documentation_5-28

May 30, 2023

1 Final Report

Kaleb Guillot

1.1 ##### May 30, 2023

This report details the work done and code produced in the FA2022 - SP2023 semesters. Throughout these two semesters, the work I have produced was done under the supervision of Dr. Tzeng. Much of the code produced caters towards the research of Fudong Lin in his deep neural network and dataset projects.

My work has focused around understanding and working with WRF-HRRR GRIB2 files. These are files in a highly compressed format containing meterological data. Their significance lies in their use by the machine learning models produced for the NSF funded PREFER project. In order to decompress and format these files, I have used both C and Python based implementations.

```
<div style="flex:1; margin-right:50px; margin-left:200px">
    <img src="figs/grib_cube.png" alt="Grib Cube Structure" style="width:80%">
</div>
<div style="flex:1; margin-left:50px; margin-right:100px">
    <img src="figs/pic_grib.png" alt="Grib Parameter Structure" style="width:58%">
</div>
```

This report will include:

- HRRR-WRF GRIB2 File Overview
- Basics of Downloading / Extracting GRIB2 Files
  - Python Based Implementation
  - C Based Implementation
- CUDA
  - ECCODES-CUDA
- Outline of customExtraction

1.2 ### HRRR-WRF GRIB2 File Overview

The Weather Research and Forecasting (WRF) Model is a numerical weather prediction system developed by a collaboration of research organizations for simulating and forecasting weather conditions. It is an open-source model, allowing for users to modify the model to fit their applications. The High Resolution Rapid Refresh (HRRR) model is a derivative of the WRF model, providing a higher resolution of predictions generated every hour. The numerical weather observations of the HRRR model span across the entire continental United States, having the ability to not only

forecast the weather, but also capturing rapidly evolving weather phenonma and severe weather predictions.

The numerical weather measurements and predictions from the HRRR model are stored in GRIdded Binary version 2 (GRIB2) file format, a highly processed and compressed file format. The steps involved to minimize the space taken up by each hourly WRF-HRRR file include preprocessing, packing, spatial differencing, quantization, compression, and encoding. The initial preprocessing removes any incorrect or irrelevant data, including data measured from outside the continential United States. During packing, the data is encoded into a binary format and the headers are added to the meta data. Spacial differencing minimizes space by encoding the differences between adjacent points rather than storing a given point's full value. Quantization lessens the precision of the data, allowing values to take up fewer bits. Compression methods are then used to further shrink the file's size. These methods include a combination of both LZ77 and Huffamn encoding. Then finally, the file is encoded into a binary-to-text scheme, represing the binary data in ASCII format.

The data in this file structure is often represented as a 3D cube, containing cartesian points on an XYZ coordinate plane. For each GRIB2 file, the file X and Y coordinates correspond to lattitudes and longitudes across the continential United States, while points in the Z plane represent the weather parameters that the HRRR model has measured. These weather parameters are sometimes called Grib Messages; they contain the name of the parameter, unit of measurement, and level above the Earth's surface. The current version of the models stores approximately 150 weather parameters.

IMPORTANT NOTE: WRF-HRRR data contains some parameters that are marked as "accumulation" measurements, such as `Total Precipitation`. This means that the real-time (files that are observed measurements, not predicted) values for that parameter will always be 0. A complete list of GRIB2 parameters can be found on the eccodes webpage, where accumulation measurements are marked as such.

1.2.1 Basics of Downloading / Extracting GRIB2 Files

The python module Herbie was used to download the WRF-HRRR GRIB2 files from a number of repositories. The files were then stored on canpc39 (now canpc40) under the directory `/mnt/wrf/`. The file used is mentioned later in the report under Outline of customExtraction.

The Herbie module was used due to its flexibility and ease of use. Through using this library, a single command could be used to download a file, and if need be, only download a subset of the file as specified by a regex string. An example of using Herbie is as follows:

[3]:
```
from herbie import Herbie

date_time = "20220101 10:00" # format as yyyymmdd hh:MM
save_dir = "/directory/to/save/wrf/files/"

herb_obj = Herbie(date_time, model="hrrr",
                  product="sfc", save_dir=save_dir,
                  priority=['pando', 'pando2', 'aws', 'nomads',
                            'google', 'azure'],
                  fxx=0).download(":(?:TMP|RH):2 m")
```

```
Found   model=hrrr   product=sfc   2022-Jan-01 10:00
UTC F00   GRIB2 @ pando
IDX @ pando
```

The first argument is a string of the date, formatted as "yyyymmdd hh:MM". The next argument, the model, specifies that the HRRR contiguous United states model will be downloaded. The `product` argument will download the 150 parameters of surface fields. The `priority` argument gives the list of repositories to search for the files. Each time that the specified file is not found for the given date, the next repository in the array is searched. The `fxx` is the forecast lead time in hours, specifying how many hours in the future the selected file will predict. Fxx is set to 0 to download the real-time values. The argument passed to the `download()` is a regex string which specifies that only a subset of the file will be downloaded. This subset specifically, are the two layers `2 metre temperature` and `2 metre relative humidity`.

Important Note: Herbie is still a very new module and is prone to problems. Oftentimes when downloading the data with Herbie, execution will halt while Herbie tries to communicate with the repository. When downloading data with this model, set an automatic timer which specifies that if a time limit exceeds, the file's download should restart and try again.

1.2.2 Python Based Implementation

A number of different methods were used to extract necessary information from a GRIB2 file. These methods involved the use of both Pygrib and Herbie. Their uses were similar, however had their distinct differences.

Using Herbie When using Herbie, the first required step would be to grab the file from the repository and load it into a herbie object using the command provided above. From this step, one has the option of download the object locally with the `download()` command, or skipping this step and loading the file into an array with the `xarray()` command.

[5]:
```
t2m_rh = herb_obj.xarray(":(?:TMP|RH):2 m")
```

Next would be to declare some points of interest and then use the built in Herbie commands to index those points

[6]:
```
lsu = (-91.1783, 30.414)
ull = (-92.014, 30.213)
coordinates = [lsu, ull]
coord_names = ["lsu", "ull"]

t2m_rh_arr = t2m_rh.herbie.nearest_points(points=coordinates, names=coord_names)

t2m_values = t2m_rh_arr.t2m.values
r2_value = t2m_rh_arr.r2.values

print("2 metre temp vals: ", t2m_values, "\n2 metre relative humidity vals: ",␣
 ↪r2_value)
```

```
/home/kaleb/miniconda3/envs/postproc/lib/python3.9/site-
packages/metpy/xarray.py:382: UserWarning: More than one time coordinate present
for variable "gribfile_projection".
  warnings.warn('More than one ' + axis + ' coordinate present for variable'
/home/kaleb/miniconda3/envs/postproc/lib/python3.9/site-
packages/metpy/xarray.py:382: UserWarning: More than one time coordinate present
for variable "t2m".
  warnings.warn('More than one ' + axis + ' coordinate present for variable'
/home/kaleb/miniconda3/envs/postproc/lib/python3.9/site-
packages/metpy/xarray.py:382: UserWarning: More than one time coordinate present
for variable "r2".
  warnings.warn('More than one ' + axis + ' coordinate present for variable'
/home/kaleb/miniconda3/envs/postproc/lib/python3.9/site-
packages/metpy/xarray.py:382: UserWarning: More than one time coordinate present
for variable "gribfile_projection".
  warnings.warn('More than one ' + axis + ' coordinate present for variable'
```

```
2 metre temp vals:  [297.42273 296.73523]
2 metre relative humidity vals:  [84.2 88.9]
```

Using Pygrib There are two methodologies when using pygrib, with one being more reliable than the other.

In both methods, the first step is to open a GRIB2 file into a pygrib object.

[7]:
```
import pygrib

wrf_file_dir = "/media/kaleb/extraSpace/wrf/2022/20220101/hrrr.20220101.00.00.
 ↪grib2"
grb = pygrib.open(wrf_file_dir)
```

The naiive implementation is to loop through the layers of the files until the parameter of interest is found. The first implementation kept an array of parameters of interest to index from the GRIB2 file:

[8]:
```
params = [1, 2, 3]

for p in params:
    layer = grb[p]
    print(layer)
```

```
1:Maximum/Composite radar reflectivity:dB (instant):lambert:atmosphere:level 0
-:fcst time 0 hrs:from 202201010000
2:3:3 (instant):lambert:cloudTop:level 0:fcst time 0 hrs:from 202201010000
3:201:201 (instant):lambert:atmosphere:level 0 -:fcst time 0 hrs:from
202201010000
```

This method, however, is problematic. The numbering of the layers is often changed at arbitrary time intervals. When extracting information from a large number of files for a wide date range, the layer numbering may change from one month to the next. The solution to this is to extract

the layers by thier names and levels, factors which do not change with time. This method uses the `select()` function to find a given layer of the grib file

[11]:
```
param_names = ['2 metre temperature', '2 metre relative humidity', 'U component␣
↪of wind']
param_levels = [2, 2, 1000]

for name, level in zip(param_names, param_levels):
    layer = grb.select(name=name, level=level)
    print(layer)
```

```
[71:2 metre temperature:K (instant):lambert:heightAboveGround:level 2 m:fcst
time 0 hrs:from 202201010000]
[75:2 metre relative humidity:% (instant):lambert:heightAboveGround:level 2
m:fcst time 0 hrs:from 202201010000]
[36:U component of wind:m s**-1 (instant):lambert:isobaricInhPa:level 100000
Pa:fcst time 0 hrs:from 202201010000]
```

Final Implementation (Fudong's Model) The second implementation was the one of choice when extracting weather data for Fudong's models. This implementation is found in the file `full_preproc_pygrib_threaded.py`. This file takes a `--fips` argument with the fips code of desired counties to extract information from. Optionally, the fips argument can be changed to `--fips state=` to extract information from an entire state. The code calls a modified version of `geo_grid.py` to take a number of fips codes as inputs, and output necessary information about the fips codes in a `wrf_output.csv` file. This file divides each county into 10km by 10km grid points, storing the latitude and longitude of the upper right and lower left corners of each grid. The county information loaded into the code and stored in a pandas dataframe.

The code will extract 7 parameters from a real-time GRIB2 file and a precipitation parameter from a 1 hour prediction GRIB2 file. The 7 parameters are 2 metre temperature, 2 metre relative humidity, downward short-wave radiation flux, u component of wind (level 1000), v component of wind (level 1000), and wind gust. After extraction, the code will take the daily averages of each metric. The minimum and maximum temperatures for each day are stored and recorded. The U and V components of wind are used to calculate wind speed, and temperature and relative humidity are used to calculate vapor pressure deficit.

The outputted files are monthly files stored by `<year>/<state>/HRRR_<state_fips>_<state_abbrev>_<year>-<m` They include the daily information about each given grid point as well as a monthly average of the entire state.

In order to run this code, a few variables must be configured:

- self.write_path: the path to write outputted monthly files to
- self.grib_path: the path of the GRIB2 files. Point at the folder containing the years.
  - For example, if a file's path is `/home/kaleb/wrf/2022/20220101/hrrr.20220101.00.00.grib2`, then configure this path to be `/home/kaleb/wrf/`
- self.herbie_path: the path of the 1 hour prediction precipitation files. Since herbie automatically places the files into a further `hrrr` directory, then point at the level above that.
  - For example, if a precipitation file's path is `/home/kaleb/precip_files/hrrr/20220101/subset_0ae4` then configure the path to be `/home/kaleb/precip_files/`

- self.repository_path: be sure that this path points to the `customExtraction` repo
- self.wrfOutput_path: this is the path to the output of the `geo_grid_recent.py` file
  - It should not need to be altered unless either file is moved
- self.max_workers: this is the max amount of threads to make when extracting and finding the monthly averages.
  - By default the max amount of threads when extracting is 24, however more can be used whenever finding monthly averages
- self.begin_date and self.end_date: the begin and end days of GRIB2 files to extract from. They are formatted as `yyyymmdd`. NOTE: end day is NOT inclusive -self.begin_hour and self.end_hour: the begin and end hour of GRIB2 files to extract from. They are formatted as `hh:MM`. NOTE: end hour IS inclusive

1.3 ### C-Based Implementation