Building Python-based operational systems for prediction of atmospheric processes

Don Morton
Arctic Region Supercomputing Center
University of Alaska Fairbanks
Boreal Scientific Computing LLC

Outline

- Evolution and overview of operational activities
- Python – the glue that has held it together and enabled deployment of complex, evolving systems
- Some examples
- Ambitions

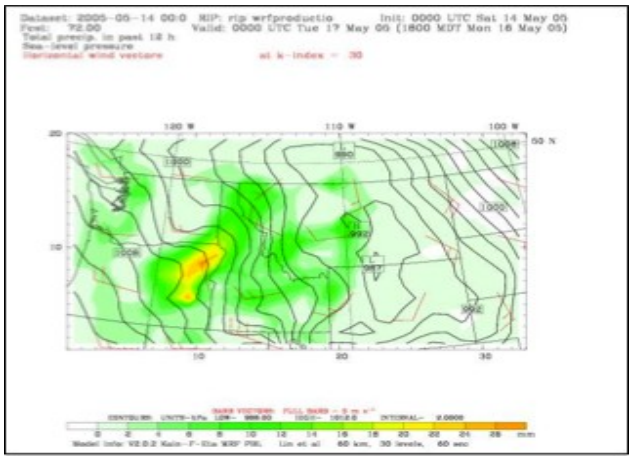
UAF UNIVERSITY OF ALASKA FAIRBANKS

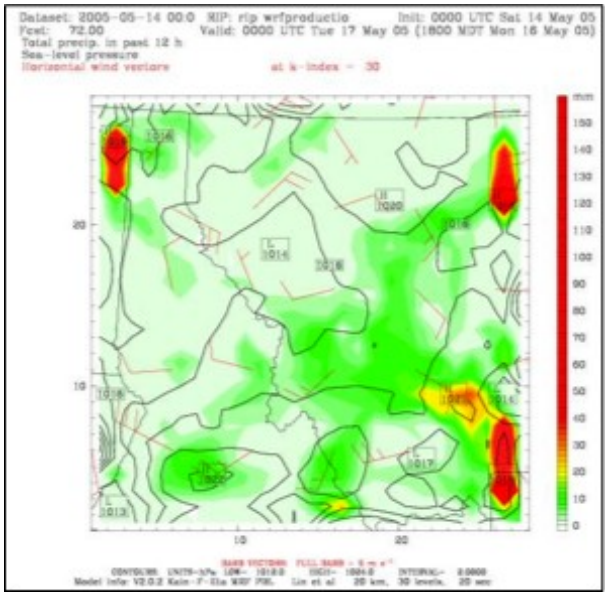
Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Beginnings of operational NWP

- Early 2004 – MM5 prototyping
- After first WRF Tutorial, June 2004
- Daily 60km with 20km nest
- Performed on home computer – Linux VM with guest PGI compiler running on Windows XP 1 GHz, *bash*-driven
- 72 hour forecast took 7.5 hours
- Moved to spare Linux machine at U. Montana





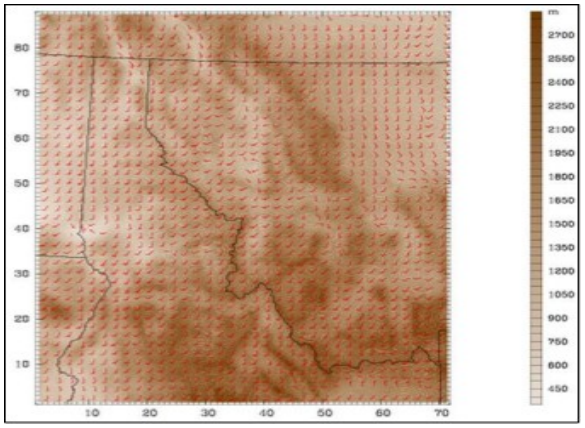
UAF UNIVERSITY OF ALASKA FAIRBANKS

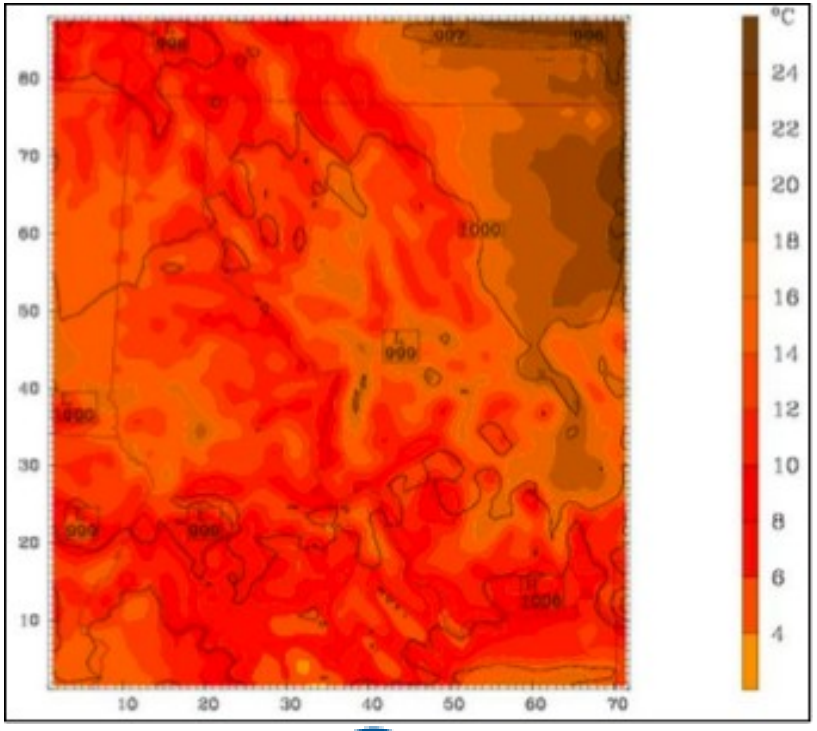
Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

2004 – Missoula WFO forecasts

- Single Linux machine
- Evolved to Gene Petrescu Memorial Supercomputer – 2-machine, 4-cpu Linux cluster
- *bash*, local web posting, one forecast per day
- Daily 7.5km (73x86x75 grid)
- 72-hour forecast in 12 hours





UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Cray XD1

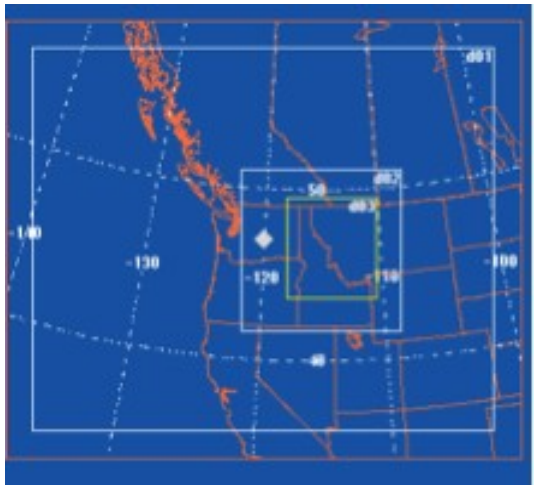
- Missoula USFS purchased small XD1, housed at NWS with usage agreement
- Started getting complicated – moved from bash to Python
- SGE queueing system
- People wanted multiple runs per day
- People wanted AWIPS
- Automated verifications
- Still a monolithic and linear structure

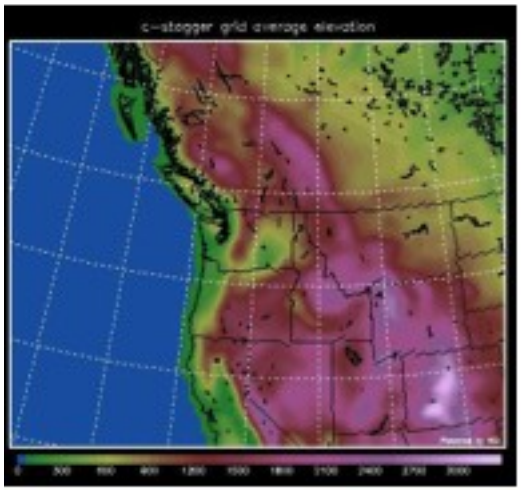
UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

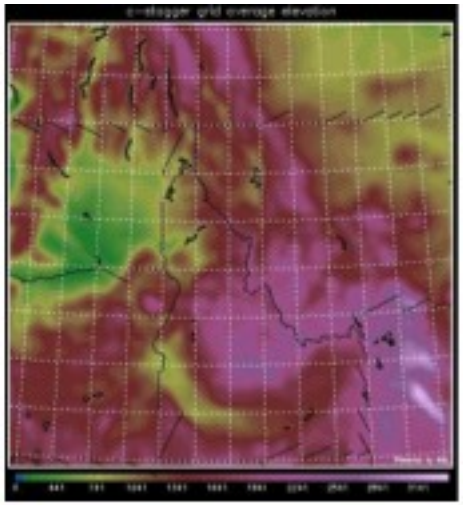
Arctic Region Supercomputing Center

XD1 MSO Production Domains

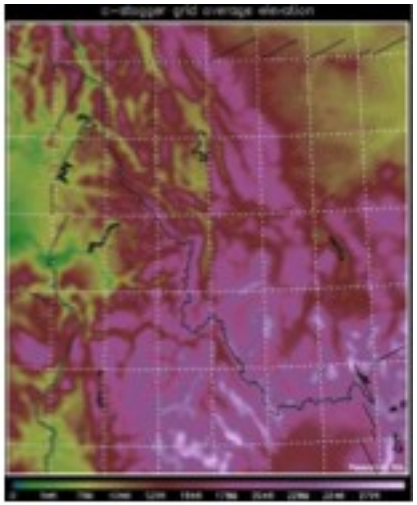




36 km



12 km



4 km

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

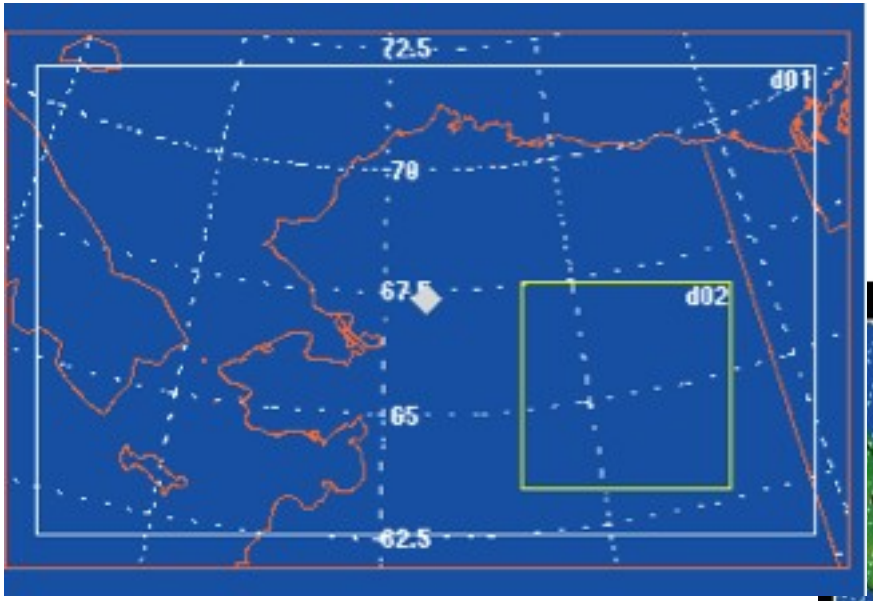
Arctic Region Supercomputing Center

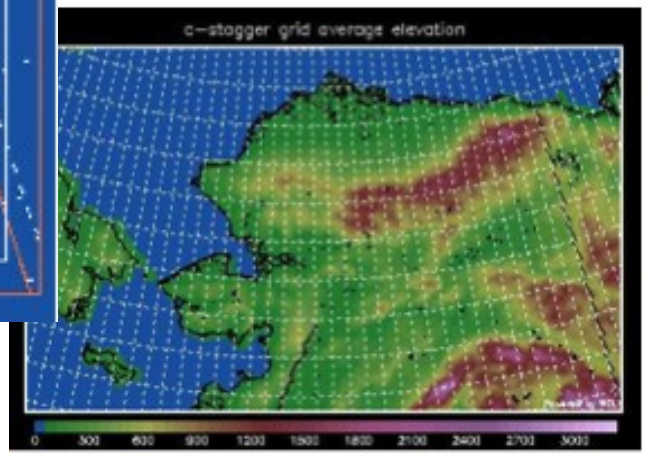
2005 – Alaska forecasts

- ARSC had acquired an XD1 and was interested (along with Cray) in these forecasts as showcase application
- Early summer - XD1 wasn't ready yet, so deployed first on IBM p655 cluster (XL compilers, LoadLeveler queueing system), with intent to move to XD1 when ready
- Late summer - Moved to XD1, more ambitious runs, more I/O issues

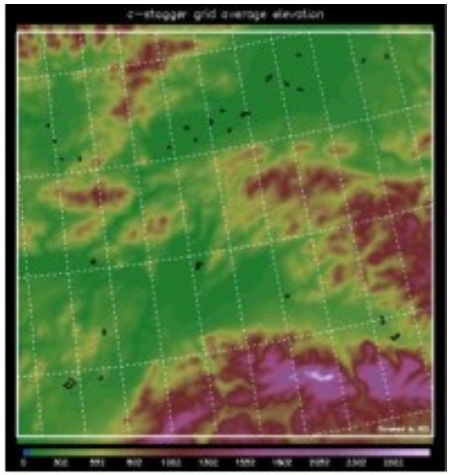
2005 – Alaska Forecasts

- At the time, focused on Northern Alaska forecast area





9 km



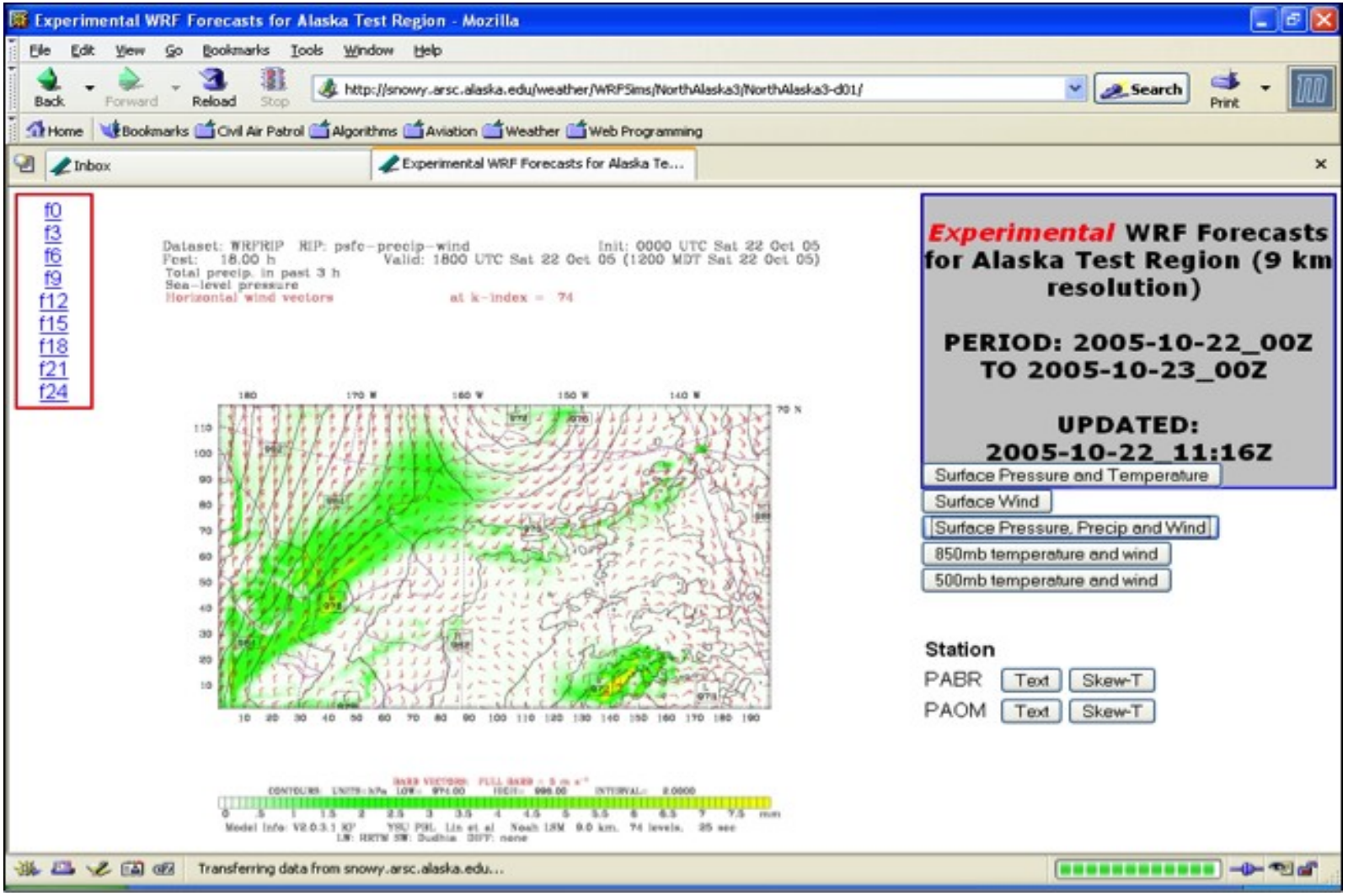
3 km

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Alaska Products

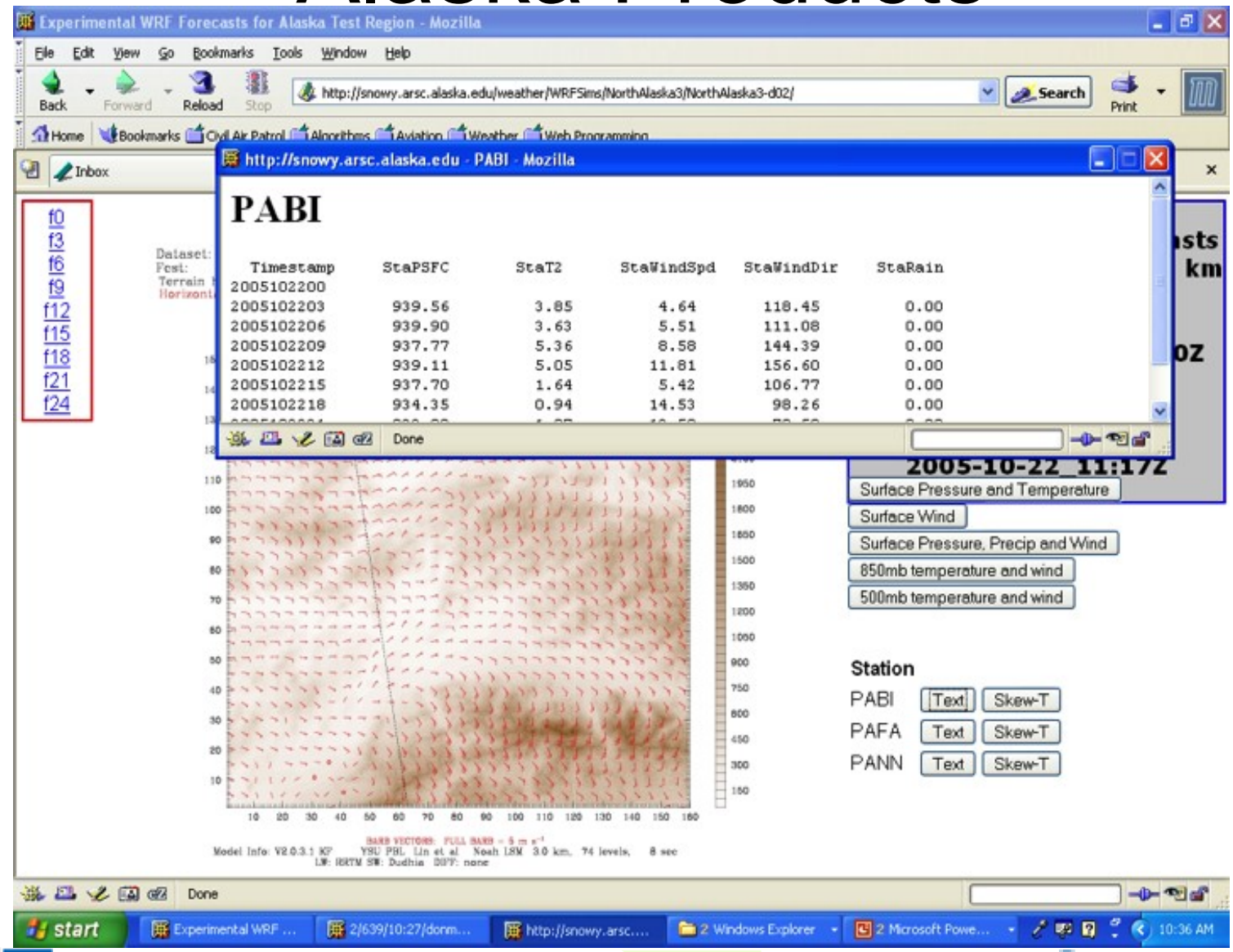


UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Alaska Products



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

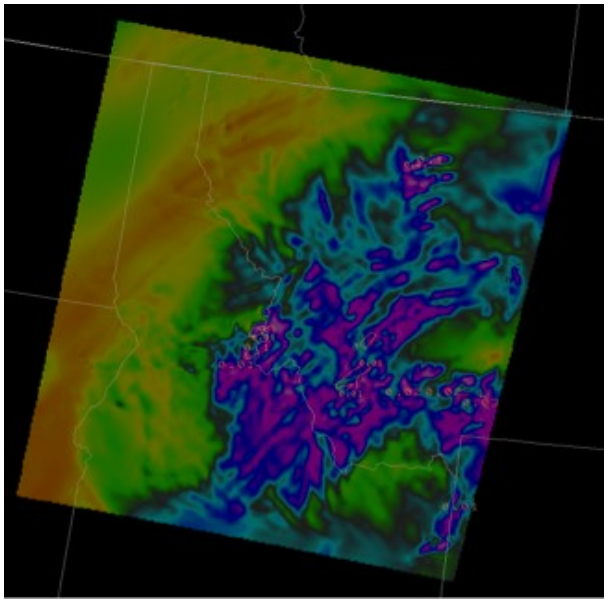
Arctic Region Supercomputing Center

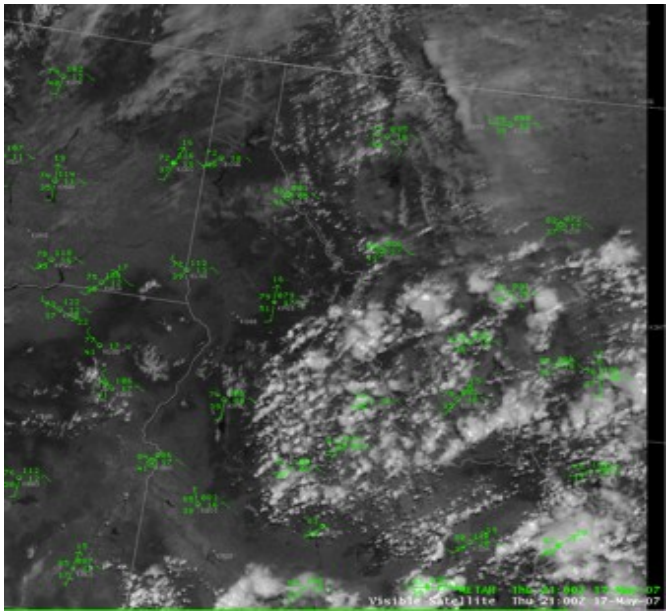
Continued MSO and FAI runs

- MSO runs in high-demand
- NWS wanted more timely runs – e.g. 00Z forecast products starting to come in around 00Z!!
- Multiple runs per day, which meant overlapping of operations
- New, temporary nests
- At ARSC – reserving resources for immediate execution (adjusting of PBS batch scripts)

MSO Runs

- Northern Rockies Mesoscale Model (NRMM) (or the Norma Gene)
- Forecasters began to note skill in summer convective activity





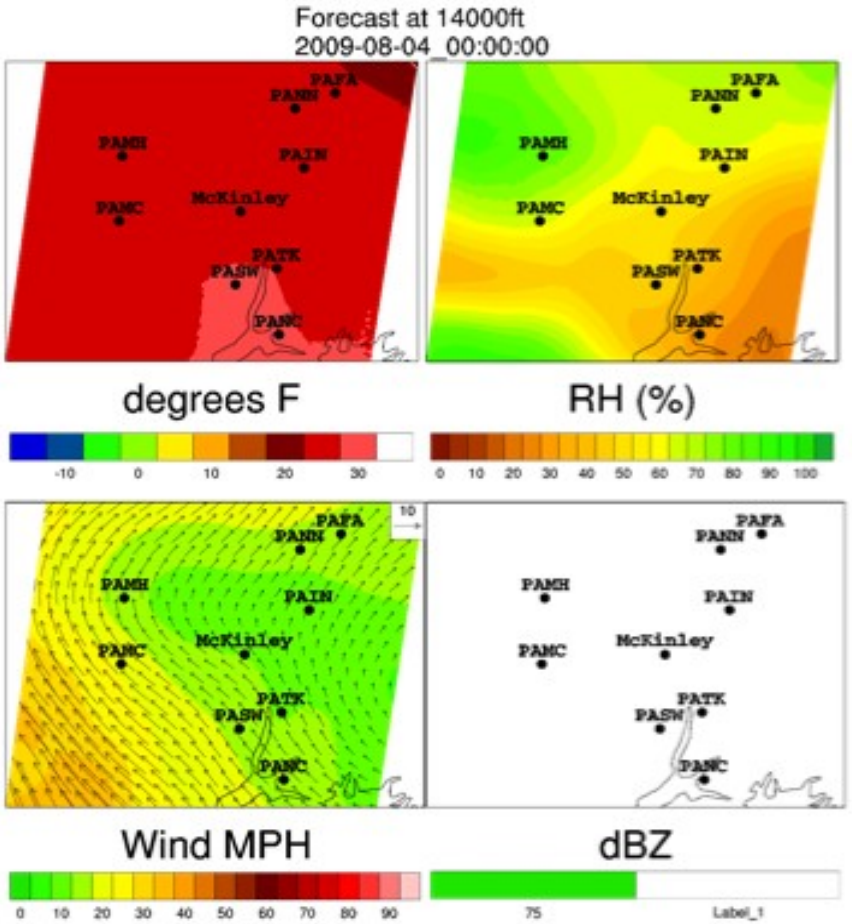
UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Mt. McKinley Forecasts

- Tallest mountain in North America, popular for science and recreation in the summer
- At 63N, experiences bitter, violent weather any time of year
- Daily forecast for 7,000, 14,000, 17,000 and 20,000 feet



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

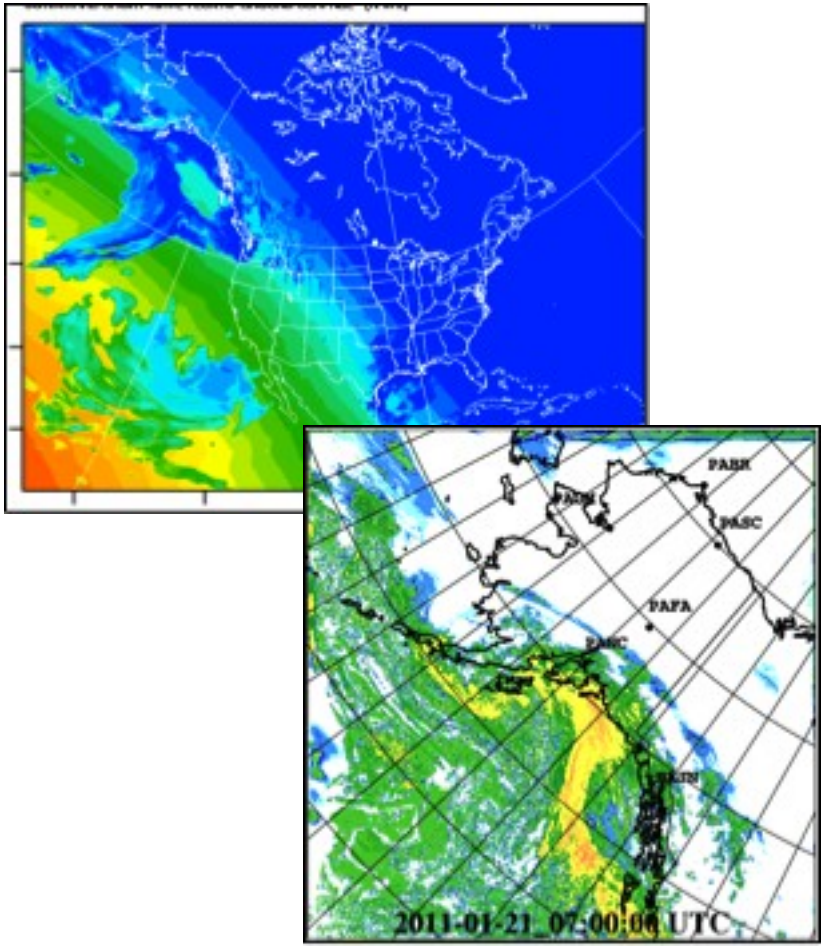
Arctic Region Supercomputing Center

Constant migrations to new systems

- System retirements
- Performance issues
- Making room for other users – time/space constraints

2009 - HRRR-AK

- 48-hr forecasts, 2x daily, 256 cores
- 1050x1050x51 grid points at 3km
- Initialised by RAP, LBC's from AK NAM 12
- Multiple streams of pre-processing now
- Real-time GRIB production during simulation for AWIPS

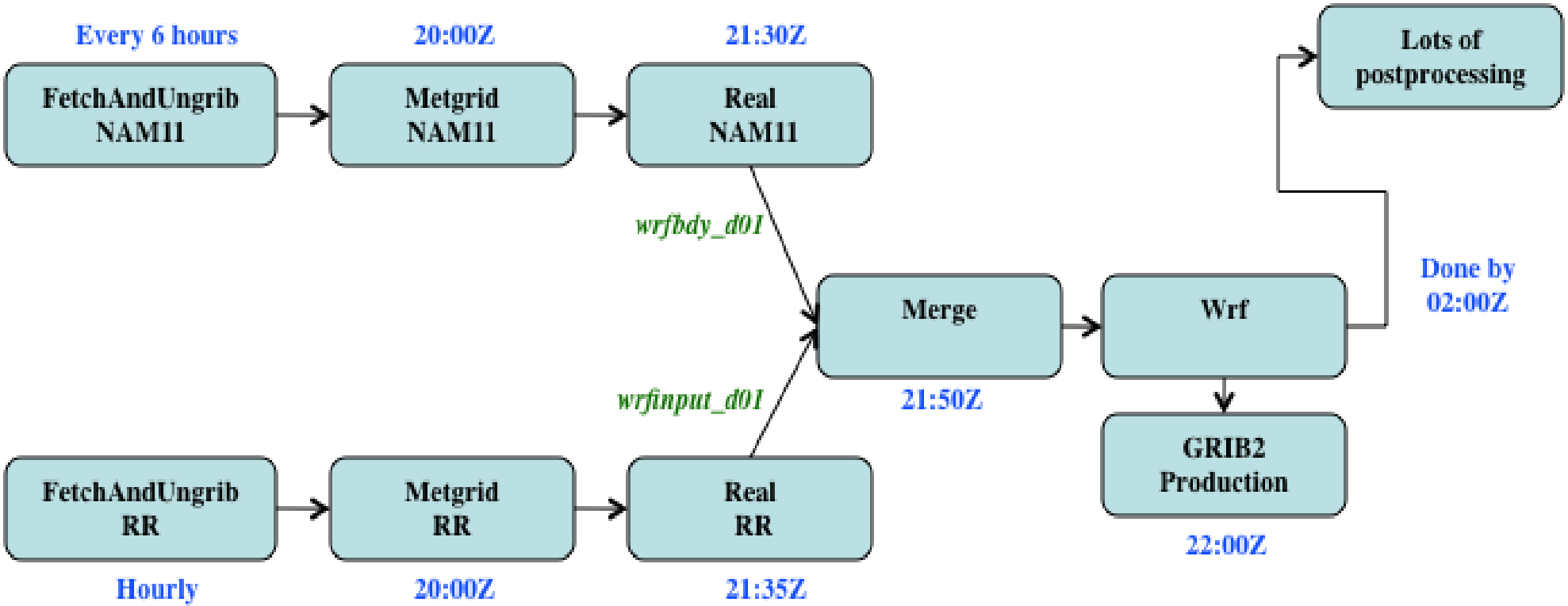


UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRR-AK Process Flow

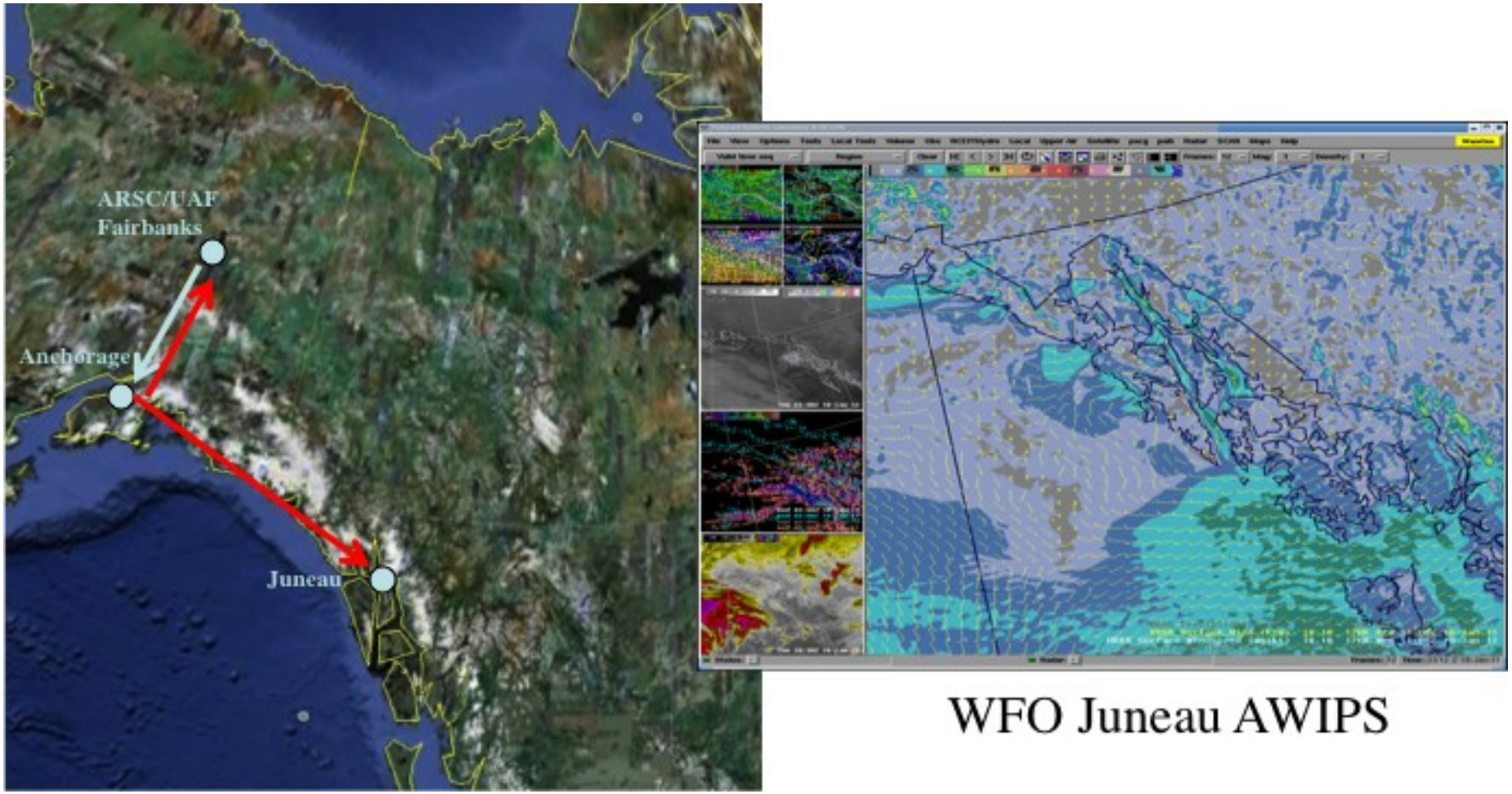


UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRR-AK Distribution



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

2010 – DTC support

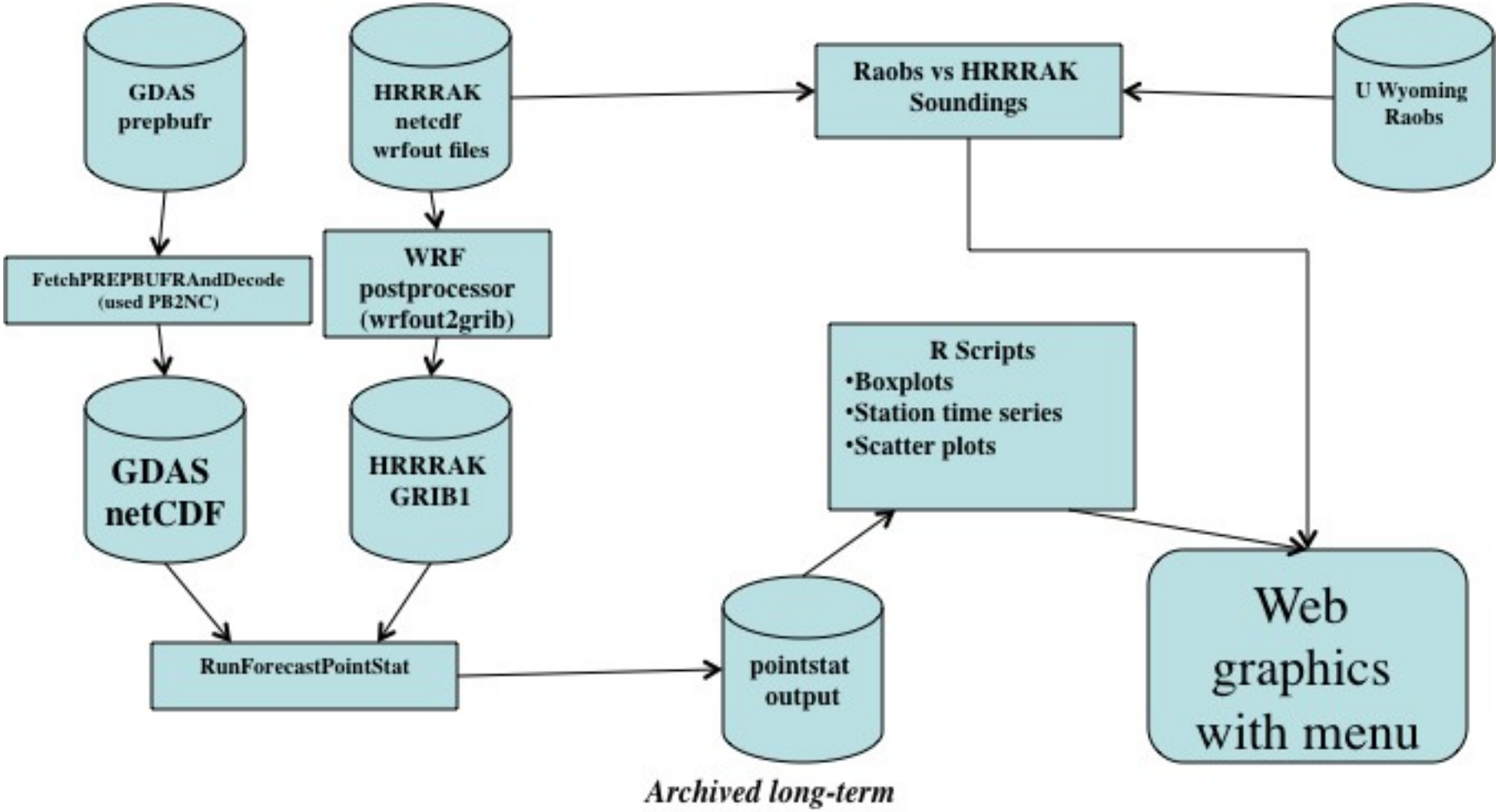
- Model Evaluation Tools (MET)
- Sounding verifications
- Longer term summaries
- Experimental introduction of Gridpoint Statistical Interpolation (GSI) data assimilation

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Verification Post Processing



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

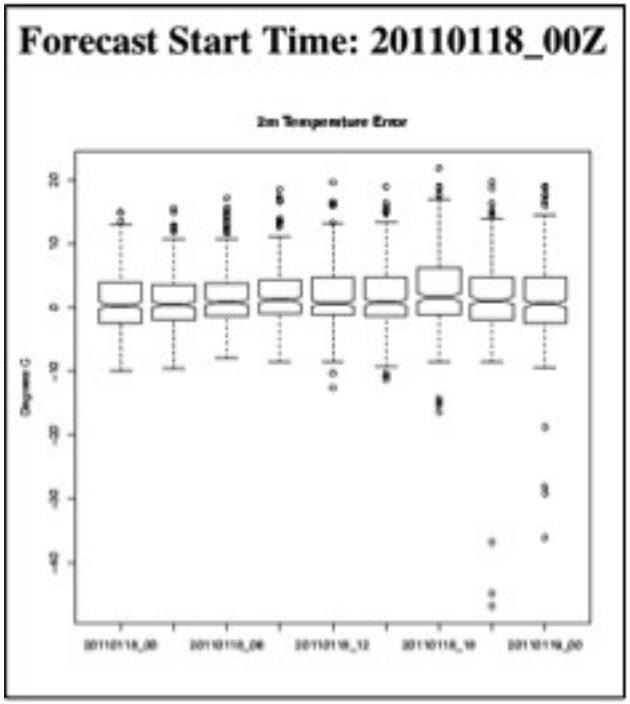
Verification Products

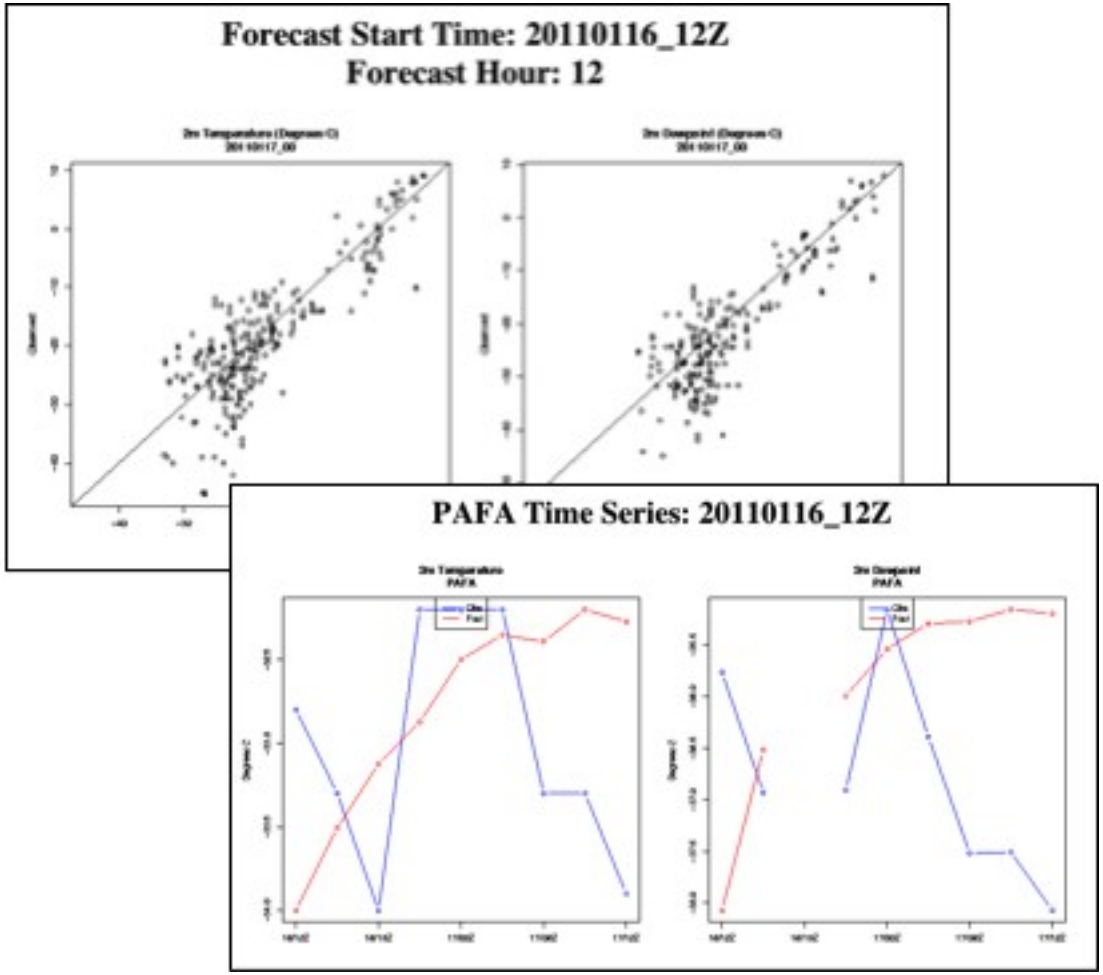
UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Verification Products



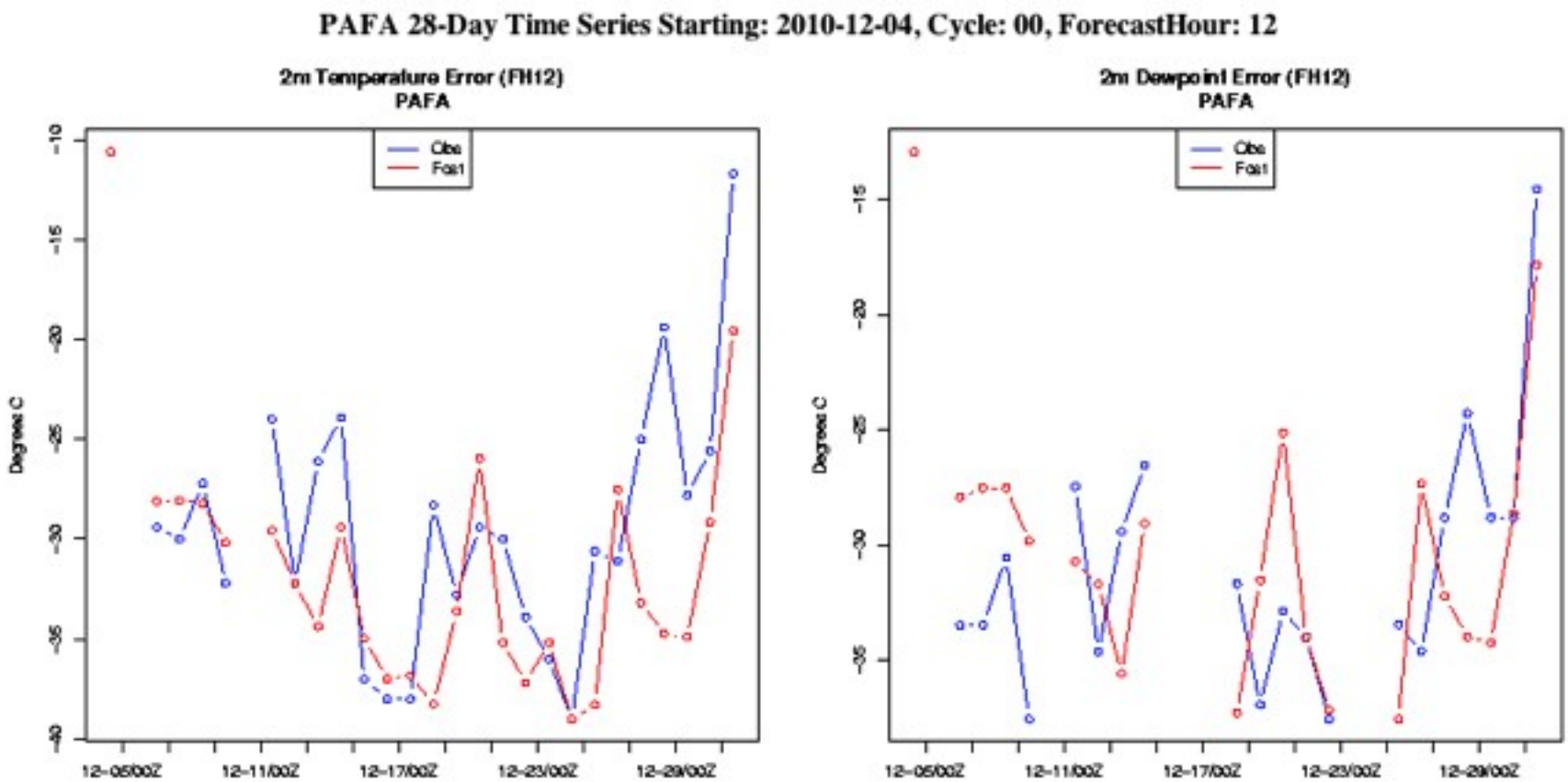


UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

28-day Verifications

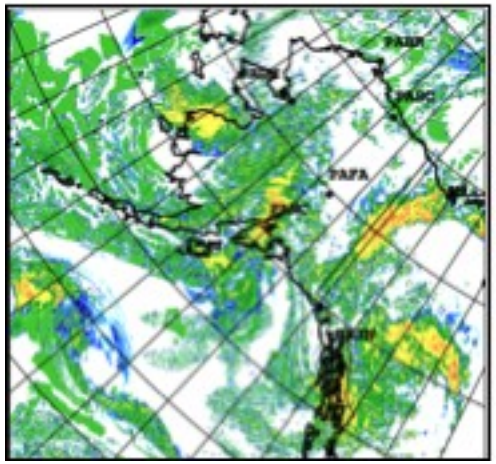


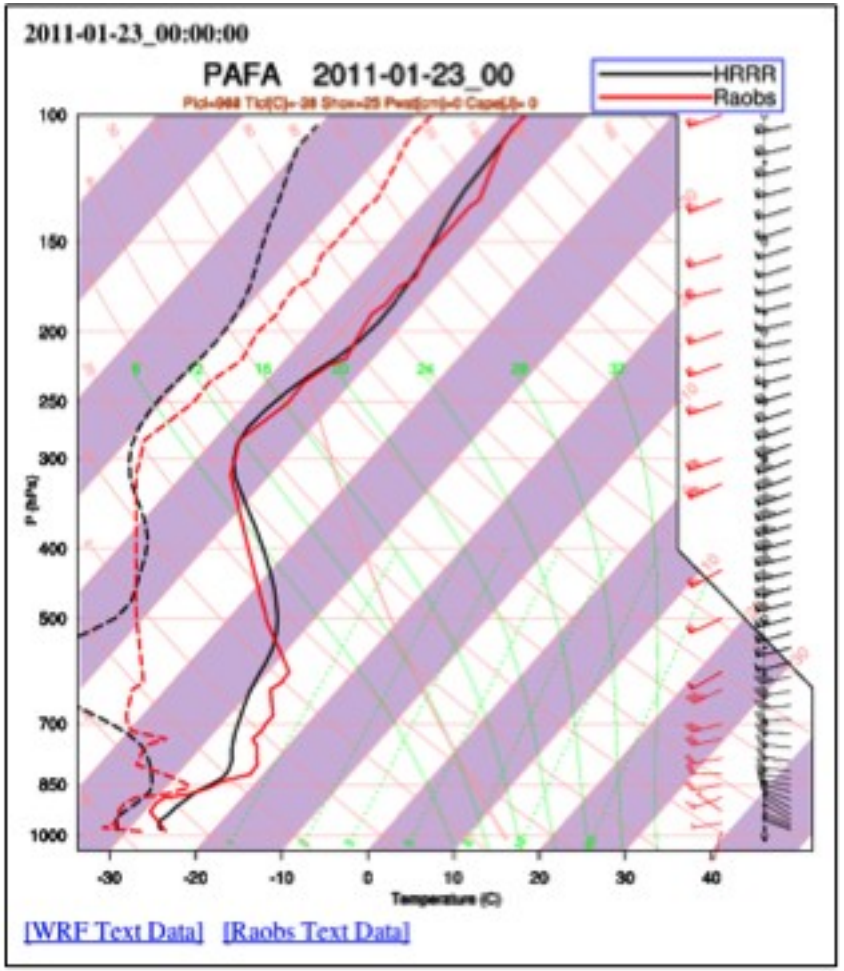
UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Raobs vs WRF Soundings



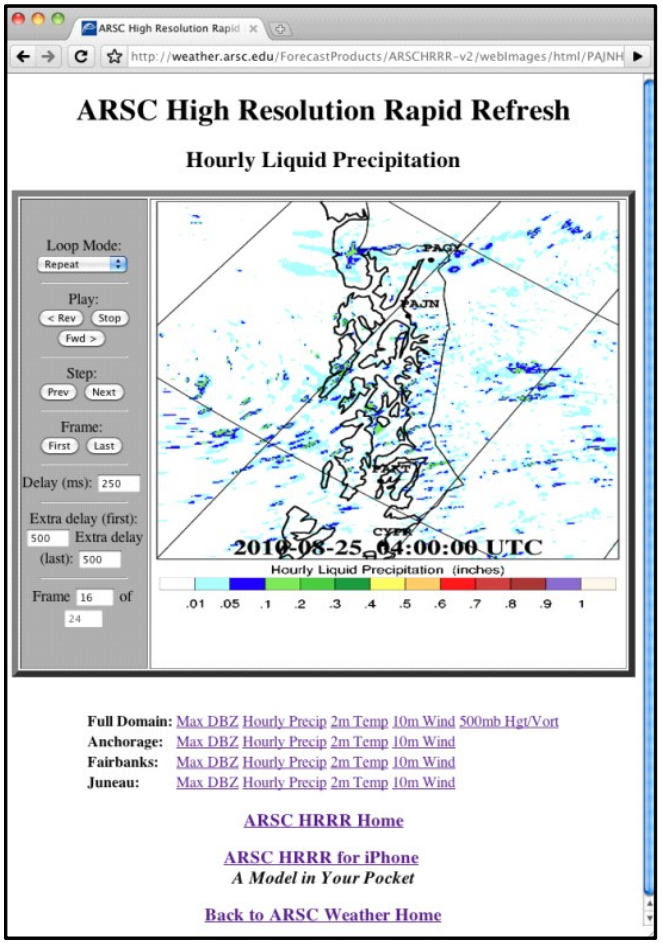


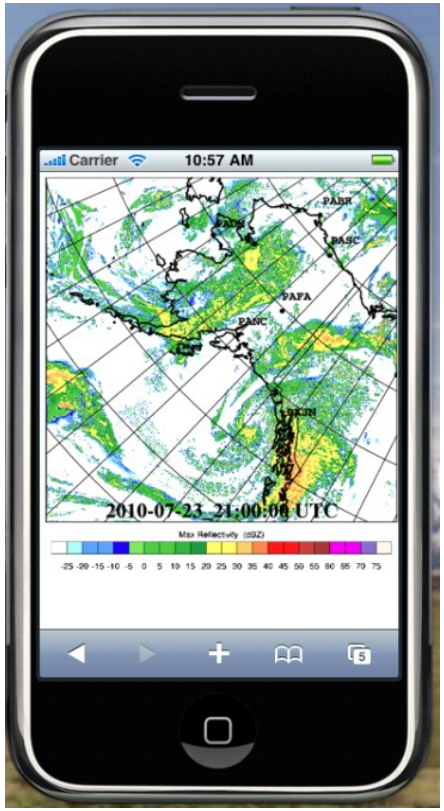
UAF UNIVERSITY OF ALASKA FAIRBANKS

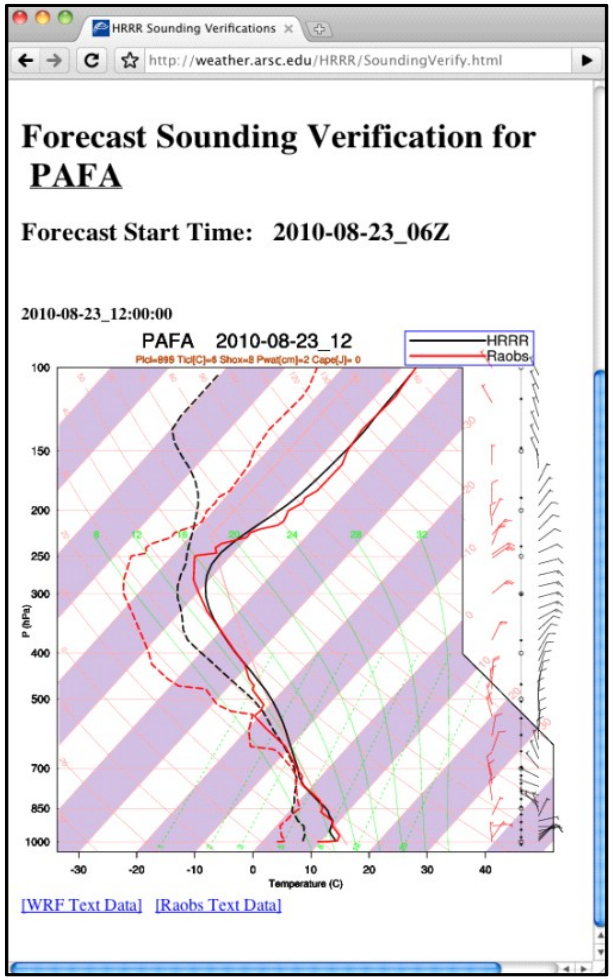
Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRR-AK Web Products







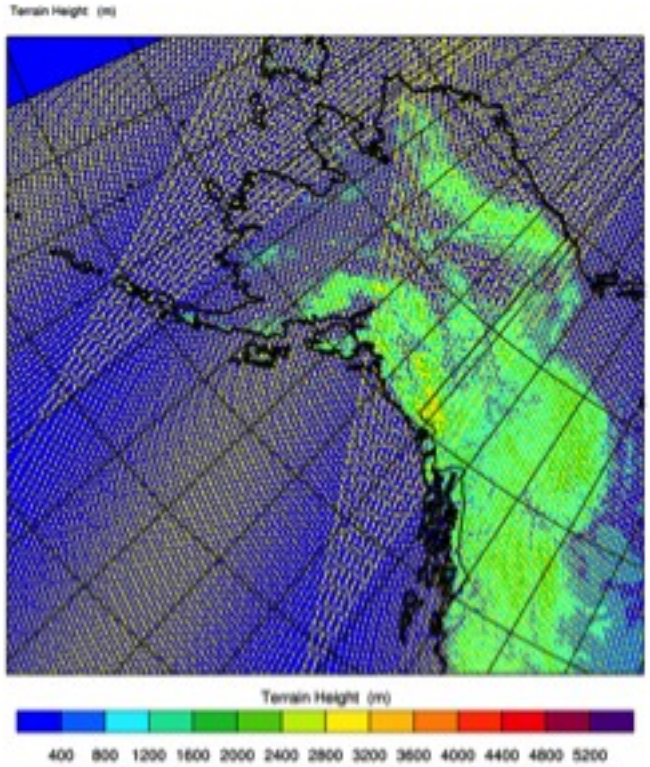
UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Assimilation of Atmospheric Infrared Sounder (AIRS) profiles

- Comes from instrument aboard NASA's polar-orbiting EOS Aqua satellite
- JPL and NASA SpoRT derive temperature and moisture profiles from radiances
- Short-term experiment to assimilate these, using Gridpoint Statistical Interpolation (GSI) system with HRRR-AK, concurrent with HRRR-AK runs

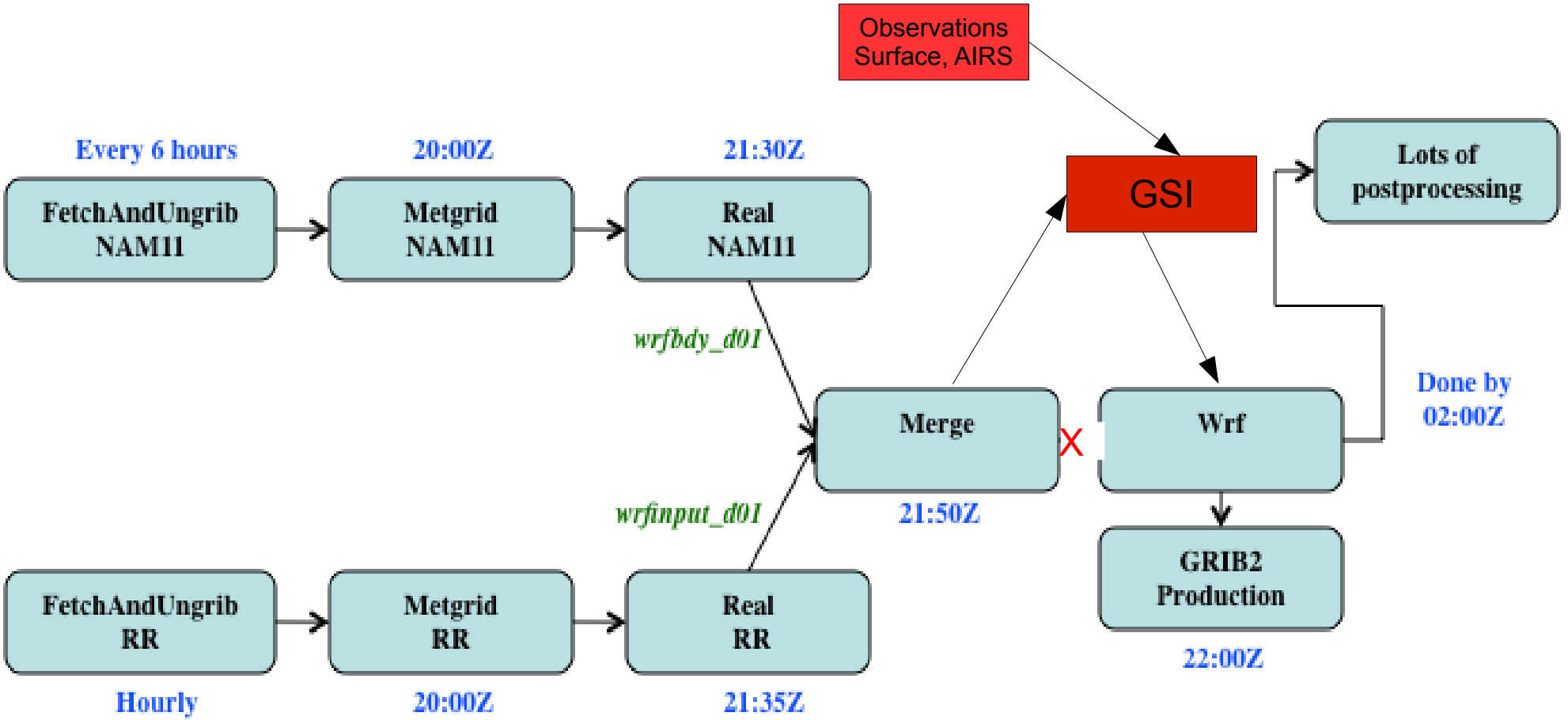


UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

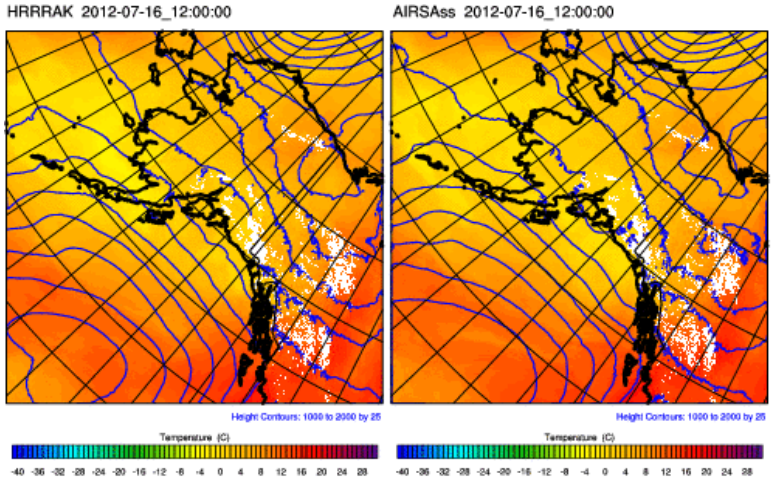
Arctic Region Supercomputing Center

Addition of GSI components

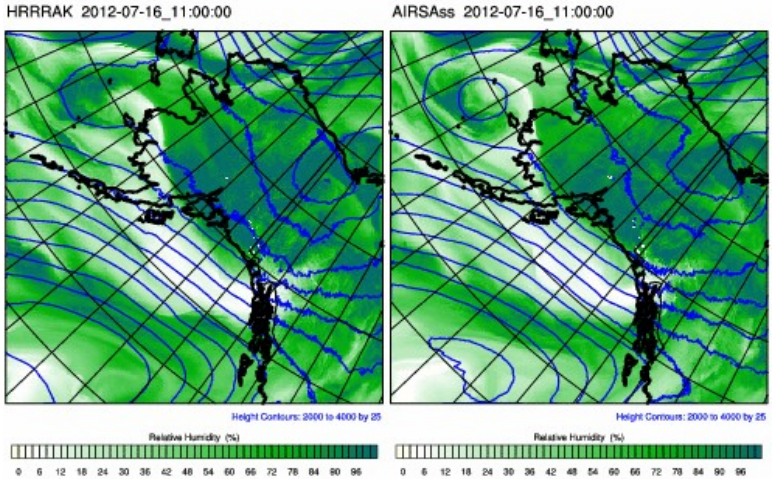


HRRR-AK / AIRSAss Comparisons

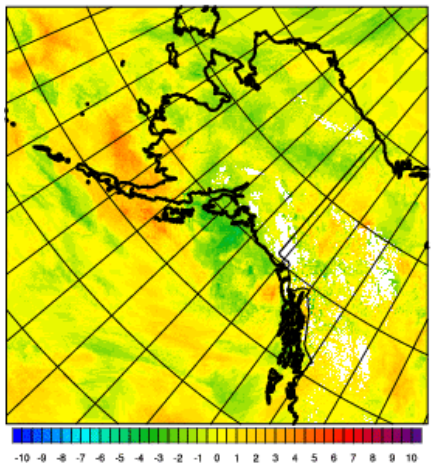
850mb T



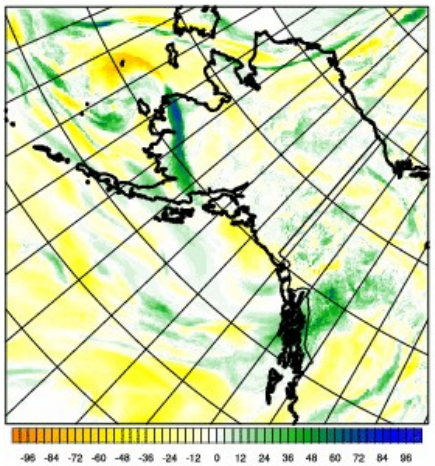
700mb RH



AIRSAss minus HRRRAK 2012-07-16_12:00:00



AIRSAss minus HRRRAK 2012-07-16_11:00:00

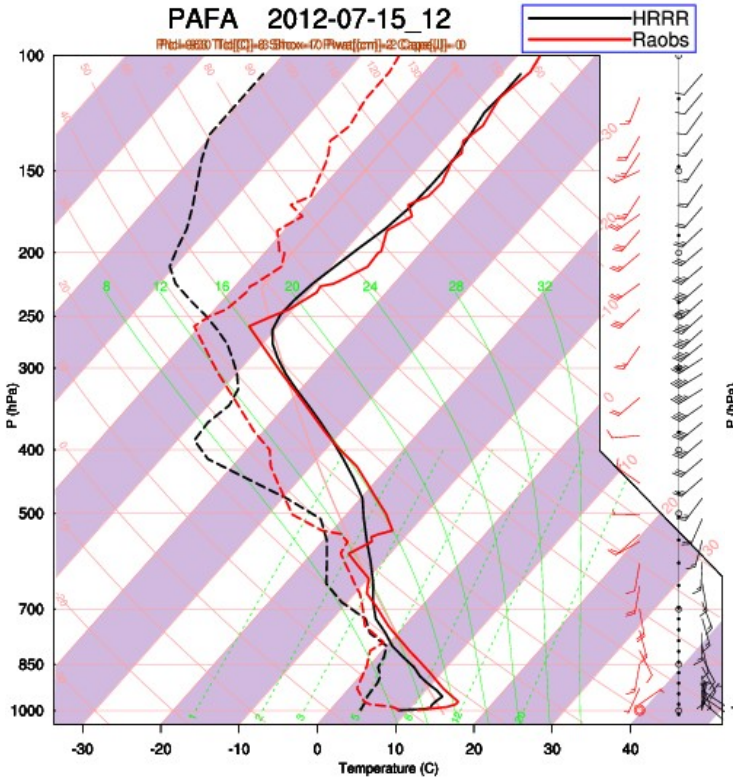


UAF UNIVERSITY OF ALASKA FAIRBANKS

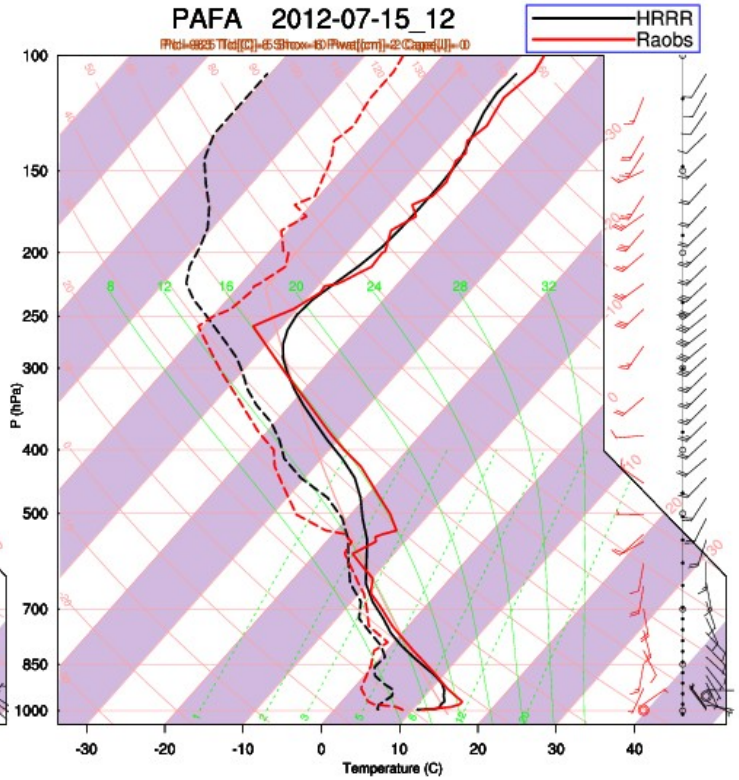
Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRR-AK / AIRSAss Sounding/RAOBS



HRRR-AK



AIRSAss

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

2011 – CTBTO, LPDM

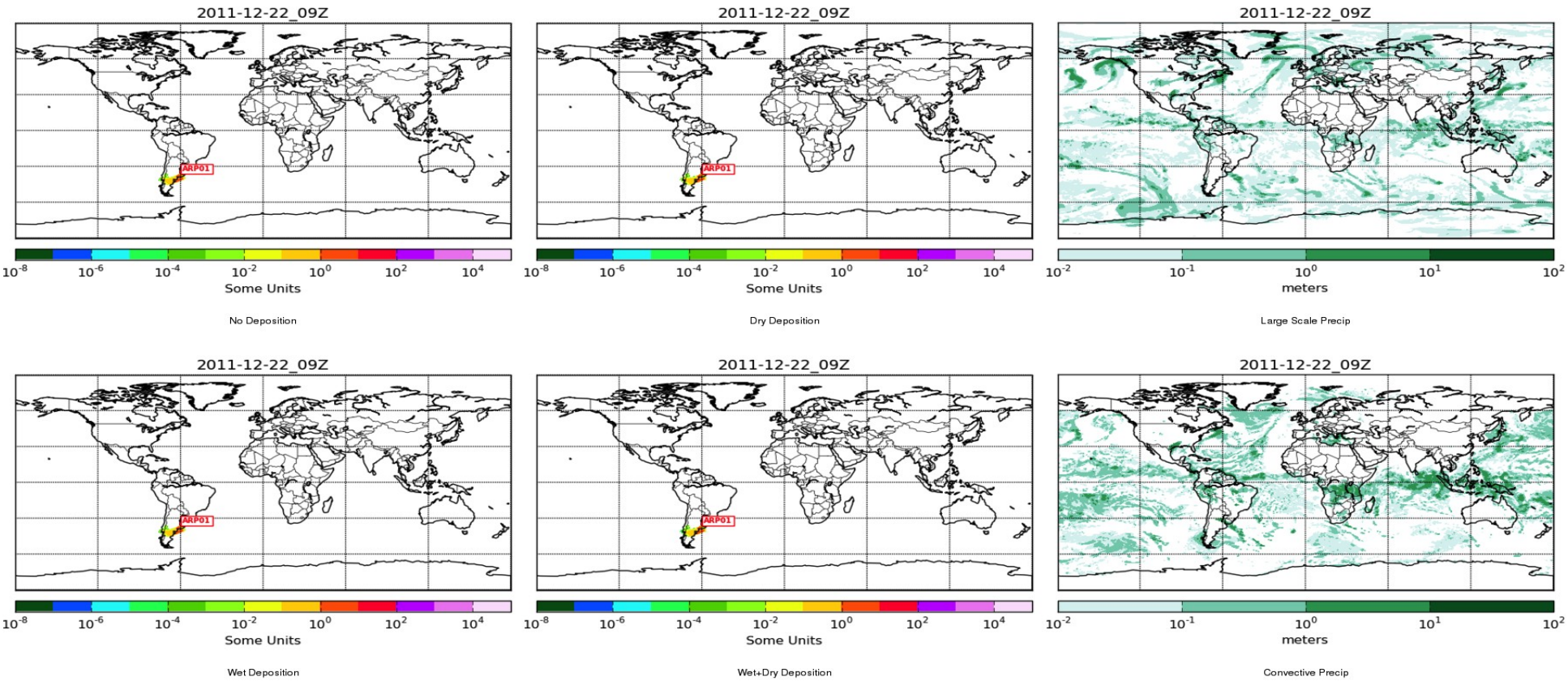
- Operational backwards simulations for United Nations Comprehensive Test Ban Treaty Organisation (CTBTO) with Lagrangian Particle Dispersion Models (LPDM)
- Forward plume simulations – operational smoke forecasts for Alaska – just proof of concept – based on FLEXPART. Required new streams of met data (GFS), new problems
- Rapid deployment of same smoke forecasts for Catalonia

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Operational Backwards Simulations

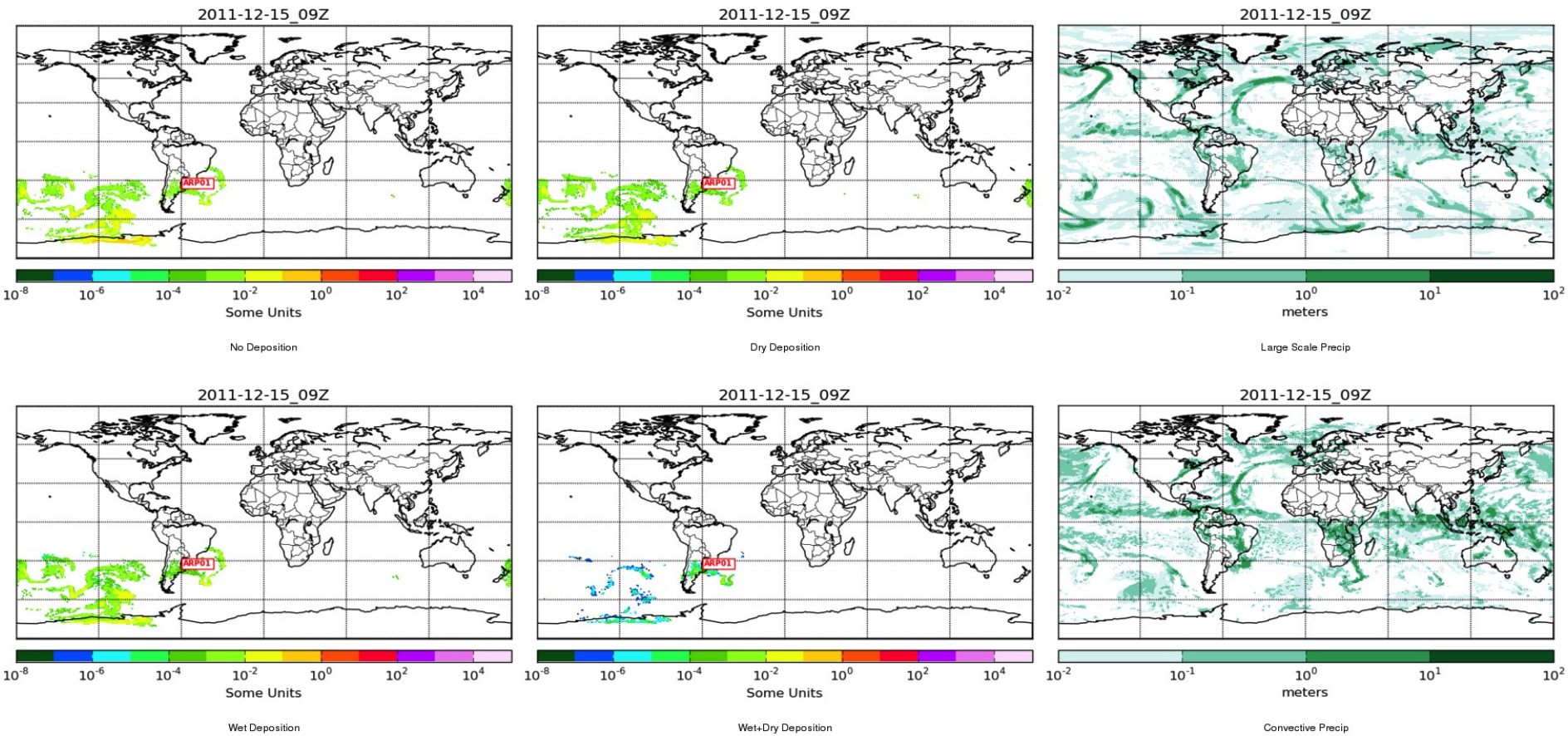


UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Operational Backwards Simulations

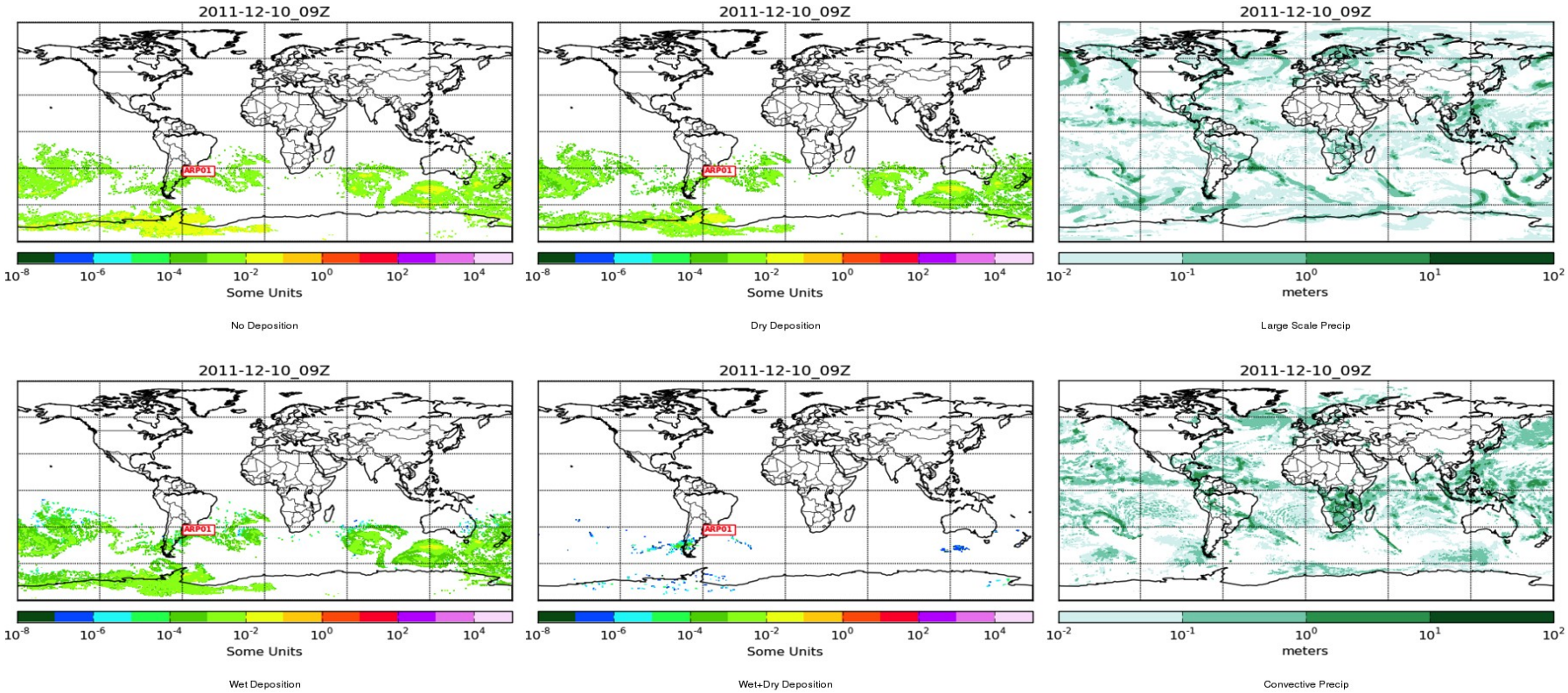


UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Operational Backwards Simulations

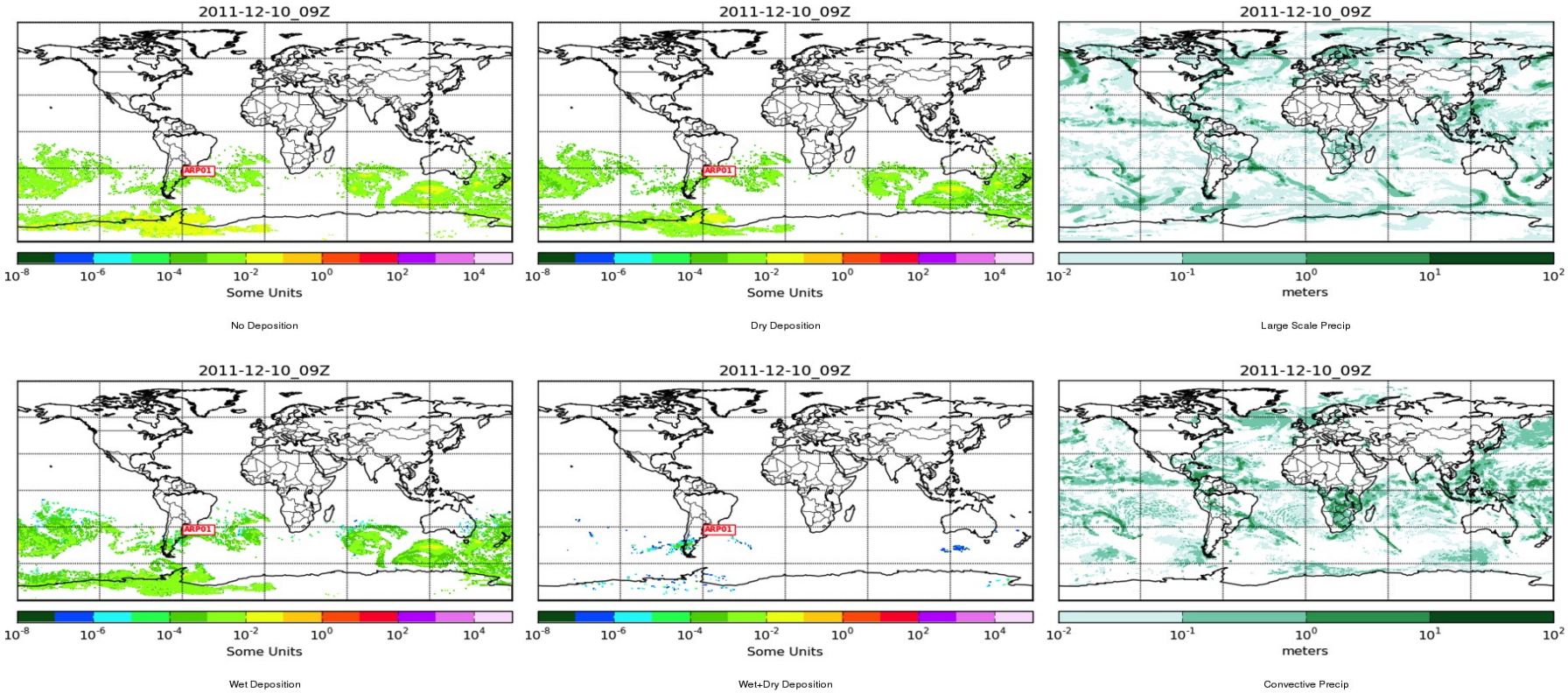


UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Operational Backwards Simulations



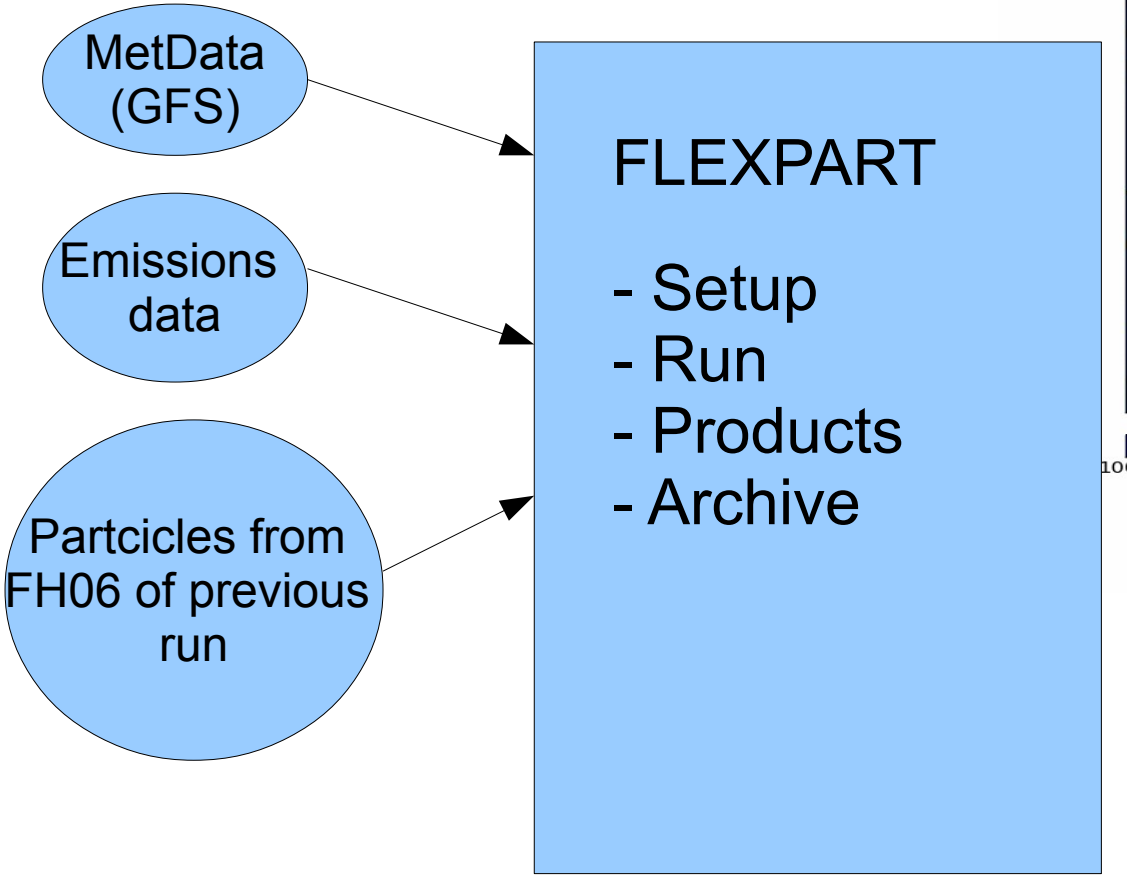
UAF UNIVERSITY OF ALASKA FAIRBANKS

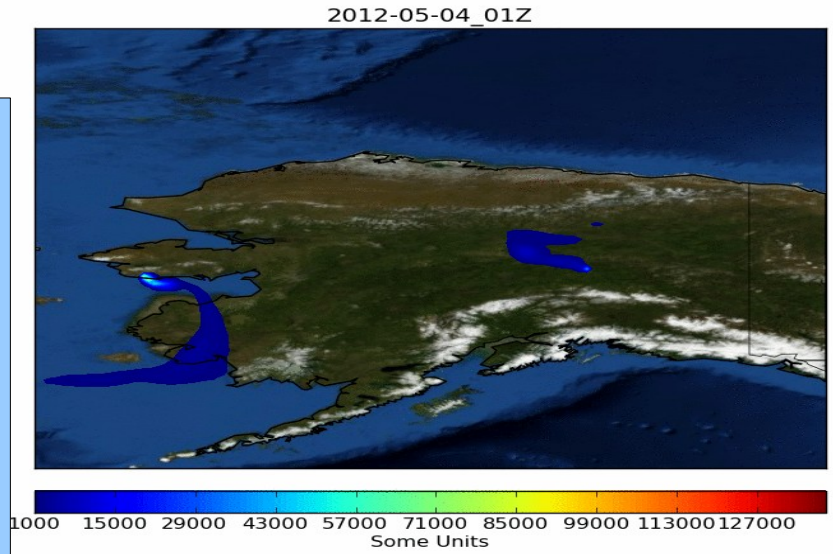
Boreal Scientific Computing LLC

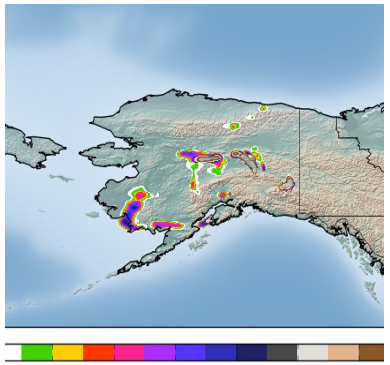
Arctic Region Supercomputing Center

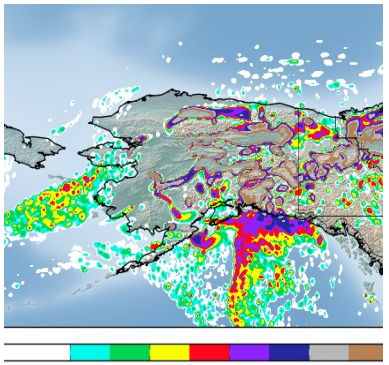
FLEXPART operational runs

- April 2012 - <u>prototyping</u> operational wildfire smoke dispersion - run every six hours









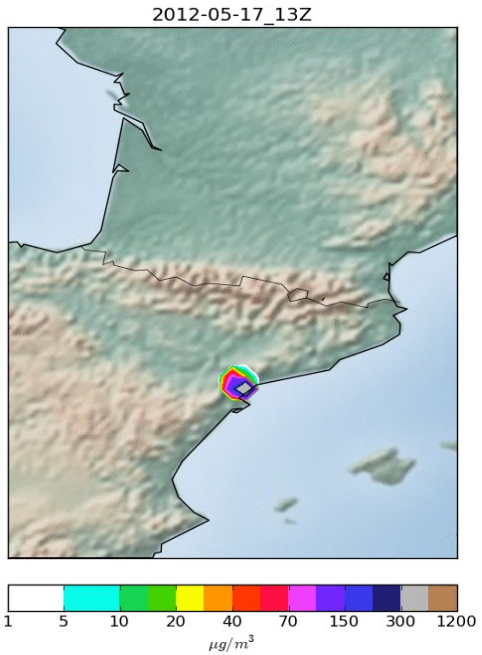
UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Fires in Catalonia!

- 18 May 2012
- Rapid deployment of operational FLEXPART for Catalonia in 2 hours
- Like the Alaska runs, ran every six hours using same GFS met data



https://www.facebook.com/don.morton.7/posts/366087773448370?notif_t=feed_comment

Meritxell Pagès ▸ Don Morton
52 minutes ago

Any information you need, do not hesitate in asking me! This fire is a shame because the area is nice...in fact, I was there last summer :-(

Like · Comment · Unfollow Post · See Friendship

Don Morton Thanks Meritxell! Delia's going to call me from Vienna soon. Right now I'm looking at setting up a 0.2 degree domain from a SW corner of 38N/3.0W and a NE corner of 45N/7.0E. Just for starters.
50 minutes ago · Like

Delia Arnold wonderful! ok, so let's flexpart fires in catalonia!!!! what do you need?
43 minutes ago · Like

Meritxell Pagès Well done even being for starters! Would be synoptic/mesoscale analisys useful for you?
43 minutes ago · Like

Delia Arnold hola meritxell? estas al chat?
See Translation
42 minutes ago · Like

Meritxell Pagès No el tinc connectat, però I'm alive!
42 minutes ago · Like

Delia Arnold hehe
42 minutes ago · Like

Delia Arnold ok, so don is talking with me right now.. he is setting all this up
41 minutes ago · Like

Delia Arnold output is at 0.2 deg, not bad... in the near future we can think about flexpart-wrf
41 minutes ago · Like

Meritxell Pagès In one second I'll connect the chat
41 minutes ago · Like

Delia Arnold juhu!!!! everyone excited and don under pressure!!!
40 minutes ago · Like

Meritxell Pagès :-)
40 minutes ago · Like

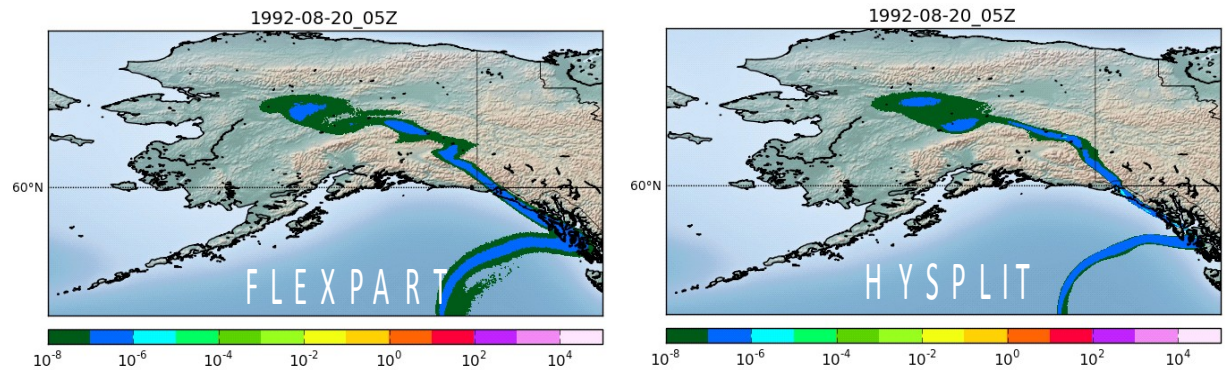
Don Morton Ok, because the 18Z GFS wasn't out yet, I initialised on the 12Z, and we're now up to Forecast Hour 5. This may take a few hours to run. In retrospect, maybe I should have initialised with 1.0 deg GFS rather than 0.5, but that's OK, we'll see how this works :). Every forecast hour is taking longer to compute because we're increasing the amount of smoke in the model. I'm going to set this up so that it automatically runs four times a day, just like the Alaska model (but we don't have any fires yet).
10 minutes ago · Like

Write a comment...

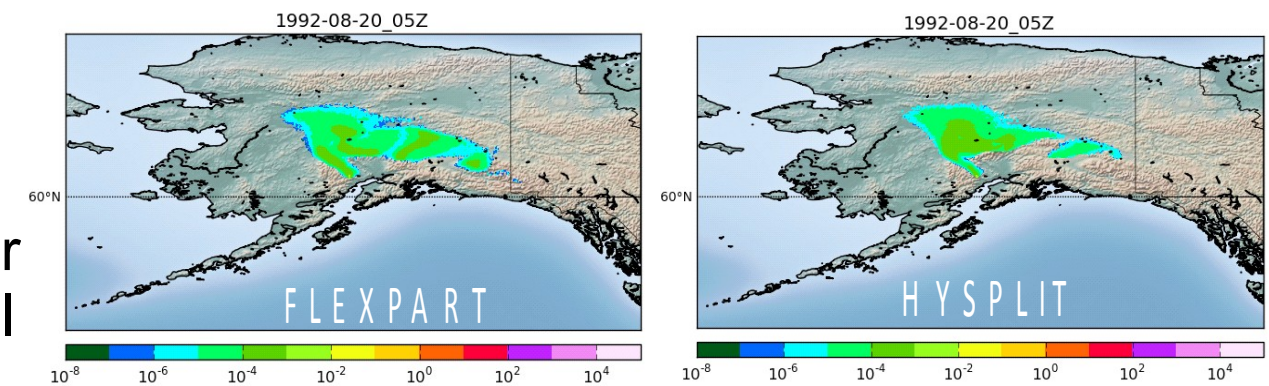
Chat (Offline)

Long-term operational vision

- Multimodel operational runs (begin with FLEXPART and HYSPLIT)
- Outputs in common framework
- Rapid deployment capabilities - literally go to a web page (or even better, have an event trigger the operations), enter coordinates, and launch a multi-model, multi-member set of operational runs until event has passed.



Total column concentration



Cumulative deposition

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Volcanic ash transport and other emergency response activities

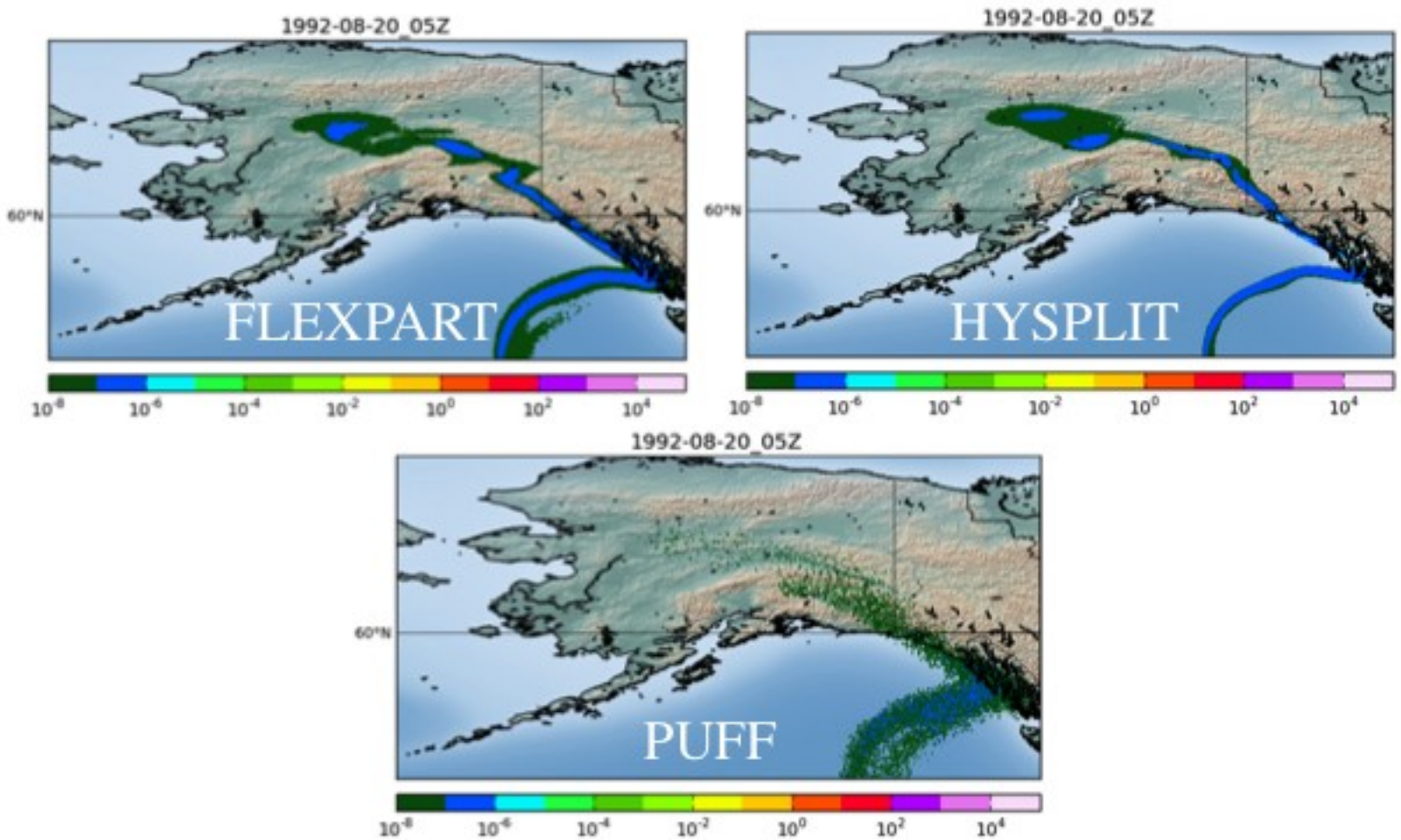
- There is a need to rapidly deploy targeted forecasts for surprise events like nuclear accidents and volcanic eruptions
- Need to be able to conglomerate outputs from different models and configurations into unified environment for rapid assessment
- Vision – literally, with the push of a button, or automated, launch an ensemble of models and configurations and display results in a common framework

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Total integrated column concentration, Mount Spurr eruption 1992

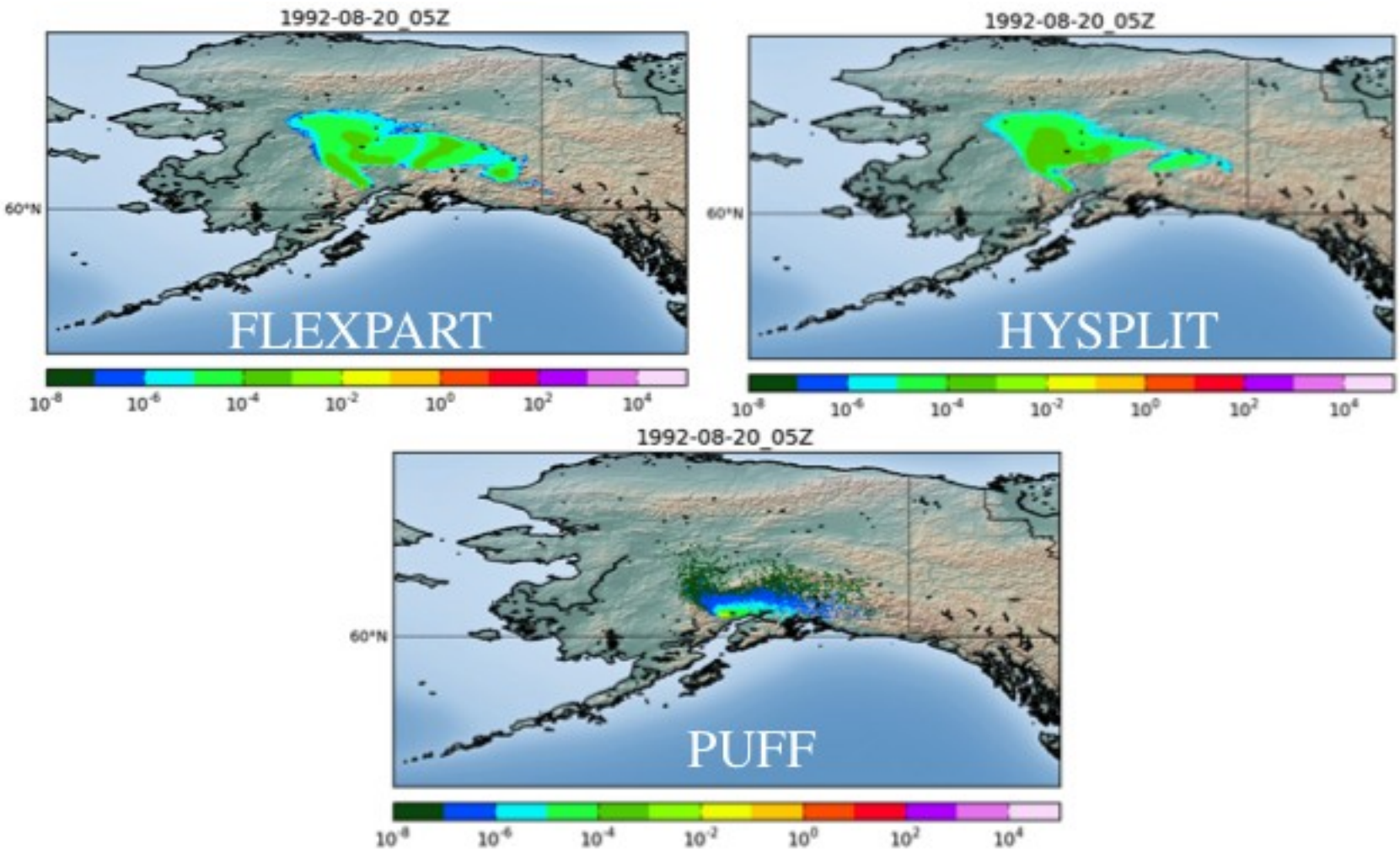


UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Cumulative deposition, Mount Spurr eruption 1992

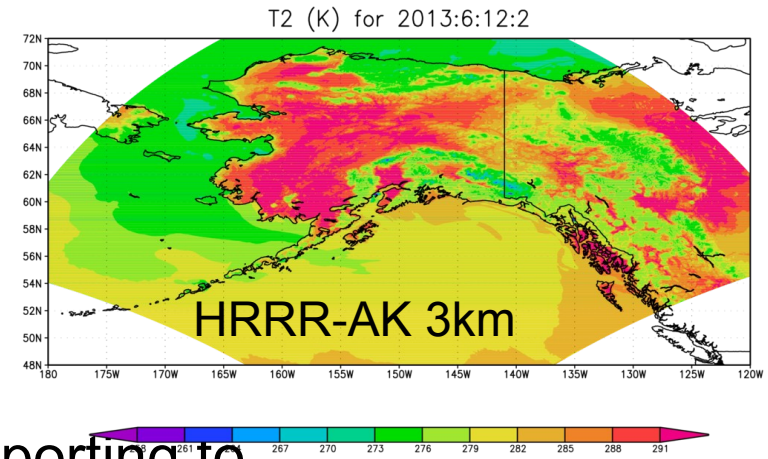


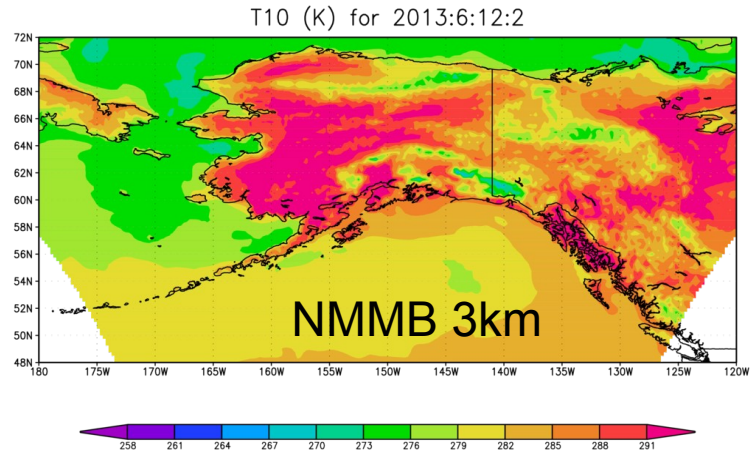
UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

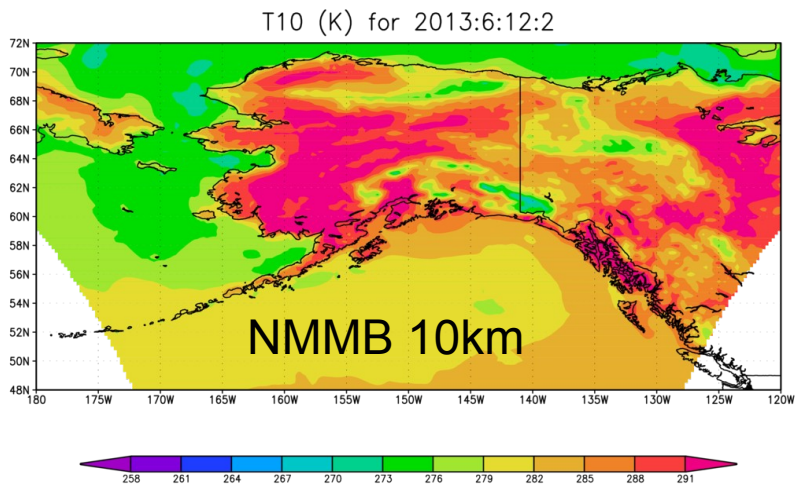
Arctic Region Supercomputing Center

Operational NEMS/NMMB for Alaska





- Learning, porting to GNU environments took weeks of work
- Setting up operationally took less than a day
- 10km NMMB 4x per day, 48 hour forecasts



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Some software engineering perspectives in a real-world constantly changing environment

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

From monolithic bash script to loosely coupled Python components

- Various learning experiences over the years - changes were often mandated, requiring "uncomfortable" hacks
- Examples
  - From one run to multiple runs per day - conflicts in timing, filesystems
  - NWS requests
    - Start a forecast (e.g. 00Z) before the forecast start time (e.g. 00Z)
    - Get GRIB output files delivered to LDM as soon as each forecast hour completes
    - Special requests for new, temporary nests

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

From monolithic bash script to loosely coupled Python components

- More Examples
  - Frequent changes from one supercomputer to another
  - Changes in filesystems often mean new locations for scratch work, performance changes, etc.
  - On supercomputers, not all filesystems are available from all processors, meaning we frequently need to stage files from one place to another
    - e.g. We typically want our input and output files on archive filesystems, but compute nodes only have access to high-performance filesystems
    - high-performance filesystems have limited storage
    - Need to do a lot of juggling
  - Timing is critical - if input data is late or incomplete, what do we do? We need to have everything cleared for next forecast

A paradigm of loosely-coupled Python components

- The Unix philosophy. Excerpts from *The Art of Unix Programming*, by Eric S. Raymond
  - Rule of Modularity: Developers should build a program out of simple parts connected by well defined interfaces, so problems are local, and parts of the program can be replaced in future versions to support new features. This rule aims to save time on debugging code that is complex, long, and unreadable.

A paradigm of loosely-coupled Python components

- The Unix philosophy
  - Rule of Composition: Developers should write programs that can communicate easily with other programs. This rule aims to allow developers to break down projects into small, simple programs rather than overly complex monolithic programs.
  - Rule of Simplicity: Developers should design for simplicity by looking for ways to break up program systems into small, straightforward cooperating pieces. This rule aims to discourage developers' affection for writing "intricate and beautiful complexities" that are in reality bug prone programs.

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

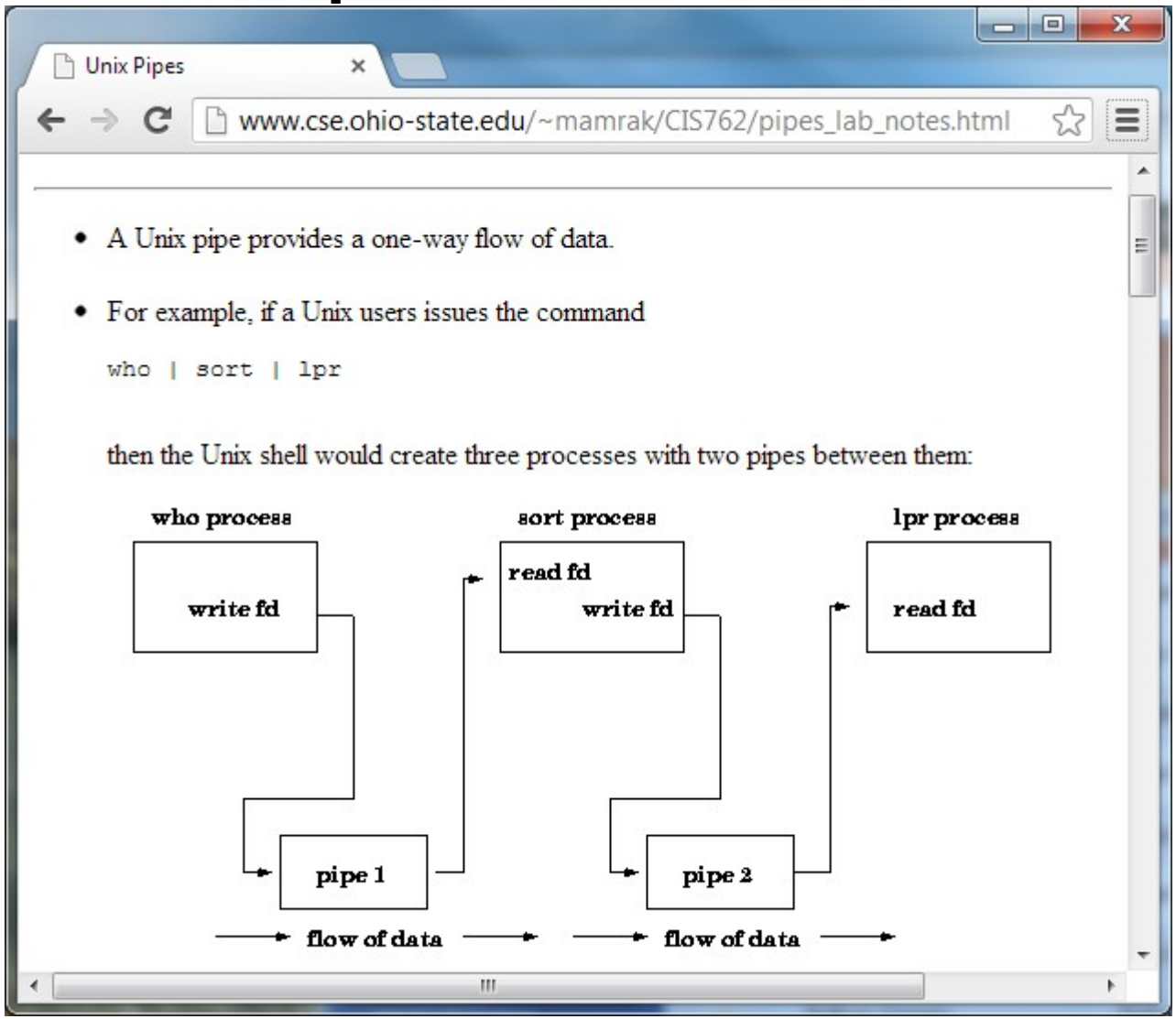
Arctic Region Supercomputing Center

A paradigm of loosely-coupled Python components

- The Unix philosophy
  - Rule of Parsimony: Developers should avoid writing big programs. This rule aims to prevent overinvestment of development time in failed or suboptimal approaches caused by the owners of the program's reluctance to throw away visibly large pieces of work. Smaller programs are not only easier to optimize and maintain; they are easier to delete when deprecated.

A paradigm of loosely-coupled Python components

- The Unix philosophy



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

A paradigm of loosely-coupled Python components

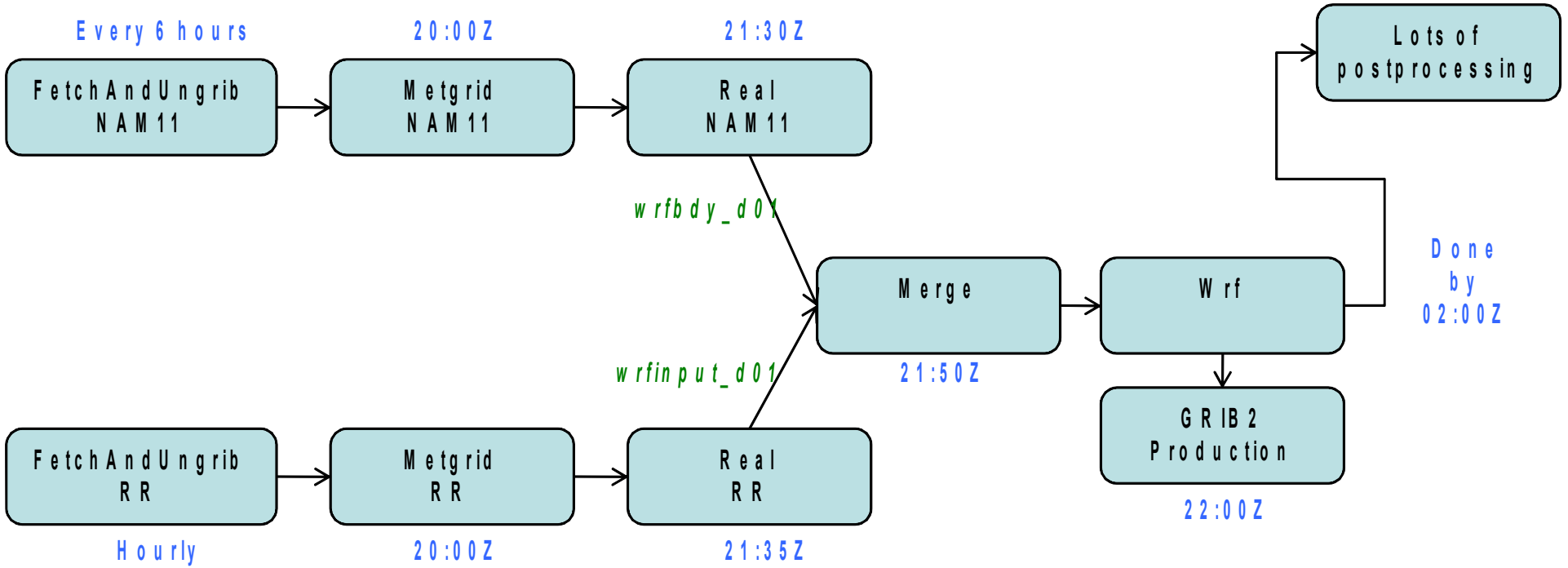
- Ideally, our operational system is built up of loosely-coupled, cohesive components
  - Each component has a specific job to do, and its function is self-contained within the component (cohesive)
  - We try to minimise interactions (side effects) by making the components loosely-coupled. Interfaces are well-designed and simple and, ideally, we think in terms of a single input stream and a single output stream

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRR-AK Process Flow (00Z Forecast)



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

In the real world

- People who want operational systems aren't necessarily interested in the investment needed for a good, robust design
- Often, we are not fully aware of our requirements when we start developing, and need to be ready to modify certain functionalities.
- Thinking in terms of small, functional, loosely-coupled components is, of course, complex, but it also gives us flexibility not found in a huge, monolithic system
- Software design and deployment is all about recognising imperfection and managing many tradeoffs

A closer look at our HRRRAK operational system

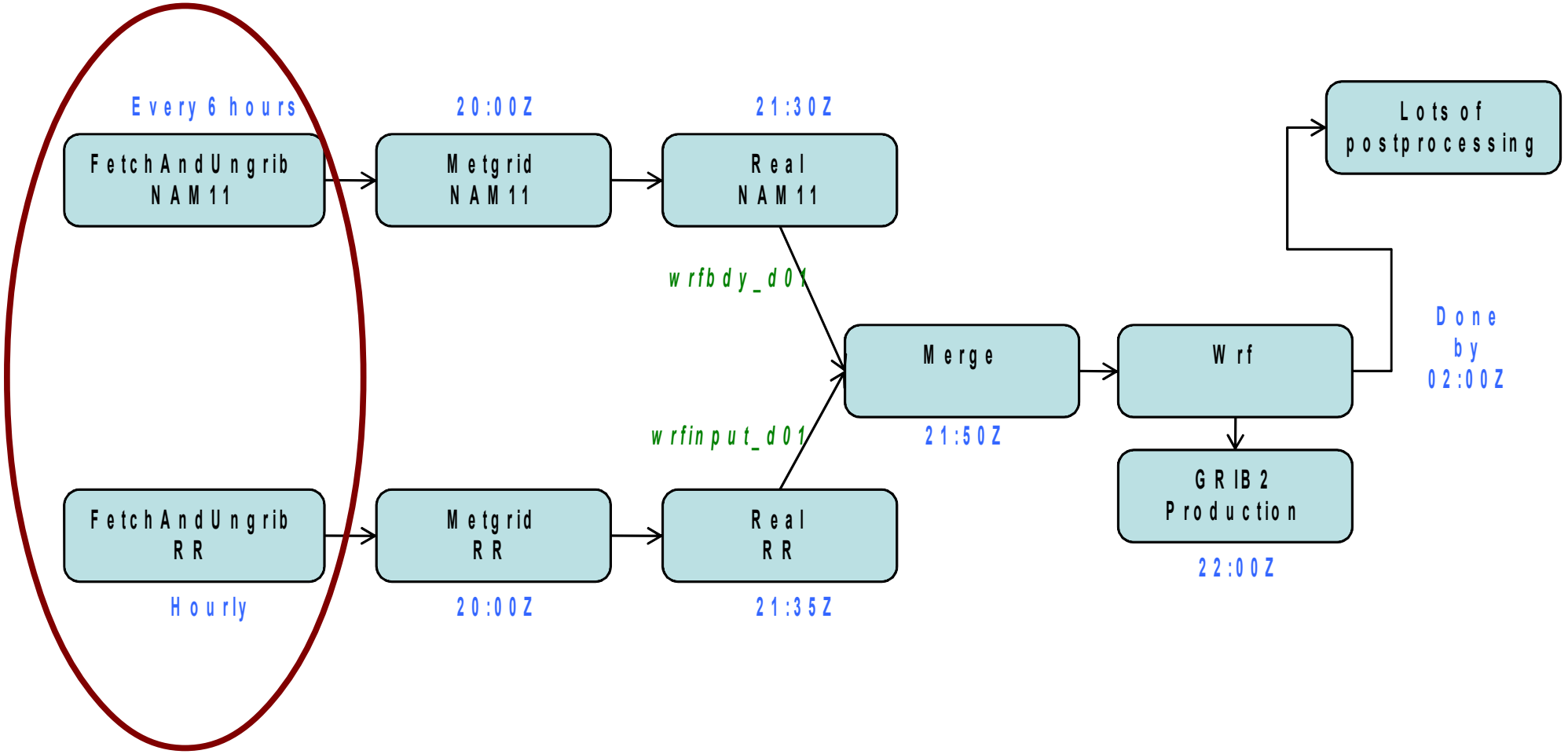
- Want to make you aware of the many issues and potential problems we face
  - We try to handle some potential problems beforehand
  - With other problems we gamble and hope for the best
  - Tradeoffs - there comes a point where increased error handling requires a huge resource commitment, and we must weigh this against the odds and the ramifications of potential problems

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRR-AK Process Flow (00Z Forecast)



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRRAK input data

- Two sources, one every six hours, one hourly
- Sometimes the sources are delayed, so we need to keep trying (but not forever)
- How do we know when a remote file is "complete?" Depends on server capabilities, but sometimes we have to download it first and make sure it exceeds a predetermined threshold size. If not, we try again a few minutes later
- Sometimes file transfers simply fail, so we need to be able to recover, and try again, but not forever

UAF UNIVERSITY OF ALASKA FAIRBANKS

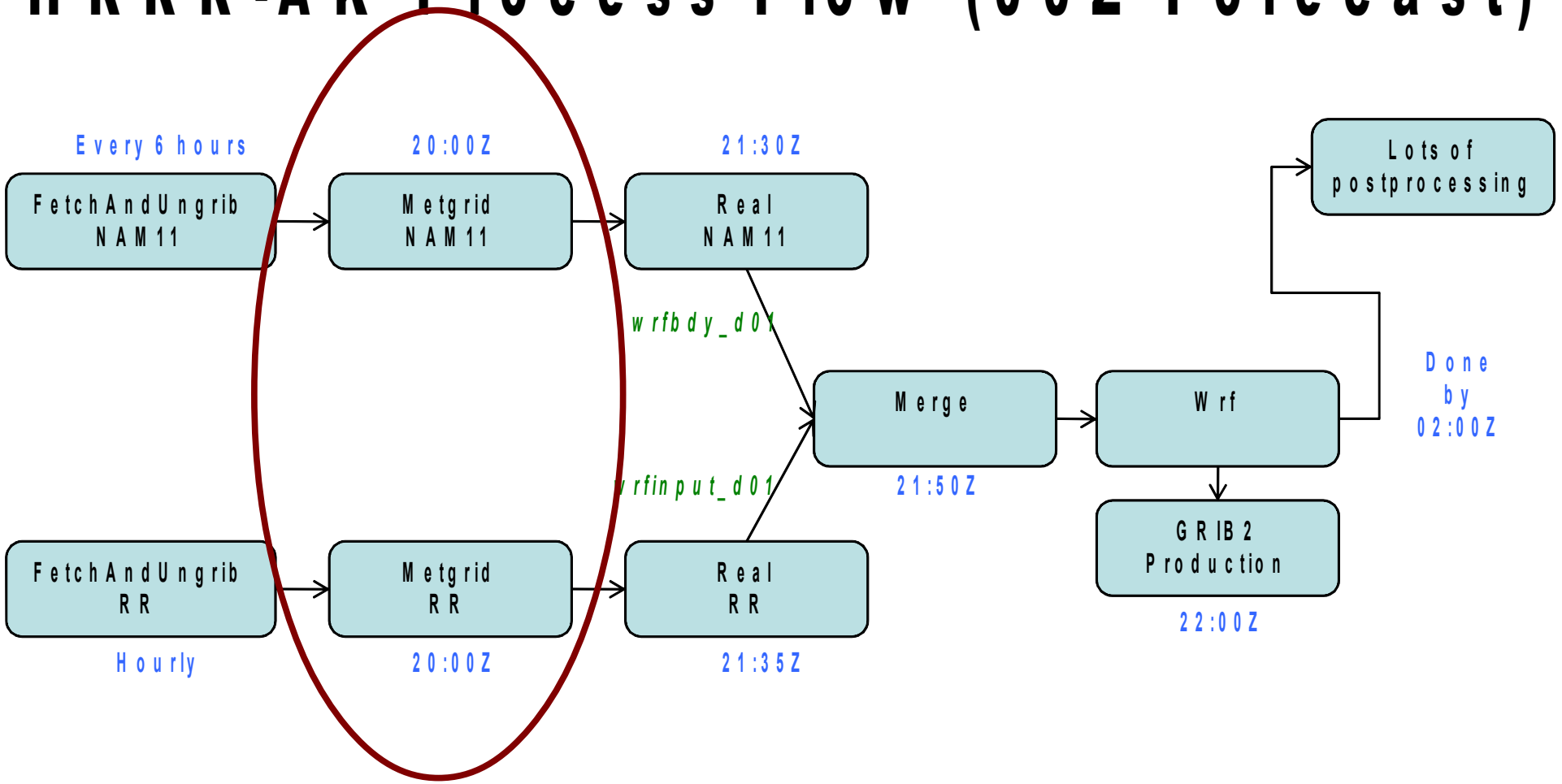
Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRRAK Input Data

```
######## Various criteria for success, retry, or giving up
# If the file size on the server is not at least this big, then don't retrieve
MIN_FILESIZE_THRESHOLD_BYTES = 50000000

MAX_PROCESS_AGE_MINUTES = 120   # Age at which to terminate process

FILESIZE_WAIT_TIME_SECONDS = 120   # How long to wait before trying filesize again
FILESIZE_WAIT_MAX_TRIES = 10       # How many times to try on filesize before
                              # giving up
FILERETRIEVE_WAIT_TIME_SECONDS = 60   # How long to wait before trying file retrieve
again
FILERETRIEVE_WAIT_MAX_TRIES = 5  # How many times to try on file retrieve before
                                 # giving up
SOCKET_TIMEOUT_SECONDS = 60      # How many seconds before a socket times out
```

HRRR-AK Process Flow (00Z Forecast)



UNIVERSITY OF ALASKA FAIRBANKS

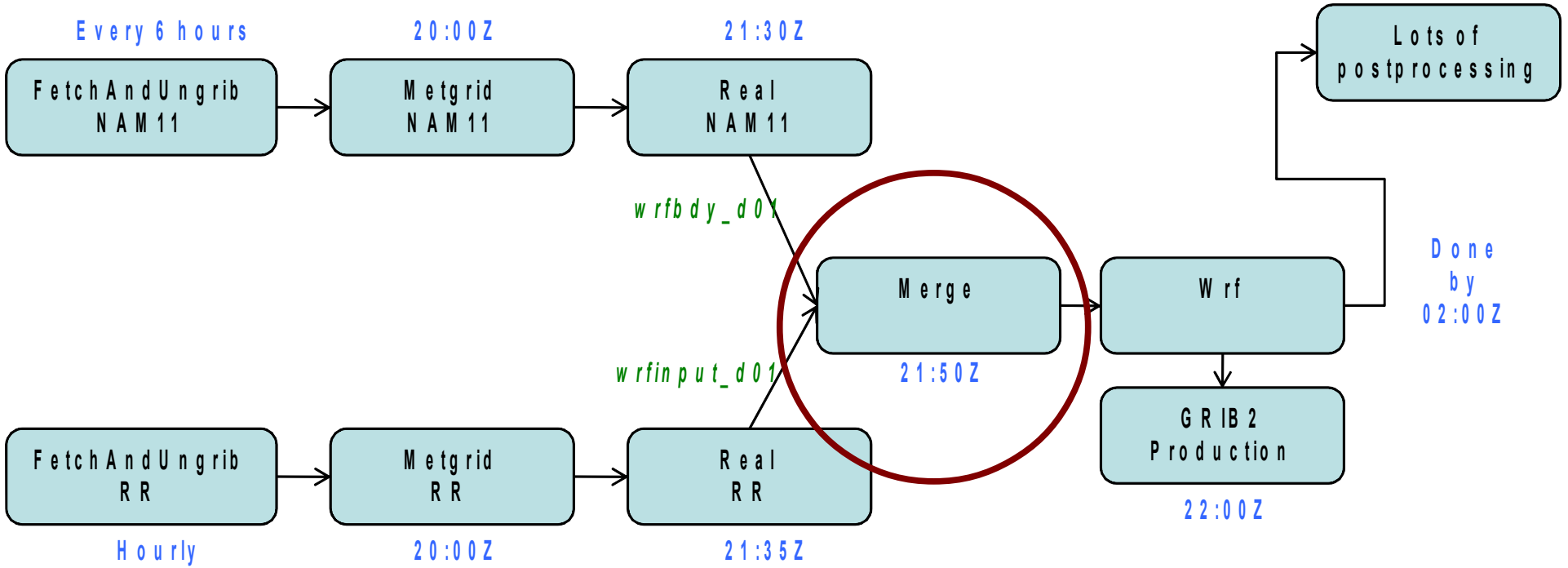
Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRRAK Metgrid

- Transforms input data to a specified computational grid
  - This is done for both NAM and RAP input data
  - Metgrid for the 00Z run needs to run at 20Z, or not much later
  - The computational grid is static and pre-specified
  - The input data may or may not be there
    - If it's there, but it's not complete, or it's not there at all what do we do? Settle for a shorter forecast? Or maybe we try the previous set of input data (try starting at forecast hour 6 of the 18Z input data). What if the previous set of input data isn't there, or is incomplete? Go back to previous one?

HRRR-AK Process Flow (00Z Forecast)



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRRAK Merge

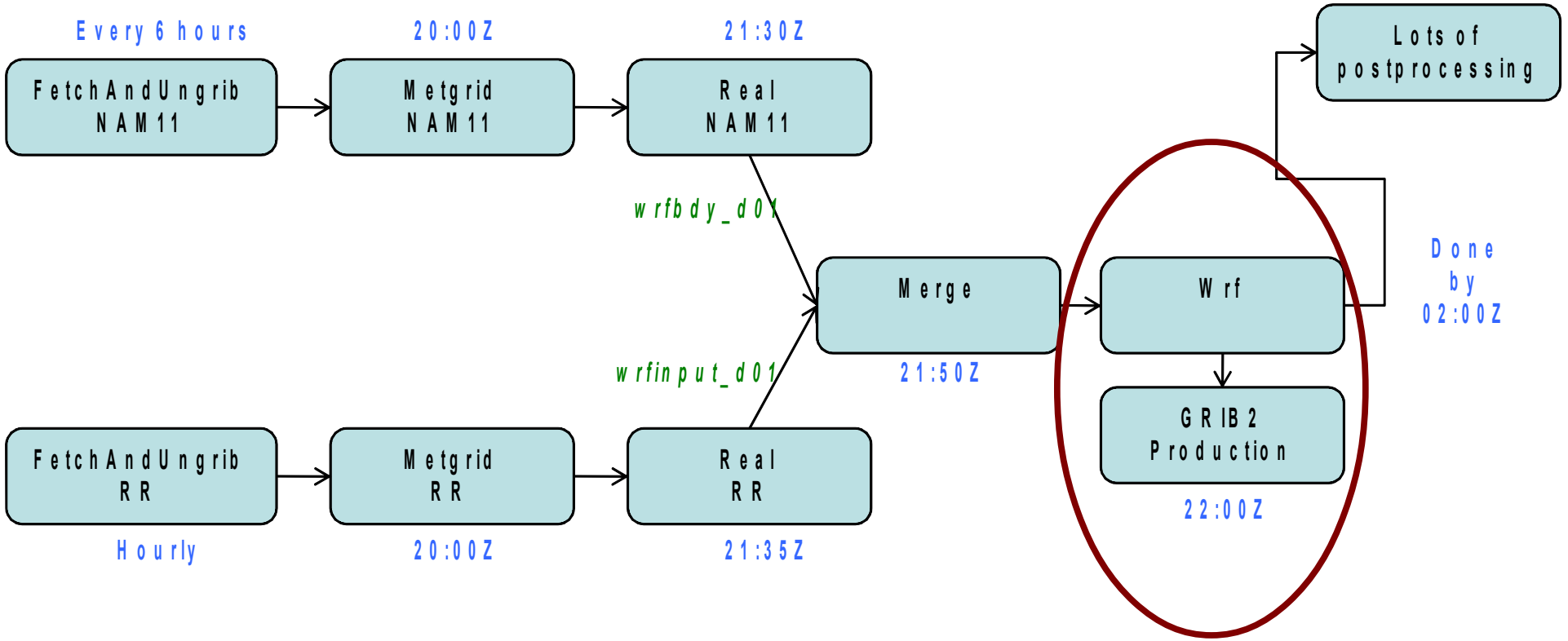
- Merges the NAM LBC's and the RAP initial conditions
  - We need the outputs of real by about 21:50Z
  - What if they're not available? Wait and try again, but not forever? We need to move with the WRF run
  - What if one is available, especially the NAM? Use its initial condition file instead of the RAP, and run the HRNAMAK instead?

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRR-AK Process Flow (00Z Forecast)



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRRAK WRF run and GRIB production

- 00Z WRF run needs its initial and lateral boundary conditions by 22Z. If not available, then we abort, because we don't want to delay the 06Z forecast
- After WRF is launched, we launch processes to monitor the presence of wrfout netCDF files. When the next one is available, numerous processes start up to create GRIB files and push them to an LDM server
- We used to do GRIB production in a single, monolithic program, but performance was horrible. Now, for each forecast hour, we launch 9 GRIB2 production processes, each using 16 cores of supercomputer

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

HRRRAK Postprocessing

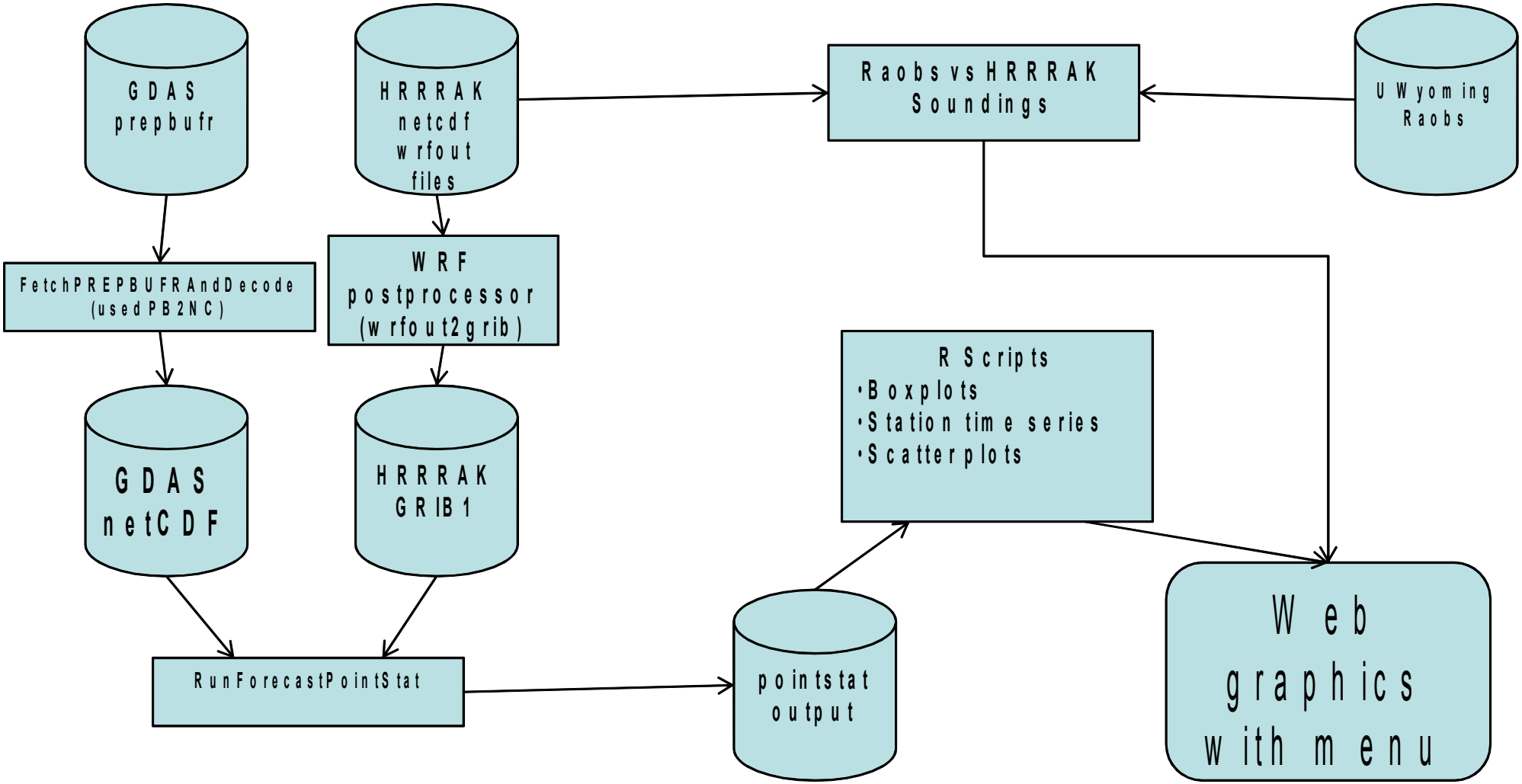
- Real-time GRIB files for forecast offices
- Close to real-time web products for public and PR purposes
- Forecast verification
  - Downloading of observations
  - Extracting point forecasts from the wrfout files
  - Producing tables and graphics of comparisons
  - Also comparing WRF soundings with raobs
- Weekly, monthly, seasonal verifications
- Archiving

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Verification Post Processing



Archived long-term

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Additional postprocessing

- Now converting wrfout files to ARL format for HYSPLIT (and AER, for time averaged fields) - every forecast
- Cleanup - each forecast uses 3-5 TBytes (3-5 million Gigabytes) of storage
  - Need to purge soon, but not too soon
  - Need to make sure they stay around long enough

Motivations for Python (reference the BAMS)

- BAMS, December 2012, Why Python is the Next Wave in Earth Sciences Computing, Johnny Wei-Bing Lin
- “Python is executable pseudocode” - programs are clear and easy to read
- Interpreted, object-oriented
- Open source
- Very portable

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Introduction to Python

- General-purpose, high-level programming language
- Design philosophy emphasises code readability
- Allows programmers to express concepts in fewer lines of code than many other languages like C and Fortran
- Supports object-oriented (e.g. Java), imperative (e.g. Fortran, C) and functional programming (e.g. LISP) styles

Introduction to Python

- With a relatively simple core, designed to be highly extensible
- Core philosophies
  - Beautiful is better than ugly.
  - Explicit is better than implicit.
  - Simple is better than complex.
  - Complex is better than complicated.
  - Readability counts.
- “To describe something as clever is *not* considered a compliment in the Python culture”, Alex Martelli

Introduction to Python

- Conceived in late 1980's with implementation beginning in 1989
- Guido van Rossum is Python's principal author and continues to play a central role. Affectionately referred to in the Python community as Benevolent Dictator for Life
- Name derived from *Monty Python's Flying Circus*
- Python 2.0 released in 2000
  - Last version 2.7
- Python 3.0 released 2008
  - Current production version 3.3.1
  - Development version 3.4

Python 2 or 3?

- Python 3.0 is know as the first ever *intentionally backward* *incompatible* Python release.
- Python 2 final version is 2.7 with statement of extended support. 2.x branch will see no new major releases.
- Python 3.x under active development, and improvements in libraries won't be reflected back into 2.x
- As a language, Python 3.x is ready, but external library support is still lacking in some areas (e.g. PIL).
- Python 2.x is better supported in IT environments
- Don's opinion - these are days where you might want to consider Python 3, but be prepared for some challenges, particularly in the area of third-party library support.

Python's “Beautiful Heart”

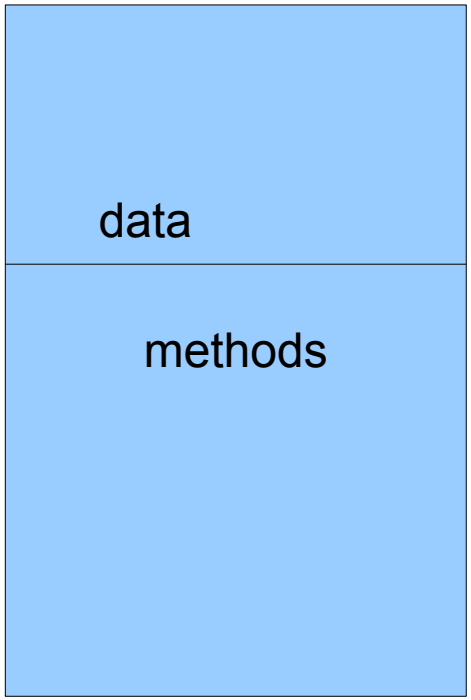
- Reference: *Programming in Python 3: A Complete Introduction to the Python Language,* Mark Summerfield
- Python is incredibly rich in features with many depths of complexity that can seem overwhelming
- However, understanding a few key features allows you to do almost anything you want, and that's where our focus will be
- Additional features will be learned over time as need and interest dictate

Python's “Beautiful Heart”

- Eight key features
  - Data Types
  - Object References
  - Collection Data Types
  - Local Operations
  - Control Flow Statements
  - Arithmetic Operators
  - Input/Output
  - Creating and calling functions

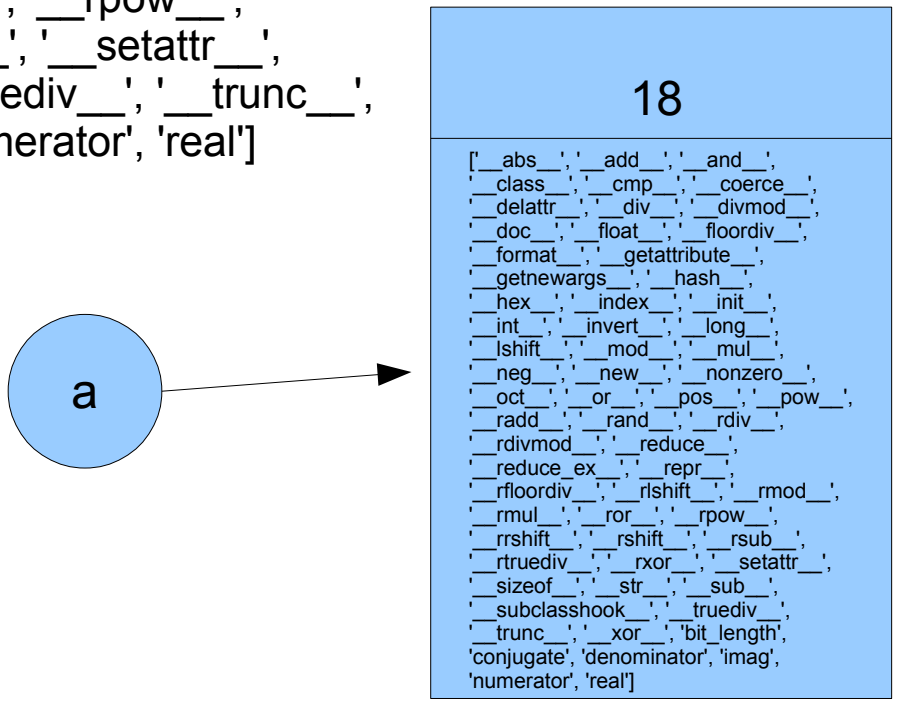
Object Implementation

- Everything in Python is an object
- In short, an object is an entity with data and methods (for accessing the data)
- But, we often use the terms variable and object interchangeably



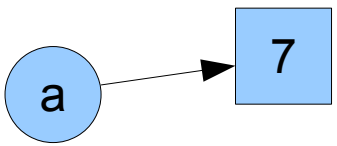
Object Implementation

```
>>> a = 18
>>> dir(a)
['__abs__', '__add__', '__and__', '__class__', '__cmp__', '__coerce__', '__delattr__',
'__div__', '__divmod__', '__doc__', '__float__', '__floordiv__', '__format__',
'__getattribute__', '__getnewargs__', '__hash__', '__hex__', '__index__', '__init__',
'__int__', '__invert__', '__long__', '__lshift__', '__mod__', '__mul__', '__neg__',
'__new__', '__nonzero__', '__oct__', '__or__', '__pos__', '__pow__', '__radd__',
'__rand__', '__rdiv__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__',
'__rfloordiv__', '__rlshift__', '__rmod__', '__rmul__', '__ror__', '__rpow__',
'__rrshift__', '__rshift__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__',
'__sizeof__', '__str__', '__sub__', '__subclasshook__', '__truediv__', '__trunc__',
'__xor__', 'bit_length', 'conjugate', 'denominator', 'imag', 'numerator', 'real']
>>> x = a.__hex__()
>>> print x
0x12
>>> y = a.__float__()
>>> print y
18.0
>>> z = a.__add__(32)
>>> print z
50
```

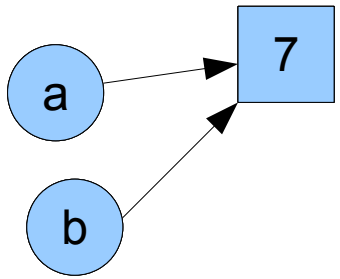


Object References

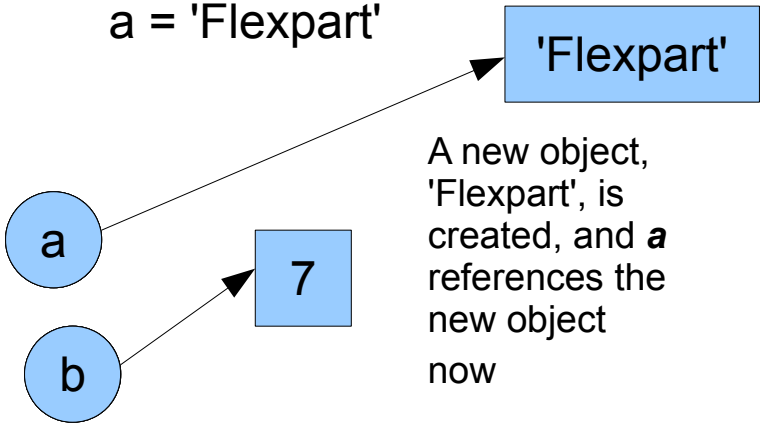
a = 7



b = a



a = 'Flexpart'



A new object, 'Flexpart', is created, and ***a*** references the new object now

Example of reading in data and storing in collection data types

Reading in real-time fire emissions data (Fire Information for Resource Management System - FIRMS)

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

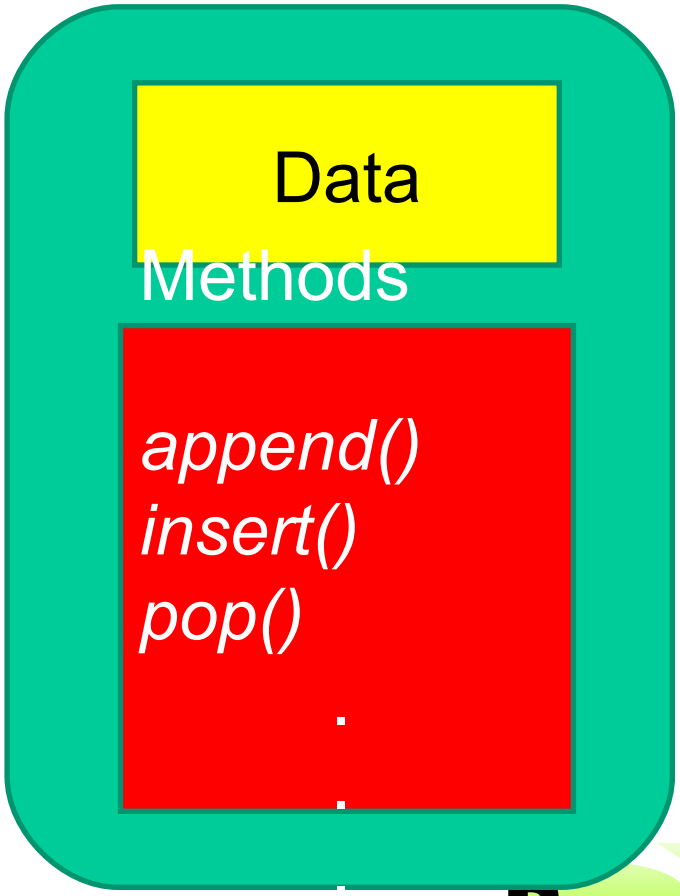
Arctic Region Supercomputing Center

Overview of modules and objects

- The core of Python is relatively small and simple, though you can do many things with just this much.
- Most of the power of the Python environment comes from its standard library in the form of modules, 3rd party packages and modules, and your own user-defined modules

Concepts of Object-Oriented Programming

Abstract *list()* class



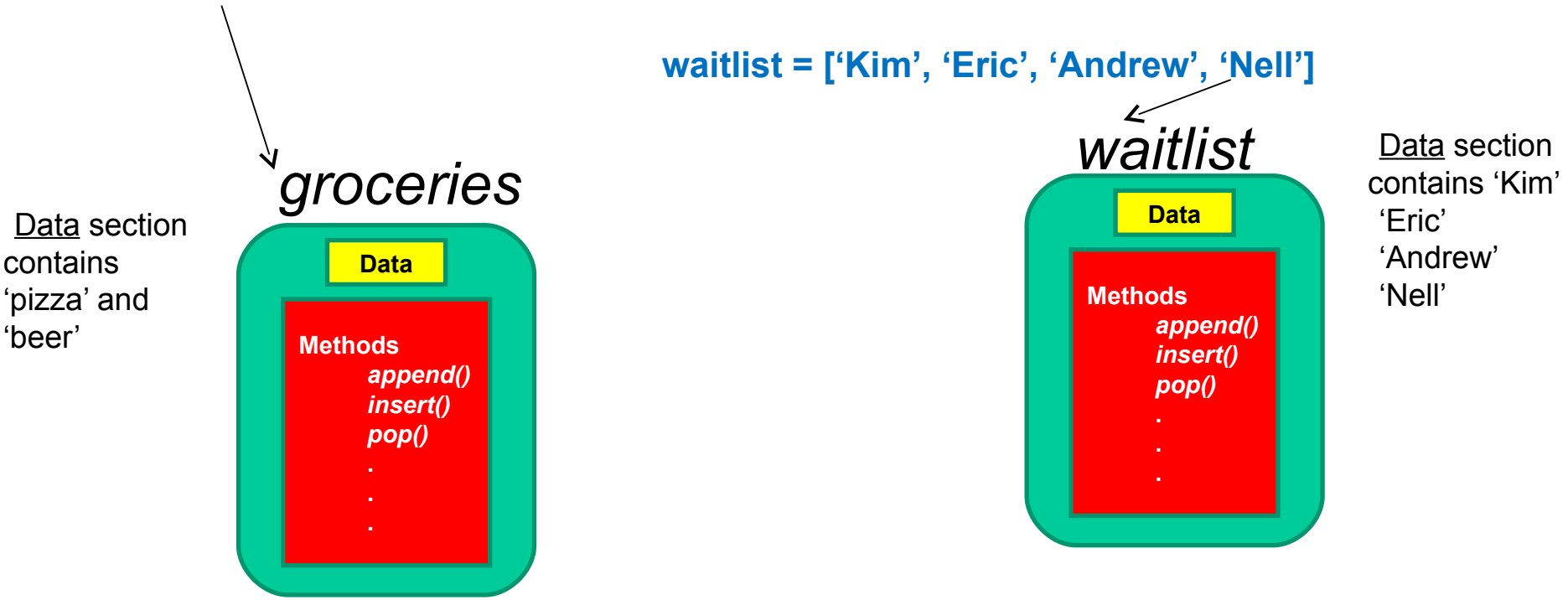
- objects from the *list()* class.
- These objects can be viewed as a black box
  - Details hidden from user
  - Well-defined and controlled access to the data, via methods

Concepts of Object-Oriented Programming

- **We can create instantiations of the list class**

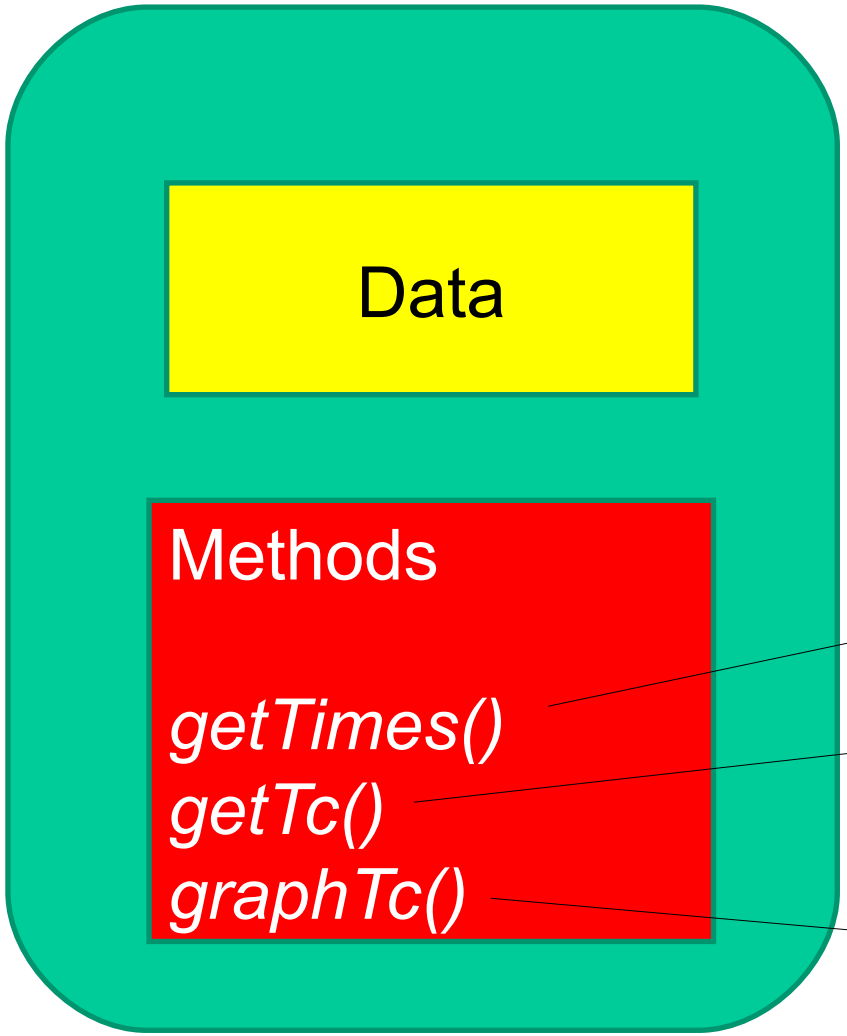
```
groceries = list()
groceries.append('pizza')
groceries.append('beer')
```

```
waitlist = ['Kim', 'Eric', 'Andrew', 'Nell']
```



Concepts of Object-Oriented Programming

Abstract *MetFetcher()* class



Returns a list of observation times

Returns a list of observed temperatures in degrees C

Displays a graph of observed temperature in degrees C vs. observation time

- **Here is an abstract representation of a simple class that I created, called MetFetcher()**
- **MetFetcher serves as a *black box*, getting recent meteorological observations from a specified weather station.**

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

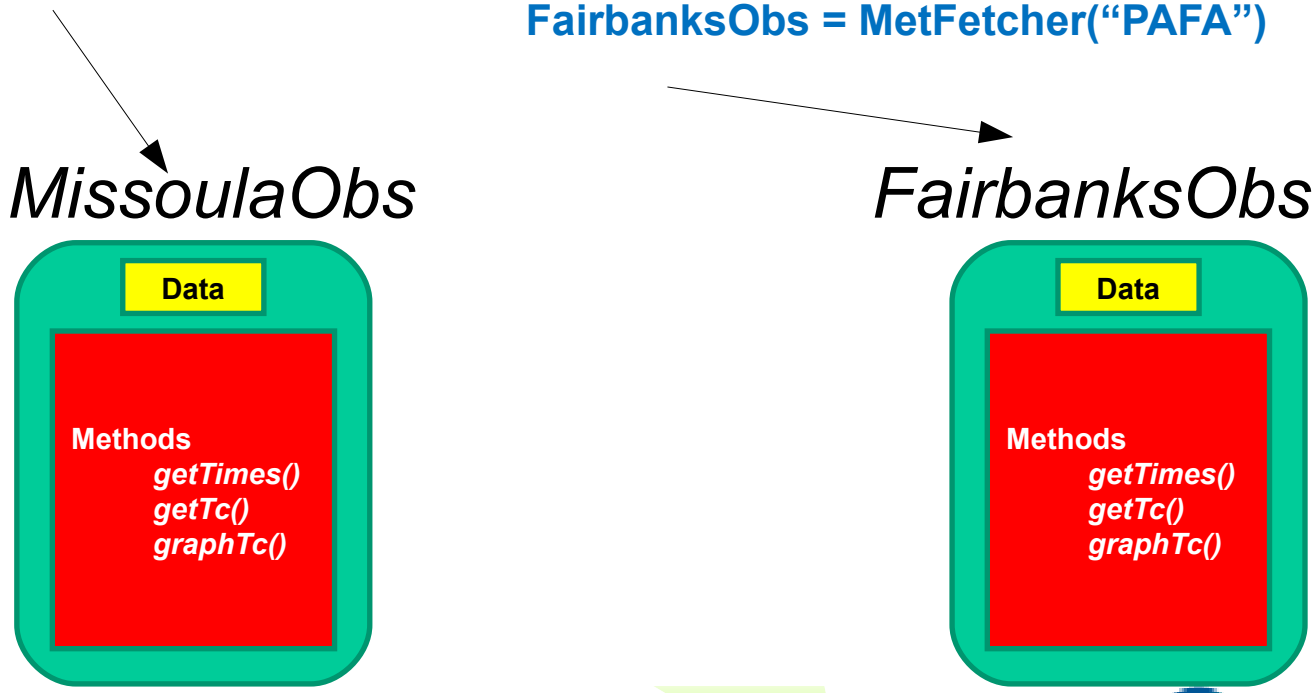
Arctic Region Supercomputing Center

Concepts of Object-Oriented Programming

- **MetFetcher serves as a black box, getting recent meteorological observations from a specified weather station.**
- **You can create <u>instances</u> of the class, as follows**

MissoulaObs = MetFetcher("KMSO")

FairbanksObs = MetFetcher("PAFA")



Data section contains met data for Missoula

Data section contains met data for Fairbanks

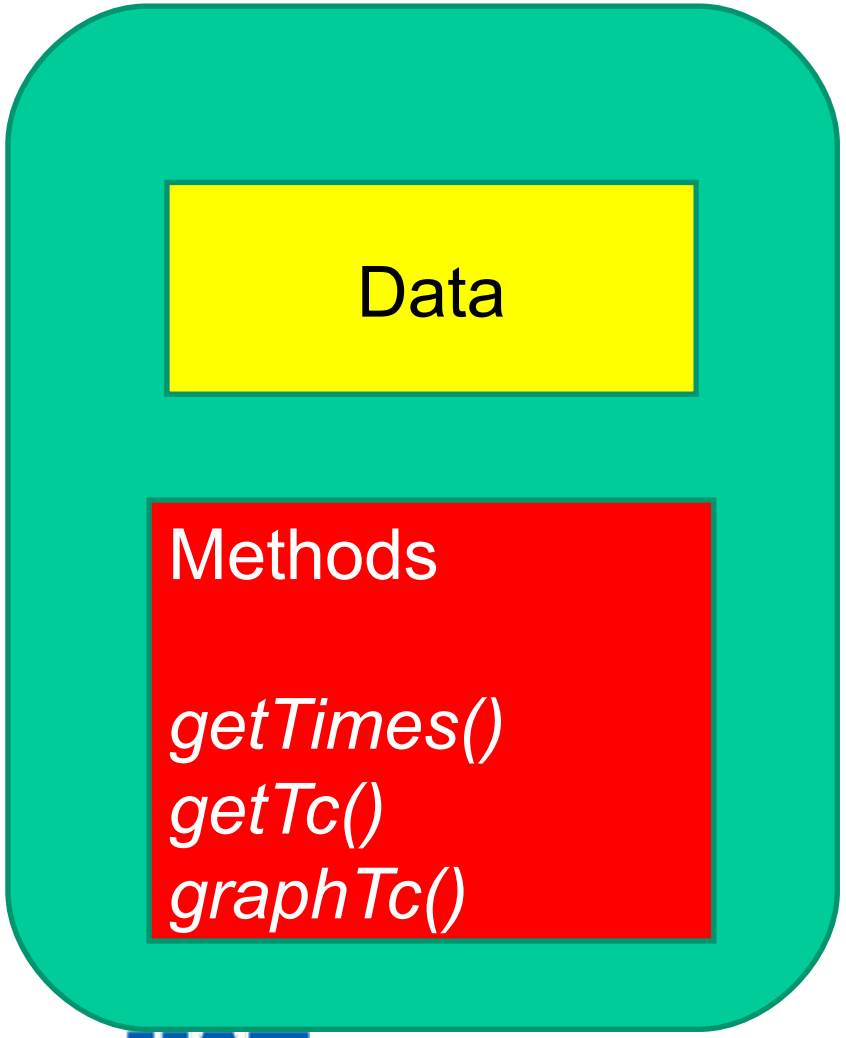
UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Concepts of Object-Oriented Programming

MissoulaObs() object



This is what some of the raw data looks like inside the object, but you don't care about that. You only care about using the supplied methods to get certain data out in a certain format

['<pre>StnId,Lat,Lon,Elev,Time,Tmp,Dwpt,RH,WndDir,WndSpd,WndGst,Vsby,WX,WX,Clds,SLP,Al
tim,StnPrs,Pcpn1hr,Pcpn3hr,Pcpn6hr,Pcpn24hr,6hrMaxT,6hrMinT,24hrMaxT,24hrMinT,QC,\n',
'KMSO,46.92,-
114.09,3199,200809051853,55,41,59,210,3,,10.00,Clear,,OVC060,1020.9,30.15,26.837,T,,,,,,,,OK,\n',
'KMSO,46.92,-114.09,3199,200809051953,55,44,66,250,3,,10.00,Lt Rain,-RA,FEW038
BKN055,1021.3,30.15,26.837,0.01,,,,,,,,OK,\n', 'KMSO,46.92,-
114.09,3199,200809052053,56,45,67,0,,,10.00,Lt Rain,-
RA,SCT050,1021.4,30.16,26.846,0.01,0.02,,,,,,,OK,\n', 'KMSO,46.92,-
114.09,3199,200809052153,55,47,74,110,7,,9.00,Lt Rain,-RA,SCT041
BKN050,1021.2,30.15,26.837,0.01,,,,,,,,OK,\n', 'KMSO,46.92,-
114.09,3199,200809052253,56,47,72,110,7,,10.00,Lt Rain,-
RA,SCT055,1020.6,30.13,26.818,T,,,,,,,,OK,\n', 'KMSO,46.92,-
114.09,3199,200809052353,57,46,67,190,5,,10.00,Clear,,OVC080,1020.4,30.12,26.809,T,,0.03,,57,52,,,
OK,\n', 'KMSO,46.92,-
114.09,3199,200809060053,56,48,75,160,7,,10.00,Clear,,OVC070,1020.2,30.11,26.800,,,,,,,,,OK,\n',
'KMSO,46.92,-114.09,3199,200809060153,56,46,69,80,5,,10.00,Clear,,BKN080
OVC090,1019.8,30.10,26.791,,,,,,,,,OK,\n', 'KMSO,46.92,-
114.09,3199,200809060253,54,46,75,180,3,,10.00,Clear,,OVC095,1019.5,30.10,26.791,,,,,,,,,OK,\n',
'KMSO,46.92,-114.09,3199,200809060353,52,45,77,320,3,,1

UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Concepts of Object-Oriented Programming

Sequence of Python statements

```
MissoulaObs  =  MetFetcher("KMSO")

theTimes  =  MissoulaObs.getTimes()
theTemps  =  MissoulaObs.getTc()

print  theTimes
print  theTemps
```

Output

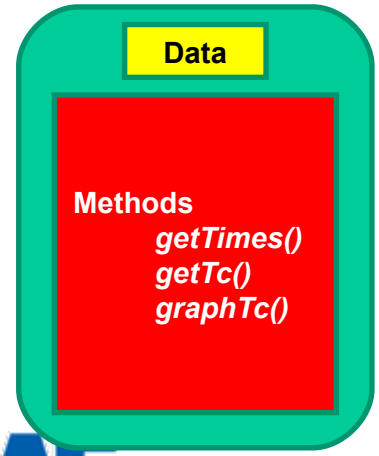
Python Shell

File Edit Shell Debug Options Windows Help

```
['200809051853', '200809051953', '200809052053', '200809052153', '200809052253',
 '200809052353', '200809060053', '200809060153', '200809060253', '200809060353',
 '200809060453', '200809060553', '200809060653', '200809060753', '200809060853',
 '200809060953', '200809061053', '200809061153', '200809061253', '200809061353',
 '200809061453', '200809061553', '200809061653', '200809061753', '200809061853',
 '200809061953', '200809062053', '200809062153', '200809062253', '200809062348',
 '200809062353', '200809062357', '200809070017', '200809070053', '200809070153',
 '200809070253', '200809070353', '200809070453', '200809070512', '200809070553',
 '200809070653', '200809070753', '200809070826', '200809070853', '200809070946',
 '200809070953', '200809071000', '200809071053', '200809071107', '200809071137',
 '200809071153', '200809071155', '200809071208', '200809071253', '200809071353',
 '200809071453', '200809071553', '200809071653', '200809071753']
[12.777777777777779, 12.777777777777779, 13.333333333333334, 12.777777777777779,
 13.333333333333334, 13.888888888888889, 13.333333333333334, 13.333333333333334,
 12.222222222222223, 11.111111111111111, 11.111111111111111, 10.555555555555555,
 10.555555555555555, 10.0, 7.777777777777786, 7.777777777777786, 8.33333333333
33339, 7.777777777777786, 6.666666666666667, 7.2222222222222223, 9.444444444444
4446, 11.666666666666668, 13.888888888888889, 16.111111111111111, 17.77777777777
7779, 19.444444444444446, 19.444444444444446, 19.444444444444446, 19.44444444444
4446, 13.888888888888889, 14.444444444444445, 15.0, 15.0, 15.0, 14.4444444444444
45, 13.888888888888889, 12.777777777777779, 13.333333333333334, 12.2222222222222
23, 12.222222222222223, 12.222222222222223, 12.222222222222223, 12.2222222222222
23, 11.111111111111111, 11.111111111111111, 10.555555555555555, 11.1111111111111
11, 11.111111111111111, 11.111111111111111, 11.111111111111111, 10.0, 10.0, 10.0
, 10.0, 10.0, 11.111111111111111, 12.777777777777779, 13.333333333333334, 14.444
444444444445]
>>>
```

Ln: 512 Col: 115

MissoulaObs



UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

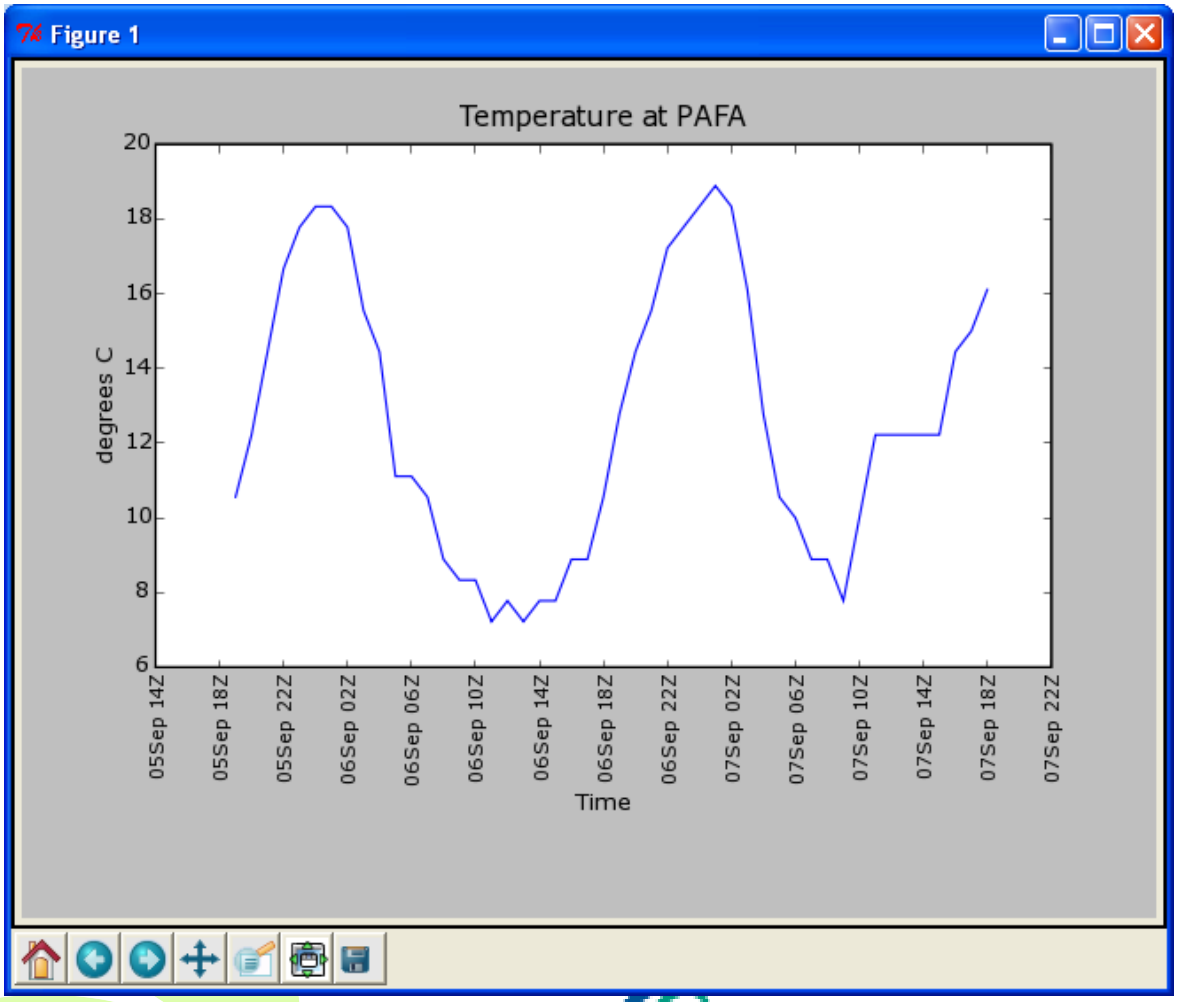
Arctic Region Supercomputing Center

Concepts of Object-Oriented Programming

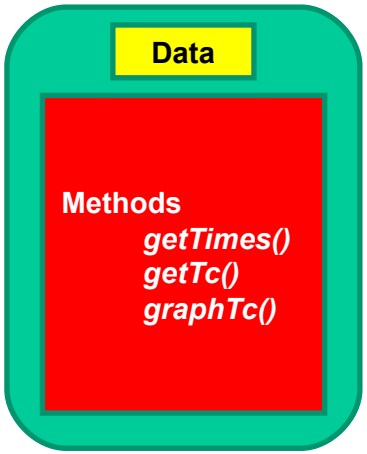
Sequence of Python statements

```
FairbanksObs  =  MetFetcher("PAFA")

FairbanksObs.graphTc()
```

Output



FairbanksObs



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Preparing for next lab

(From operational Python workshop, April 2012, Vienna, Austria)

- I have prepared a class, *FIRMS*, for retrieving satellite data to be used as fire emissions
- You will import the class, study it, use it, and make minor enhancements to it
- This class will be a part of the overall FLEXPART operational system

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

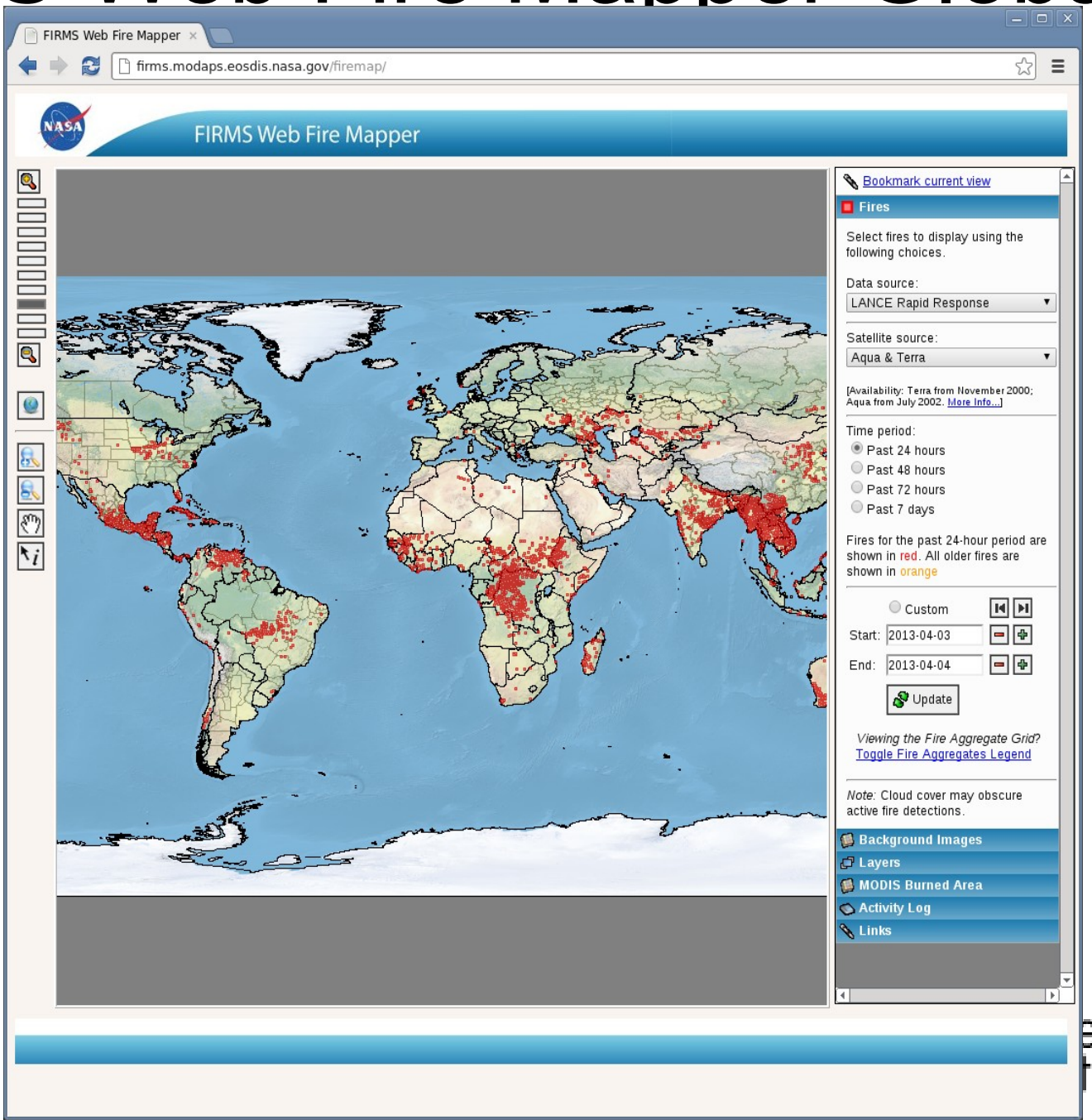
FIRMS Data Access page

FIRMS: Fire Information for Resource Management System

Delivers MODIS hotspots / fire information in easy to use formats.

UAF UNIVERSITY OF ALASKA FAIRBANKS

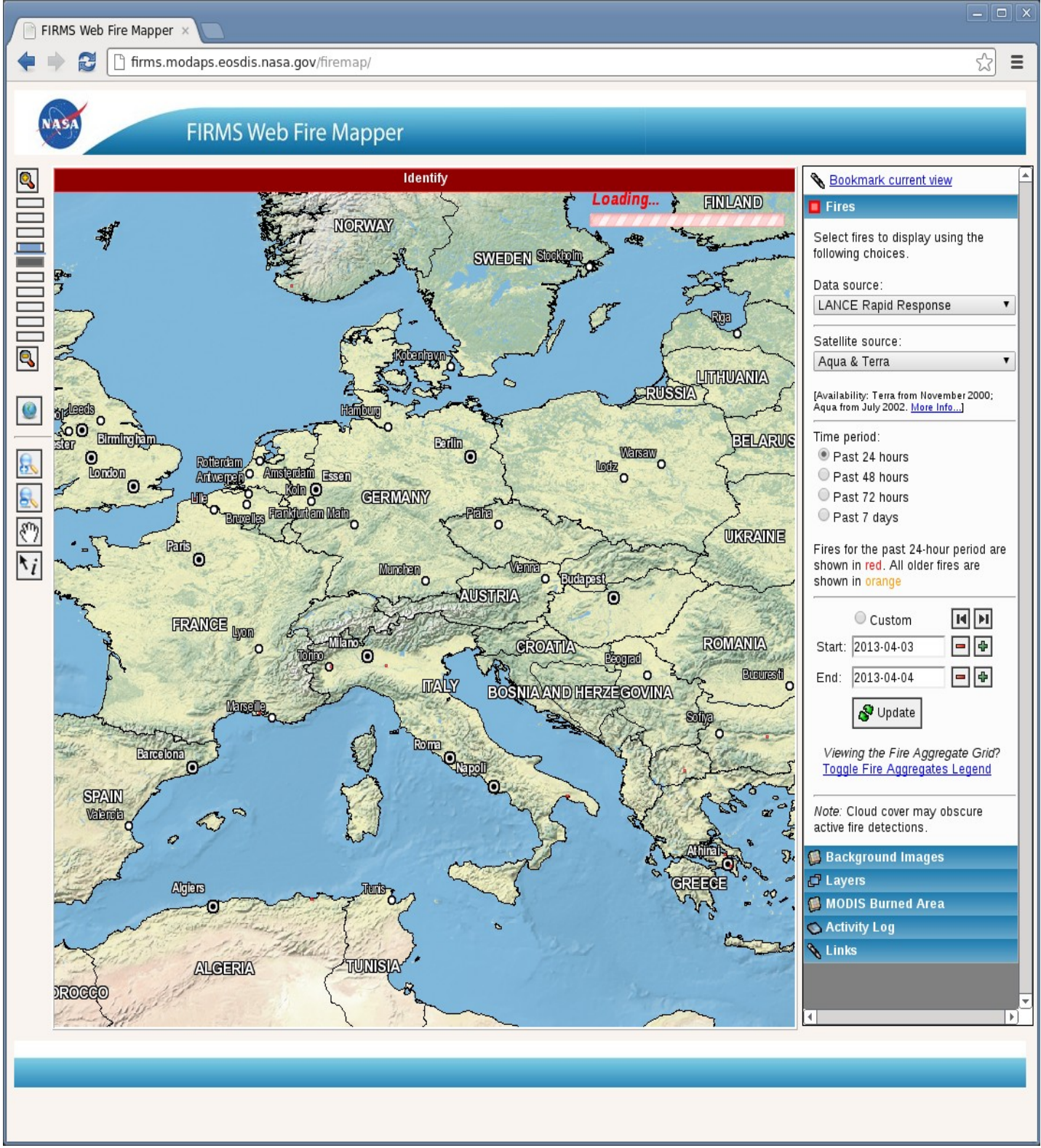
FIRMS Web Fire Mapper Global



UAF UNIVERSITY OF ALASKA FAIRBANKS

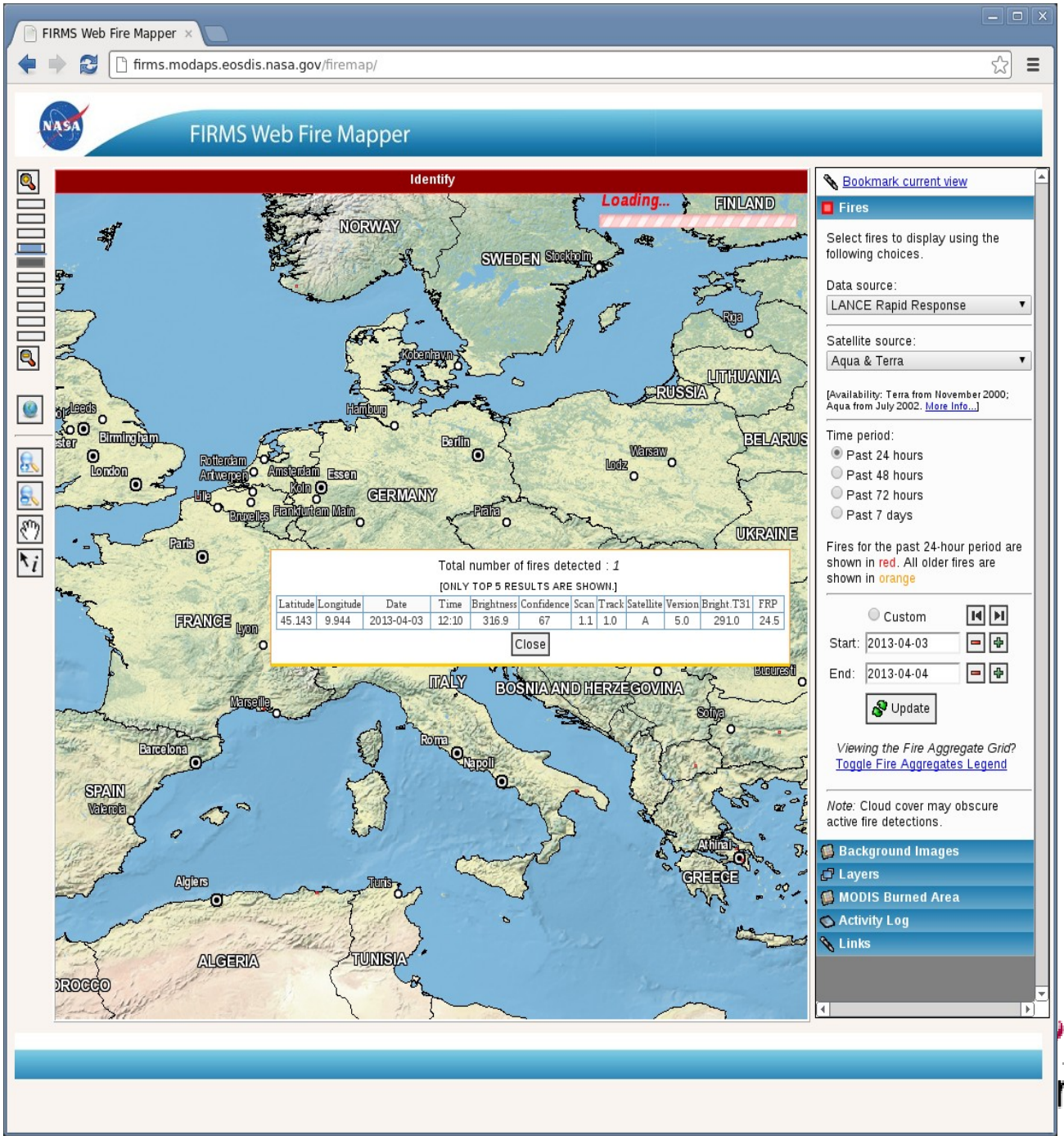
egion
ting Center

Web Fire Mapper Zoom EU



UNIVERSITY OF ALASKA FAIRBANKS

Web Fire Mapper Zoom EU Italy Info



UAF UNIVERSITY OF ALASKA FAIRBANKS

Arctic Region
rcomputing Center

FIRMS Raw EU Data

morton@localhost:~/Administrative/ZAMGCourse2013/LabExerciseData/FIRMS

```
latitude,longitude,brightness,scan,track,acq_date,acq_time,satellite,confidence,version,bright_t31,frp
40.513,17.215,305.2,1.1,1,2013-04-03, 0105,A,22,5.0       ,283.2,16.7
36.88,6.953,309.3,1.8,1.3,2013-04-03, 0105,A,65,5.0       ,283.9,46.3
49.786,32.784,312.4,1.1,1,2013-04-03, 0845,T,54,5.0       ,279.7,22.3
47.859,33.448,315.3,1,1,2013-04-03, 0845,T,54,5.0       ,292.5,16.3
36.879,6.943,341.8,1,1,2013-04-03, 1030,T,69,5.0       ,297.9,53
36.877,6.954,333.8,1,1,2013-04-03, 1030,T,65,5.0       ,298.2,38.6
47.676,34.736,313.9,1,1,2013-04-03, 1030,A,59,5.0       ,294.8,12.9
47.627,34.32,313.2,1.1,1,2013-04-03, 1030,A,57,5.0       ,294.3,12.8
48.81,32.536,321.2,1.1,1.1,2013-04-03, 1030,A,74,5.0       ,289.7,31.5
49.89,34.229,314.2,1,1,2013-04-03, 1030,A,60,5.0       ,286,19.2
43.434,4.887,310.3,2.1,1.4,2013-04-03, 1210,A,31,5.0       ,296.3,32.1
45.143,9.944,316.9,1.1,1,2013-04-03, 1210,A,67,5.0       ,291,24.5
45.106,7.725,311.1,1.3,1.1,2013-04-03, 1210,A,17,5.0       ,293.6,14.5
51.031,2.27,322.7,1.7,1.3,2013-04-03, 1210,A,57,5.0       ,283.3,64.9
57.316,-5.573,379.6,1.2,1.1,2013-04-03, 1200,T,87,5.0       ,280.4,212.6
56.879,-5.52,310.4,1.2,1.1,2013-04-03, 1200,T,38,5.0       ,288.3,22.7
54.091,-8.969,313.7,1.1,1,2013-04-03, 1205,T,59,5.0       ,282.7,23.8
36.869,6.951,316.5,2.3,1.5,2013-04-03, 1205,A,66,5.0       ,299.1,49.5
36.878,6.943,326.7,2.3,1.5,2013-04-03, 1205,A,0,5.0       ,295.2,93.9
52.134,-9.446,331.1,1.3,1.1,2013-04-03, 1350,A,84,5.0       ,284.8,61.8
53.356,-9.302,311.3,1.4,1.2,2013-04-03, 1350,A,48,5.0       ,284.4,26.8
54.082,-8.977,314.6,1.4,1.2,2013-04-03, 1350,A,61,5.0       ,282.5,35.7
                                                                    1,1           Top
```

We can download raw text data for past 24 hours, and for different regions - e.g. http://firms.modaps.eosdis.nasa.gov/active_fire/text/Europe_24h.csv

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Use of FIRMS data

- We will use FIRMS data - in a tutorial setting - to estimate fire emissions. Simplifying assumptions, for academic purposes
  - Just to get relative emissions values, we will multiply a pixel's Fire Radiance Power (FRP) to represent the mass emissions over the duration of the FLEXPART simulation
  - We assume that FRP is constant throughout the simulation
  - Though the data contains confidence measures, we will assume all data is good

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Use of FIRMS data

- The approach - use a Python class, FIRMS, in which to store and access this data.

MODIS/Active fire data
Methods: FIRMS(sourceOfData) printFullListing() setRegion(boundingBox) printRegionalListing() getNumRegionalReleases() getRegionalData() plotRegion(imageFilename)

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

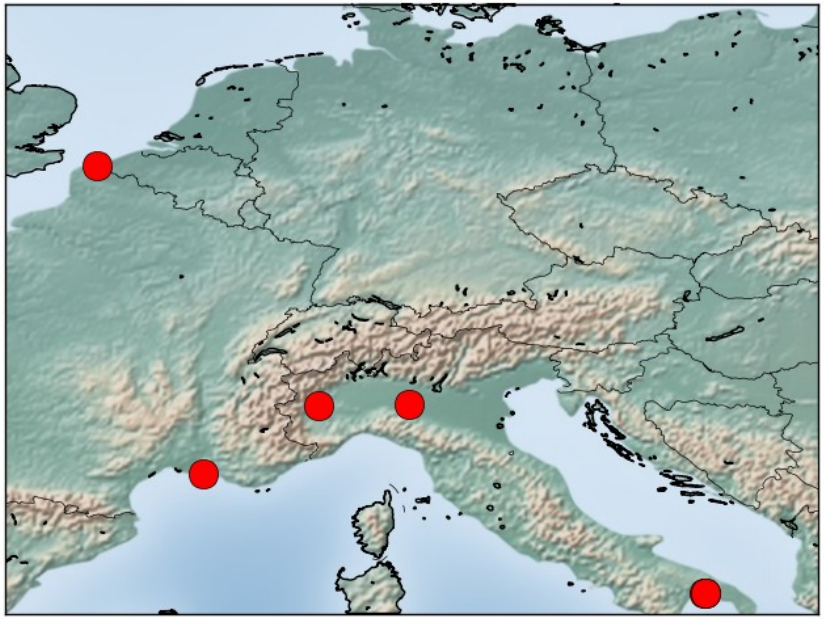
Sample program using FIRMS.py

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

import FIRMS


# European source data
EU_SOURCE_DATA =
'http://firms.modaps.eosdis.nasa.gov/active_fire/text/Europe_24h.csv'

# Alaskan source data
AK_SOURCE_DATA =
'http://firms.modaps.eosdis.nasa.gov/active_fire/text/Alaska_24h.csv'

# Instantiate an object with Europe fire data - this will automatically go
# to the source, retrieve, and store the data
EUDat = FIRMS.FIRMS(EU_SOURCE_DATA)

# Instantiate an object with Alaska fire data
AlaskaDat = FIRMS.FIRMS(AK_SOURCE_DATA)
```

Sample program using FIRMS.py

```
# Print the full listing of the Alaska data
AlaskaDat.printFullListing()

# Create a bounding box (lower left and upper right corners) for EU region
llLat = 40.0;  llLon = 0.0;  urLat = 55.0;  urLon = 20.0

# Store this in a tuple (LL_LAT, LL_LON, UR_LAT, UR_LON)
boundingBox = (llLat, llLon, urLat, urLon)

# Set region, then print listing
EUDat.setRegion(boundingBox)
EUDat.printRegionalListing()

# Plot the region
EUDat.plotRegion('EUFires.png')
```

FIRMS EU BB (40,0,55,20)



UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Miscellaneous Python Solutions

- Submission of jobs into batch queue systems
- Image production with NCL
- Python Numerical and visualisation capabilities
- File retrievals – timeouts and retries
- Process logging

Abstracting batch queue submission for portability

```
import BatchJob
.
.
.
def runMetgrid(metgridDir, numTasks, queueName, groupName, wallMinutes, binaryPath):
    print 'in runMetgrid()...'
    os.chdir( metgridDir )

    theJob = BatchJob.BatchJob('metgridrun', 'pacman_4')

    theJob.setCores(numTasks)
    theJob.setQueue(queueName)
    theJob.setGroup(groupName)
    theJob.setWallTimeMinutes(wallMinutes)
    theJob.setRunCommand(binaryPath)

    theStatements = []
    theStatements.append('echo \'Start: \' `date`')
    theJob.setPreJobStatements(theStatements)

    theStatements = []
    theStatements.append('echo \'Finish: \' `date`')
    theJob.setPostJobStatements(theStatements)

    theJob.blockingSubmit()
```

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Implementation of blocking submit

```
    def blockingSubmit(self):
        print 'In blockingSubmit()'
        print '.... creating the batch script...'
        if self._systemName in ['pacman_4']:
            self._createBatchScript_pacman_4()
            jobID = str( self._submitJob_pacman_4() )
            self._jobSubmitTime = time.time()
            print 'Submitted Job: ' + jobID
            # Sleep for 10 seconds, just giving job plenty of time to
            # be viewable in qstat
            time.sleep(10)

            # Wait for job to finish, or timeout
            # As long as status is 'Q' or 'R', keep waiting, unless timeout
            batchJobDone = False
            while not batchJobDone:
                jobStatus = self._jobStatus_pacman_4(jobID)
                #print 'jobStatus:  ' + jobStatus
                if jobStatus in ['Q', 'R']:
                    time.sleep(self._blockingSleepTimeMinutes*60)
                    if self._isTimedOut():
                        print __name__ + '.blockingSubmit(): block timed out...'
                        batchJobDone = True
                else:
                    batchJobDone = True


        elif self._systemName in ['tana']:
            ...
```

PBS Batch Files

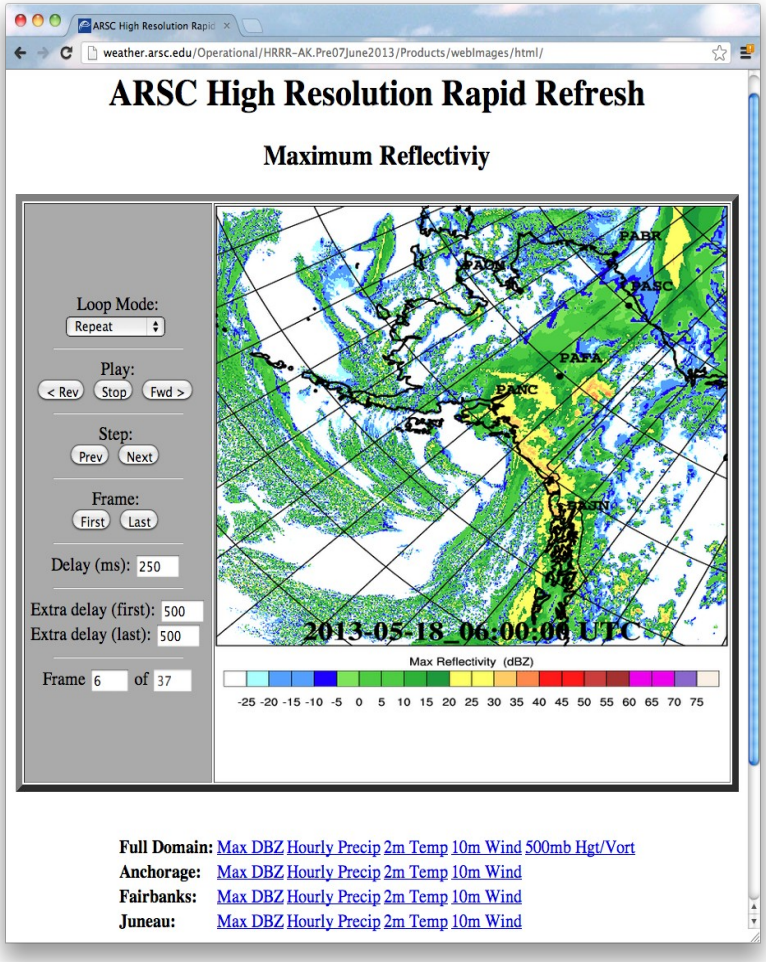
pacman – Penguin computing cluster

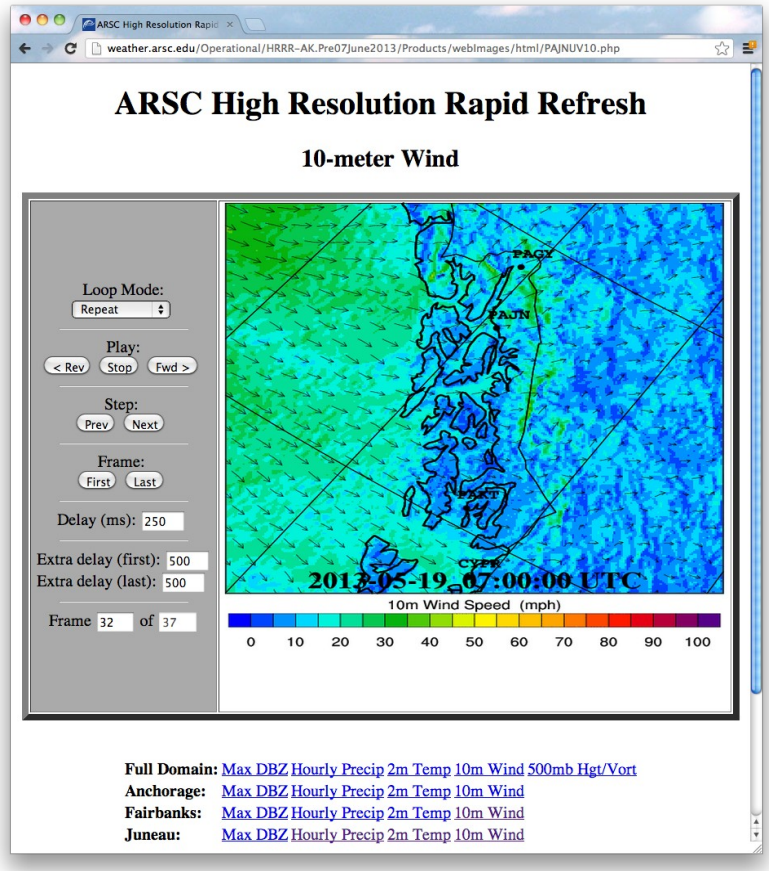
```
#PBS -q arscwthr
#PBS -l nodes=4:ppn=4
#PBS -l walltime=00:20:00
#PBS -j oe

cd $PBS_O_WORKDIR
echo 'Start: ' `date`
mpirun -np 16 ./metgrid.exe
echo 'Finish: ' `date`
```

tana – Cray XE6

```
#PBS -q standard
#PBS -l mppwidth=16
#PBS -l walltime=00:20:00
#PBS -W block=true
#PBS -j oe

cd $PBS_O_WORKDIR
echo 'Start: ' `date`
aprun -n 16 ./metgrid.exe
echo 'Finish: ' `date`
```

Image Production






```
[wrfuser@pacman1:~/WX/Operational/HRRR-AK/PostProcScripts/gifImages]$ ls
CommonDefs.py   fullDBZ.ncl            fullUV10.ncl   localHourlyPrecip.ncl
Common.py       fullHourlyPrecip.ncl   gifDriver.py   localT2.ncl
full500mb.ncl   fullT2.ncl             localDBZ.ncl   localUV10.ncl
```

UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

localT2.ncl excerpt

```
;;;;;;;;  These are used for operational - comment out when testing ;;;;;;;;
  wrfoutFile = getenv("wrfoutFile")  ; Full path to the wrfout file
  imageName = getenv("imageName")    ; File name of image file (without ext)
  REGION_NAME = getenv("regionName")
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  ;; Set up the locale-specific parameters
  if (REGION_NAME .eq. "PAJN") then
     ; Zoom box
     LL_LAT = 52.0
     LL_LON = -135.0
     UR_LAT = 62.0
     UR_LON = -132.0

     LOCATIONS = (/"PAJN", "CYPR",   "PAGY",  "PAKT"/)
     LATS =      (/58.35,   54.30,   59.46,   55.36/)
     LONS =      (/-134.58, -130.43, -135.32, -131.71/)

  else
     if (REGION_NAME .eq. "PAFA") then
         .
         .
         .
```

```
def createImages(field, domainList, NCL_SCRIPT_DIR, wrfoutFileList,
                 WRFOUT_DIR, nestNum, CONVERT_BIN, ARCHIVE_PRODUCT_DIR,
                 MOBILE_ANIMATED_GIF_DIR, WEB_GIF_DIR, wrkdirname):

    for domainName in domainList:
        plotName = domainName + field
        os.environ['regionName'] = domainName

        if domainName == 'Full':
            nclScriptName = 'full' + field + '.ncl'
        else:
            nclScriptName = 'local' + field + '.ncl'


        nclScript = NCL_SCRIPT_DIR + '/' + nclScriptName
        for file in wrfoutFileList:

            # Call NCL script to produce eps file
            os.environ['wrfoutFile'] = WRFOUT_DIR + '/' + \
                                       file + '.nc'

            imageFileName = 'image-' + plotName + '-d0' + \
                        str(nestNum) + '-' + str(imageNum).zfill(3)

            os.environ['imageName'] = imageFileName

            print 'producing: ' + imageFileName

            theCommand = NCL + ' ' + nclScript
            print 'executing: ' + theCommand
            os.system(theCommand)

            # Create the GIF
            outputFilename = imageFileName + '.gif'
            gifCommand = CONVERT_BIN +                  \
                        ' -compress LZW ' +                 \
                        ' -geometry 100%x100% ' +               \
                        ' -background white '

            gifCommand = gifCommand + imageFileName + '.eps ' + \
                        outputFilename
            print 'executing: ' + gifCommand
            os.system(gifCommand)

            ImageNum += 1
```

UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

Python Numerical and Visualisation

- Python is an interpreted language with little emphasis on optimisation, especially for high performance applications
- However, numerous external libraries are available to expand Python's reach into the HPC community
- The lineage of scientific libraries is confusing

Python Scientific Libraries (History)

- *Numeric*, a Python array package originated in 1995 to improve Python for array operations. Reasonably complete and stable, but now obsolete
- *Numarray* was a complete rewrite of *Numeric*, but is also deprecated now (last release 2006)
- SciPy, in 2005 had a subproject to take the best of Numeric and Numarray. This was separated and called NumPy
- NumPy compatible with Python versions 2.4-2.7 and 3.1+

Numerical Python (NumPy)

- Working with arrays and matrices in a natural way
- Package contains large list of mathematical functions
- If available, LAPACK is used for linear algebra routines, enhancing performance.
- Originally part of SciPy, later separated and used by SciPy for array and matrix processing

NumPy - www.numpy.org

- NumPy code is cleaner than native Python code for numerical operations
  - Many operations work on entire matrices, requiring fewer loops
  - Many of the algorithms are mature
- Arrays stored more efficiently than in base Python (which would use a list of lists).
- Performance scales with number of elements in arrays

Python versus NumPy example

- Initialisation and summation of two vectors

Python

```
def pythonsum(n):
    a = range(n)
    b = range(n)
    c = []

    for i in range( len(a) ):
        a[i] = i**2
        b[i] = i**3
        c.append( a[i] + b[i] )

    return c
```

NumPy

```
def numpysum(n):
    import numpy

    a = numpy.arange(n)**2
    b = numpy.arange(n)**3
    c = a + b

    return c
```

seconds

N	Python	NumPy
1000	707	171
4000	2829	274

(NumPy 1.5 Beginner's Guide, Ivan Idris, 2011)

Simple NumPy matrix multiplication example

- Create 2 1x4 float arrays
- Reshape to 2x2
- Matrix multiplication

```
import numpy as np

A = np.array( np.arange(1,5), float )
B = np.array( np.arange(5,9), float )

A = A.reshape(2,2)
B = B.reshape(2,2)

print 'A: ' + str(A)
print 'B: ' + str(B)

C = np.dot(A,B)

print 'C: ' + str(C)
```

```
A: [[ 1.  2.]
 [ 3.  4.]]
B: [[ 5.  6.]
 [ 7.  8.]]
C: [[ 19.  22.]
 [ 43.  50.]]
```

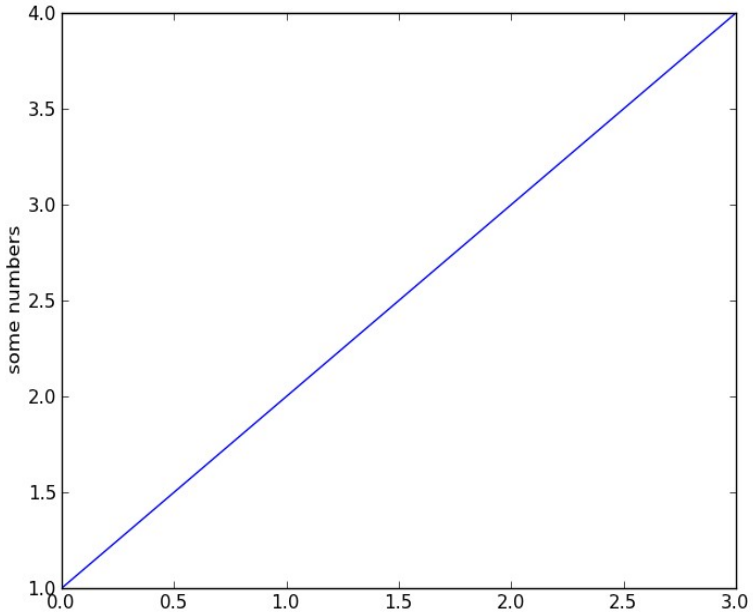
Matplotlib - matplotlib.org

- Library for making 2D plots of arrays in Python
- Origins in emulating MATLAB graphics commands, but is independent of MATLAB, and can be used in a Pythonic, object oriented framework
- Makes heavy use of NumPy for high performance

matplotlib.pyplot

- Collection of command-style functions that make matplotlib work like MATLAB
- Stateful - keeps track of current figure and plotting area

```
import matplotlib.pyplot as plt

plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```

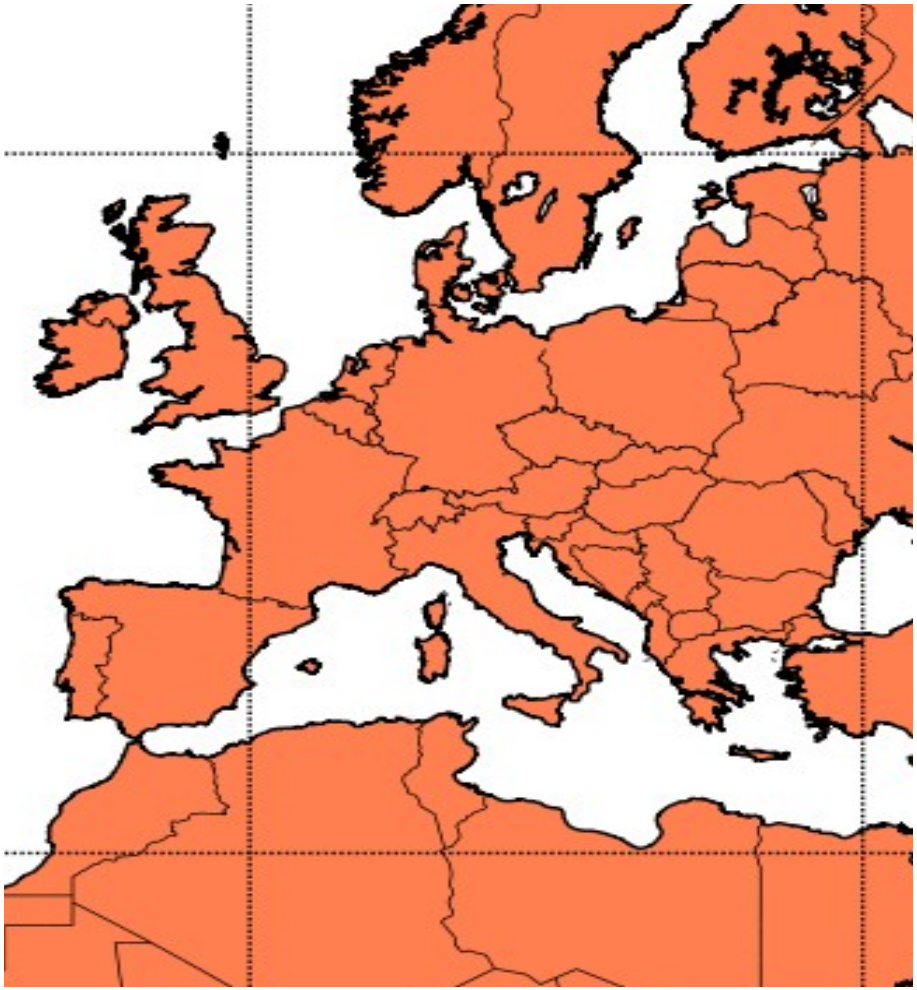


Basemap toolkit - matplotlib.org/basemap

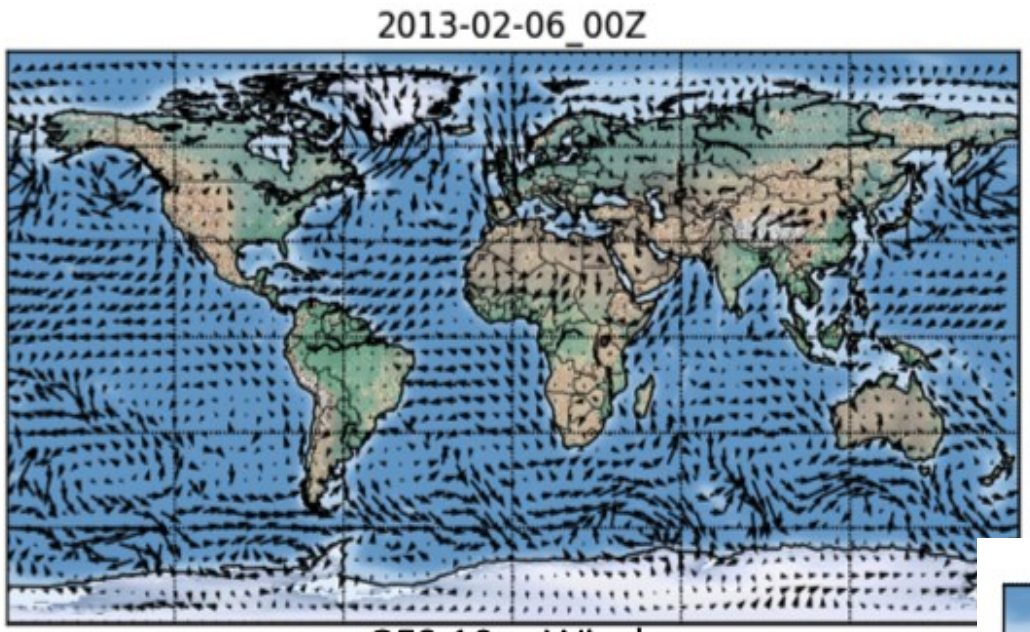
- Library for plotting 2D data on maps in Python
- Similar to MATLAB mapping toolbox, and others
- Does no plotting on its own. Transforms coordinates to one of many provided map projections. Matplotlib then plots contours, etc. in the transformed coordinates

Basemap example - just plotting a map

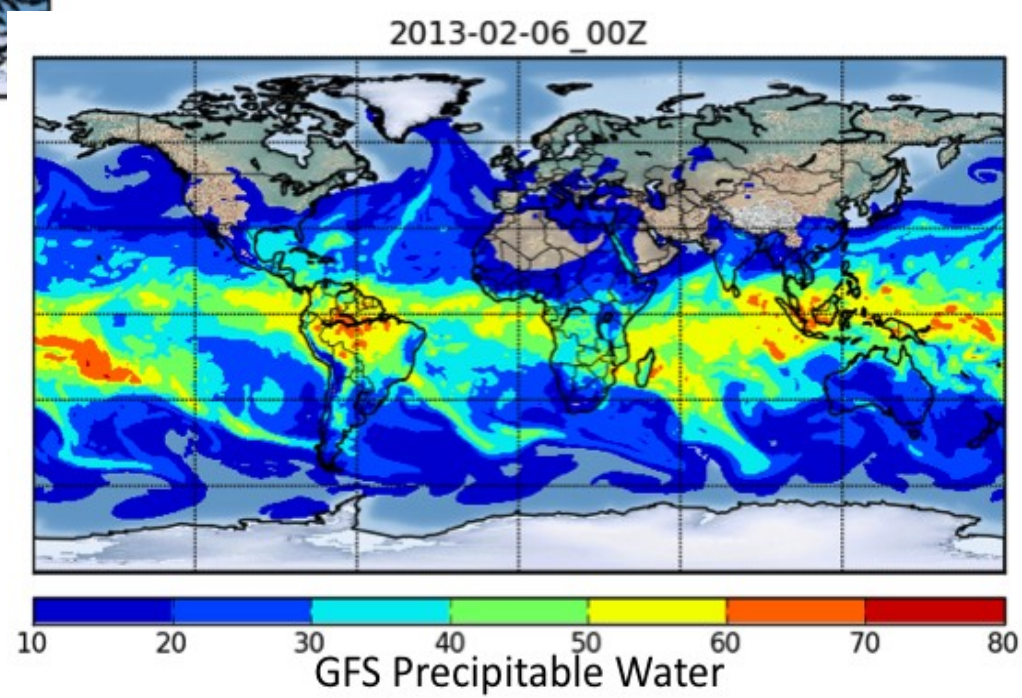
```
#!/usr/bin/env python

import matplotlib as mpl
mpl.use('Agg')
from mpl_toolkits.basemap import Basemap, cm
import matplotlib.pyplot as plt
import numpy as np

map = Basemap(projection='merc',
    resolution = 'h', area_thresh = 1000.0,
    llcrnrlon=-20.0, llcrnrlat=20,
    urcrnrlon=40, urcrnrlat=65)

map.drawcoastlines()
map.drawcountries()
map.fillcontinents(color='coral')
map.drawmapboundary()

map.drawmeridians(np.arange(0, 360, 30))
map.drawparallels(np.arange(-90, 90, 30))

plt.savefig("out.png")
plt.close()
```



Matplotlib GFS plotting



GFS 10m Winds



Plotting GFS winds with matplotlib

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

import os
import sys

import gribapi

import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt
import pylab as pl
from mpl_toolkits.basemap import Basemap, cm
import numpy as np


OUTPUT_FILENAME = 'quickGFSWind.png'

def main(argv=sys.argv):

    if len(argv) != 2:
        print 'Usage:  gfsWindVis.py  <GFS File Path>'
        sys.exit()
    else:
        MET_INPUT_FILE = argv[1]

    if not os.path.isfile(MET_INPUT_FILE):
        print 'Unable to find MET_INPUT_FILE: ' + MET_INPUT_FILE
        sys.exit()

    FH = open(MET_INPUT_FILE)

    # Turn on the multi-field support
    gribapi.grib_multi_support_on()
```

UAF UNIVERSITY OF ALASKA FAIRBANKS

Boreal Scientific Computing LLC

Arctic Region Supercomputing Center

```
    while 1:
        gid = gribapi.grib_new_from_file(FH)
        if gid is None: break

        shortName = gribapi.grib_get(gid, 'shortName')

        if shortName == '10u':
            print 'U WIND!!!'
            Ni = int( gribapi.grib_get(gid, 'Ni') )
            Nj = int( gribapi.grib_get(gid, 'Nj') )
            dataDate = gribapi.grib_get(gid, 'dataDate')
            dataTime = gribapi.grib_get(gid, 'dataTime')
            fcastTime = gribapi.grib_get(gid, 'forecastTime')
            latLL = gribapi.grib_get(gid, 'latitudeOfFirstGridPointInDegrees')
            lonLL = gribapi.grib_get(gid, 'longitudeOfFirstGridPointInDegrees')
            latUR = gribapi.grib_get(gid, 'latitudeOfLastGridPointInDegrees')
            lonUR = gribapi.grib_get(gid, 'longitudeOfLastGridPointInDegrees')


            # Store this stuff in a dictionary so it's easy to pass to
            # a function as an arg
            gridInfoDict = { \
                   'LL_LAT' : latLL,  \
                   'LL_LON' : lonLL,  \
                   'UR_LAT' : latUR,  \
                   'UR_LON' : lonUR,  \
                   'NI'     : Ni,     \
                   'NJ'     : Nj      \
                           }

            print "Grid info: %5d  %5d  (%f/%f)   (%f/%f)" % \
                   (Ni, Nj, latLL, lonLL, latUR, lonUR)

            theVarGrid = gribapi.grib_get_values(gid)
            theUGrid = np.reshape(theVarGrid, (Nj, Ni) )
            theUGrid = np.flipud(theUGrid)
        # END if  10u
```

```
        if shortName == '10v':
            print 'V WIND!!!'
            theVarGrid = gribapi.grib_get_values(gid)
            theVGrid = np.reshape(theVarGrid, (Nj, Ni) )
            theVGrid = np.flipud(theVGrid)
        # END if  10v

    # END while 1

    dataDateStr = '%08d' % dataDate
    dataTimeStr = '%02d' % dataTime
    fcastTimeStr = '%02d' % fcastTime
    timeStampTitle = dataDateStr[0:4] + '-' + \
                     dataDateStr[4:6] + '-' + \
                     dataDateStr[6:8] + '_' + \
                     dataTimeStr + 'Z + ' + fcastTimeStr
    createWindFrame(gridInfoDict, theUGrid, theVGrid, \
                    timeStampTitle, OUTPUT_FILENAME)


# END main()
```

UAF UNIVERSITY OF ALASKA FAIRBANKS

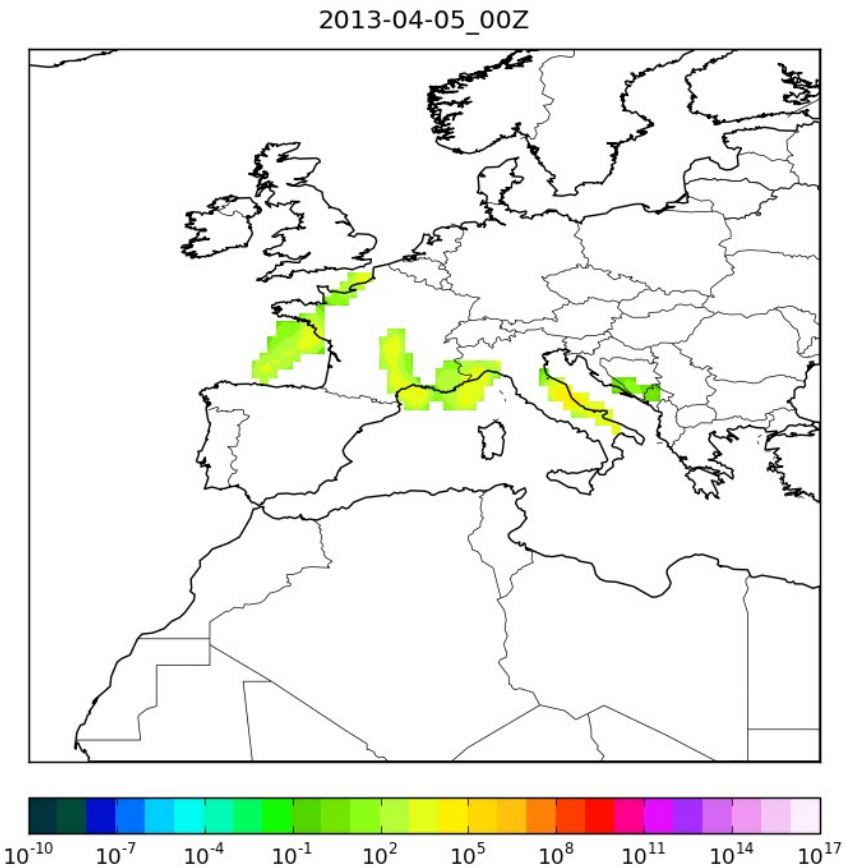
Boreal Scientific Computing LLC

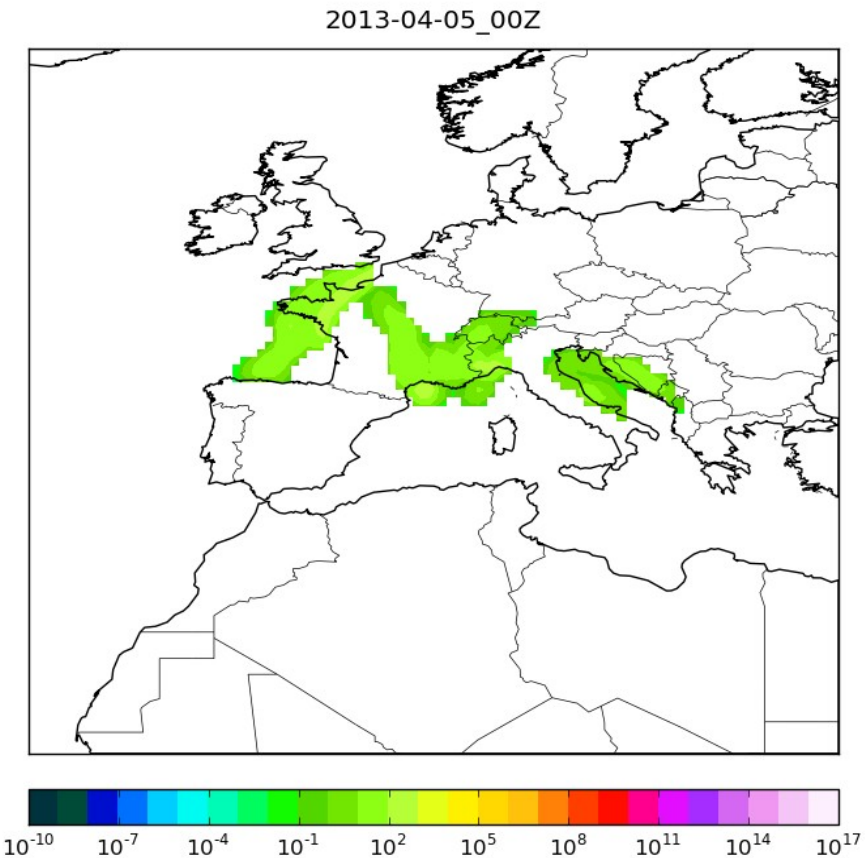
Arctic Region Supercomputing Center

```
def createWindFrame(gridInfo, theUGrid, theVGrid, timeStampTitle, \
                    fullFilename):

    # Creates a single image snapshot (currently defaults to PNG)

    #---------------  arguments  --------------------
    # gridInfo:     a dictionary containing grid info
    # theUGrid, theVGrid:      2D horizontal grids of the data to be plotted
    # timeStamp:    YYYYMMDDHH timestamp associated with the data
    # fullFilename: full path and filename of image (without the extension)

    # Define the type and boundaries of this map
    m = Basemap(projection='cyl', \
                llcrnrlat=gridInfo['UR_LAT'], urcrnrlat=gridInfo['LL_LAT'],\
                llcrnrlon=gridInfo['LL_LON'], urcrnrlon=gridInfo['UR_LON'])
    QUIVER_THIN = 5

    # add in features to the map
    m.drawcoastlines()
    m.drawcountries()

    # Create matrix of lats and lons for this  map
    lons, lats = m.makegrid( gridInfo['NI'], gridInfo['NJ'] )

    # Create the x,y values (in meters) for lons and lats
    x, y = m(lons, lats)
```

```
    thin = QUIVER_THIN
    xThin, yThin = m(lons[::thin,::thin], lats[::thin,::thin])
    uThin = theUGrid[::thin,::thin]
    vThin = theVGrid[::thin,::thin]
    Q = m.quiver(xThin, yThin, uThin, vThin, color="blue")

    '''
    # Use the timestamp as a title (for now).
    thePlottedTimestamp = timeStamp[0:4] + '-' + timeStamp[4:6]
    thePlottedTimestamp += '-' + timeStamp[6:8] + '_'
    thePlottedTimestamp += timeStamp[8:10] + 'Z'
    '''
    plt.title(timeStampTitle)


    plt.savefig( fullFilename )
    plt.close()

# END createFrame()

#-------------------------------------------------------------

if __name__ == "__main__":
    main()
```

Plotting concentrations from dispersion models (e.g. FLEXPART)





FLEXPART output/ directory

dates	grid_conc_20130404140000_001
grid_conc_20130404010000_001	grid_conc_20130404150000_001
grid_conc_20130404020000_001	grid_conc_20130404160000_001
grid_conc_20130404030000_001	grid_conc_20130404170000_001
grid_conc_20130404040000_001	grid_conc_20130404180000_001
grid_conc_20130404050000_001	grid_conc_20130404190000_001
grid_conc_20130404060000_001	grid_conc_20130404200000_001
grid_conc_20130404070000_001	grid_conc_20130404210000_001
grid_conc_20130404080000_001	grid_conc_20130404220000_001
grid_conc_20130404090000_001	grid_conc_20130404230000_001
grid_conc_20130404100000_001	grid_conc_20130405000000_001
grid_conc_20130404110000_001	header
grid_conc_20130404120000_001	receptor_conc
grid_conc_20130404130000_001	trajectories.txt

Plotting concentration from a FLEXPART output file

- Assuming a simple, one-species, forward, non-nested run
- Use pflexible (from NILU) Python module to extract necessary data from FLEXPART output header and desired file
  - This is an evolving module that provides access to all of the variables stored in the FLEXPART output
  - Ultimate goal is to extract a 2D horizontal slice, plus projection parameters
- Use matplotlib and basemap to overlay a colour-filled contour on the region

Plotting concentration from a FLEXPART output file

- Most of the work is in the plotting of a single snapshot FLEXPART output
  - Get 2D grid via plexible and store in 2D numpy array
  - Get grid and projection information via pflexible
  - Use Basemap to define the map region
  - Define a 2D grid over the map region, based on FLEXPART output grid specifications
  - Create a colour-filled contour plot based on FLEXPART output grid values and, mapping it to the region map
  - Add in features like colourbar, title, etc.

quickFLEXVis.py Excerpts

```
# Read the general header information common to all files
H = pf.Header(FLEXPARTOutputDir, nested=False)
...............
# Extract the grid
G = pf.read_grid(H, nspec_ret=0, date=timeStamp)
theTuple = (0, timeStamp)
theGrid = G[theTuple]
gridShape = theGrid.shape     # 4-tuple (x, y, z, p)
...............
theHorizSlice = theGrid.grid[:, :, LEVEL_NUMBER, 0]
theHorizSlice = np.transpose(theHorizSlice)
...............
m = Basemap(projection='cyl', lon_0=lon_0, lat_0=lat_0, \
     lat_ts=lat_0,\
                llcrnrlat=lat_ll, urcrnrlat=lat_ur,\
                llcrnrlon=lon_ll, urcrnrlon=lon_ur,\
                rsphere=6371200.,resolution='l',area_thresh=10000)
...............
cs = m.contourf(x, y, theHorizSlice, levels,
norm=mpl.colors.LogNorm(vmin=LO_CONTOUR_VALUE,
vmax=HI_CONTOUR_VALUE), cmap=COLORMAP)
...............
plt.savefig( imageFilename )
plt.close()
```

Timeouts and retries

- Often, the data we want isn't available when we try to retrieve it, or, it may not be available in full
- For flexibility, we try to give the data several chances to come in completely
- Example - fetching of GRIB data from a remote server

Various hardwired criteria

```
######## Various criteria for success, retry, or giving up
# If the file size on the server is not at least this big, then don't retrieve
#MIN_FILESIZE_THRESHOLD_BYTES = 100000000
MIN_FILESIZE_THRESHOLD_BYTES = 50000000

MAX_PROCESS_AGE_MINUTES = 120   # Age at which to terminate process

FILESIZE_WAIT_TIME_SECONDS = 120   # How long to wait before trying filesize again
FILESIZE_WAIT_MAX_TRIES = 10       # How many times to try on filesize before
                              # giving up
FILERETRIEVE_WAIT_TIME_SECONDS = 60   # How long to wait before trying file retrieve
again
FILERETRIEVE_WAIT_MAX_TRIES = 5  # How many times to try on file retrieve before
                                 # giving up
SOCKET_TIMEOUT_SECONDS = 60      # How many seconds before a socket times out

....................................
    # Set the timeout for any sockets created
    socket.setdefaulttimeout(SOCKET_TIMEOUT_SECONDS)

....................................

    # Get process start time (seconds since epoch) for later use
    PROCESS_START_TIME = time.time()
    currTimeString = time.ctime( PROCESS_START_TIME )
    print '[' + currTimeString + '] Start of driving script'
```

Code excerpts

```
# Form timestamp string for current time (will use with ungribbing)
    # Format is YYYYMMDDHHmm
    currentTimestampStr = startTimeObj.offsetZuluByMinutes( forecastMinute )


    # At this point, I want to connect with the FTP server and find out
    # how big the existing file is (or if it exists).  If all is OK I
    # should grab and store.  If not, I should try again in a bit for some
    # max amount of time.  If the file doesn't appear in the expected
    # amount of time, then I should either abort this process completely,
    # or consider iterating to the next one.
```

Code excerpts

```
# First, just try to get the filesize.  Once we have a filesize
    # that makes the threshold, we assume file is present.
    fileIsPresent = False
    numFilesizeTries = 0
    serverFileSize = 0
    while not fileIsPresent and numFilesizeTries < FILESIZE_WAIT_MAX_TRIES:
        theFTPSession = ftplib.FTP(FTP_SERVER_NAME)
        theFTPSession.login(FTP_SERVER_USER, FTP_SERVER_PASSWD)
        try:
            serverFileSize = theFTPSession.size(expectedFullPathname)
            if serverFileSize > MIN_FILESIZE_THRESHOLD_BYTES:
                fileIsPresent = True
            else:
                print 'File found but size does not exceed threshold'
                print '     File: ' + expectedFullPathname
                print '     Size: ' + str(serverFileSize) + ' bytes'
        except ftplib.all_errors:
            numFilesizeTries += 1
            theReason = 'FTP filesize failed: ' + expectedFullPathname
            busyWait(FILESIZE_WAIT_TIME_SECONDS, theReason)
            evaluateProcessTermination(PROCESS_START_TIME, MAX_PROCESS_AGE_MINUTES)
```

Code excerpts

```
# If file passed the previous test (we got filesize and it makes
   # file size threshold), try to retrieve it.
   if fileIsPresent:

       readyForNext = False
       numRetrieveTries = 0
       while not readyForNext and numRetrieveTries < FILERETRIEVE_WAIT_MAX_TRIES:
           # Retrieve the file
           print 'Retrieving: ' + expectedFullPathname

           fullLocalPathName = LOCAL_STORAGE_DIR + '/' + expectedFilename
           file = open(fullLocalPathName, 'wb')

           try:
               theFTPSession.retrbinary('RETR ' + expectedFullPathname, \
                                        open(fullLocalPathName, 'wb').write)

               # Clean up - close the file and connection
               file.close()
               print 'Closing FTP connection...'
               theFTPSession.close()
               readyForNext = True

           except ftplib.all_errors:

               # Wait
               print 'Closing FTP connection...'
               theFTPSession.close()
               print 'Waiting'
               theReason = 'FTP file retrieve failed: ' + expectedFullPathname
               busyWait(FILERETRIEVE_WAIT_TIME_SECONDS, theReason)
               evaluateProcessTermination(PROCESS_START_TIME,
                                          MAX_PROCESS_AGE_MINUTES)
               numRetrieveTries += 1
```

Code Excerpts

```
def busyWait(seconds, reason):
    # Puts process to sleep for "seconds" secs, stating "reason" first

    print 'busyWait() for ' + str(seconds) + ' seconds'
    print '    reason: ' + reason
    print ' '
    time.sleep(seconds)

#-------------------------------------------

def evaluateProcessTermination(processStartTime, processKillAge):
    # Compares current process age against designated max age, then
    # kills if we've passed the threshold.  The following code currently
    # assumes units in minutes

    # Get current time in seconds since epoch
    currentTime = time.time()

    # Get current age (seconds), then convert to minutes
    currentAge = (currentTime - processStartTime) / 60.0

    # If we've exceeded max age, exit
    if currentAge > processKillAge:
        print 'evaluateProcessTermination() - process exceeds max age
(minutes)'
        print '      processKillAge: ' + str(processKillAge)
        print '      currentAge    : ' + str(currentAge)
        print '      RIP...'
        sys.exit(1)
```

Logging

http://code.activestate.com/recipes/577025-loggingwebmonitor-a-central-logging-server-and-mon/

Logging Server Status Page - Opera

Archivo Editar Ver Marcadores Widgets Fuentes Correo Chat Herramientas Ayuda

LoggingWebMo... | Logging Server ... | PostImage.org -...

http://127.0.0.1:9021/summ.

Google

Logging Server Status Page

Logging Server Start Time	2010-02-02 06:26:39
Logging Server Up Time	0:23:11
Log Records Total	1375

Most Recent Log Records

Date	Channel	Level	Location	Message
2010-02-02 06:49:50	myapp.area2	ERROR	<module> (test.py:25)	The five boxing wizards jump quickly.
2010-02-02 06:49:50	myapp.area2	WARNING	<module> (test.py:24)	Jail zesty vixen who grabbed pay from quack.
2010-02-02 06:49:50	myapp.area1	INFO	<module> (test.py:23)	How quickly daft jumping zebras vex.
2010-02-02 06:49:50	myapp.area1	DEBUG	<module> (test.py:22)	Quick zephyrs blow, vexing daft Jim.

100%

Logging Web Monitor

AS LoggingWebMonitor - a c

code.activestate.com/recipes/577025-loggingwebmonitor-a-central-logging-server-and-mon/

ActiveState Community | Code | Docs | Downloads ▼ | more ▼ Welcome, guest | Sign In | My Account | Store | Cart

ActiveState Code » Recipes

Search

Add a Recipe Languages Tags Authors Sets

LoggingWebMonitor - a central logging server and monitor. (Python recipe)

1

LoggingWebMonitor listens for log records sent from other processes running in the same box or network. Collects and saves them concurrently in a log file. Shows a summary web page with the latest N records received.

Download Copy to clipbaord Python, 409 lines

```
"""
LoggingWebMonitor - a central logging server and monitor.

Listens for log records sent from other processes running
in the same box or network.  Collects and saves them
concurrently in a log file.  Shows a summary web page with
the latest N records received.

Usage:

- Add a SocketHandler to your application::

    from logging.handlers import SocketHandler, DEFAULT_TCP_LOGGING_PORT
    socketh = SocketHandler(servername, DEFAULT_TCP_LOGGING_PORT)
    logging.getLogger('').addHandler(socketh)

  where servername is the host name of the logging server ('localhost'
  if run on the same box)

- Start an instance of this script (the logging server).
```

Created by Gabriel Genellina on Tue, 2 Feb 2010 (MIT)

◄ Python recipes (3997) ►

◄ Gabriel Genellina's recipes (9) ►

Tags

client_server

debugging

distributed

logging

remote

sysadmin

web

► Show machine tags (19)

Required Modules

os

sys

cpickle

logging

Logging Web Monitor

AS LoggingWebMonitor - a c

code.activestate.com/recipes/577025-loggingwebmonitor-a-central-logging-server-and-mon/

Steve Romanow 2 years, 11 months ago

@Gabriel: I got it, it was a typo.

Don Morton 2 years, 7 months ago

This server and client are FANTASTIC! I have 20-30 Python scripts that get launched several times a day - all of them need to work for data fetching and processing to culminate in numerical weather forecasts for Alaska. They don't all run on the same machine, and I even have plans to run some of these scripts redundantly on different machines. Each script writes its own log file, but of course if something goes wrong it's an incredibly tedious task to track it back to the particular script that failed.

With only trivial modifications to this code, I've been able to set up a logging server, and then build a Python "LoggerClient" class that each script instantiates. So, now I can refer to the single web page, or the log file, to get a feel for which scripts have run correctly and which haven't.

I had planned on doing this with low-level file locking, but of course that can be very problematic, and requires the use of a filesystem common to all processes (which, in itself, can be problematic). With less than a day of effort, this code has done everything I wanted!

rtmie 2 years, 6 months ago

Great stuff, thanks for this useful script.

You may need to add a thr_recv.join() after the recv.shutdown() in the main to ensure that no sockets are left in a wait state

Logging

```
import LoggerClient

# Set up the logger client
theLogger = LoggerClient.LoggerClient('FetchAndUngrib RRNatFull')

.............................................

    logMessage = 'Started.  START_TIME: ' + START_TIME + ', '
    logMessage += 'NUM_FORECAST_HOURS: ' + str(NUM_FORECAST_HOURS)
    theLogger.logInfo(logMessage)
```

Ambitions

- A multi-machine process organiser and timer
  - Trying to get out of "crontab hell"
- System for rapidly running and combining multiple model runs and configurations into a single output mechanism – e.g. Emergency response system for volcano eruptions, nuclear power plant disasters, etc.
- Seamless utilisation of cloud computing environments