Computational performance improvements in UFS weather forecast system

NATIONAL WEATHER SERVICE

Jun Wang[1*], Jeffrey Whitaker[2], Edward Hartnett[3, 7], James Abeles[4], Gerhard Theurich[5], Wen Meng[4], Cory Martin[6], Jose-Henrique Alves[8], Fanglin Yang[1], Arun Chawla

UFS Users' Workshop, 7/27/2020

[1]NOAA/NWS/NCEP/EMC, [2]NOAA/ESRL/PSD, [3]NOAA/ESRL/GSD, [4]IMSG@NOAA/NWS/NCEP/EMC, [5]ESMF/NRL/SAIC
[6]RedLine@NOAA/NWS/NCEP/EMC, [7]CIRES, University of Colorado[8], SRG@NOAA/NWS/NCEP/EMC

Outline

- Model description
- Computational performance improvements
- Future work

GFSv16 model Description

- GFSv16 uses the **ufs-weather-model** as its weather forecast model.
- The ufs-weather-model is using the NOAA Environmental Modeling System (**NEMS)** infrastructure.
- In GFSv16, the FV3GFS is one way coupled to WW3 model. FV3GFS is running at C768L127 resolution, with vertical resolution doubled compared to GFSv15. The wave component uses the NOAA WAVEWATCH III model (WW3) with a global grid mosaic [WW3DG, 2019]. WW3 global core resolution is increased from ½ degree to ⅙ degree.
- The model produces forecast history files in NetCDF format and post-processed grib2 files out to 16 days, 4 times a day with hourly output for the first 120 hours and then three-hourly output up to 384 hours

Computational improvements in GFSv16

- Inline post on write grid component
- Parallel writing compressed NetCDF history files
- Scalability improvement in WW3
- Data transfer improvement between ESMF model components

Inline Post

- Inline post is a new capability developed on the write grid components in atmosphere grid component fv3gfs.
- An interface was generated to set up the variables that are required to compute POST products from model output fields. It saves the time of reading model output files in POST processing and reduces the IO activity in the system.
- The post code for product computation and output GRIB file generation were made into a library that can be called on the write grid component.
- When model outputs lossy compressed history files, the inline POST generates more accurate results compared to standalone POST.

experiments	C96L64 (6 tasks)	C192L64 (12 tasks)	C768L127 (84 tasks)
Single master file size	51MB	180MB	2.5GB
Inline post time	4s	7s	39s
Offline post time	12s	17s	211s

Writing out compressed data in NetCDF format

C768L127 fcst output	Nemsio No compression	Netcdf No compression	Netcdf Lossless (deflate=1,nbit=0)	Netcdf Lossy (deflate =1, nbit=20)	Netcdf Lossy(deflate=1,nbit=14)	Netcdf Lossy (deflate=1, nbits=14),parallel writing, default decomposition chunksize	Netcdf Lossy (deflate=1, nbits=14),parallel writing Layer chunkcsize
A 3D file size (total fcst)	33.6GB (7TB)	33.6GB (7TB)	23.6GB (5TB)	13.5GB (2.8TB)	6.3GB (1.3TB)	6.3GB (1.3TB)	6.3GB (1.3TB)
Write Time	79s	300s	960s	680s	400s	43s	34s

- HDF1.10.6 and new NetCDF 4.7.4 are installed, ESMF is rebuilt with new NetCDF library
- Model can write out netcdf file with serial or parallel writing
- Chunk sizes for 2D and 3D fields are defined in the model configuration file.
- For restart files, proper io_layout setting is critical to write out restart files without delaying forecast imtegration

Scalability improvement in WW3

- The scalability issue was observed in WW3

  - When increasing the MPI tasks, WW3 total running time also increased. In GFSv16 test WW3 running time is 24140s when using 700 tasks. The running time is 12616s when using 315 tasks.

- Two updates were made in WW3

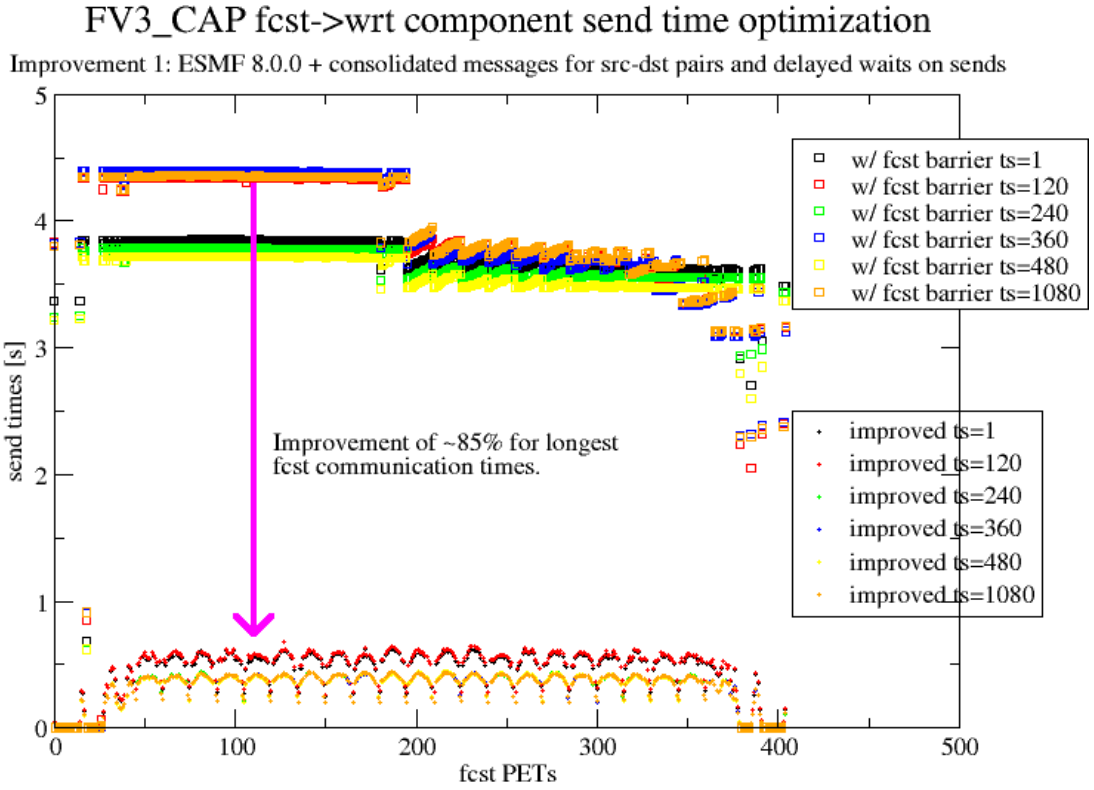
  - In WW3 it is found that a field gather call is done within a loop of number of MPI tasks for every MPI task. The code is changed to do field gather once on root task and root task broadcast the data to all the test MPI tasks.
  - Fixes have been made to ensure the threading is working properly in WW3.

- Performance improvement

  - When tripling the number of MPI tasks, WW3 runs now take about 36% of original running time while before the fixes it took about 94% of original running time.

Data transfer improvement between ESMF model components

- All messages going to the same dst PET are first written to a single buffer.
- This includes all the messages from the referenced RHs, as well as messages from the RHs for all the FieldBundles.
- The messages are then sent via single MPI_Isend(), and the MPI_Wait() is delayed until the buffer is needed again for sending to the same dst PET.
- This increases the memory requirement, but decouples the sends from the receives.



Future work

It is critical to improve model computational performance when running with resource consuming high resolution models

- Check scalability of each subcomponent in the coupled system
- Develop efficient IO strategy to meet the requirements for next generation of high resolution weather and climate forecast runs
  - **What to output**: add new capabilities to write out model outputs on different grid with different content and at different frequency
  - **How to output:**
    - Work with community to improve compression algorithms and maintain required precision and writing
    - Improve performance of writing restart files
  - **Workflow:** move other IO intensive downstream jobs to write grid component to save time fort the end to end GFS forecast system and to reduce IO stress on the whole computer system

Thank you!