Orchestrating a Climate Modeling Data Pipeline using Python and RAM Disk

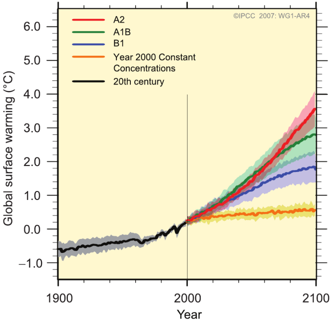
A Brief Overview of the WRF Tools Package

Andre R. Erler

PyCon Canada

November 7^{th}, 2015

Outline



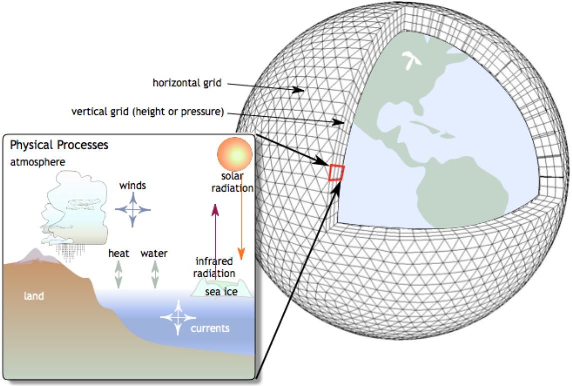
IPCC AR4 (2007) projections for global surface temperature under different scenarios.

Global Climate Models are the main tool to predict climate change.

Global Climate Models

Climate models compute energy, mass, and momentum fluxes on a relatively coarse computational grid.

Schematic of a Global Climate Model (GCM):



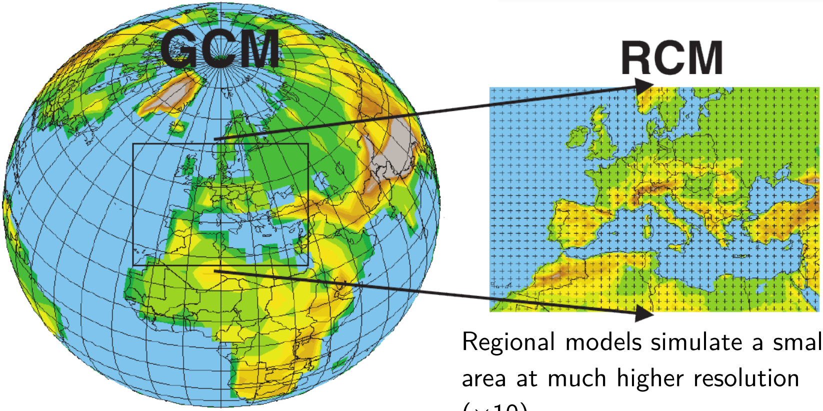
Center for Multiscale Modeling of Atmospheric Processes, CSU

Regional Climate Models

GCM resolution is coarse and many regional details are not resolved (e.g. the Rocky Mountains and the Great Lakes).

Regional Impacts

Regional impacts of Climate Change are modeled with high-resolution regional climate models (RCM).



Giorgi (2006)

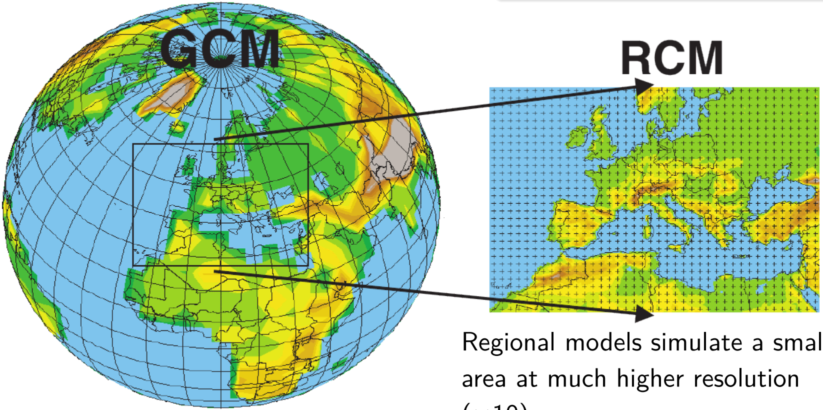
Regional models simulate a small area at much higher resolution ($\times 10$).

Regional Climate Models

GCM resolution is coarse and many regional details are not resolved (e.g. the Rocky Mountains and the Great Lakes).

Regional Impacts

Regional impacts of Climate Change are modeled with high-resolution regional climate models (RCM).

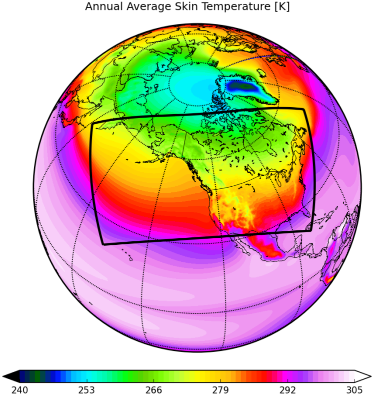


Giorgi (2006)

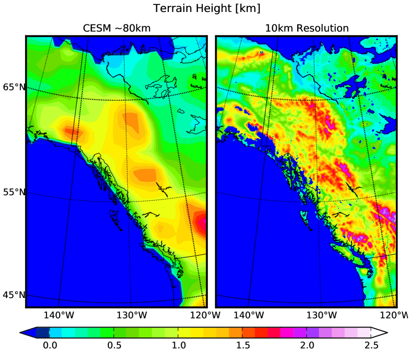
Regional models simulate a small area at much higher resolution ($\times 10$).

Global Model: CESM

The Community Earth System Model is used as driving model.



Average Surface Temperature and Outline of the WRF Domain



Topography of Western Canada.
Left: CESM at ∼80 km
Right: WRF at 10 km

Regional Model: WRF

The Weather Research and Forecast model is our regional model.

WRF is actually pronounced “Worf”, like Lt. Worf in *Star Trek: The Next Generation* (left)

The WRF model is a limited-area numerical weather prediction model developed by the National Center for Atmospheric Research

Running the Regional Climate Model (WRF)

- The coupling process between GCM and RCM is "off-line" (asynchronous)
- A pre-processing system converts GCM output into RCM (wrf-)input files
- A RCM simulation is split into $\sim$ 200 separate jobs
- The RCM runs continuously, each job submitting the next

The WRF Tools Package

Python

- Run pre-processing tool chain (WPS)
- Initialize WRF jobs
- Run post-processing

Shell Script

- Submit pre-processing
- Run the WRF job, submit next job
- Archiving to tape

WRF Tools enables continuous and autonomous operation

Running the Regional Climate Model (WRF)

- The coupling process between GCM and RCM is "off-line" (asynchronous)
- A pre-processing system converts GCM output into RCM (wrf-)input files
- A RCM simulation is split into $\sim$ 200 separate jobs
- The RCM runs continuously, each job submitting the next

The WRF Tools Package

Python

- Run pre-processing tool chain (WPS)
- Initialize WRF jobs
- Run post-processing

Shell Script

- Submit pre-processing
- Run the WRF job, submit next job
- Archiving to tape

WRF Tools enables continuous and autonomous operation

Running the Regional Climate Model (WRF)

- The coupling process between GCM and RCM is “off-line” (asynchronous)
- A pre-processing system converts GCM output into RCM (wrf-)input files
- A RCM simulation is split into $\sim$ 200 separate jobs
- The RCM runs continuously, each job submitting the next

The WRF Tools Package

Python

- Run pre-processing tool chain (WPS)
- Initialize WRF jobs
- Run post-processing

Shell Script

- Submit pre-processing
- Run the WRF job, submit next job
- Archiving to tape

WRF Tools enables continuous and autonomous operation

Running the Regional Climate Model (WRF)

- The coupling process between GCM and RCM is “off-line” (asynchronous)
- A pre-processing system converts GCM output into RCM (wrf-)input files
- A RCM simulation is split into $\sim$ 200 separate jobs
- The RCM runs continuously, each job submitting the next

The WRF Tools Package

Python

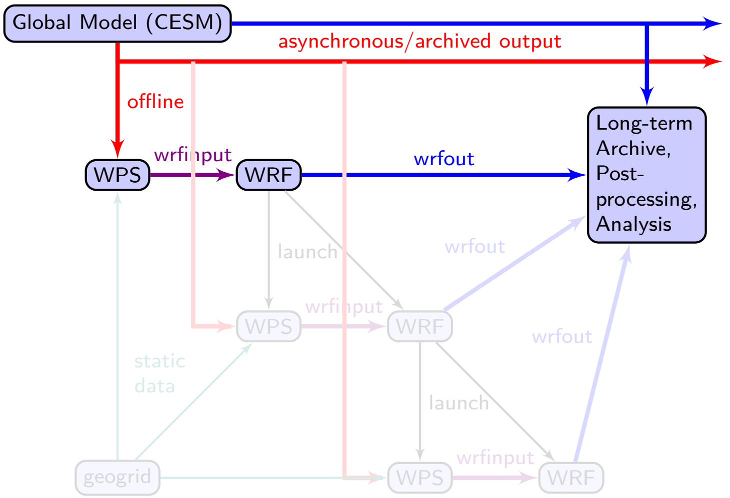
- Run pre-processing tool chain (WPS)
- Initialize WRF jobs
- Run post-processing

Shell Script

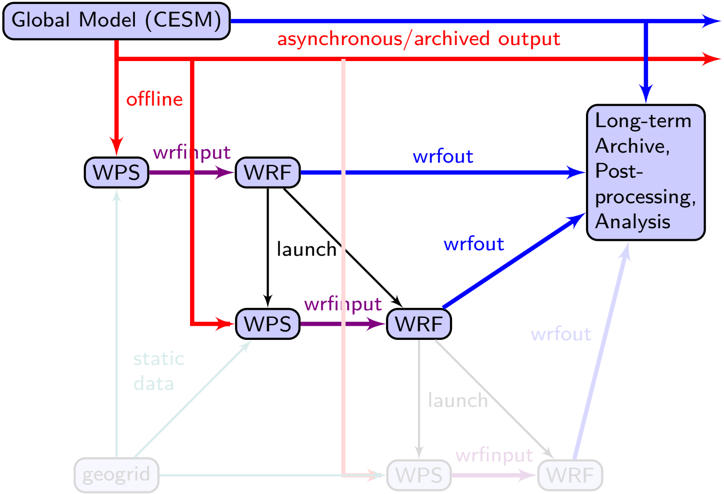
- Submit pre-processing
- Run the WRF job, submit next job
- Archiving to tape

WRF Tools enables continuous and autonomous operation

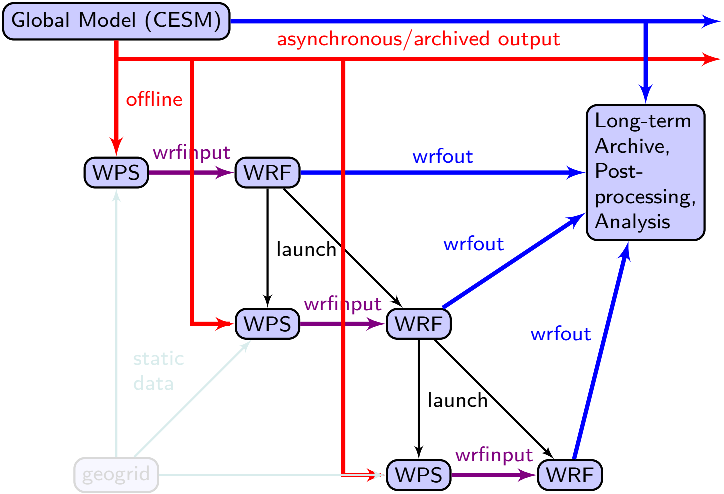
The Data Pipeline



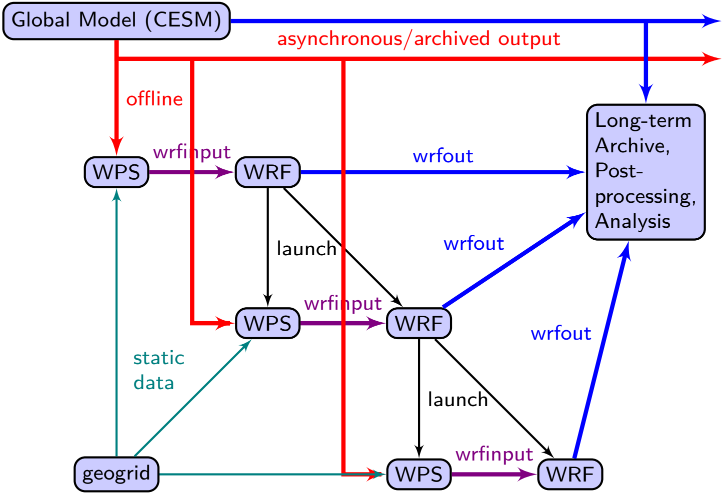
The Data Pipeline



The Data Pipeline



The Data Pipeline



WPS: A Collection of FORTRAN Legacy Tools

WPS Components

1. `geogrid.exe`
   static / geographic data
2. `ungrib.exe / unccsm.exe`
   convert driving data to WRF IM Format
3. `metgrid.exe`
   interpolate to WRF grid
4. `real.exe`
   generate boundary condition files

FORTRAN legacy tools read from and write to temporary files:

- Strongly I/O limited in a HPC cluster environment

The Solution (on Linux)

Run on RAM-disk!

- speedup $\sim \times 10$
- requires 64 GB RAM

Using Python driver script

WPS: A Collection of FORTRAN Legacy Tools

WPS Components

1. `geogrid.exe`
   static / geographic data
2. `ungrib.exe` / `unccsm.exe`
   convert driving data to WRF IM Format
3. `metgrid.exe`
   interpolate to WRF grid
4. `real.exe`
   generate boundary condition files

FORTRAN legacy tools read from and write to temporary files:

- Strongly I/O limited in a HPC cluster environment

The Solution (on Linux)

Run on RAM-disk!

- speedup $\sim \times 10$
- requires 64 GB RAM

Using Python driver script

PyWPS: A Driver Module for WPS

- Collect required input data from GCM archive
- Run applicable pre-processing tools on RAM disk
- Assemble WRF input files

Why Python?

- Easier with complex logic
- Classes for different datasets/GCMs

PyWPS Imports

- `multiprocessing` for parallelization
- `re` to find input files
- `fileinput`, `sys` to edit configurations files
- `subprocess` to launch FORTRAN tools
- `shutils`, `os` to handle temporary files

PyWPS: A Driver Module for WPS

- Collect required input data from GCM archive
- Run applicable pre-processing tools on RAM disk
- Assemble WRF input files

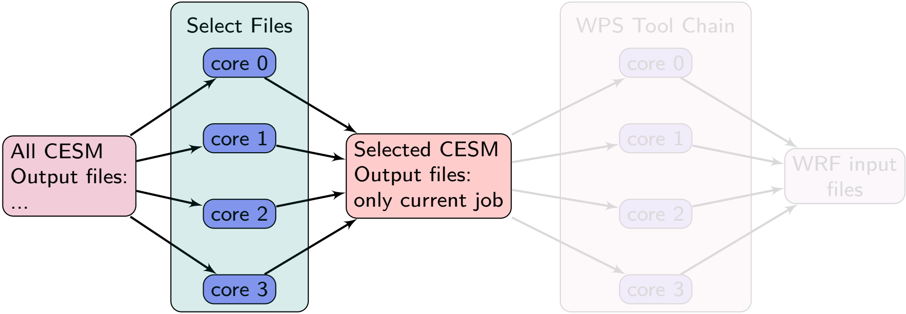
Why Python?

- Easier with complex logic
- Classes for different datasets/GCMs

PyWPS Imports

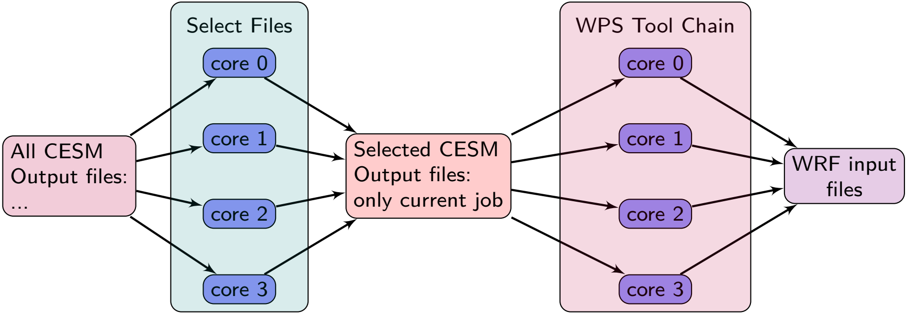
- `multiprocessing` for parallelization
- `re` to find input files
- `fileinput`, `sys` to edit configurations files
- `subprocess` to launch FORTRAN tools
- `shutils`, `os` to handle temporary files

PyWPS: The Program Flow & Parallelization



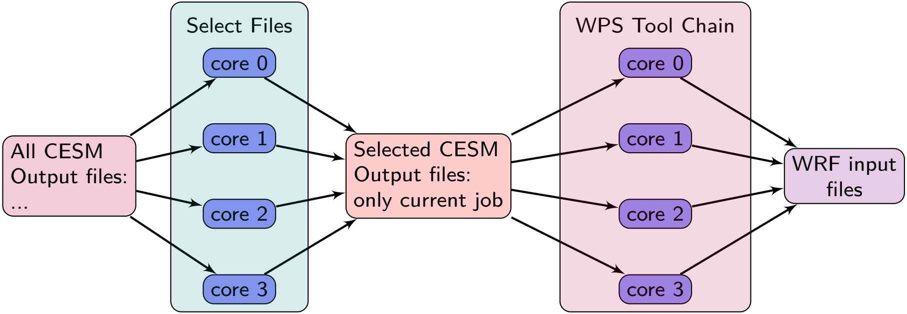
The WPS Tool Chain:

PyWPS: The Program Flow & Parallelization



The WPS Tool Chain:

PyWPS: The Program Flow & Parallelization



The WPS Tool Chain:

The Class Structure

Dataset/GCM specific parameters:

- Input file types/names
- Interpolation tables/grid
- Variables / frequency

Multiple Datasets

- *Inheritance* for common procedures
- *Polymorphism* for different procedures

```
class Dataset(object):
  prefix = ''  # file prefix
  vtable = 'Vtable'
  gribname = 'GRIBFILE' # input
  ungrib_exe = 'ungrib.exe'
  ungrib_log = 'ungrib.exe.log'
  ...
  def __init__(self, ...):
    # type checking
    ...
  def setup(self, src, ...):
    ...
  def cleanup(self, tgt):
    ...
  def extractDate(self, fname):
    # match valid filenames
    ...
  def ungrib(self, date, mytag):
    # generate file for metgrid
    ...
```

The Class Structure

Dataset/GCM specific parameters:

- Input file types/names
- Interpolation tables/grid
- Variables / frequency

Multiple Datasets

- *Inheritance* for common procedures
- *Polymorphism* for different procedures

```
class Dataset(object):
  prefix = ''  # file prefix
  vtable = 'Vtable'
  gribname = 'GRIBFILE' # input
  ungrib_exe = 'ungrib.exe'
  ungrib_log = 'ungrib.exe.log'
  ...
  def __init__(self, ...):
    # type checking
    ...
  def setup(self, src, ...):
    ...
  def cleanup(self, tgt):
    ...
  def extractDate(self, fname):
    # match valid filenames
    ...
  def ungrib(self, date, mytag):
    # generate file for metgrid
    ...
```

Summary & Conclusion

Python

- Use Python for flow control (manage legacy tools)
- Parallelization relatively easy (within one node)
- Class structure is versatile and makes maintenance easier

RAM-disk

- Scientific Programming: dealing with legacy tools
  Often in FORTRAN, often relying on disk I/O
- Use RAM-disk to avoid unnecessary disk I/O

Thank You! $\sim$ Questions?

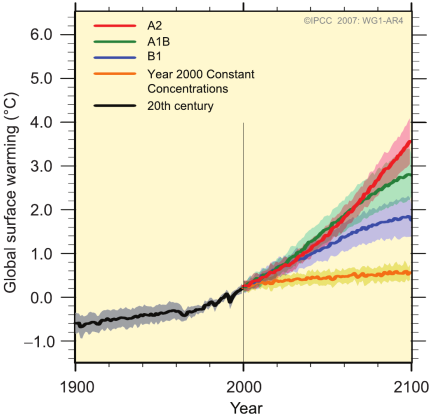
List of Publications using WRF Tools

- **Erler, Andre R.**, W. Richard Peltier, Marc d'Orgeville (under review), Projected Changes in Hydro-Climatic Extremes for Western Canada, Journal of Climate.
- Marc d'Orgeville, W. Richard Peltier, **Andre R. Erler** (accepted), Uncertainty in Future Summer Precipitation on the Great Lakes Basin due to Drought in the South-Western US, Journal of Geophysical Research.
- **Erler, Andre R.**, W. Richard Peltier, Marc d'Orgeville, 2015, Dynamically Downscaled High Resolution Hydro-Climate Projections for Western Canada, Journal of Climate.
- Marc d'Orgeville, W. Richard Peltier, **Andre R. Erler**, Jonathan Gula, 2014, Climate change impacts on Great Lakes Basin precipitation extremes, Journal of Geophysical Research.

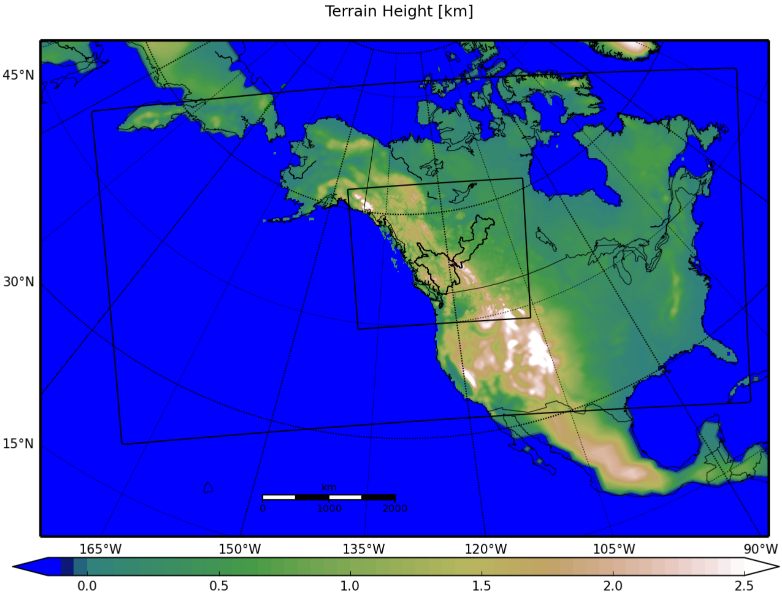
Regional Climate Projections

Experimental Setup

- GCM & RCM run for 15 years (model time)
  - Historical (1979 - 1994)
  - Mid-21st-Century (2045-2060)
  - End-21st-Century (2085-2100)
- GCM & RCM use RCP 8.5 GHG concentration scenarios
- RCM runs with different physical parameterizations
- Both models run in an initial condition ensemble with 4 members each



IPCC AR4 climate projections based on different scenarios; the RCP 8.5 is very similar to the older A2 scenario

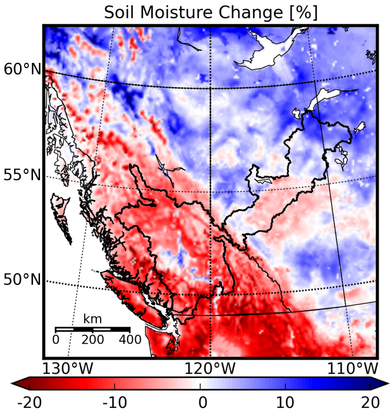

Terrain Height [km]
45°N
30°N
15°N
165°W
150°W
135°W
120°W
105°W
90°W
km
0
1000
2000
0.0
0.5
1.0
1.5
2.0
2.5

Summary of Results

- Significant increase in winter precipitation (extremes, ∼30%)
- Small increase in summer, but more increase in evaporation

Hydrological Impacts

- Climate change impacts in ARB/Alberta likely benign
- 50% reduction in peak snowmelt and spring runoff in FRB/BC...
- ... but increased flood risk due to precipitation extremes in fall



End-century soil moisture changes in late summer

- Late summer drying west of Continental Divide, but not east