Running WRF Operationally at AWS Using OBSGRID

Richard L. Carpenter, Jr.
Weather Decision Technologies, Inc., Norman, Oklahoma, USA

ABSTRACT

We describe our experience running WRF operationally in a commercial environment. We run WRF at a traditional data center and at AWS. The AWS implementation is built with CfnCluster and uses the Slurm job scheduling system. We examine performance and cost saving options, including I/O quilting and compressing the WRF netCDF input and history files. Our preprocessing system ingests MADIS observations and uses OBSGRID for both quality control of the observations for FDDA and for gridded initial conditions. Our workflow management system, WRFControl, handles all aspects of forecast execution, including data ingest, job scheduling and submission, post-processing, plotting, dissemination, monitoring, and alerting.

1. Introduction

Since its inception in 2000, Weather Decision Technologies, Inc. (WDT) has run NWP systems operationally. In 2004, WDT began using WRF as its operational mesoscale model. The delivery of a WRF-based system to the government of Paraguay marked the first commercial international application of WRF (Carpenter et al., 2004). WDT subsequently installed WRF systems at Dubai International Airport, the Philippines, and Thailand (Shaw et al., 2008).

WDT also performs hundreds of WRF runs daily in support of applications including aviation, marine exploration and shipping, renewable energy, and winter weather forecasting. These runs are a mix of hourly "rapid-refresh" runs and longer-range forecasts. Grid spacings range from 12 km down to 1 km nests for some domains.

The composition of the forecast system has changed over the years, but in general has always included:

- various data assimilation techniques, including 3DVAR, LAPS, and FDDA;
- high-performance computing (HPC) on Linux clusters of various sizes;
- a robust workflow management system known as WRFControl.

Here we describe our current operational system and discuss lessons learned over the years. In particular, we describe experience running WRF at Amazon Web Services (AWS), and with coupling our FDDA processes with WRF's OBSGRID utility.

2. Initialization Using OBSGRID and FDDA

We use WRF Version 3.9.1 with ¼° GFS data as a background. We download and run UNGRIB on the first 30 hours of each GFS run. For later GFS forecast hours we process every third hour, interpolating to hourly, in order to conserve bandwidth and processing time. For selected runs, we use ⅛° ECMWF forecasts instead of GFS. We also use NCEP's 1/12° sea surface temperature analysis.

Previously we used a combination of LAPS and observation nudging (FDDA) for data assimilation. A custom data ingest format allowed the use of non-governmental observations. We decided to stop using LAPS owing to its complexity and lack of ongoing support. Instead, we are using OBSGRID, WRF's objective analysis utility, for data ingest and quality control.

Data ingest begins by obtaining the latest observations from MADIS. We ingest the following data types: METAR, MARINE, RAOB, ACARS, MAP (profiler), and SATWND (Figure 1). (We do not

use mesonet data, which we have found can cause WRF to crash during FDDA.) We use MADIS2LITTLER to convert the observations to Little-R format for processing by OBSGRID.

OBSGRID is run for each hour that FDDA is to be applied. The records are then sorted by time and placed into a single master FDDA file (*OBS_DOMAIN101*). OBSGRID's gridded analyses are also used instead of METGRID's (overwriting the *met_em_** files with the *metoa_em_** files). For nested grid runs, we currently apply FDDA only on the parent grid.

Our OBSGRID and FDDA settings generally follow the recommendations of Reen (2016). Typically we apply FDDA for two hours before the nominal start time.

The typical workflow of a WRF job is as follows:

- Run METGRID for the duration of the run, using every third hour of UNGRIB as input.
- Run METGRID on every hour of the FDDA period (e.g., 3 times for a 2-hour FDDA period).
- Run OBSGRID on every hour of the FDDA period.
- Replace the METGRID analyses with the OBSGRID analyses.
- Concatenate and time-sort the OBSGRID output into a single *OBS_DOMAIN101* file.
- Run REAL.
- Run WRF.

3. Running WRF at a Traditional Data Center and at AWS

WDT in late 2017 retired its HPC cluster and began leasing dedicated nodes at the University of Oklahoma's OU Center for Supercomputing Education and Research (OSCER). The facility uses 20-core nodes with Haswell processors and InfiniBand interconnects, along with high-speed Lustre storage.

During periods of scheduled maintenance at OSCER, we shift operations to AWS. We are currently using 18-core Haswell *c4.8xlarge* nodes at AWS. (We have found performance per-core to be similar at OSCER and AWS when running WRF across several nodes. We have not yet had much experience with the newest *c5* Skylake nodes, which have custom, more advanced, interconnects.) We use CfnCluster to build the clusters at AWS.

Also at AWS, we use the Intel Fortran compiler to compile WRF for distributed memory. For maximum optimization, we select the "SNB with AVX mods" configuration option and set *DM_FC=-xHOST* in *configure.wrf*.

We use I/O quilting, in which one or more processors are reserved for writing history files (*nio_tasks_per_group=2*). We find this to be less cumbersome than a splitfiles/joinfiles approach. Rather than dedicate an entire node to I/O, we are conservative in the number of nodes we use. For instance, on 18-core nodes, we run WRF calculations on 16 cores and use the remaining 2 cores for I/O. Compared with using entire nodes for WRF calculations and none for I/O tasks, this approach improves the performance of a run by a few percent, more so in cases of high-frequency output.

The amount of disk space consumed during a WRF run can be quite large. The history files alone from a run on a $501 \times 401 \times 36$ grid with hourly output to 84 hours can occupy 48 GB of space. We therefore explored the option of outputting compressed WRF history and REAL input netCDF files by compiling with *NETCDF4=1*. This compression reduces the file sizes by 60-70%, although run-time performance can suffer by as much as 10-20%. Some users may consider the significant savings in disk space to be worth the decrease in run-time performance.

We use the Slurm job scheduler at both OSCER and AWS. At AWS, we modify the configuration to use both core and memory (*CR_Core_Memory*) as consumable resources. This allows multiple post-processing jobs to run on a single node.

4. Modifications to WRF

We make a few minor modifications to the WRF code. The first involves the use of "ready" files with I/O quilting. We modify the WRF code to allow the "ready" files to be written when quilting is on. The approach is not perfect, and we wait some number of seconds after the "ready" file is written before safely accessing the history file.

Other minor modifications to the WRF code include:

- Adding downward solar radiation (*ts_swdown*) to the *tslist* (time series) output. This involves a change to the Registry.
- Using a value of 6371.2 km rather than 6370 km for the radius of the earth in WPS and WRF. This improves compatibility with GRIB files from NCEP.

5. WRFControl

Our workflow management software, WRFControl, handles all aspects of forecast execution, including data ingest, job scheduling and submission, post-processing, plotting, dissemination, monitoring, and alerting. WRFControl is the direct successor to the Perl-based system that was previously used to orchestrate ARPS and WRF runs. It consists of about 35,000 lines of Python script plus various configuration files written in YAML.

We have also built a custom WRF post-processor, written in Fortran. Key products include aviation variables such as turbulence, icing, and flight conditions; and lightning flash density.

We generally use NCAR Graphics for plotting gridded data, and Matplotlib (Python) for plotting meteograms. Data and plots are transferred to AWS S3 object storage for web display and retrieval by clients. A continually updating job status page (Fig. 2) and basic WRF plots (Fig. 3) are hosted in S3.

REFERENCES

Carpenter, R., G. Bassett, K. Brewster, D. Weber, Y. Wang, J. Brotzge, K. Thomas, F. Kong, and D. Jahn, 2004: A Globally Relocatable Numerical Weather Prediction System Based on WRF and ADAS. *Extended Abstracts, 20th Conf. on Weather Analysis and Forecasting and 16th Conf. on Numerical Weather Prediction,* AMS, Seattle.

Reen, B., 2016: A Brief Guide to Observation Nudging in WRF, NCAR. http://www2.mmm.ucar.edu-/wrf/users/docs/ObsNudgingGuide.pdf.

Shaw, B., P. Spencer, R. Carpenter, and C. Barrere, 2008: Implementation of the WRF model for the Dubai International Airport Aviation Weather Decision Support System (AWDSS). *Extended Abstracts, 13th Conf. on Aviation, Range, and Aerospace Meteorology,* New Orleans, AMS.

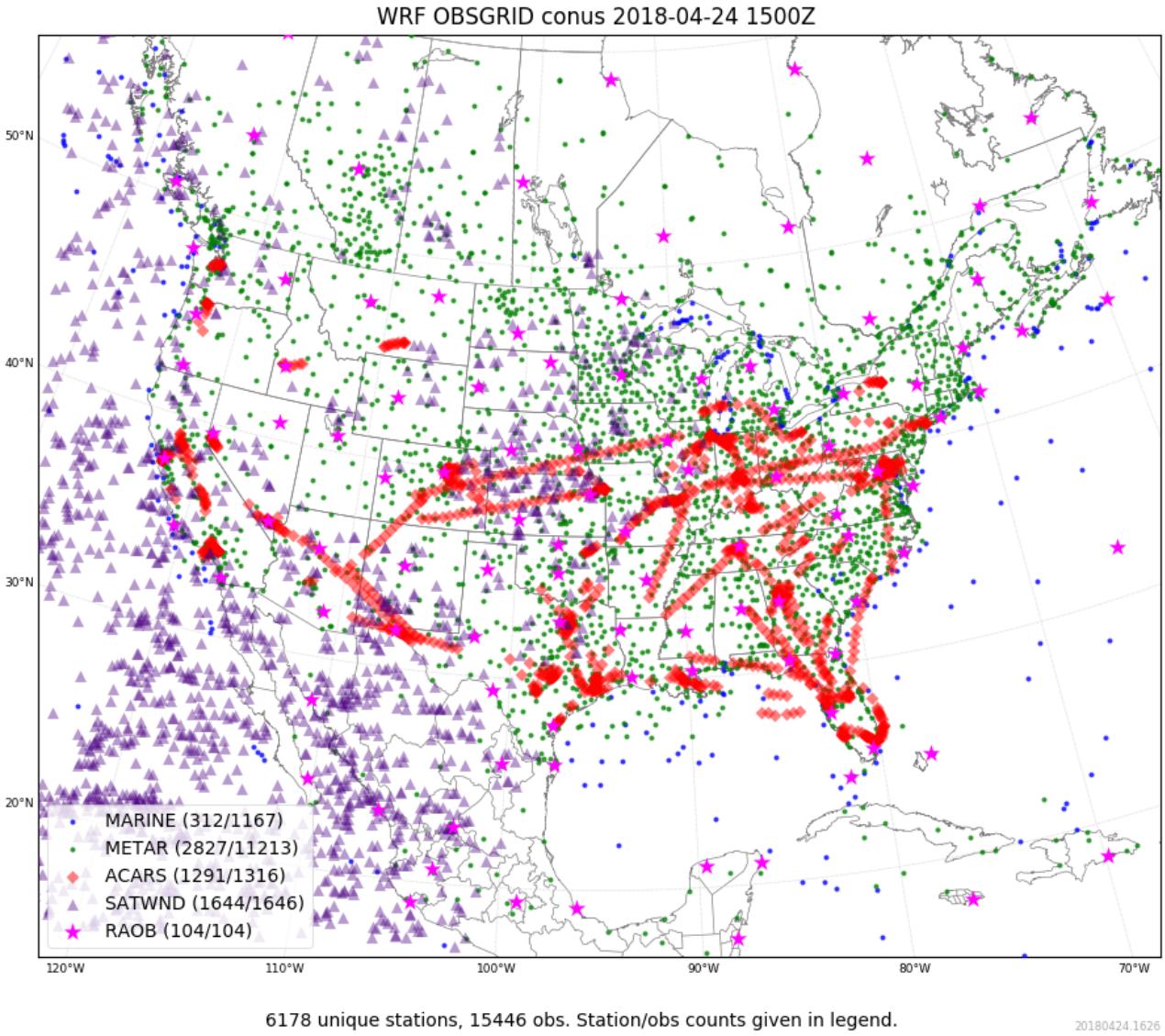

Figure 1. Observation locations and counts processed by OBSGRID for a sample initialization of WRF over CONUS. Numbers in parentheses are the counts of unique locations and total observations for each data type.

WRF Status

Past 24 hours

Updated Tue 10 Apr 2018 14:12 UTC

		Mon 9 Apr										Tue 10 Apr													
		14Z	15Z	16Z	17Z	18Z	19Z	20Z	21Z	22Z	23Z	00Z	01Z	02Z	03Z	04Z	05Z	06Z	07Z	08Z	09Z	10Z	11Z	12Z	13Z	14Z
noram_rr	18h 36+2p	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	m
conus*	120h 36+2p		A						A						A						A					
europe*	84h 36+2p					A						A						A						40%		
eu_rr	18h 36+2p	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	m
us_ca_3km*	84h 36+2p					A						A						A						49%		
colombia12_3_1	72h 36+2p					A						A						A						68%		
RHSsj6_1	36h 20p		A						A						A						A					
RHS_6_1a	36h 20p					A						A						A						A		
e_asia_rr	12h 18+2p	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	m
ausnz_rr	12h 18+2p	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	Q
alaska	3h 1p	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	S
gmex4	84h 156+2p				A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	
stnaz12_4	84h 117+2p				A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	94%	
ok124	84h 117+2p				A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	92%	
okc_2km	36h 117+2p				A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A		
okc_4km	84h 36+2p	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	S

Legend	
	Not scheduled
S	Scheduled
L	Launched
Q	Job queued
m	METGRID running
r	REAL running
W	WRF running
P	Post-processing finishing
C	Job finished
A	All products available
D	Final post-processing delayed
X	Failed
*	Late/Other failure
?	Unknown

Figure 2. Example showing the status of WRF runs over a 24-hour period. Indicators shown include: S = scheduled; Q = queued; m = running METGRID; percentage = fraction of GRIB files available; A = all products available. Other indicators not shown include running REAL, running WRF, job finishing, job delayed, and job failed.

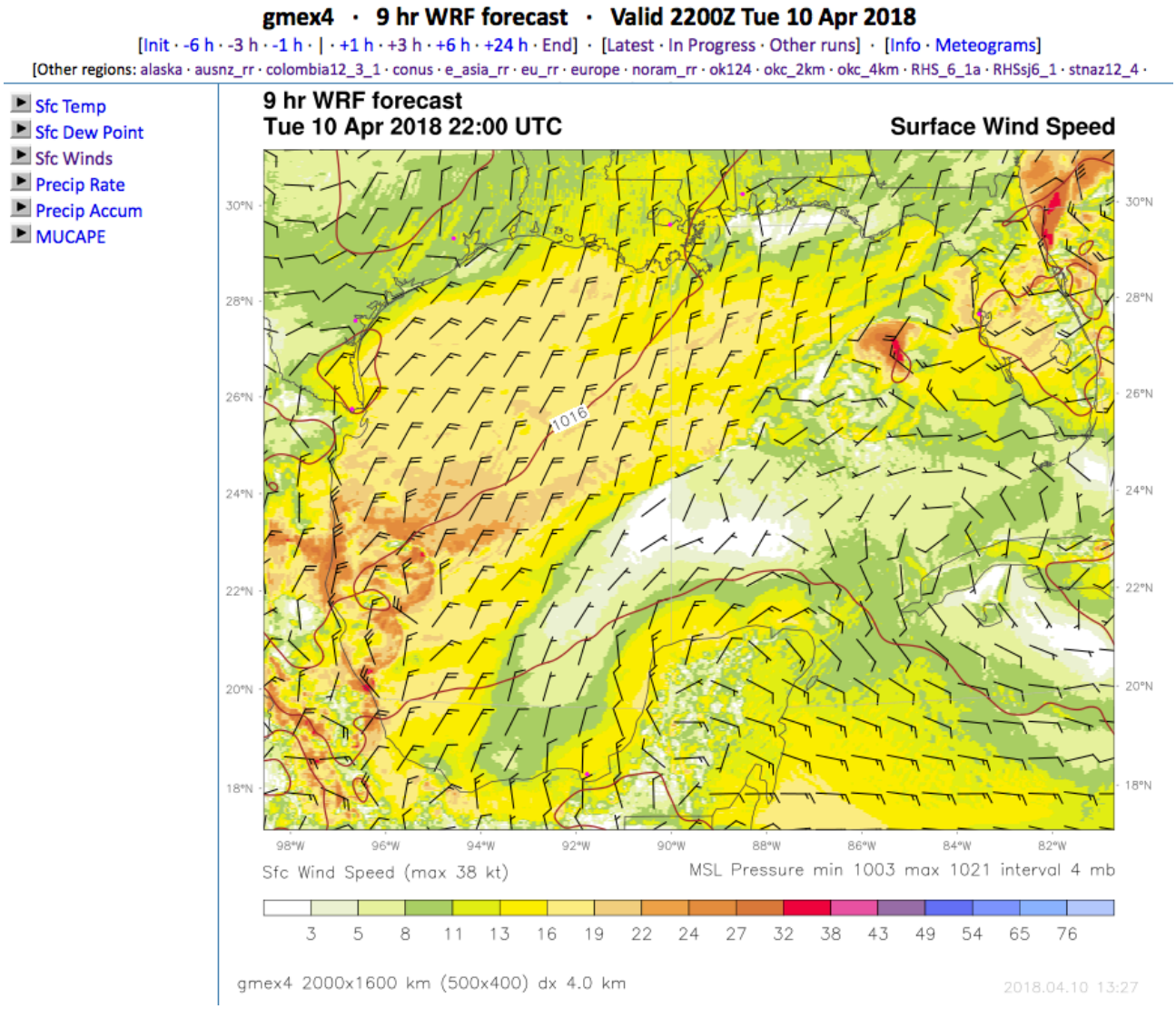

Figure 3. Example of the WRF plot viewer, showing a forecast on a 4-km grid of winds over the Gulf of Mexico. This domain is run hourly out to 84 hours in support of marine exploration and transportation operations.