="1717424" msgsnd="0" msgrcv="0" nsignals="0" nvcsw="7676" nivcsw="185495"/>
    <status raw="0"><regular exitcode="0"/></status>
    <statcall error="0">
      <file name="/storage/local/data1/condor/execute/dir_1227464/glide_bSxwfe/execute/dir_1724937/pegasus.XRZ1p3/job-wrapper.sh">23212F62696E2F626173680A0A736574</file>
      <statinfo mode="0100755" size="1305" inode="16648869" nlink="1" blksize="4096" blocks="8" mtime="2016-11-28T12:10:53-06:00" atime="2016-11-28T14:27:48-06:00" ctime="2016-11-28T14:27:48-06:00" uid="12740" user="osg" gid="9652" group="osg"/>
    </statcall>
    <argument-vector>
      <arg nr="1">100</arg>
      <arg nr="2">0</arg>
      <arg nr="3">gamma</arg>
      <arg nr="4">62</arg>
      <arg nr="5">VERITAS</arg>
      <arg nr="6">corsika.tar.gz</arg>
      <arg nr="7">corsika75000Linux_QGSII_urqmd</arg>
      <arg nr="8">13213</arg>
    </argument-vector>
  </mainjob>
  <jobids condor="547839.0"/>
  <cwd>/storage/local/data1/condor/execute/dir_1227464/glide_bSxwfe/execute/dir_1724937/pegasus.XRZ1p3</cwd>
  <usage utime="0.013" stime="0.085" maxrss="828" minflt="2448" majflt="0" nswap="0" inblock="0" outblock="0" msgsnd="0" msgrcv="0" nsignals="0" nvcsw="1" nivcsw="12"/>
  <machine page-size="4096">
    <stamp>2016-11-28T14:27:48.909-06:00</stamp>
    <uname system="linux" nodename="fnpc4593.fnal.gov" release="2.6.32-642.6.2.el6.x86_64" machine="x86_64">#1 SMP Tue Oct 25 15:06:33 CDT 2016</uname>
    <linux>
      <ram total="65319608" free="1071948" shared="0" buffer="148224"/>
      <swap total="8388604" free="7741364"/>
      <boot idle="45893257.760">2016-11-09T16:40:54.260-06:00</boot>
      <cpu count="32" speed="2000" vendor="AuthenticAMD">AMD Opteron(tm) Processor 6128</cpu>
      <load min1="26.35" min5="27.70" min15="24.33"/>
      <procs total="881" running="23" sleeping="854" waiting="3" zombie="1" vmsize="65009304" rss="14780272"/>
      <task total="1273" running="24" sleeping="1243" waiting="5" zombie="1"/>
    </linux>
```

5,5 Top

() login02.osgconnect.net

Data Staging Configurations

- Condor I/O (HTCondor pools, OSG, …)
  - Worker nodes do not share a file system
  - Data is pulled from / pushed to the submit host via HTCondor file transfers
  - Staging site is the submit host
- Non-shared File System (clouds, OSG, …)
  - Worker nodes do not share a file system
  - Data is pulled / pushed from a staging site, possibly not co-located with the computation
- Shared File System (HPC sites, XSEDE, Campus clusters, …)
  - I/O is directly against the shared file system

pegasus-transfer

- Pegasus' internal data transfer tool
- Supports many different protocols
- Directory creation, file removal
  - If protocol supports, used for cleanup
- Two stage transfers
  - e.g. GridFTP to S3 = GridFTP to local file, local file to S3
- Parallel transfers
- Automatic retries
- Checkpoint and restart transfers
- Credential management
  - Uses the appropriate credential for each site and each protocol (even 3rd party transfers)

Protocols

- HTTP
- SCP
- GridFTP
- iRods
- Amazon S3
- Google Storage
- SRM
- FDT
- stashcp
- cp
- ln -s

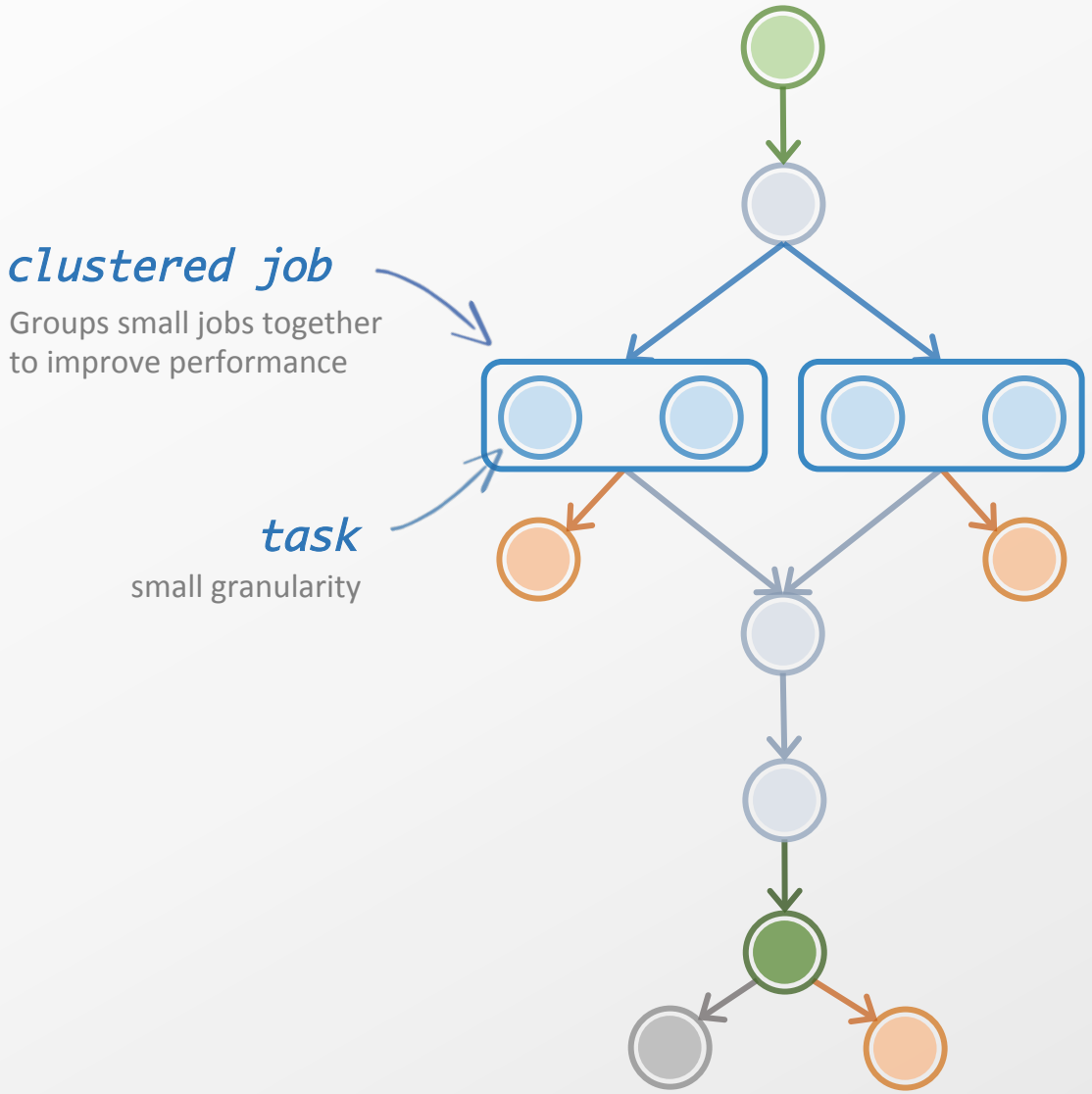
A few more features…

Performance, why not improve it?

workflow restructuring

workflow reduction

hierarchical workflows

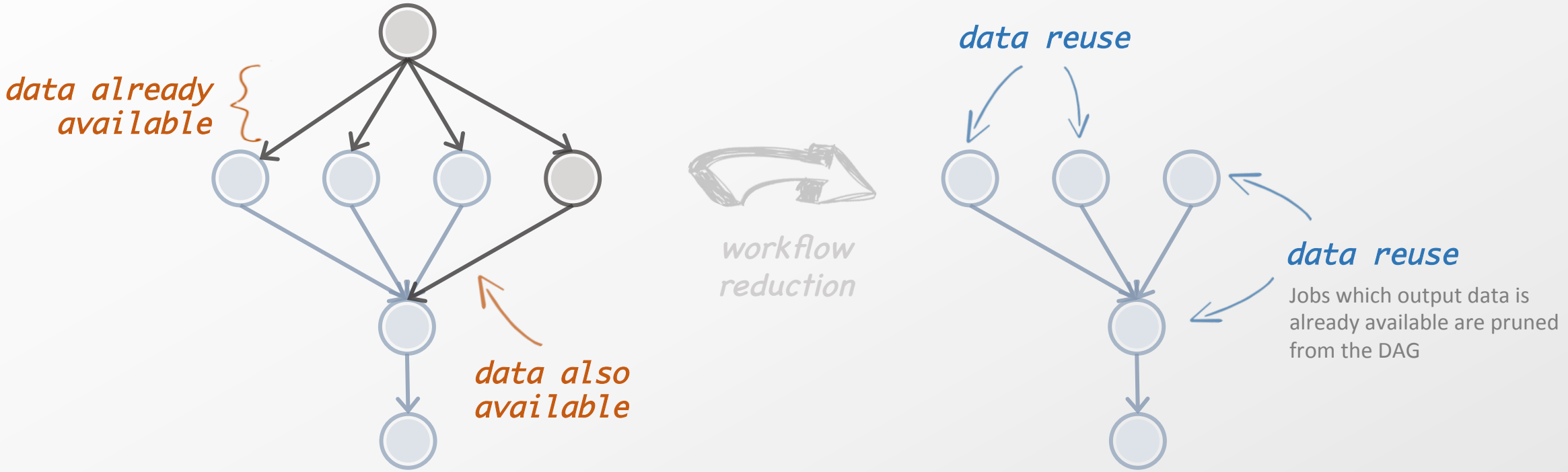


What about data reuse?

workflow restructuring

workflow reduction

hierarchical workflows



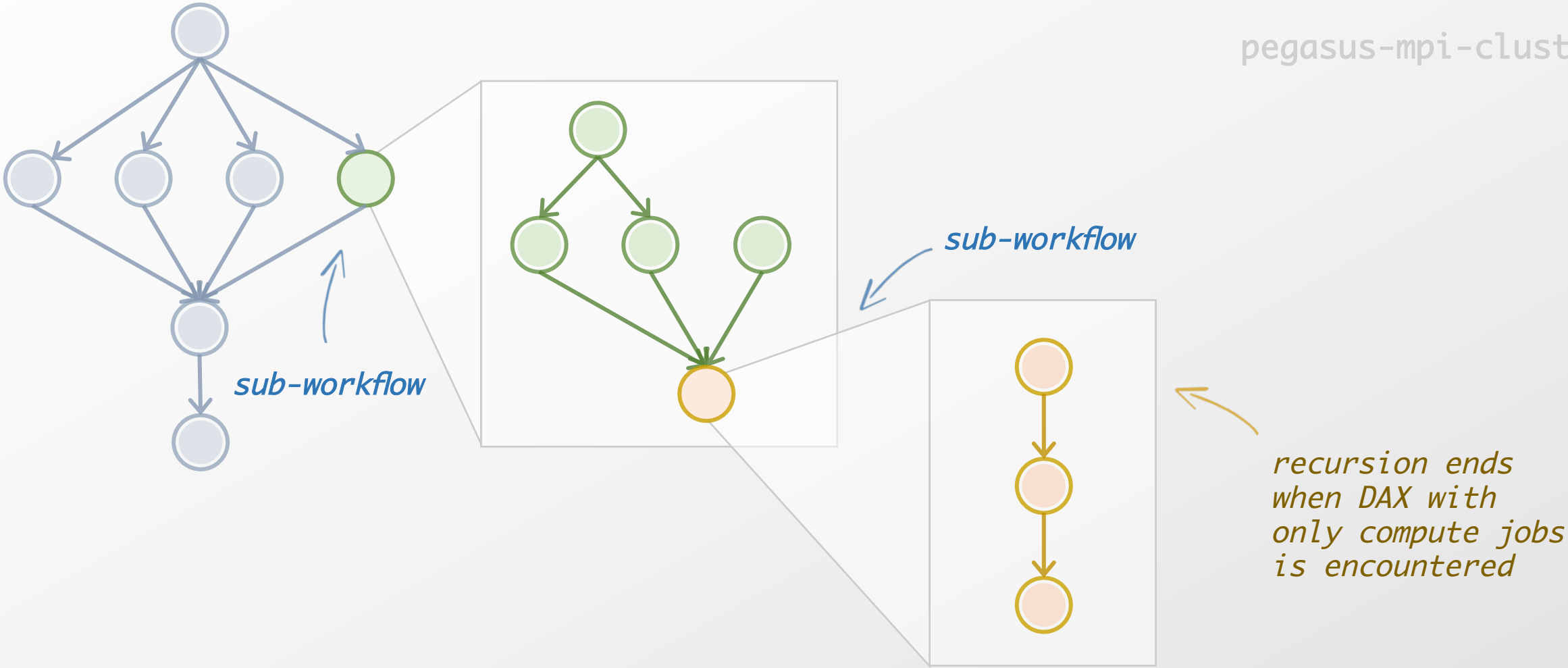
Pegasus also handles **large-scale workflows**

workflow restructuring

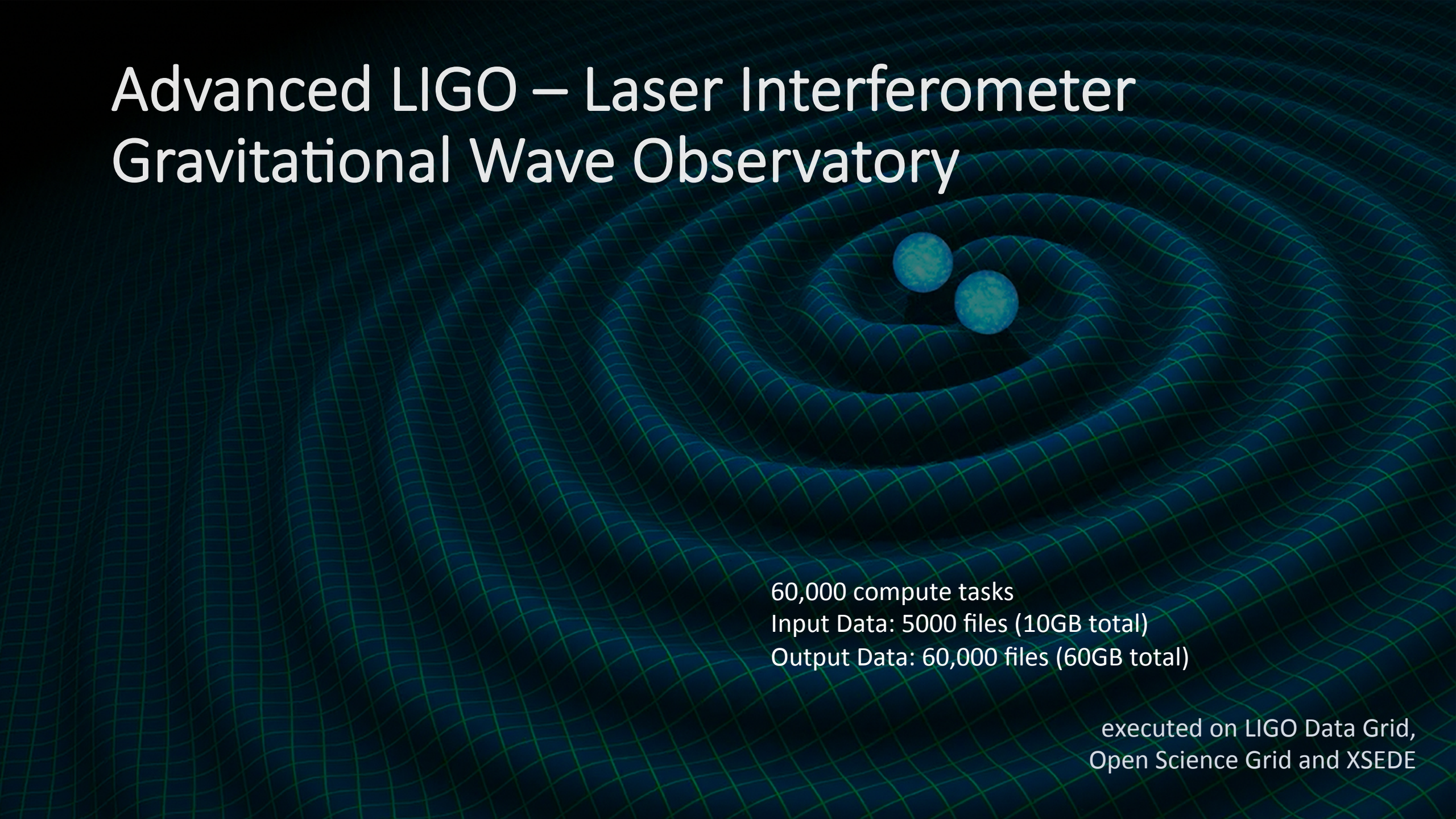
workflow reduction

hierarchical workflows

pegasus-mpi-cluster



Advanced LIGO – Laser Interferometer Gravitational Wave Observatory

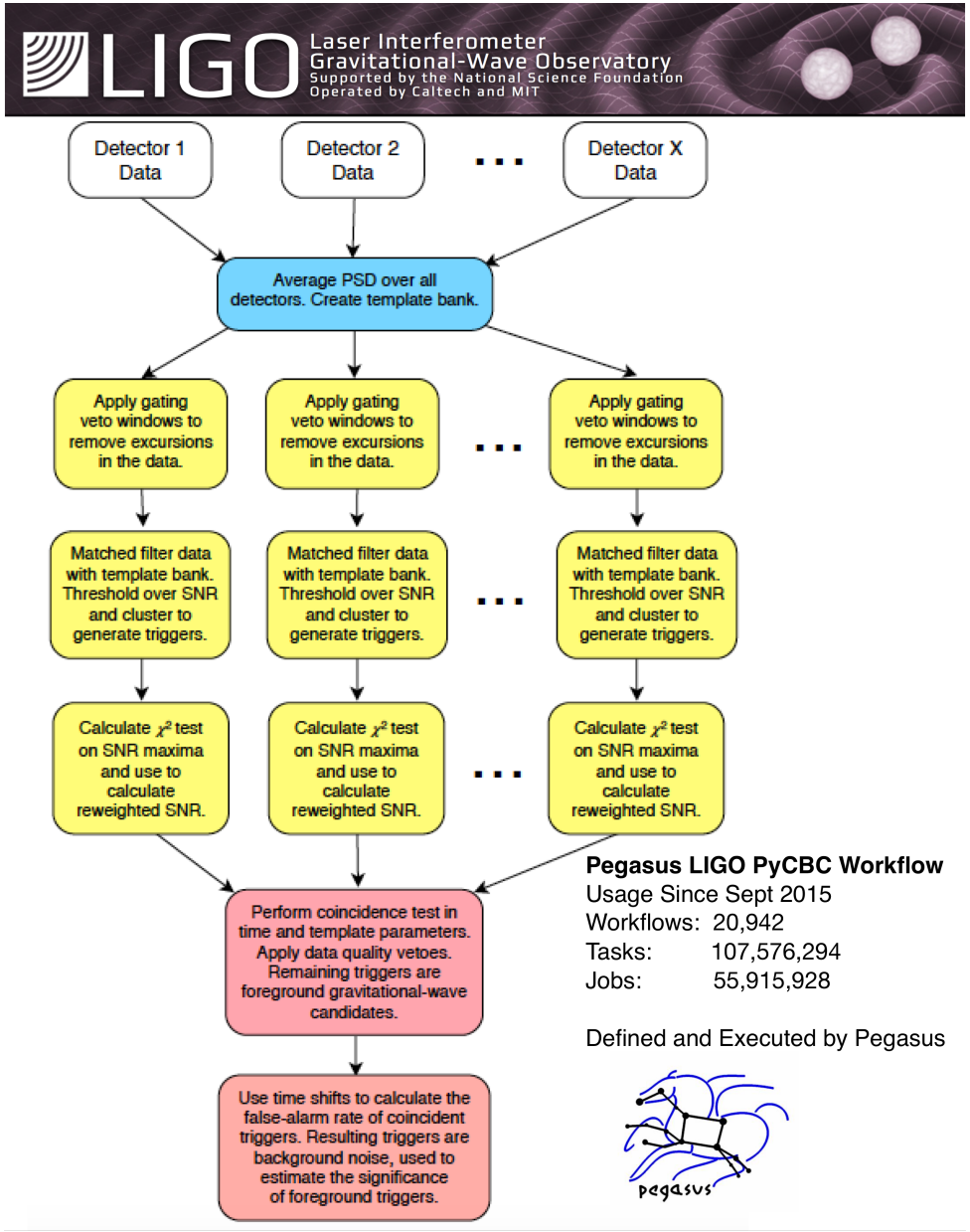


60,000 compute tasks
Input Data: 5000 files (10GB total)
Output Data: 60,000 files (60GB total)

executed on LIGO Data Grid,
Open Science Grid and XSEDE

Advanced LIGO PyCBC Workflow

- One of the main pipelines to measure the statistical significance of data needed for discovery.
- Contains 100's of thousands of jobs and accesses on order of terabytes of data.
- Uses data from multiple detectors.
- For the detection, the pipeline was executed on Syracuse and Albert Einstein Institute Hannover
- A single run of the binary black hole + binary neutron star search through the O1 data (about 3 calendar months of data with 50% duty cycle) requires a workflow with 194,364 jobs. Generating the final O1 results with all the review required for the first discovery took about 20 million core hours



PyCBC Papers: An improved pipeline to search for gravitational waves from compact binary coalescence. *Samantha Usman, Duncan Brown et al.*
The PyCBC search for gravitational waves from compact binary coalescence, *Samantha Usman et al* (https://arxiv.org/abs/1508.02357)
PyCBC Detection GW150914: First results from the search for binary black hole coalescence with Advanced LIGO. *B. P. Abbott et al.*

Benefits to LIGO provided by Pegasus- Expanded Computing Horizons

- No longer limited to a single execution resource
  - Non Pegasus LIGO pipelines can often only run on LIGO clusters
  - Input is replicated out of band , in a rigid directory layout.
  - Rely on the shared filesystem to access data.

- Pegasus made it possible to leverage Non LDG Computing Resources

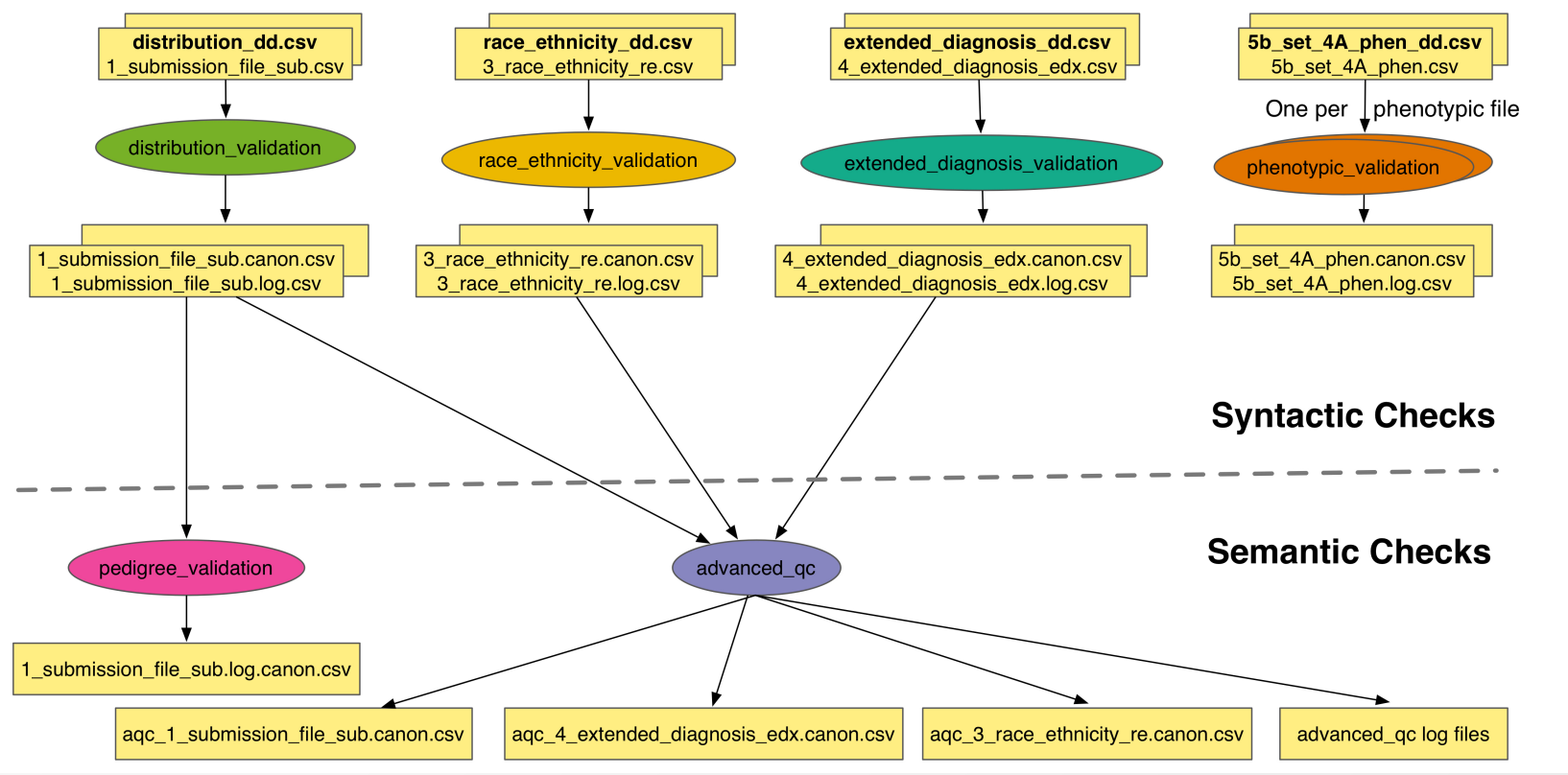
  - Open Science Grid
    - Dynamic – Best Effort Resource with no shared filesystem available

  - Large NSF Supercomputing Clusters XSEDE
    - No HTCondor
    - Geared for Large MPI jobs, not thousands of single node jobs
    - LIGO tried to setup XSEDE cluster as a LDG site but mismatch in setup.
    - Pegasus enabled LIGO to use XSEDE without changes at LIGO or at XSEDE

  - VIRGO Resources in Europe
    - Clusters with no shared filesystem and different storage management infrastructure than LDG
    - No HTCondor

Automated Quality Control Workflows for NIMH-NRGR

The NIMH Repository and Genomics Resource (NIMH-RGR) is large NIH funded center that facilitates psychiatric genetic research by providing a collection of over 150,000 well characterized, high quality patient and control bio-samples.

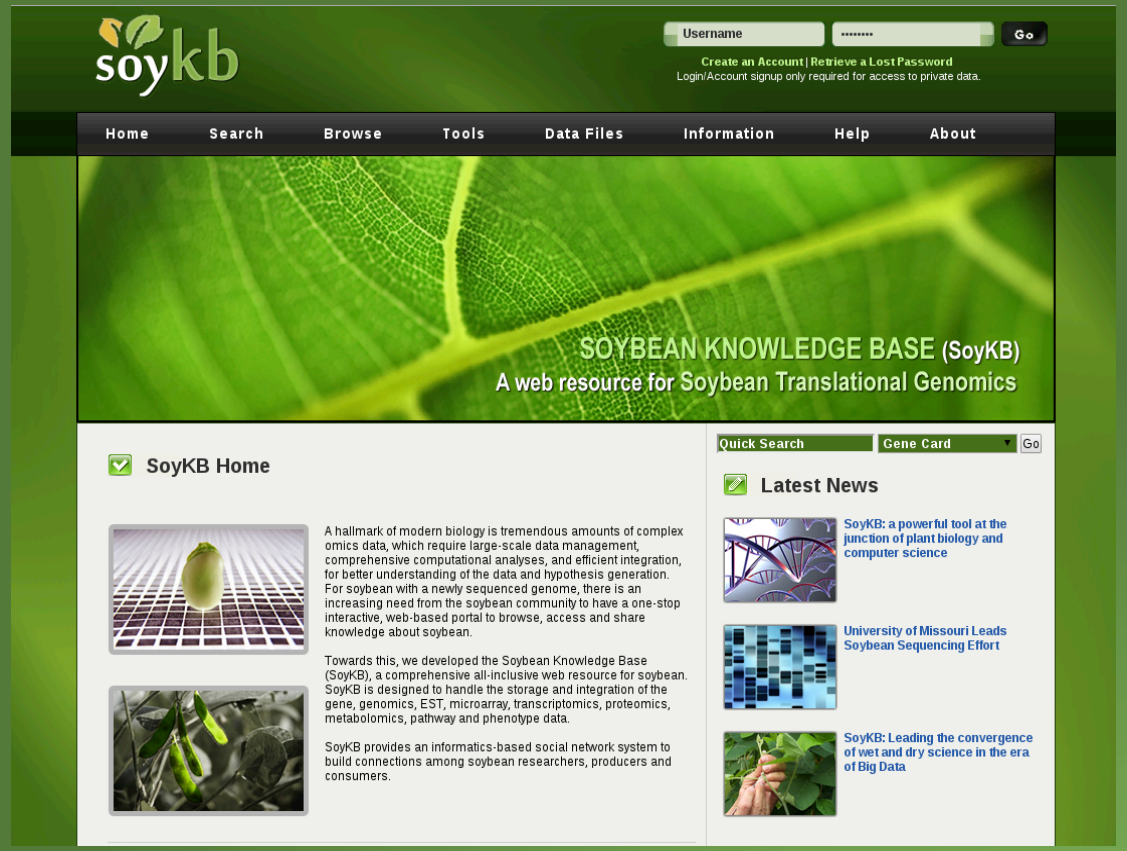


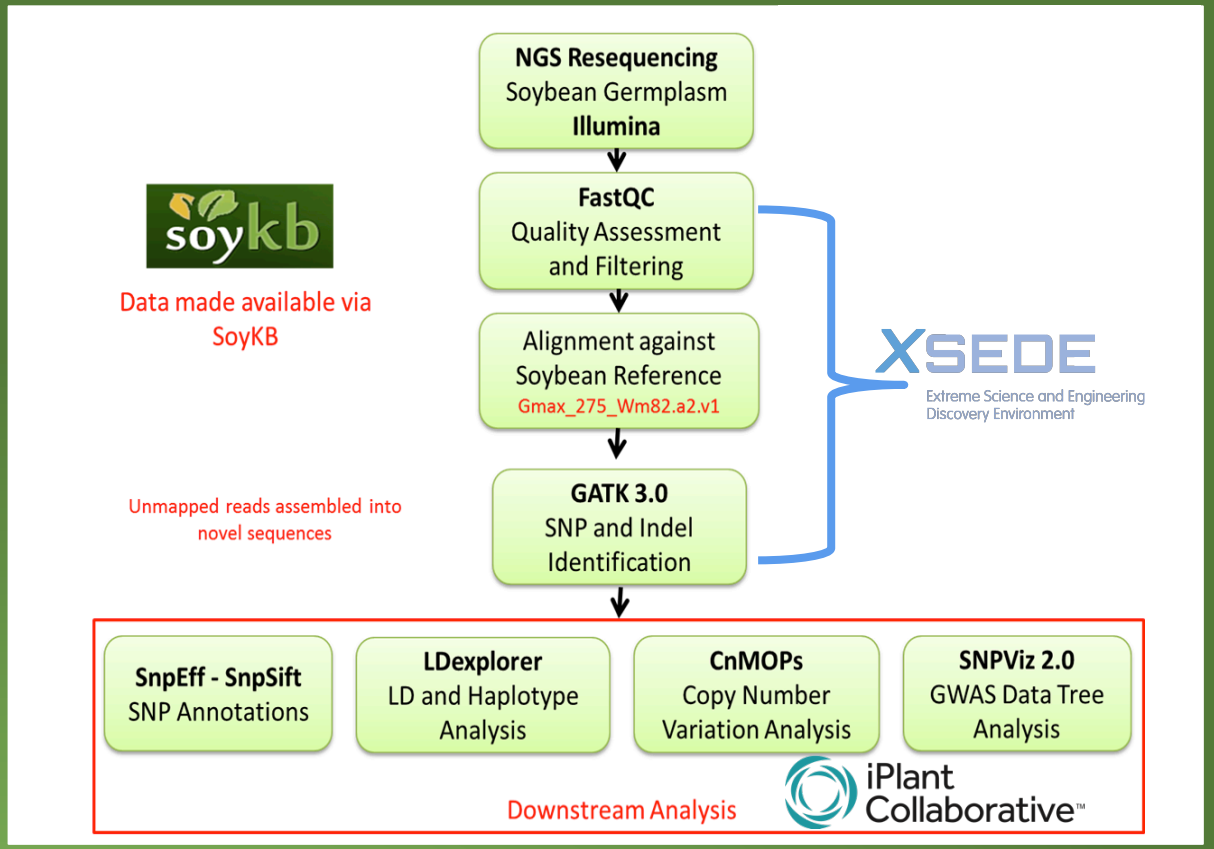
Pipeline to automate quality control checks on incoming phenotypic data to the center

Earlier curation process was manual done by analysts at the center, that was often error prone

Web based system developed on top of Pegasus WMS by the center.

https://www.nimhgenetics.org/submit_data/

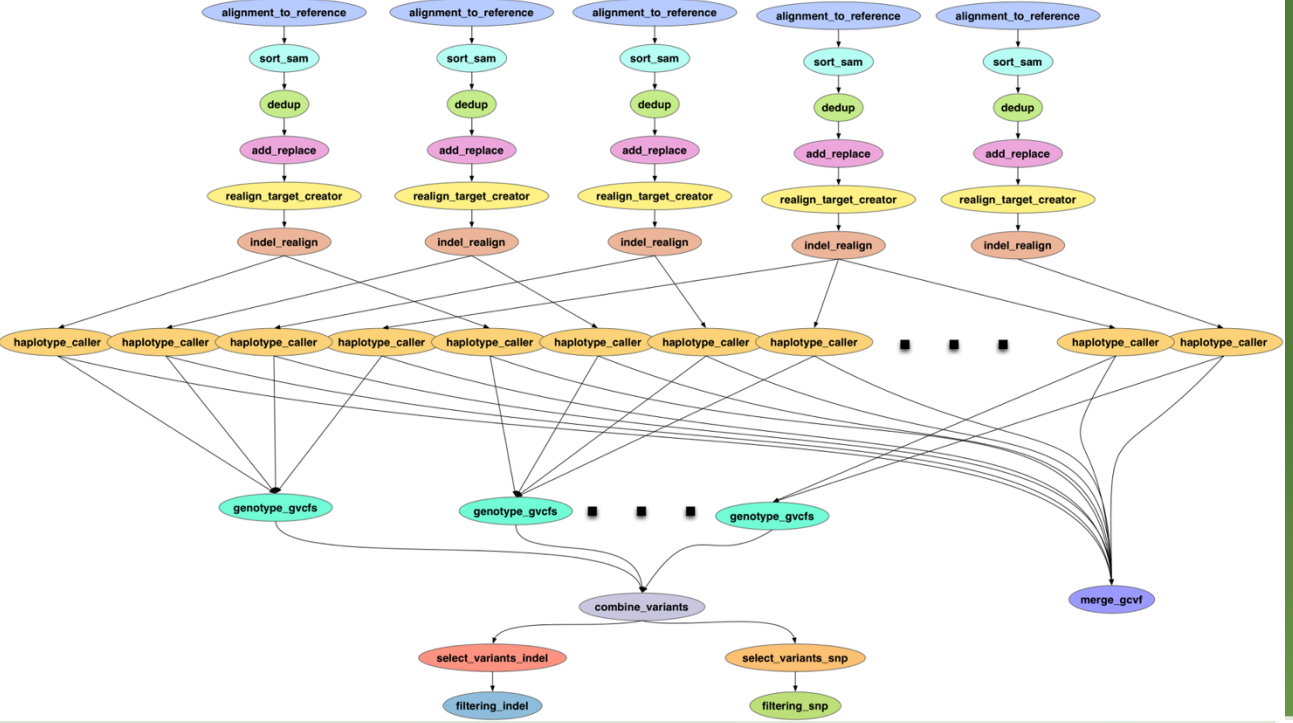




http://soykb.org

XSEDE Allocation
PI: Dong Xu
Trupti Joshi, Saad Kahn, Yang Liu, Juexin Wang, Badu Valliyodan, Jiaojiao Wang

https://github.com/pegasus-isi/Soybean-Workflow



Task	Base Code	Cores (Threads)	Memory (GB)
Alignment_to_reference	BWA	7	8
Sort_sam	Picard	1	21
Dedup	Picard	1	21
Add_replace	Picard	1	21
Realign_target_creator	GATK	15	10
Indel_realign	GATK	1	10
Haplotype_caller	GATK	1	3
Genotype_gvcfs	GATK	1	10
Merge_gvcf	GATK	10	20
Combine_variants	GATK	1	10
Select_variants	GATK	14	10
Filtering	GATK	1	10

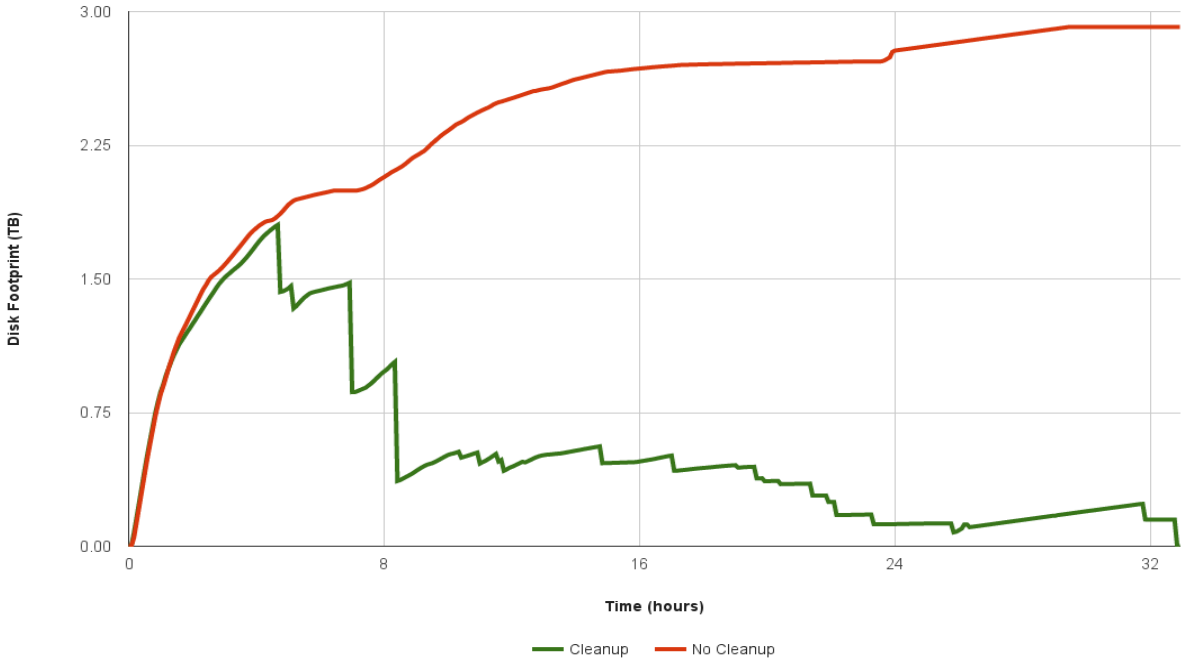
TACC Wrangler as Execution Environment

Flash Based Shared Storage

Switched to glideins (pilot jobs) - Brings in remote compute nodes and joins them to the HTCondor pool on in the submit host - Workflow runs at a finer granularity

Works well on Wrangler due to more cores and memory per node (48 cores, 128 GB RAM)

Pegasus



Pegasus est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website

http://pegasus.isi.edu

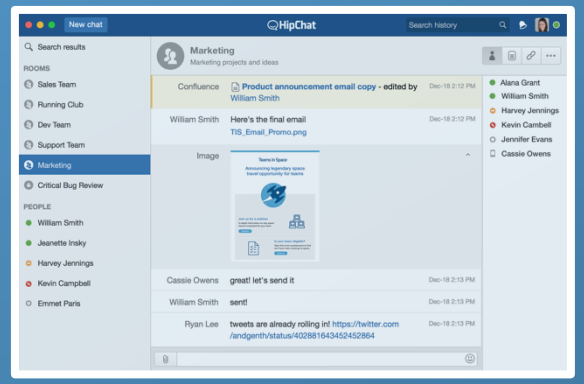
Users Mailing List

pegasus-users@isi.edu

Support

pegasus-support@isi.edu

HipChat



Pegasus est. 2001

Automate, recover, and debug scientific computations.

Thank You

Questions?

Mats Rynge
rynge@isi.edu

USC Viterbi
School of Engineering
Information Sciences Institute

Meet our team


Ewa Deelman


Karan Vahi


Mats Rynge


Rajiv Mayani


Rafael Ferreira da Silva

NSF U.S. DEPARTMENT OF ENERGY NIH

Defining the spatial properties of precipitation features using data from the WSR-88D network

CAREER Award
BCS-1053864

Dr. Corene Matyas

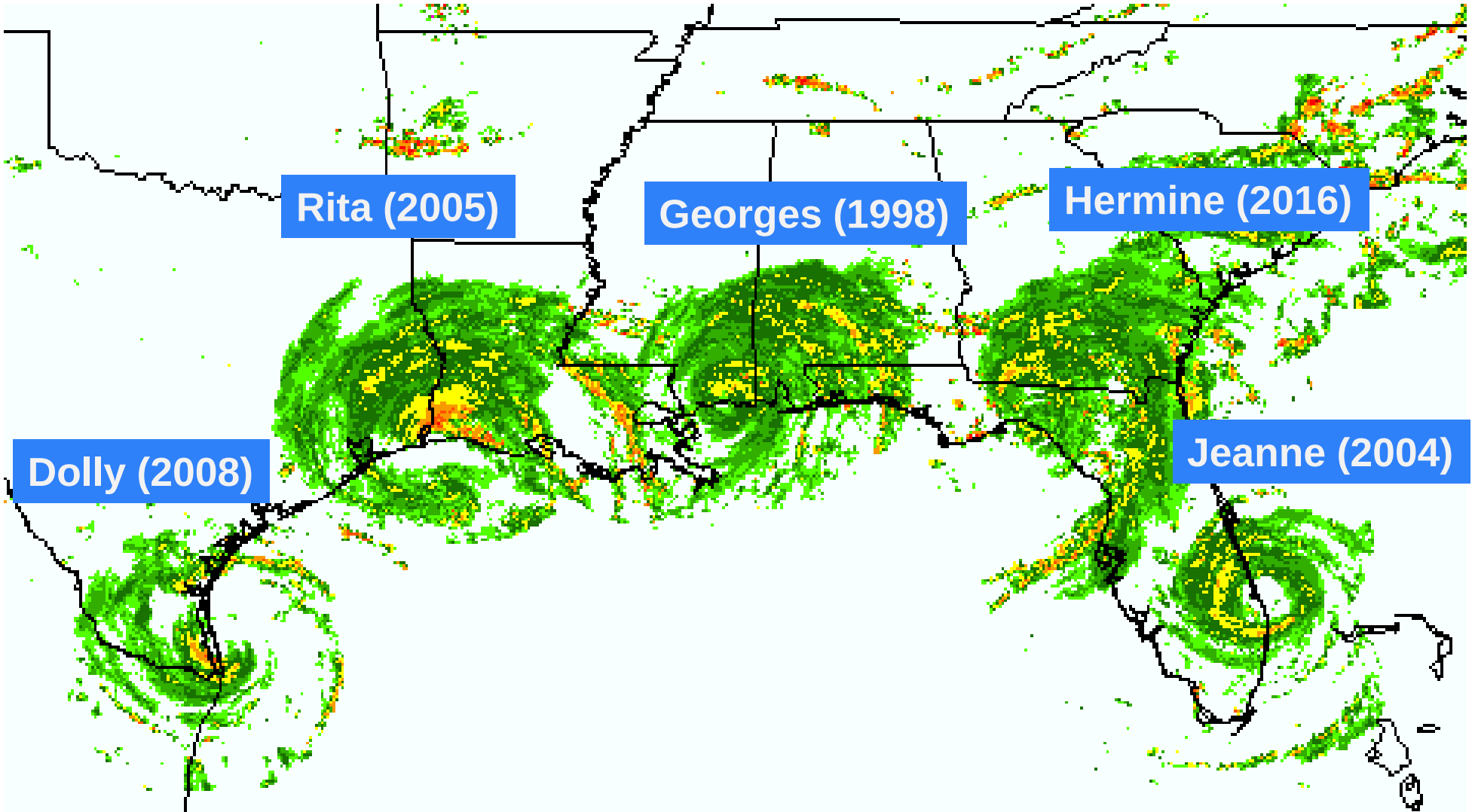
Jingyin Tang, Ph.D. Candidate

Department of Geography

University of Florida

hurricane.geog.ufl.edu

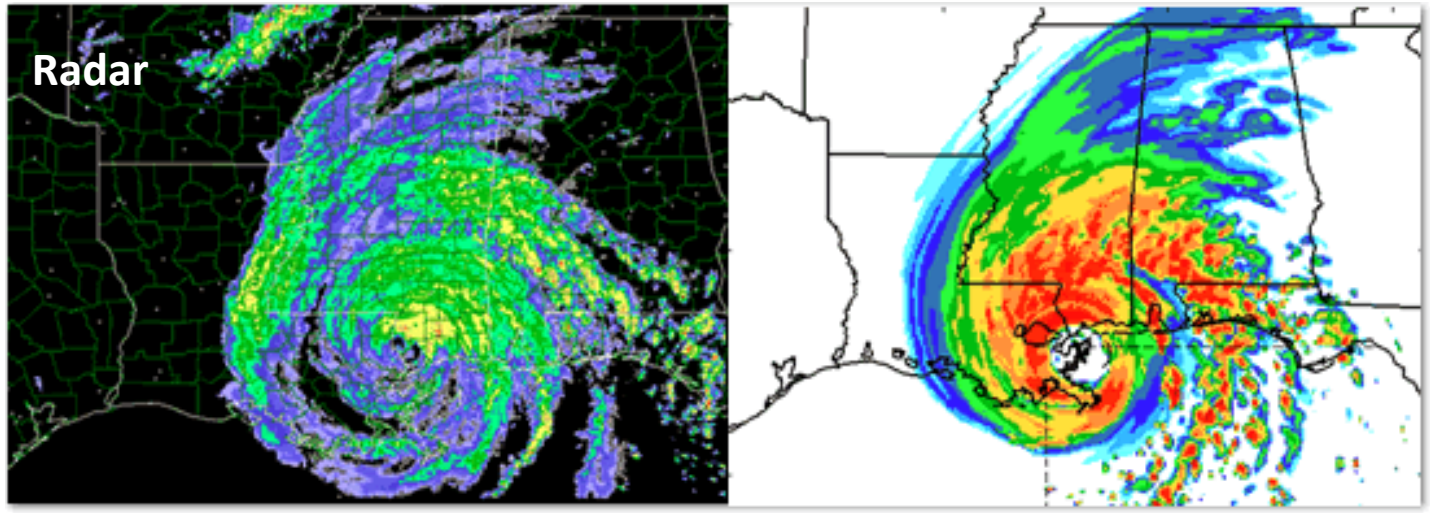
Research Goal: Define and Predict Spatial Properties of Tropical Cyclone (TC) Rain Fields



Motivation: Address Model Inaccuracies at Landfall

- Poor understanding of the dominant physical mechanisms that lead to weakening PRIOR to landfall

Hurricane Katrina (2005)



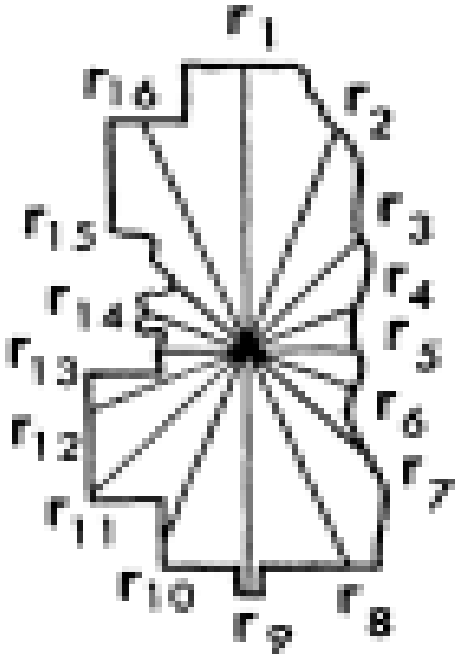
WRF-ARW (4-km)

- Understanding distribution of convection is key

Geographers Measure Space

Spatial attributes: location, symmetry, area, compactness, smoothness, fragmentation

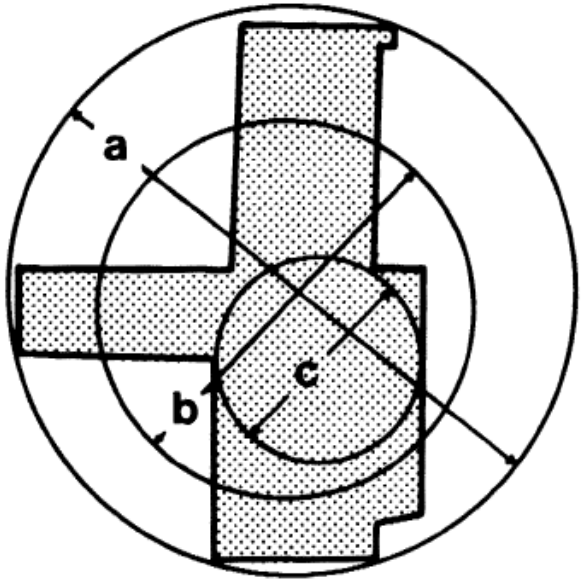
E.g., Boyce and Clark 1964; Lee and Sallee 1970; Frolov 1975; MacEachren 1985; Wentz 2000; Williams and Wentz 2008; Li et al. 2014



Dispersion about centroid (Boyce and Clark 1964)

Applications to tropical cyclones

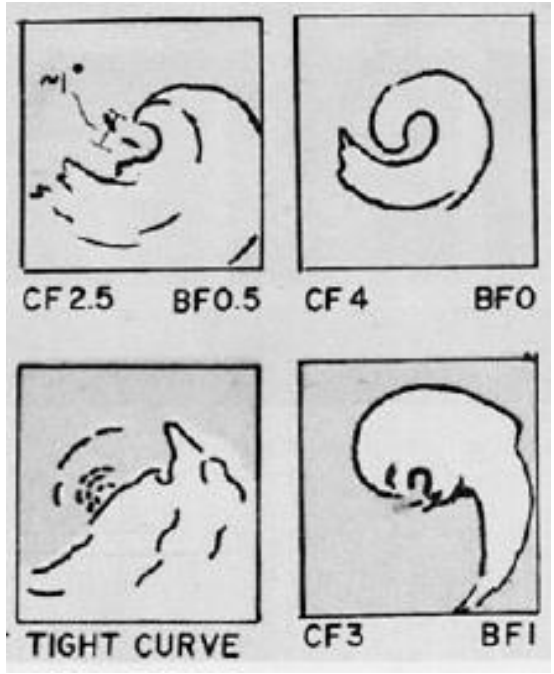
- Intensity
- Vertical wind shear
- Dry air entrainment



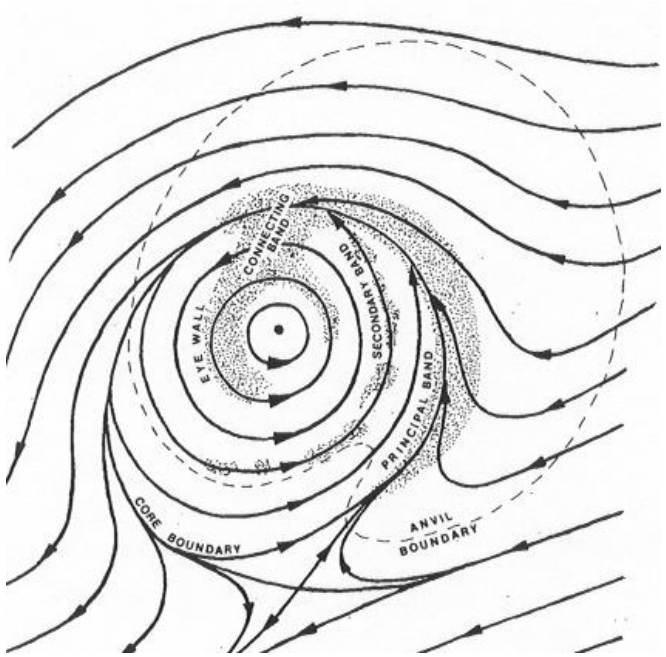
Circumscribed, same area, inscribed circles (MacEachren 1985)

TC Shapes: Indicators of Health and External Forcings

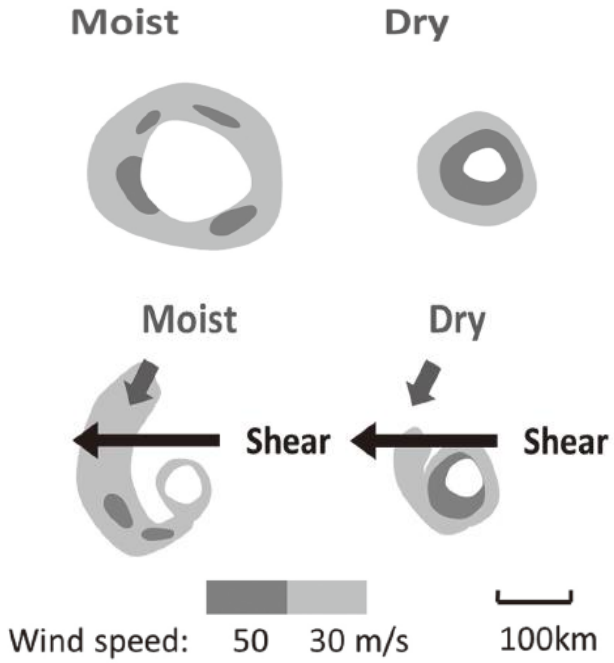
TC intensity analysis
(Dvorak 1975)



Common rainband
configurations
(Willoughby et al. 1984)

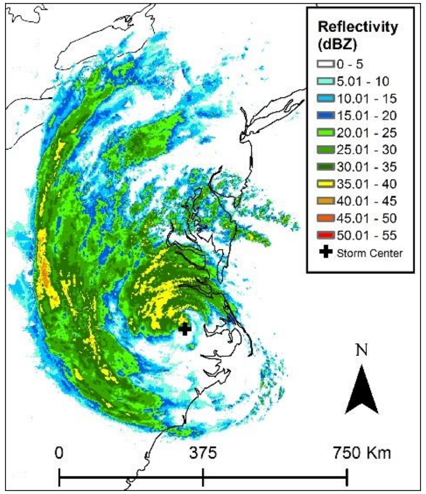


Wind shear and moisture
(Ying and Zhang 2012)



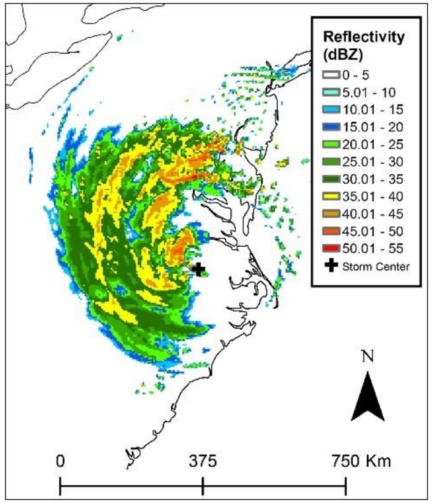
Shape Metrics: Observed vs. Simulated Reflectivity

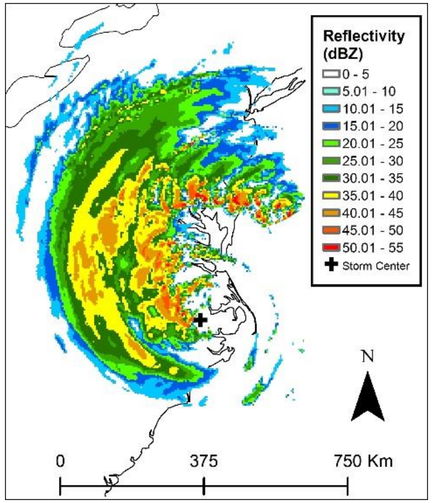
- Use shape metrics to examine spatial distribution of reflectivity
- Hurricane Isabel (2003) landfall NC, examine for 18 hours
- Reflectivity from WSR-88D network
- Simulated reflectivity Weather Research and Forecasting model (4 variations)
- 3 km grid, 3.5 km altitude



WSR-88D

Cumulus Kain-Fritsch

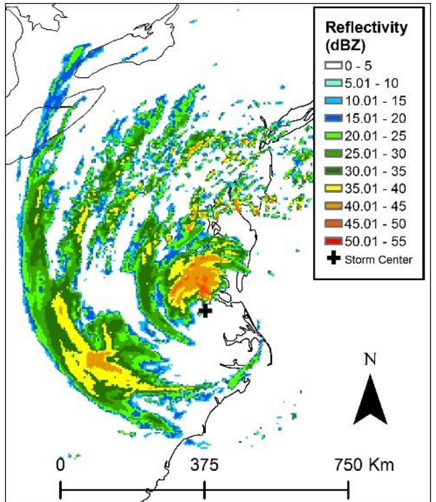


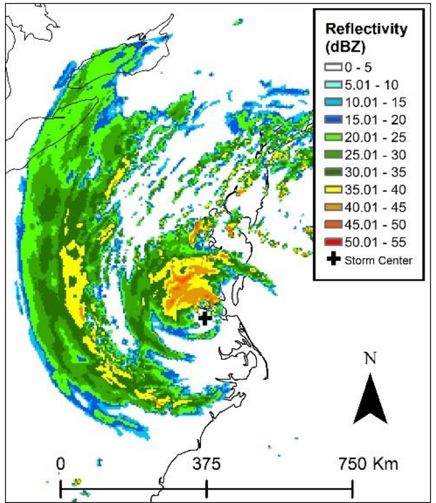


Hydrometeors

Single **Double**

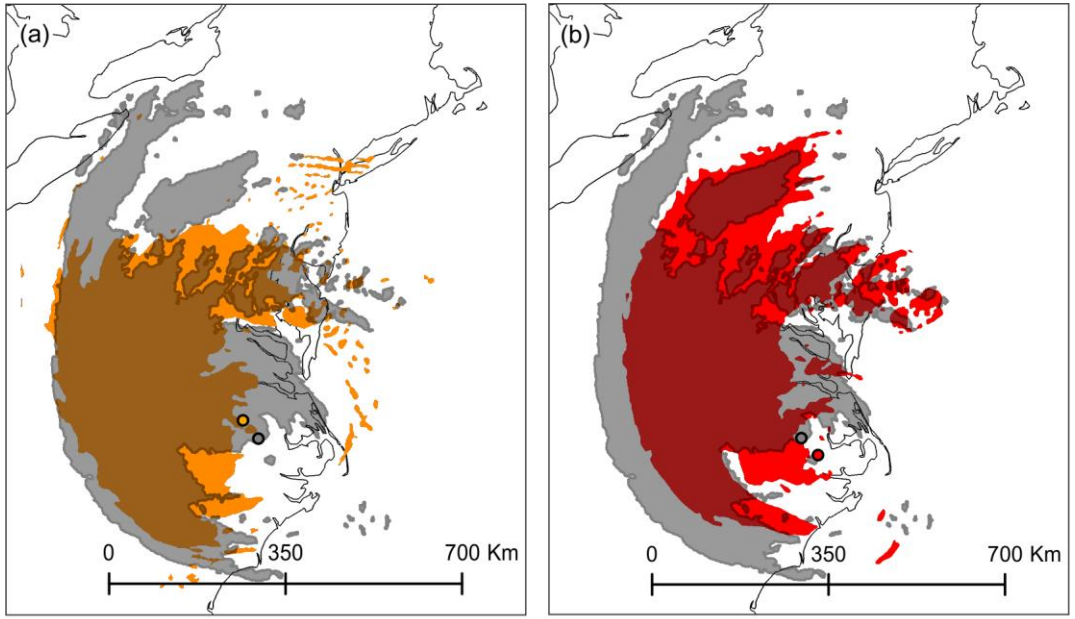
Cumulus Tiedtke

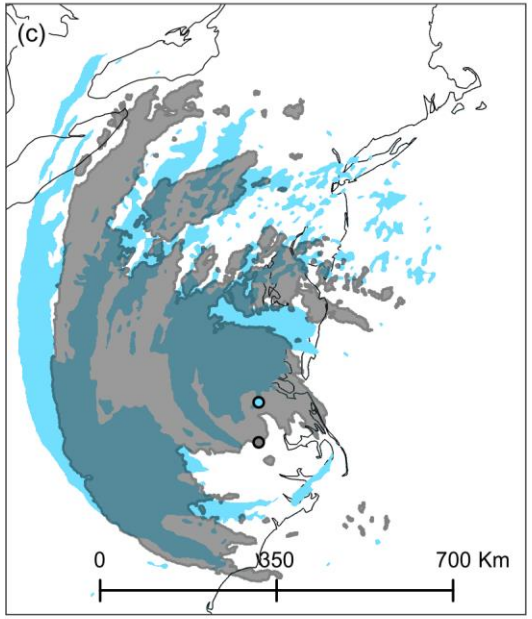


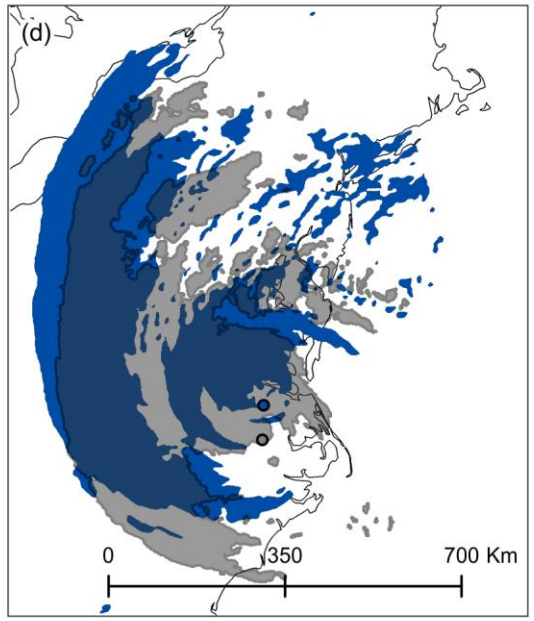


Shape Metrics

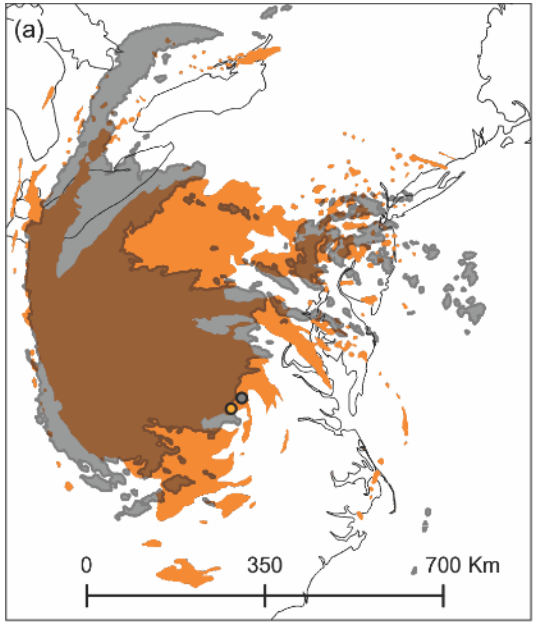
Metric	Equation	Near 0	Near 1	Literature
Elongation	$E = \frac{(Length - Width)}{Length}$	Circular	Elliptical	Matyas (2007, 2009) *Professional Geographer, J. of Appl. Met. and Clim.*
Solidity	$S = \frac{Area}{Convex\ Area}$	Empty	Filled	Jiao et al. (2012) *ISPRS J. Photogram. and R.S.*
Closure	$C = \frac{no.1°\ angles\ intersecting\ polygons}{360}$	Exposed	Enclosed	Modified arc-length from Matyas (2007) *Prof. Geog.*
Dispersion	$D = \sum_{i=1}^{NP} \frac{Area_i}{\sum_j^{NP} Area_j} \left(\frac{r_{centroid,i}}{r_{search}}\right)$	Centralized	Dispersed	Zick and Matyas (2016) *Annals of the AAG*
Fragmentation	$F = 1 - \sum_{i=1}^{NP} \frac{Area_i}{ConvexArea_i} \left[1 - \frac{NP-1}{NP+\log_{10} Area}\right]$	Cohesive	Fragmented	Zick and Matyas (2016) *Annals of the AAG*

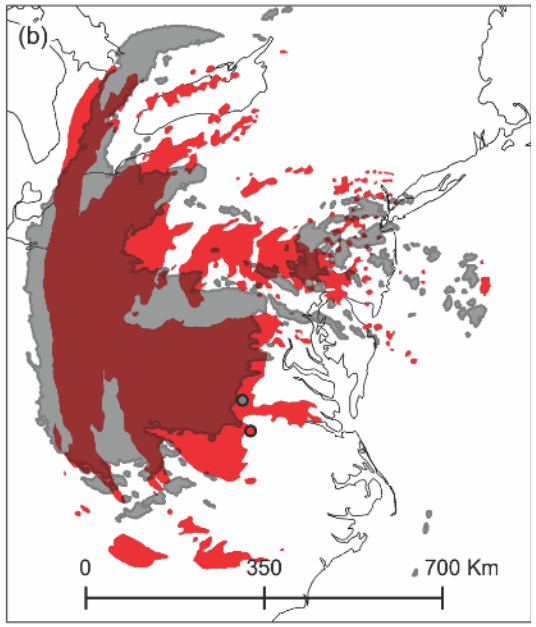


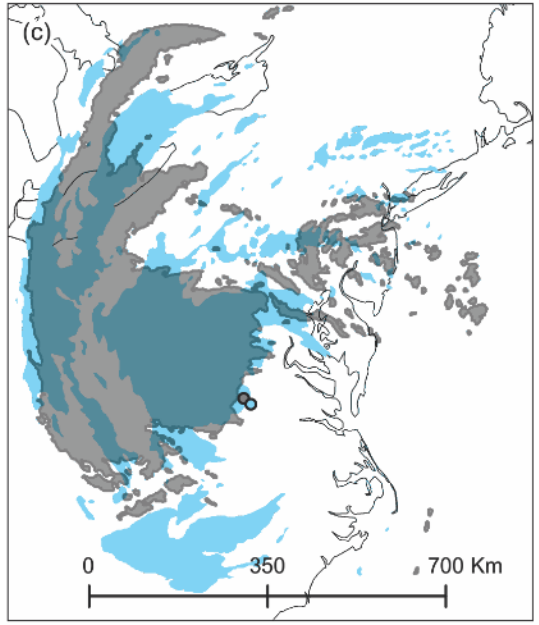


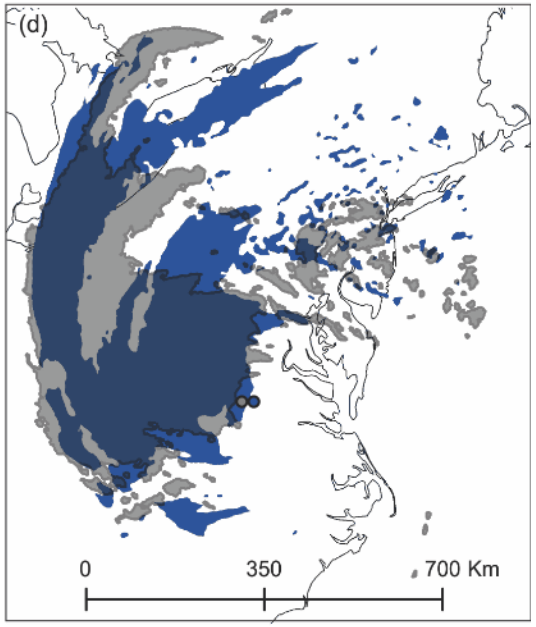


20 dBZ regions







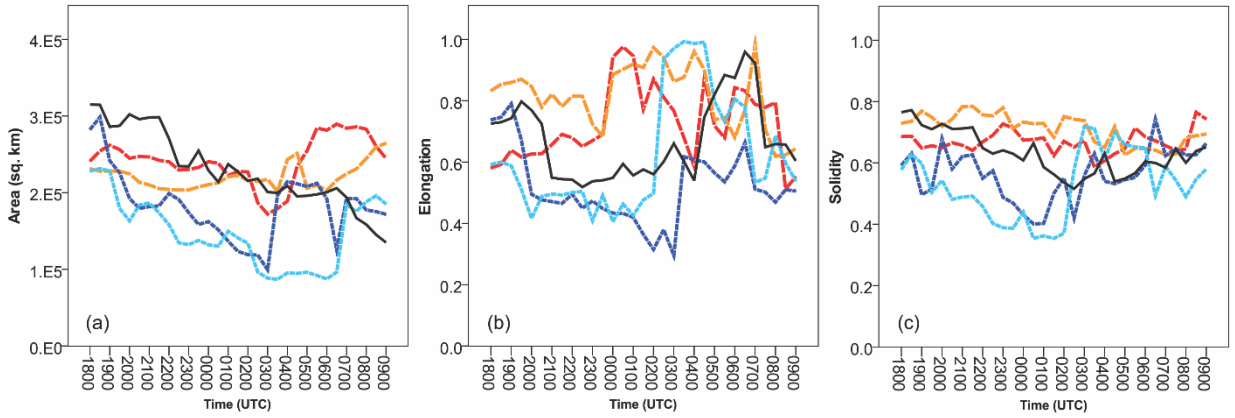


WSR KFM KFS TM TS Storm Center

1800 UTC 18 Sept

WSR KFM KFS TM TS Storm Center

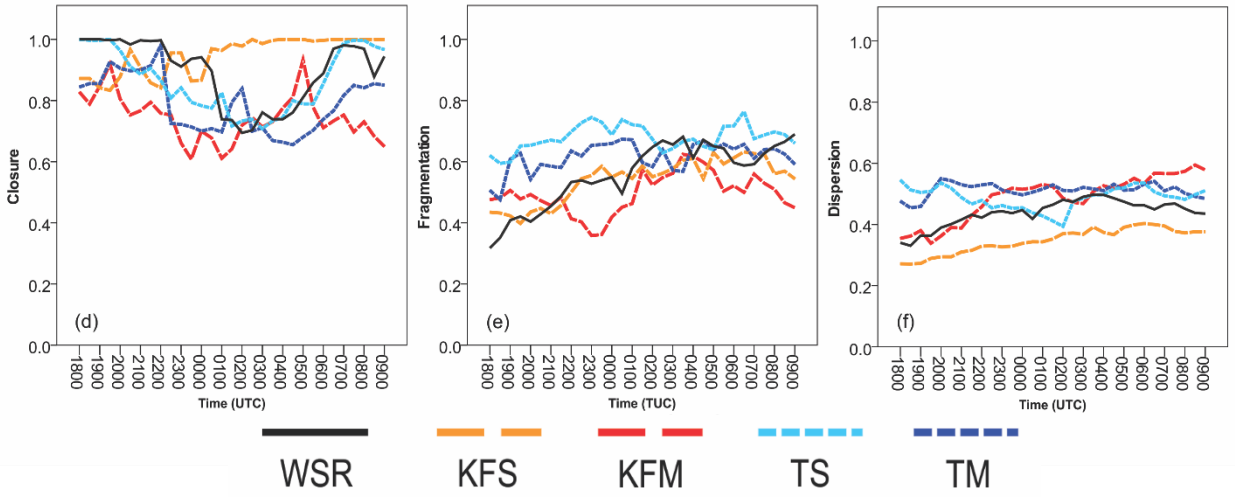
0900 UTC 19 Sept



20 dBZ Polygons (Light rainfall)

A, B, C: Largest polygon only

D, E, F: All polygons



Spearman's rank correlation coefficients between WSR polygons bounded by 20 dBZ and those produced by four simulations and time

	Area	Elongation	Solidity	Dispersion	Closure	Fragmentation
KFM	-0.183	0.100	0.180	**0.513**	**0.516**	**0.497**
KFS	-0.234	-0.097	*0.398*	**0.750**	-0.317	0.608
TM	0.204	**0.492**	0.272	0.225	**0.789**	0.232
TS	0.341	*0.434*	-0.160	-0.205	**0.844**	0.063
Time	**-0.962**	-0.226	**-0.605**	**0.680**	**-0.528**	**0.854**

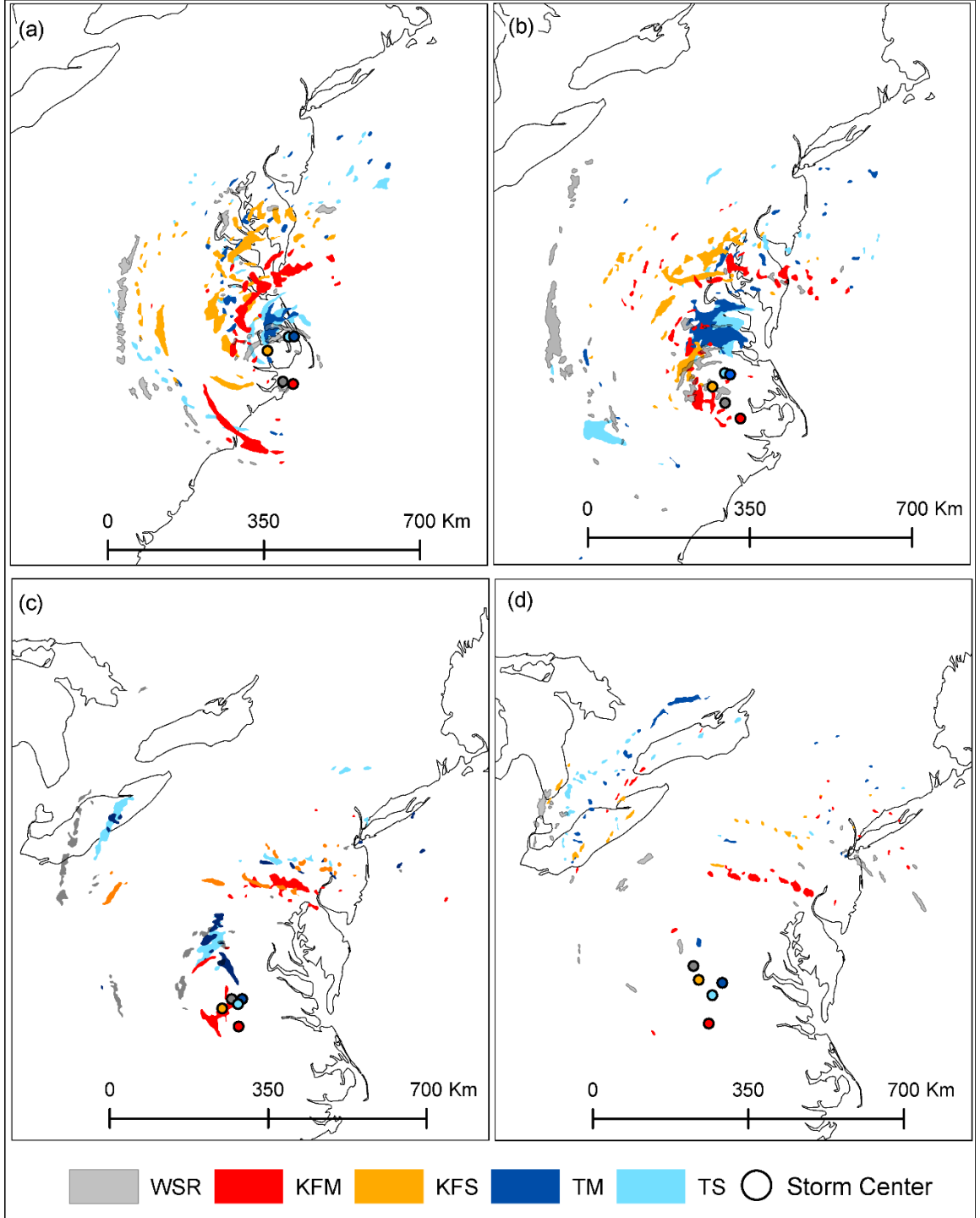
Significant at α = 0.01

Significant at α = 0.05

40 dBZ regions

1800 UTC
18 Sept.

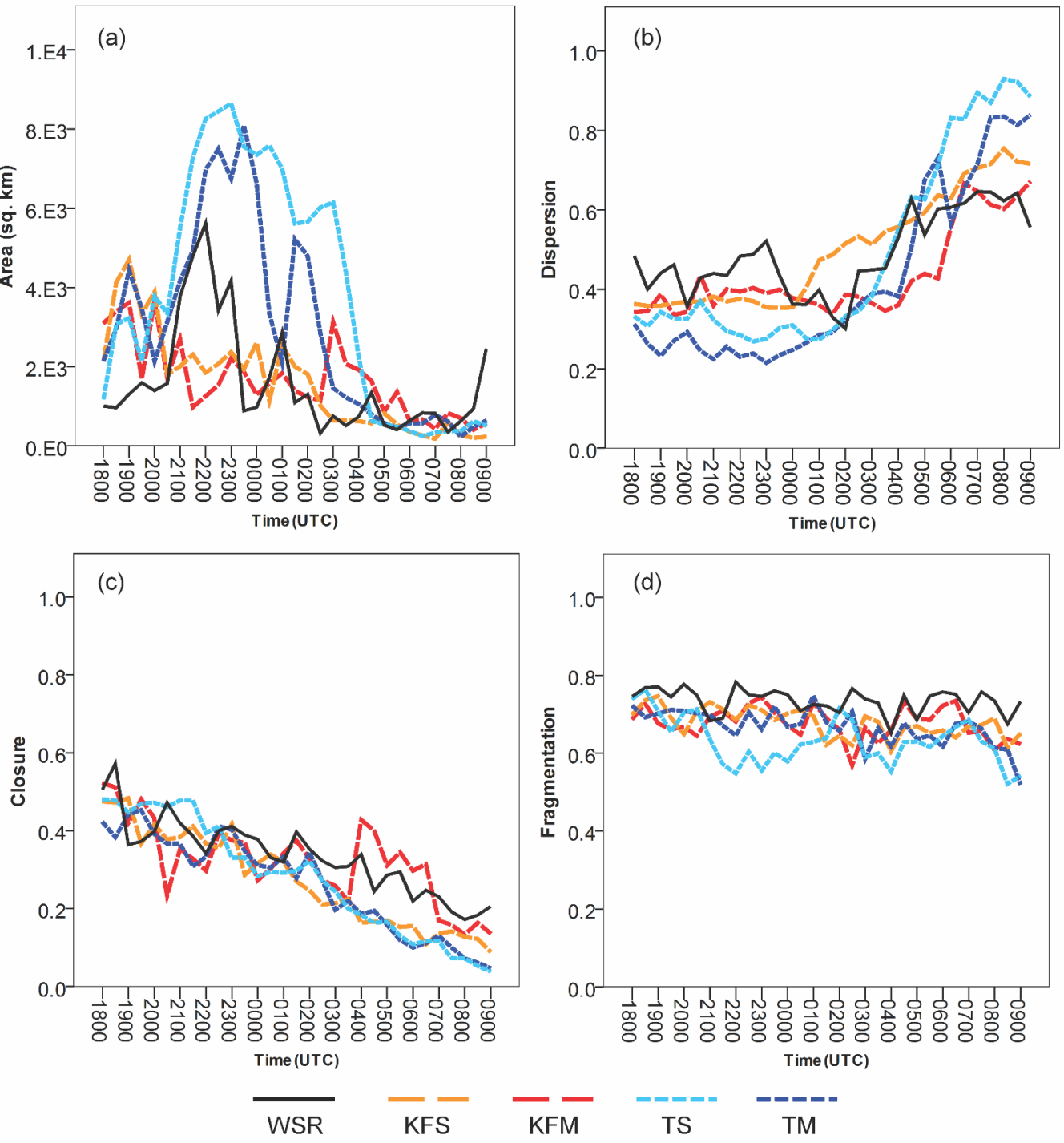
2230 UTC
18 Sept.



0400 UTC
19 Sept.

0900 UTC
19 Sept.

Matyas, Zick, Tang
Submitted to *Monthly Weather Review*



40 dBZ Polygons (Heavy rainfall)

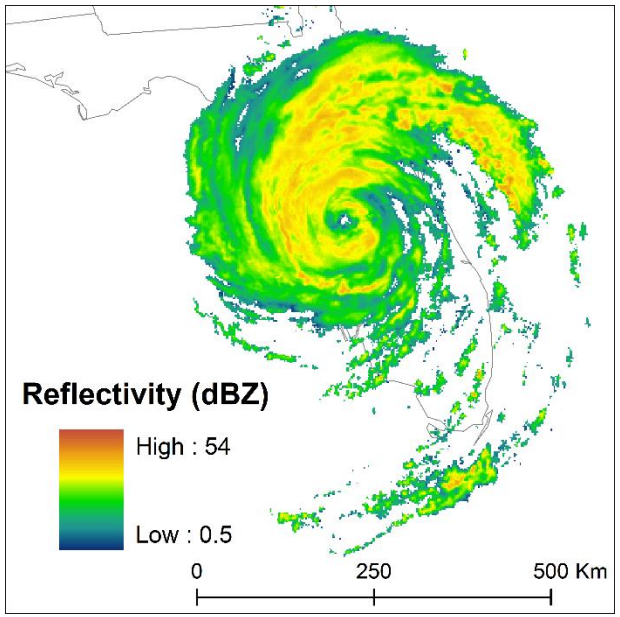
Spearman's rank correlation coefficients between WSR polygons bounded by 40 dBZ and those produced by four simulations and time

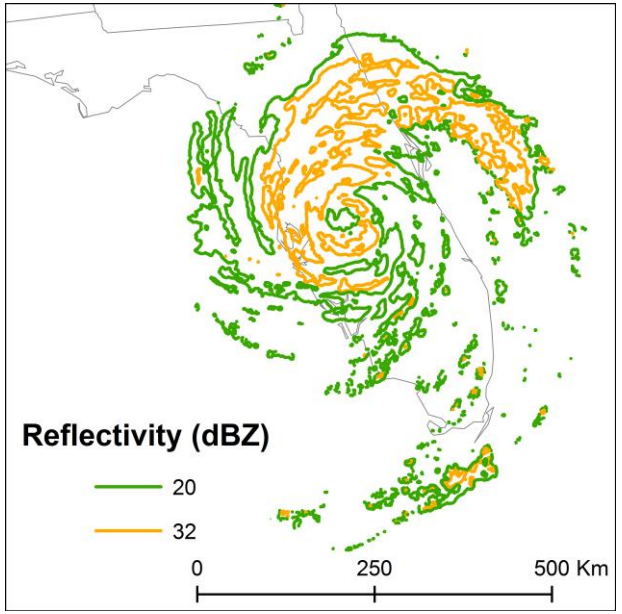
	Area	Dispersion	Closure	Fragmentation
KFM	0.27	0.151	**0.661**	**0.659**
KFS	0.146	*0.315*	**0.873**	**0.581**
TM	0.281	**0.498**	**0.907**	**0.639**
TS	*0.353*	**0.565**	**0.915**	**0.709**
Time	**-0.416**	0.104	**-0.934**	**-0.728**

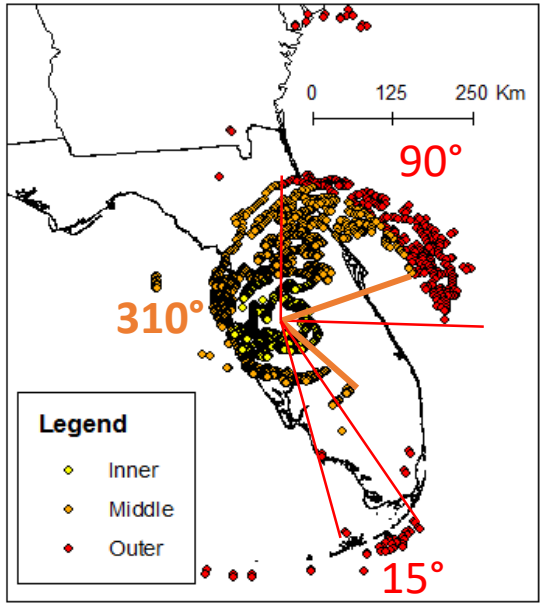
Significant at α = 0.01

Significant at α = 0.05

Another Way to Measure Closure/Exposure







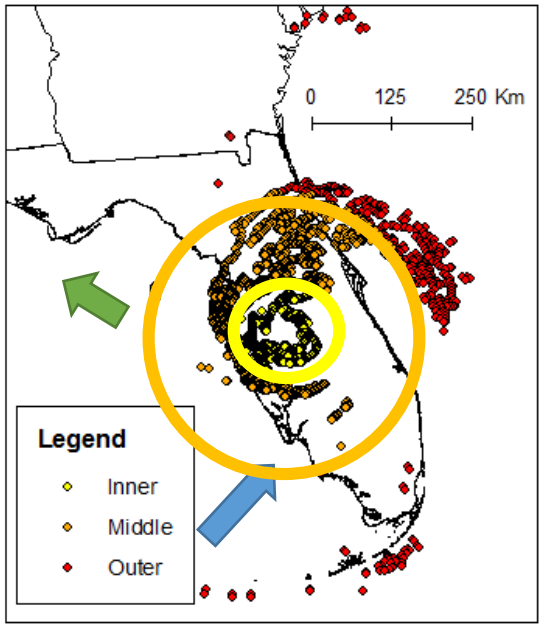
1. Contour 20 and 32 dBZ regions
2. Intersect contours with radial lines every 1°
3. Record intersections as points
4. Sweep over radial distance (storm size)
5. Identify angles where points are found
6. Filled areas must be at least 10° wide
7. Repeat every 10 minutes
8. Relate time series to vertical wind shear, storm motion

Radius of Outermost Closed Isobar: 370 km Middle region: ½ ROCI Outer: ROCI + 100 km

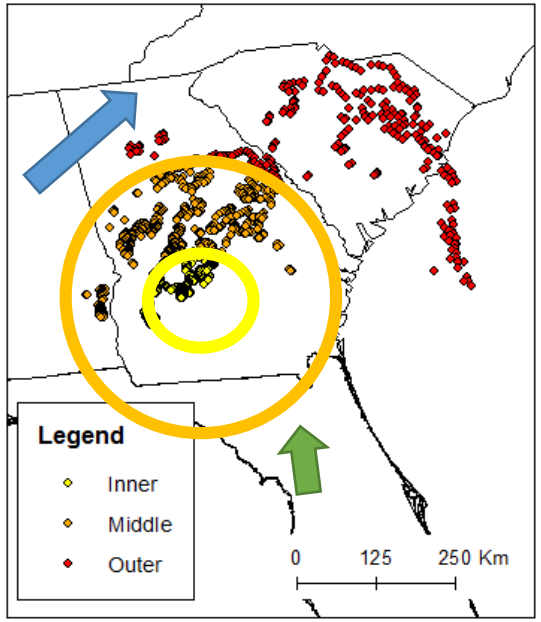
32 dBZ Contour Intersections

Storm Motion

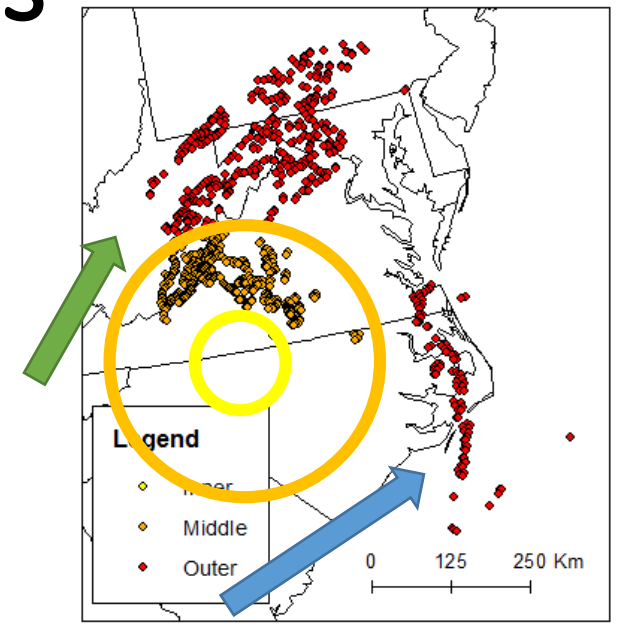
200-850 hPa Shear



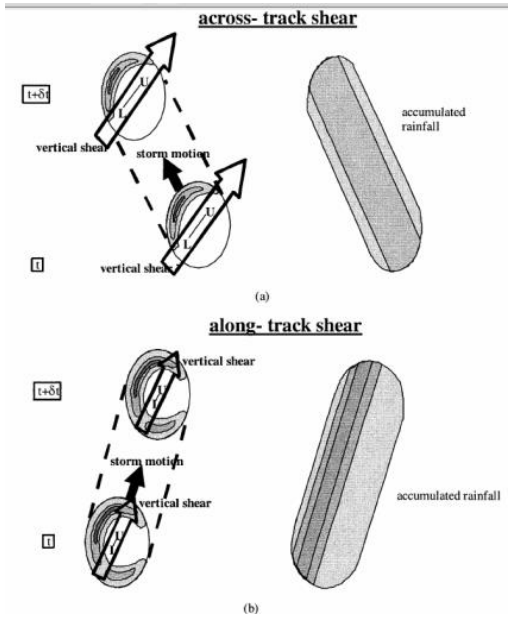
9/26 1500 UTC



9/27 1500 UTC



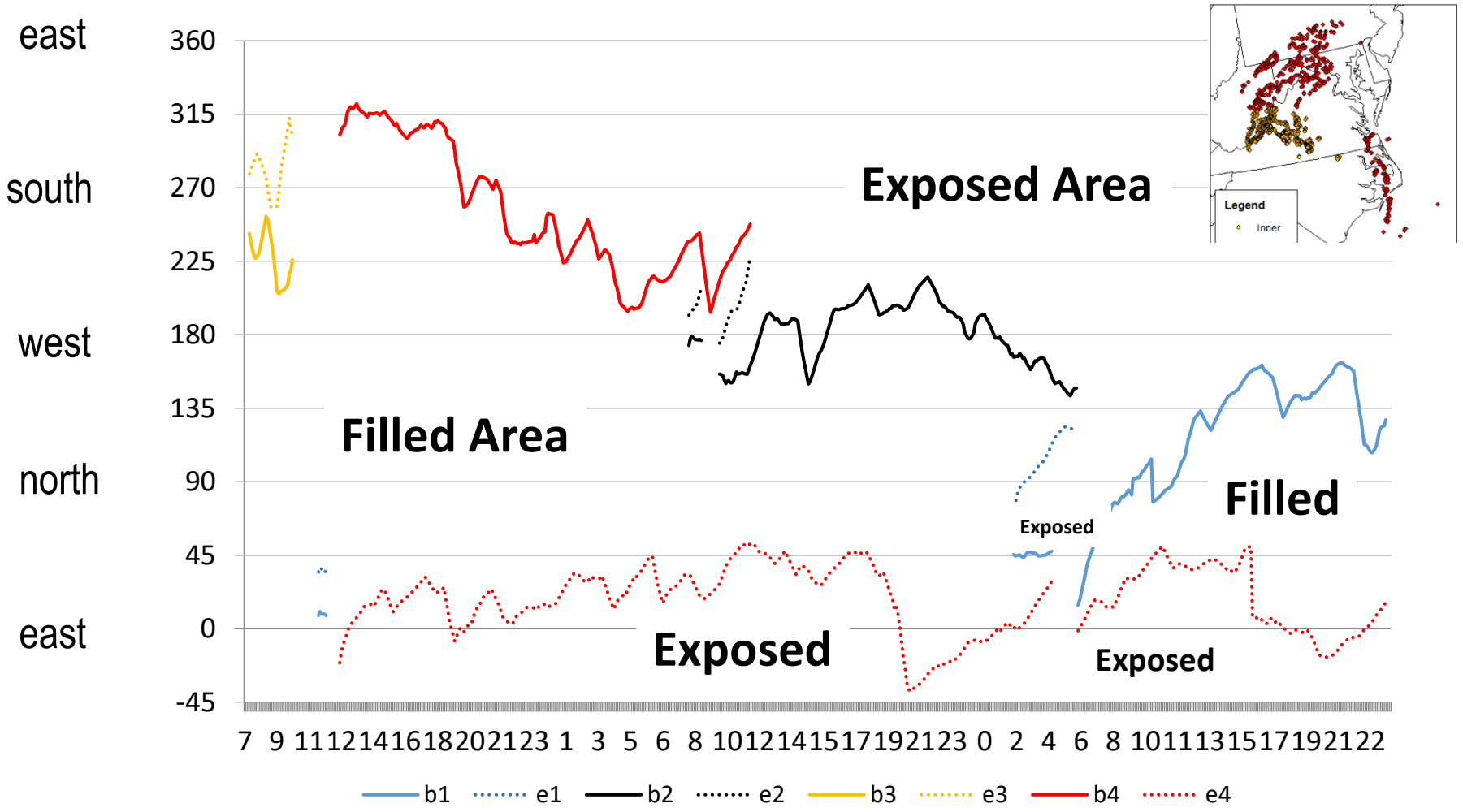
9/28 1500 UTC



Rogers et al. (2003) Fig. 21

- Exposure of middle and inner regions increases with time
- Rainfall focused downshear left, ahead of storm center
- Relationship corresponds well to published research

32 dBZ Middle Region (74-222 km)



- Middle region shrinks in coverage from 360° to 90°
- Core becomes exposed from southeast to west
- Rainfall remains north and east of center
- Using storm size worked well to subdivide radially

Summary

- Perspective from Geography: need to better predict spatial patterns of rainfall
- Shape metrics: accurate calculations require high-resolution data
- Implications for better predictions of TC rainband structures
- Comparisons with model output
- Applications for other weather systems
- Future directions: 3D shapes with tracking over time

Contact: Matyas@ufl.edu

Technical Notes about Fast Gridding

- We frequently perform the gridding algorithm on radar products.
- Many applications can create single/multiple radar gridding products.
- Our goal:
  - Performance
  - Performance
  - Performance
- Some lessons we learned from development of multi-radar gridding algorithms.

Contact: jtang8756@ufl.edu

Strategy Comparison

- Two-step method vs one-step method
  - Two-steps: ① create individual grids for each radar ② combine grids into a large mosaic
  - One-step: Directly put all gates on the final mosaic
- Interpolation:
  - Linear interpolation
  - Angular interpolation
- Architecture
  - Sequential
  - Parallel (single node)
  - Distributed

Single Radar or Multi Radar

Single radar/Mosaic

1. Assume the radar at $(0,0,z)$: Azimuthal Equidistant Projection. North: Y-axis
2. Combine:
    1. Inverse project back to the Earth Spheroid (lat/lon)
    2. Forward project to a new geographic projection (e.g. Lambert Conformal Conic)
    3. Resampling/rectify on new grid

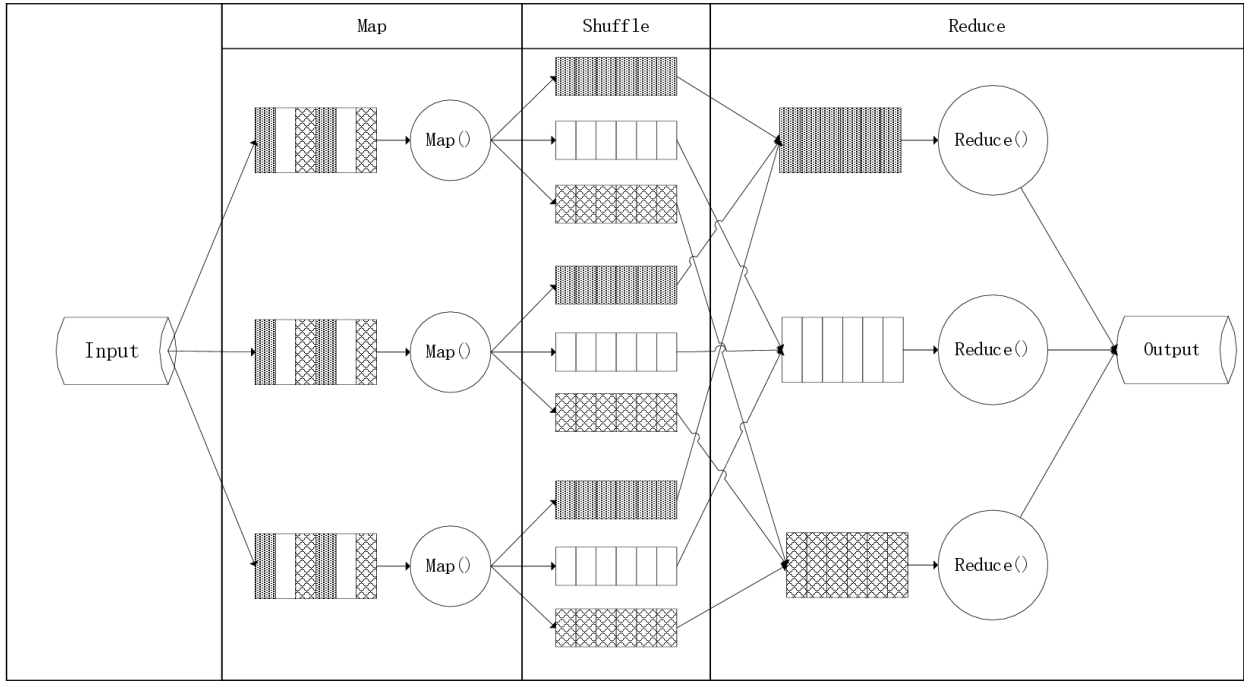
- Problem
    - ① Synchronize cells in overlapped area
    - ② Expensive computations on inverse/forward projection
    - ③ How to keep interpolation consistent due to geographical distortions.

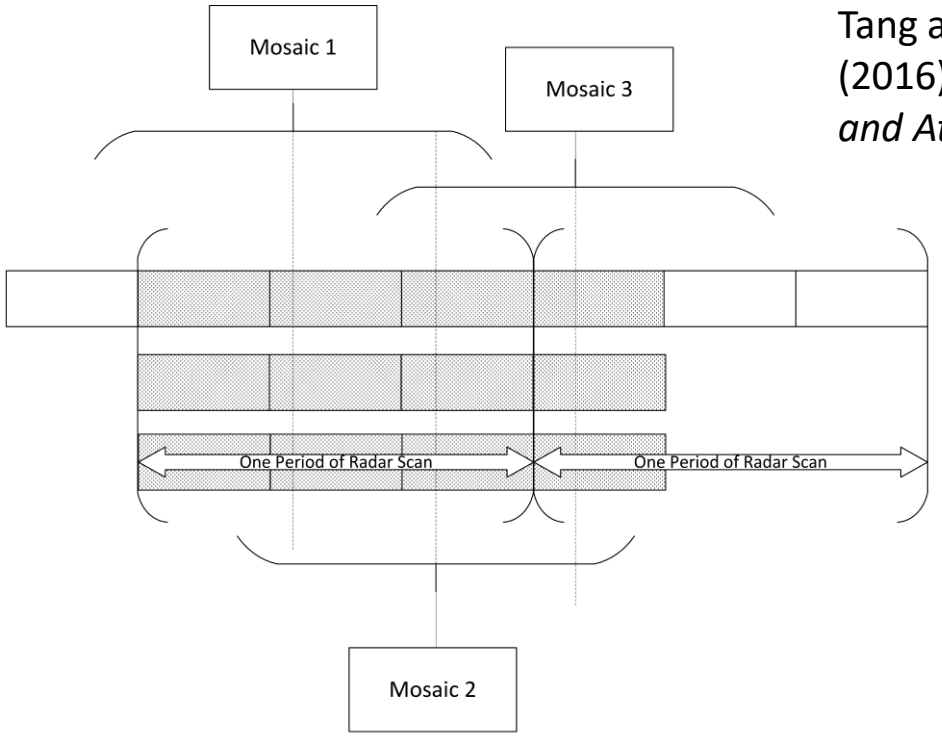
Multi-radar

1. Gate → **Spheroid** → Geographic Projection
2. Gate ~~→ **Spheroid** →~~ Geographic Projection
3. Synchronize multi-radar samplings (e.g. temporal weight, take newest)

- Problem
    - ① Large memory demands
    - ② Complicated software architecture
    - ③ 0-deg Azimuth is not along Y-axis (geographic convergence
    - ④ Beams may scale and curve on the map (geographical distortion)

First Try: Multi-radar using MapReduce





Tang and Matyas (2016) *J. Ocean and Atmos. Tech.*

Key: (Time, Elevation, Azimuth, Gate) → Grid Index (T, i, j, k)
Value: Radar Moments

Implementation: Scala Language and Apache Spark
Distributed Platform: 16 nodes, 4 CPU/node

NetCDF-java *improperly* structures super-resolution scans (0.25deg) in Level-II
Duplicate data to synchronize over time
A conformal geographic projection can save time, as no forward projection (lat/lon to x/y) required, slightly up-scaled

Getting Inspiration

Radx

- Multi-threading
- 3D linear interpolation
- Simple design

Map Reduce

- Highly scalable
- No shared data between each “worker”

Py-ART

- Propagation strategy: “map_gate_to_grid”
- No need to lookup for “nearest” gates.

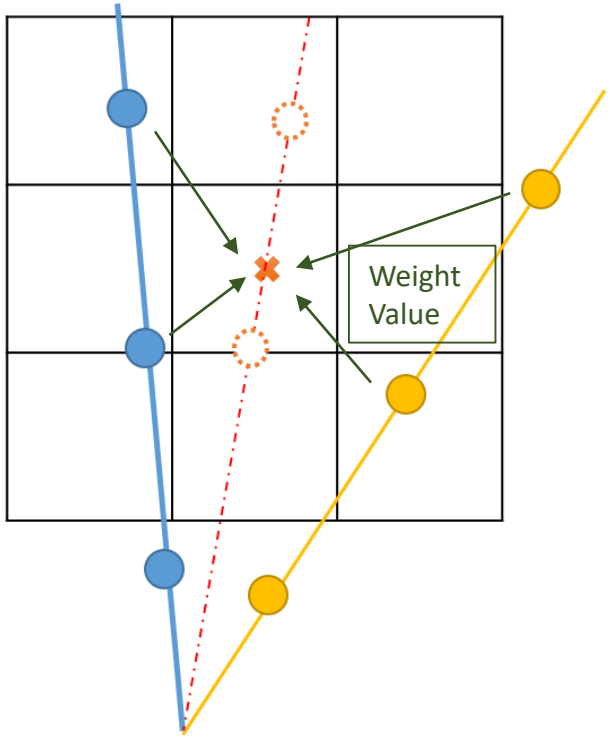
WDSS-II

- Elevation/Gate angular interpolation
- Producer-Consumer Pattern
- Caching

New Gridding Algorithm

Multi-radar gridding on earth spheroid:

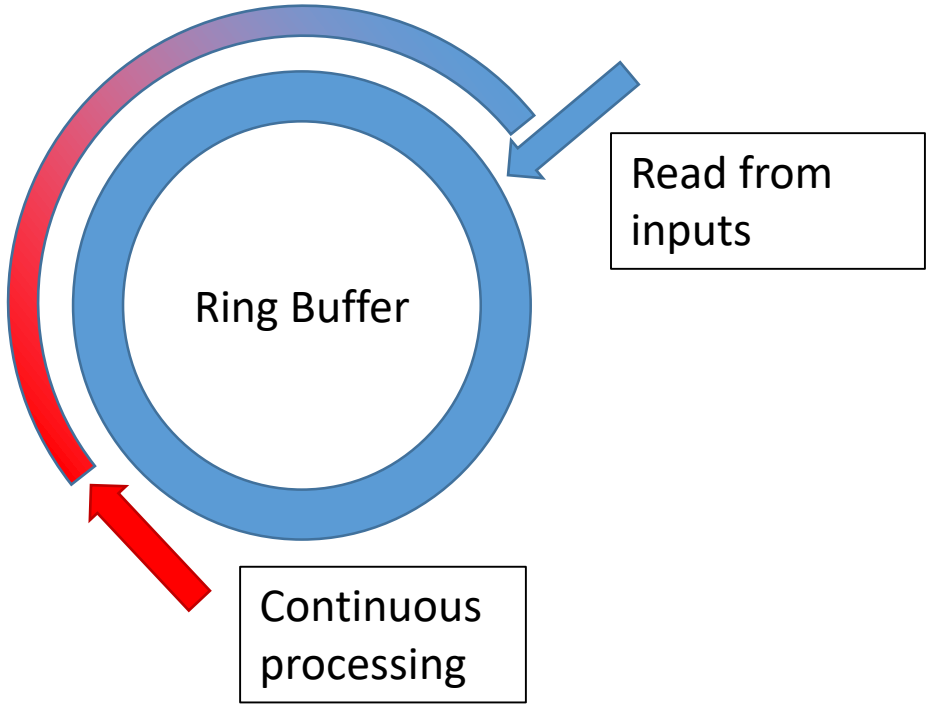
Gate → **WGS 1984**



Combined angular, distance, and temporal interpolation

Propagation strategies

Cache

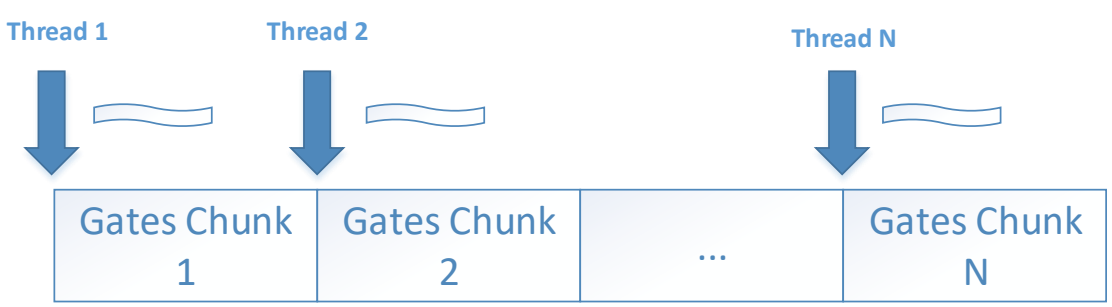


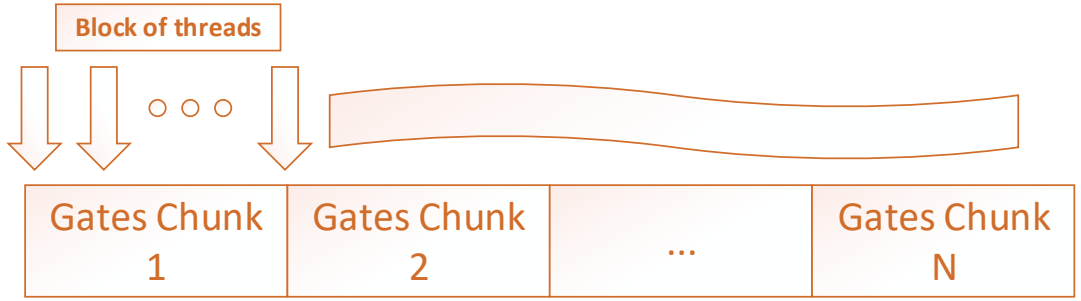
Asynchronous processing using a ring buffer

Multi-threading parallel

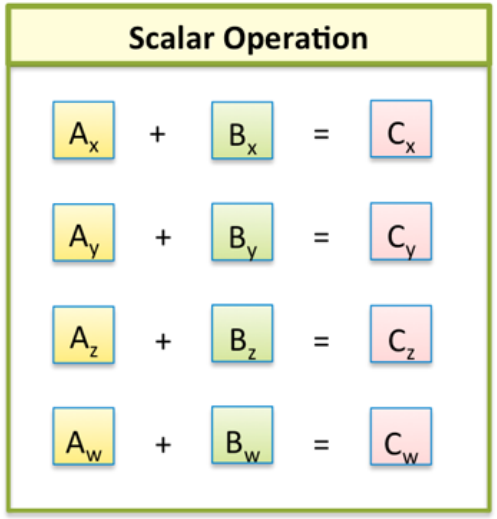
Advanced Performance Tuning

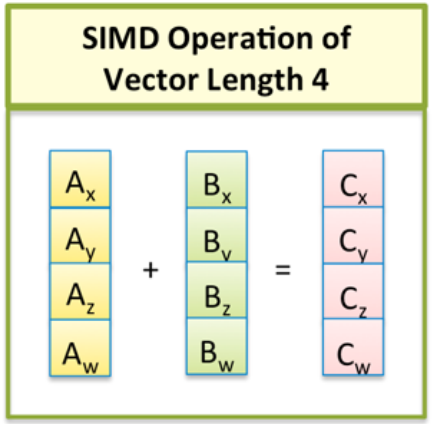
- Avoid Cache Missing



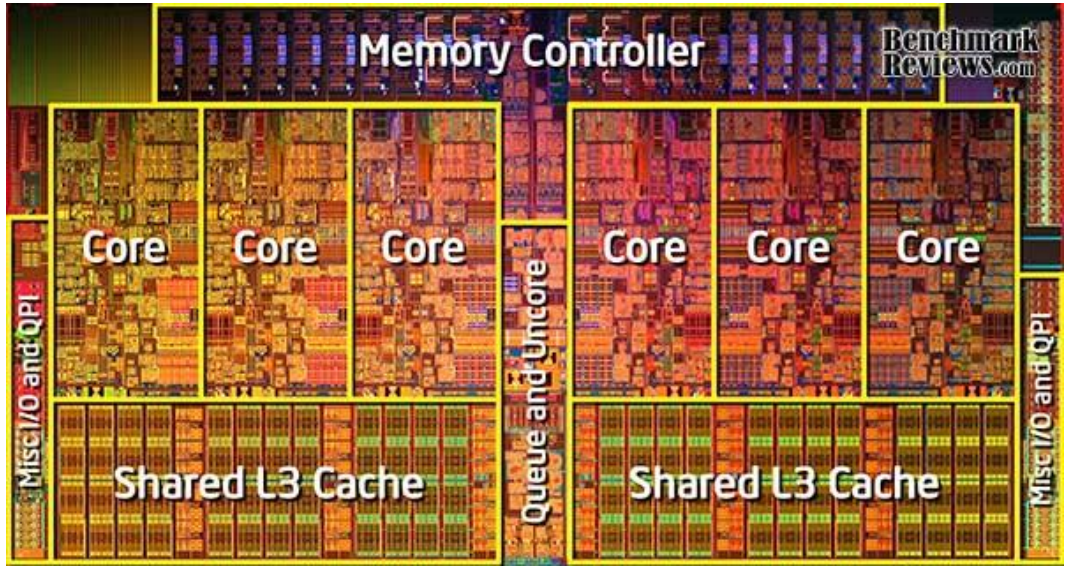


- Vectorization





Intel® Architecture currently has SIMD operations of vector length 4, 8, 16



Performance and Profiling Remarks

- <0.01 sec to create a single CAPPI layer for a single radar
  - Two Xeon 2650v3 CPUs, 128G Memory, SSD Raid 6
  - 0.005° x 0.005° x 250m (60 levels)
- <5 minutes to create nationwide mosaic
  - Lak (2013)* reported a 32-node configuration for similar resolution.
- Easy to migrate to GPU.
  - 2 minutes to create nationwide mosaic on my home PC (i7, 32G, GTX 1070)
  - No advanced tuning

*Lakshmanan V, Humphrey T W. A MapReduce technique to mosaic continental-scale weather radar data in real-time[J]. IEEE Journal Of Selected Topics In Applied Earth Observations And Remote Sensing, 2014, 7(2): 721-732.

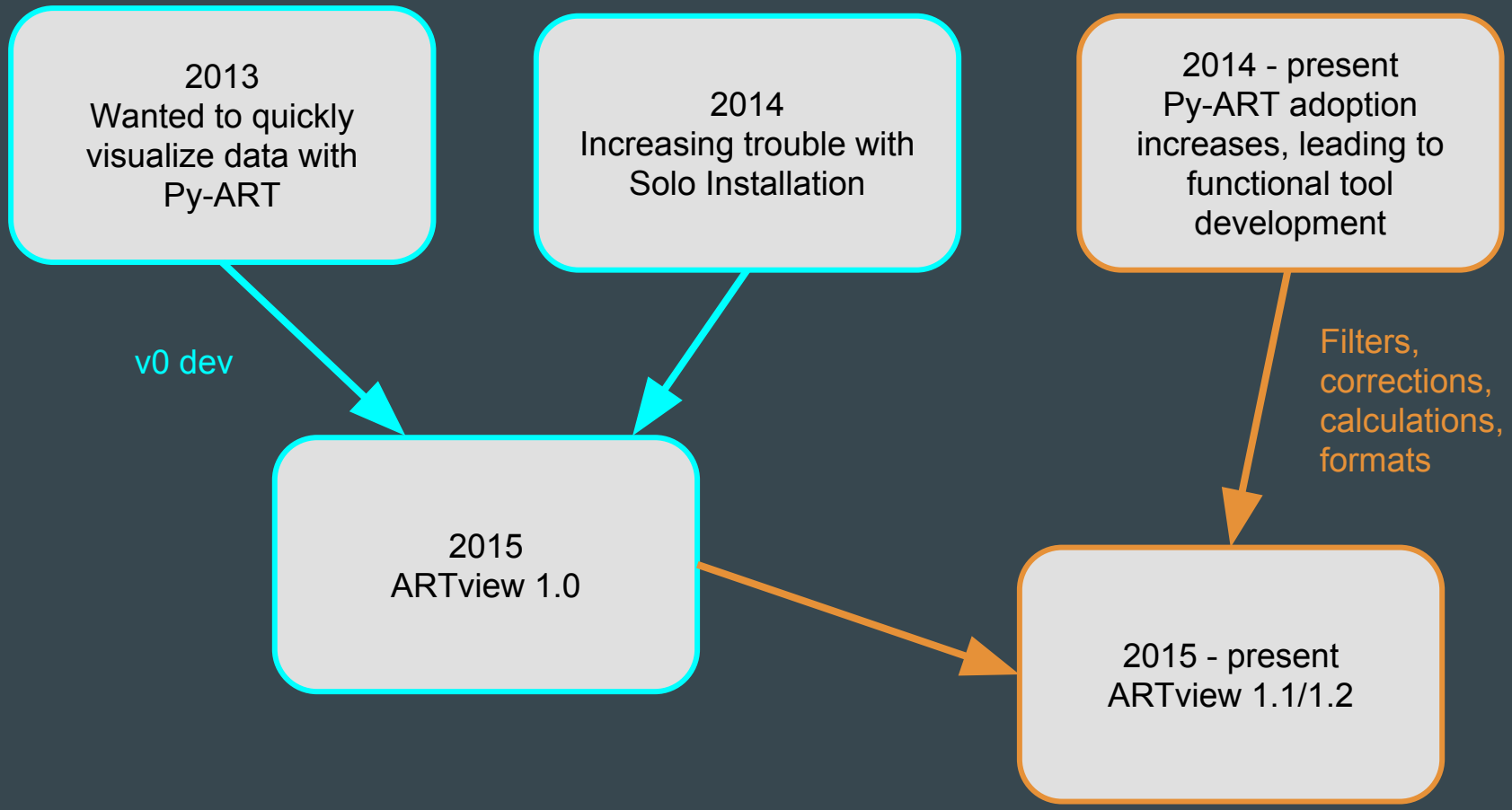
Contact: jtang8756@ufl.edu

ARTview: Radar analysis GUI

•••

Nick Guy
Not developed as part of The Climate Corporation work

Development History



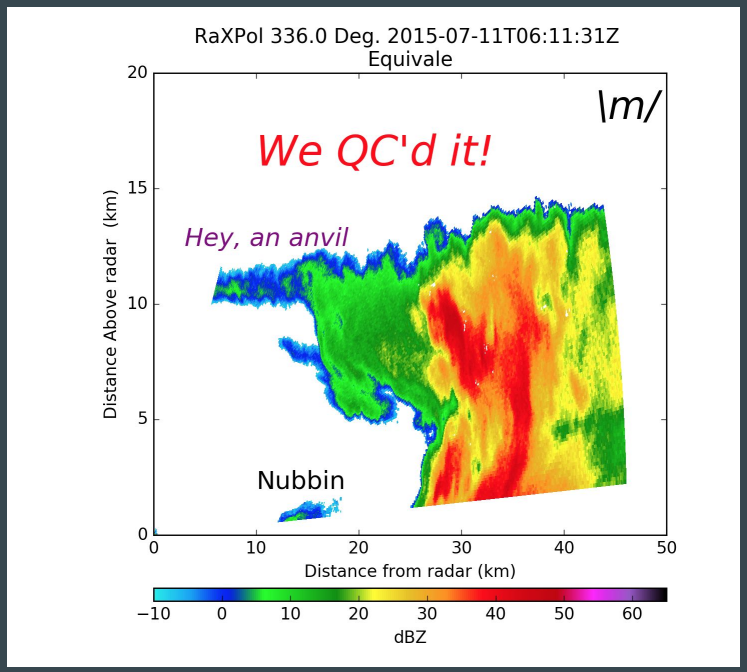
Contributions from others critical!

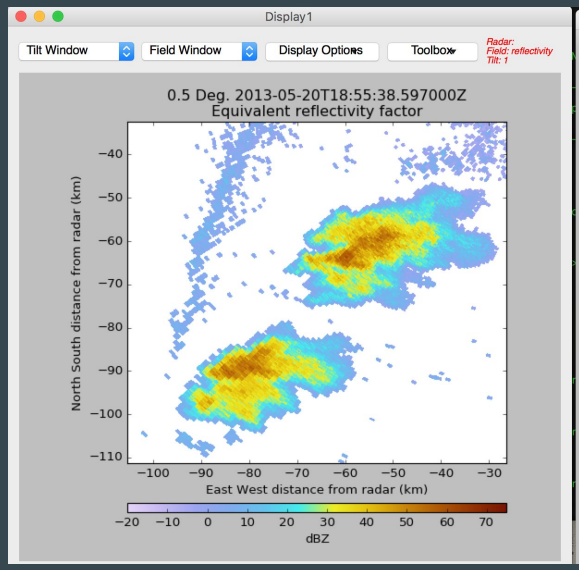
Design and Philosophy

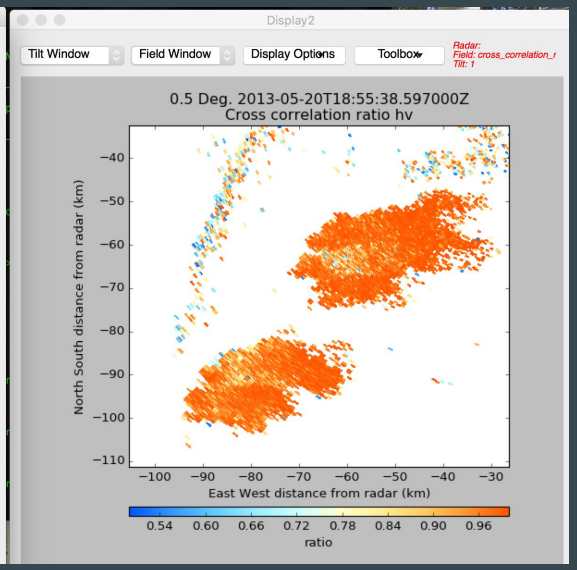
- Leverage Py-ART input/output/data model
- Employ robust, modern GUI interface
- Modularity
- Design the environment to be extensible
- Documentation, including video tutorials

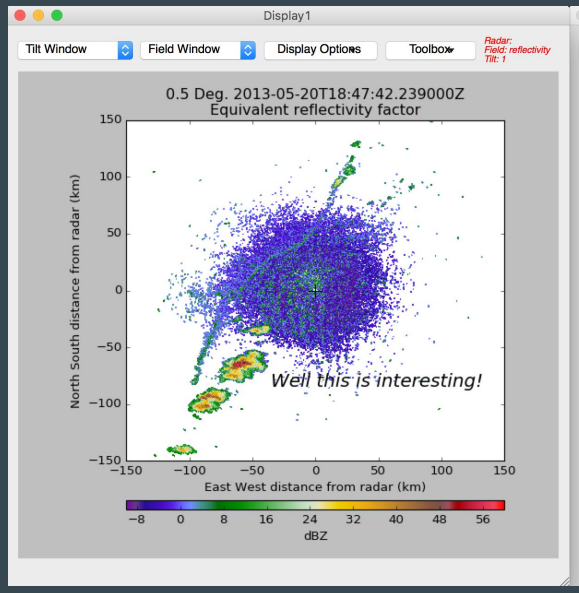
- Allow quick exploration of data, make it “approachable”
- Data and analysis visualizations
- Provide interface that is both familiar to Solo users, as well as easily navigable by all to Py-ART tools and common operations

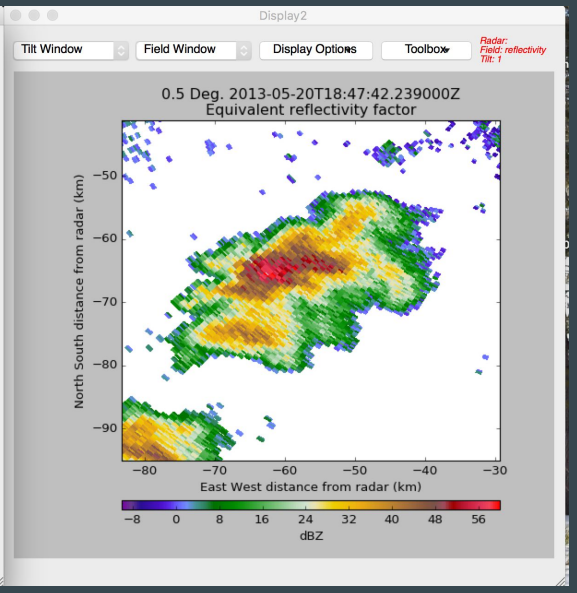
Visualizations



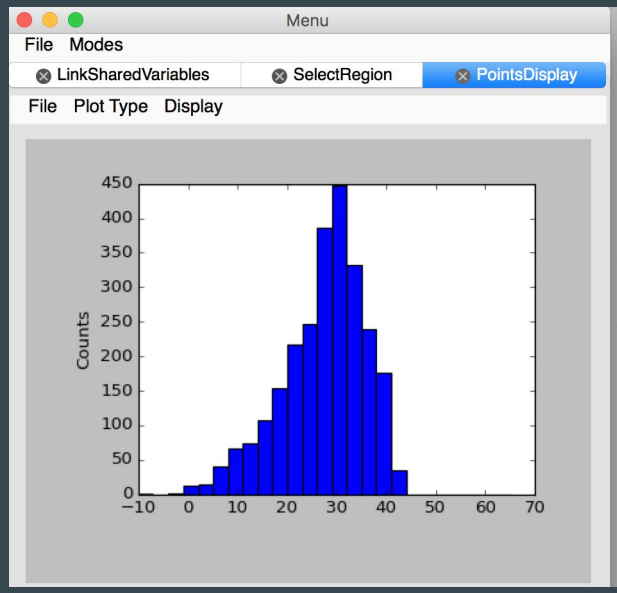


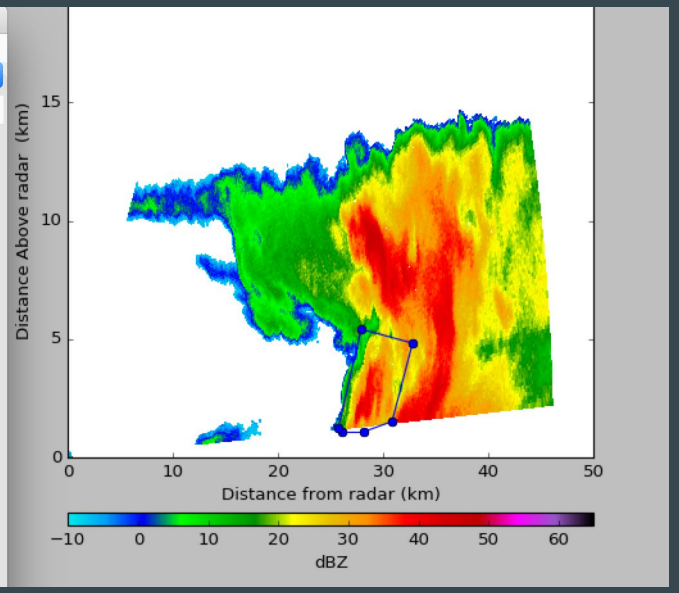


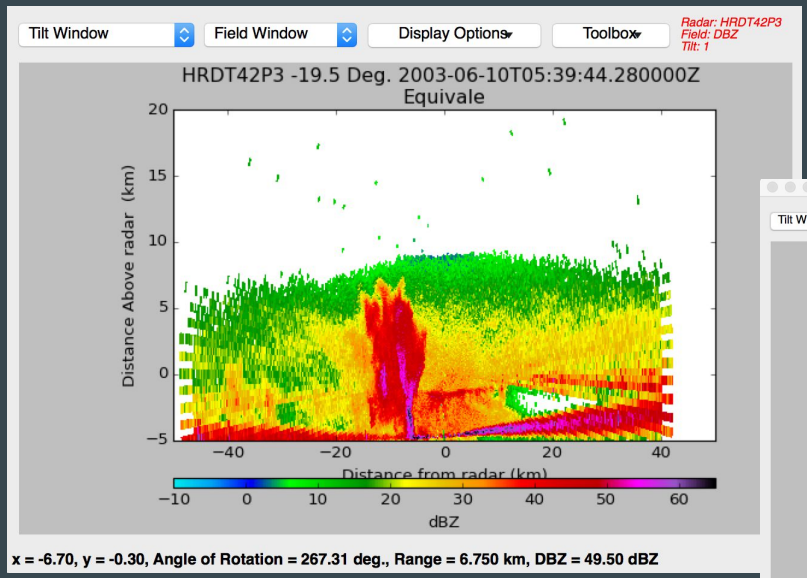


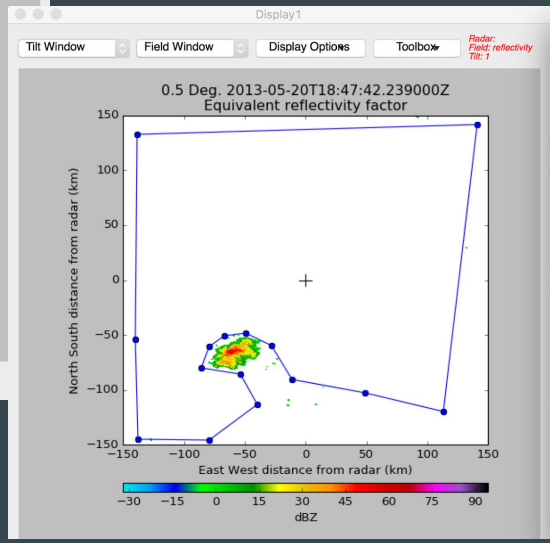


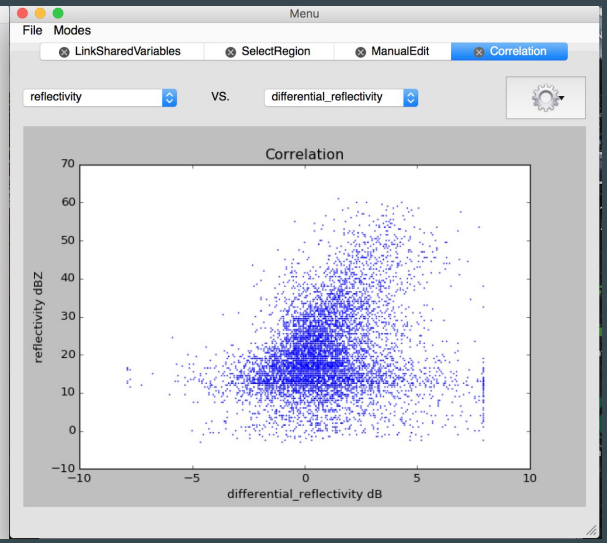
Data Exploration





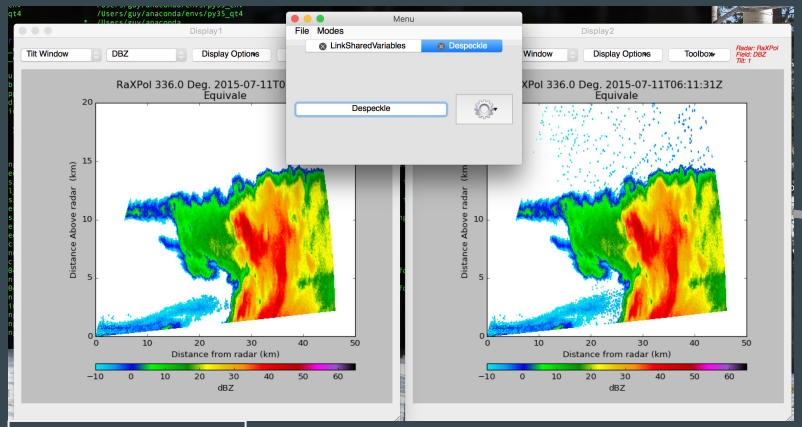




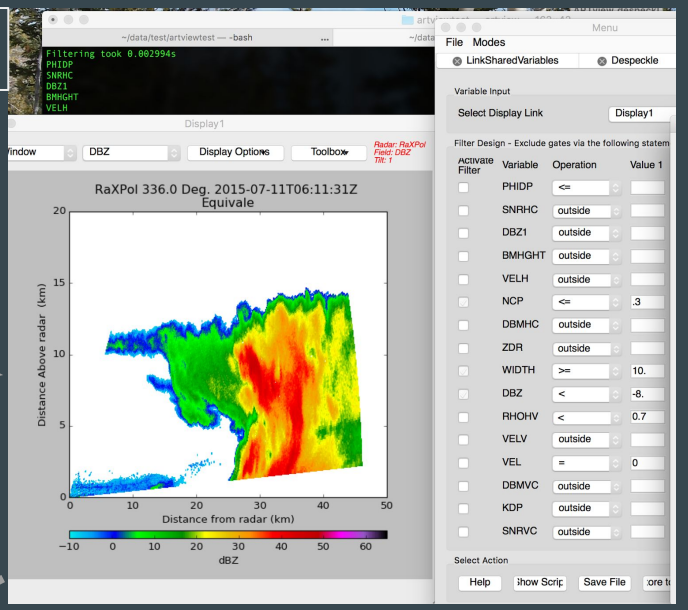


Data Editing

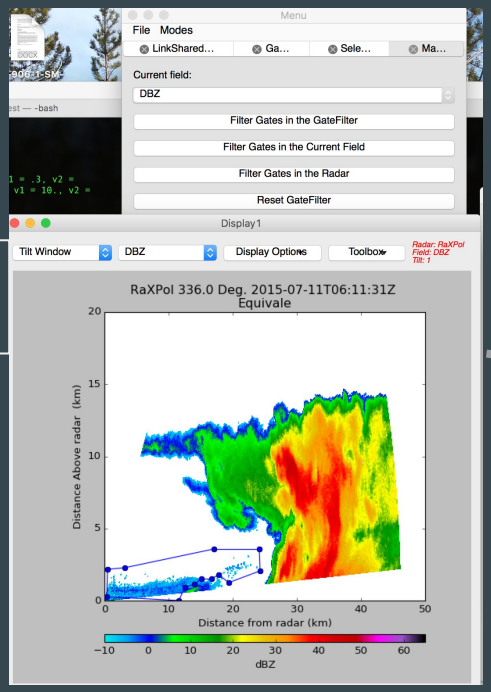
Despeckle

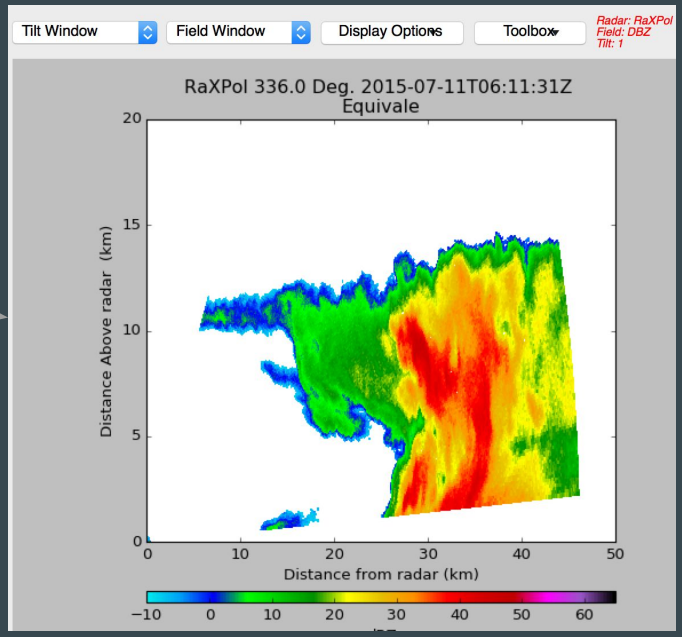


Filter



Polygon Removal





State of ARTview

- v1.2 Released
- Installable via conda package manager
- Currently nearing the end of v1.3 development cycle

- Many Py-ART functions are included
- Can query individual points or selectable polygons of data
- Create images, add text
- Easy file navigation

What's Next for ARTview?

- v1.3 software release mid-summer
- Short course at the 2017 American Meteorological Society Radar conference along with other open source software packages

- Proper testing
- Increase coverage of Py-ART functions
- Include user-requested features (we're working on it!)
  - Batch functionality
  - Cross-sections
  - etc.
- Improved user interface
- Better documentation

- Roadmap integration with LROSE

Final thoughts

ARTview is completely unfunded, it's a labor of love of the developers because we like coding and radar data.

We hope that people find this tool useful and contribute their ideas.

The weather radar community is in a unique spot, with a boom in open source projects and many people with the skill set to contribute.

Thank You!

?

A limited view of other ongoing projects

Airborne Weather Observations Toolkit
http://nguy.github.io/AWOT/

PyDisdrometer
https://github.com/josephhardinee/PyDisdrometer

CSU RadarTools
https://github.com/CSU-Radarmet/CSU_RadarTools

PyTDA
SingleDop
MultiDop
PyBlock
wradlib