Using Cloud Computing to Analyze Model Output Archived in Zarr Format

TAYLOR A. GOWAN,[a] JOHN D. HOREL,[a] ALEXANDER A. JACQUES,[a] AND ADAIR KOVAC[a]

[a] *Department of Atmospheric Sciences, University of Utah, Salt Lake City, Utah*

(Manuscript received 28 July 2021, in final form 2 January 2022)

ABSTRACT: Numerical weather prediction centers rely on the Gridded Binary Second Edition (GRIB2) file format to efficiently compress and disseminate model output as two-dimensional grids. User processing time and storage requirements are high if many GRIB2 files with size O(100 MB, where B = bytes) need to be accessed routinely. We illustrate one approach to overcome such bottlenecks by reformatting GRIB2 model output from the High-Resolution Rapid Refresh (HRRR) model of the National Centers for Environmental Prediction to a cloud-optimized storage type, Zarr. Archives of the original HRRR GRIB2 files and the resulting Zarr stores on Amazon Web Services (AWS) Simple Storage Service (S3) are available publicly through the Amazon Sustainability Data Initiative. Every hour, the HRRR model produces 18- or 48-hourly GRIB2 surface forecast files of size O(100 MB). To simplify access to the grids in the surface files, we reorganize the HRRR model output for each variable and vertical level into Zarr stores of size O(1 MB), with chunks O(10 kB) containing all forecast lead times for 150 × 150 gridpoint subdomains. Open-source libraries provide efficient access to the compressed Zarr stores using cloud or local computing resources. The HRRR-Zarr approach is illustrated for common applications of sensible weather parameters, including real-time alerts for high-impact situations and retrospective access to output from hundreds to thousands of model runs. For example, time series of surface pressure forecast grids can be accessed using AWS cloud computing resources approximately 40 times as fast from the HRRR-Zarr store as from the HRRR-GRIB2 archive.

SIGNIFICANCE STATEMENT: The rapid evolution of computing power and data storage have enabled numerical weather prediction forecasts to be generated faster and with more detail than ever before. The increased temporal and spatial resolution of forecast model output can force end users with finite memory and storage capabilities to make pragmatic decisions about which data to retrieve, archive, and process for their applications. We illustrate an approach to alleviate this access bottleneck for common weather analysis and forecasting applications by using the Amazon Web Services (AWS) Simple Storage Service (S3) to store output from the High-Resolution Rapid Refresh (HRRR) model in Zarr format. Zarr is a relatively new data storage format that is flexible, compressible, and designed to be accessed with open-source software either using cloud or local computing resources. The HRRR-Zarr dataset is publicly available as part of the AWS Sustainability Data Initiative.

KEYWORDS: Data processing/distribution; Databases; Numerical weather prediction/forecasting; Data science

1. Introduction

The global weather enterprise relies on millions of large, two-dimensional data fields (grids) created each day by operational numerical weather prediction (NWP) models (Benjamin et al. 2018). The perceptions, uses, and values for that vast amount of information depend in part on its accessibility and how it is disseminated to end users (Lazo et al. 2009). Advances in computing processing power and storage have allowed operational centers to run models at finer spatial scales and higher temporal frequency, yet only a small fraction of the information available from the models is typically available to end users (Benjamin et al. 2018). Pragmatic decisions are made by operational forecast centers in order to disseminate global and regional model output for dozens of parameters by restricting ranges and frequencies of valid times and horizontal and vertical grid spacings. Those decisions have been heavily influenced by internal and external limitations on storing and accessing the hundreds of gigabytes (GB) of model output generated by each model run. These challenges are not unique to the weather sector; many disciplines are struggling to overcome the "volume, variety, and velocity" of data cubes (datasets in space and time) available from Earth observation systems (Giuliani et al. 2020; Yao et al. 2020). Improved data cube cyber infrastructures are recognized to be needed for environmental datasets to allow the ingestion, storage, access, analysis, and use of data elements ordered by geolocation and other shared attributes (Nativi et al. 2017).

The National Oceanic and Atmospheric Administration (NOAA) Big Data Program (BDP) began in 2015 to address agency-wide issues to access the tens of terabytes (TB) of observations and model output created each day within the agency (Ansari et al. 2018; NOAA 2020). With support from the NOAA BDP, the NOAA Cooperative Institute for Climate and Satellite–North Carolina (CICS-NC) has implemented a data hub architecture to facilitate transfer of key NOAA environmental datasets to infrastructure-as-a-service

Denotes content that is immediately available upon publication as open access.

Corresponding author: Taylor A. Gowan, taylor.mccorkle@utah.edu

DOI: 10.1175/JTECH-D-21-0106.1

(IaaS) providers for data storage. This includes over 130 data streams, including such high-demand datasets as current and historical Next Generation Weather Radar (NEXRAD) products from 160 sites in the United States (Ansari et al. 2018). IaaS providers [e.g., Amazon Web Services (AWS); Google Cloud Platform; IBM; and Microsoft Azure] have the capacity to store enormous datasets and provide public access and computing resources for end users to postprocess these data streams within their IaaS environment to reduce the time and cost to access the information using cloud (offsite on the Internet) or local compute resources (Molthan et al. 2015; Siuta et al. 2016). The commitment by NOAA and IaaS providers to facilitate public access to these datasets addresses many of the FAIR principles for data repositories: findability, accessibility, interoperability, and reusability (Wilkinson et al. 2016).

The Google Cloud Platform and AWS Simple Storage Service (S3) began providing public access during 2020 to output from the High-Resolution Rapid Refresh (HRRR) model of the National Centers for Environmental Prediction (NCEP). The HRRR is a convection-allowing model that was developed by the Earth Systems Research Laboratory and is run operationally every hour by the NCEP's Environmental Modeling Center. HRRR output is available for dozens of surface and upper-atmospheric variables at 3-km grid spacing for a 1.9 million gridpoint domain that covers the contiguous United States (CONUS; Benjamin et al. 2016; Blaylock et al. 2017a). As of December 2021, the HRRR archives provided by Google and AWS are larger than 2 petabytes (PB) in total storage and are growing at a rate of over 700 GB per day.

From 2016 to 2020, more than 1000 registered operational and research users relied on the only publicly accessible archive of HRRR model output that was managed by researchers at the University of Utah. This archive utilized S3-type storage procedures provided by the Center for High Performance Computing (Blaylock et al. 2017b, 2018). By 2020, the archive grew to over 160 TB and continuing to maintain and expand the HRRR archive at the University of Utah was no longer feasible. We began exploring alternative approaches and formats to store HRRR model output for diverse applications that rely on the GRIB2 model output available now from the IaaS providers.

International standards were established by the World Meteorological Organization (WMO) to efficiently store and disseminate NWP model output in hypercube-structured file formats with built-in compression algorithms. The Gridded Binary Second Edition (GRIB2) format has been in use during the past several decades to archive two-dimensional files that are efficiently compressed using a method similar to JPEG image compression (Silver and Zender 2017). While GRIB2 files effectively help to store and transmit large amounts of meteorological data as two-dimensional slices, they can be cumbersome to work with and rely on WMO-defined tables that are unfamiliar to users in other disciplines (Wang 2014). Many users rely on software tools to transform GRIB2 files into other self-describing formats such as netCDF-4 (Silver and Zender 2017). Decoding the two-dimensional slices in GRIB2 format requires high memory use that contributes to inefficiencies when, for example, an end user may only be interested in certain parameters for all forecast times available from a specific model run within a local or regional subdomain. It is possible to access individual variables within GRIB2 files by selecting their byte range or specifying a bounding box for domain subsets, but doing so requires first loading each file into memory and performing additional postprocessing to build a file index. Once such an index is created, individual variables over their whole domain can be selected from GRIB2 files in object storage and accessing subdomains is possible for files stored locally (Blaylock et al. 2017b, 2018).

Researchers generally use high-level programming environments that rely on MATLAB, Interactive Data Language (IDL), or Python to examine, postprocess, and visualize operational model data. Data science and machine learning (ML) techniques applied to operational model output typically require multivariate training datasets with long periods of record for which alternative data structures beyond GRIB2 are necessary (Vannitsem et al. 2021). As summarized by McGovern et al. (2017), these big data and ML methods have been used to improve forecasts of weather and pollution parameters such as storm duration (Cintineo et al. 2014), severe wind (Lagerquist 2016), large hail (Adams-Selin and Ziegler 2016), precipitation type (Reeves et al. 2014; Elmore et al. 2015), aviation turbulence (Sharman 2016), orographic precipitation (Arulraj and Barros 2021), and ozone concentrations (Wang et al. 2022). To continue applying ML and artificial intelligence techniques to the ever-growing model output repositories, it will be critical to have data in structures that allow for flexible dissection in space, time, and across many forecast model runs or ensemble members (McGovern et al. 2017).

An alternative storage format, Zarr, is described in this study as a means to archive HRRR output. Zarr is a relatively new storage format, developed in 2016 for use in a malaria genome project. It chunks and compresses *N*-dimensional datasets for flexible storage in memory, on disk, or within cloud platforms (Vance et al. 2019; Miles et al. 2020; Abernathey et al. 2021). The Zarr format is being used for promising ML and big data applications in other disciplines, for example, the Lyft level-5 self-driving dataset (Houston et al. 2020), the MalariaGEN project (Pearson et al. 2019), and the Pangeo project (Eynards-Bontemps et al. 2019; Signell and Pothina 2019; Abernathey et al. 2021).

In the weather enterprise, the Met Office has adopted Zarr as its file storage format of choice for the over 200 TB of data produced by high-resolution NWP models each day (McCaie 2019). Additionally, Unidata developers of netCDF have extended its netcdf-c library to access Zarr data in a storage format referred to as NCZarr (Heimbigner 2021). The Open Geospatial Consortium has recognized Zarr as a storage format of high interest but has not yet approved the proposal to specify Zarr, version 2, as an official community standard (https://www.ogc.org/search/content/zarr).

The HRRR model output in Zarr format developed in this study (hereinafter HRRR-Zarr) is one approach to extract and disseminate model output intended for common ML workflows that may require specific variables from one to thousands of model runs at specific locations. HRRR-Zarr makes it practical to access relatively small fractions of data rather than attempting to retrieve that data from the original

TABLE 1. Selected characteristics of HRRR CONUS versions from 2014 to present available from IaaS cloud providers.

HRRR CONUS		Forecast length for initialization times		No. of GRIB2 output files			
Version	First date	0000, 0600, 1200, and 1800 UTC	Other hours	Surface	Pressure	Native	Subhourly
1	30 Sep 2014	15	15	102	659	778	26
2	23 Aug 2016	18	18	135	687	1110	41
3	12 Jul 2018	36	18	151	701	1126	44
4	2 Dec 2020	48	18	173	711	1136	196

GRIB2 formatted files. The capability to do so is possible since HRRR-Zarr formatted objects are being created by our group, stored in the AWS S3 environment, and made publicly accessible as part of the AWS Sustainability Data Initiative, complementing the HRRR GRIB2 model archive available there (Amazon 2021). While the HRRR-Zarr archive is optimized for use via AWS Cloud services, it is important to note that it can be accessed from any machine.

The remainder of this paper will be organized in the following manner. We first detail the HRRR model specifications, Zarr capabilities and limitations, AWS HRRR-Zarr archive structure, and comparisons of the time required to access GRIB2 files versus Zarr formatted objects. Next, we explore potential use cases for the HRRR-Zarr dataset, for both research and operational applications. We will detail the benefits of the HRRR-Zarr format in a general sense, as well as demonstrate its utility in analyzing a high-impact meteorological event from September 2020 that included record-breaking downslope windstorms in two states, devastating wildfire spread, and an early season snowstorm. The final section presents a summary and future work.

2. Data and methods

a. The High-Resolution Rapid Refresh model

The HRRR is a 3-km, convection-allowing model that is run operationally by NCEP's Environmental Modeling Center (Benjamin et al. 2016). It was developed by the Earth Systems Research Laboratory and was first run operationally in September 2014. The latest version of the HRRR model (version 4, deployed 2 December 2020), is initialized each hour, with hourly forecasts out to either 18 or 48 h depending on the initialization time (Table 1). The operational HRRR domain covers the entire CONUS (Fig. 1), with a separate domain for Alaska (McCorkle et al. 2018).

NOAA BDP and CICS-NC staff manage the distribution of HRRR model output to IaaS providers Google and AWS. Public cloud platforms, such as Google and AWS, provide object stores rather than file systems to access massive amounts of data. Hence, we rely on the archive and real-time HRRR GRIB2 data available publicly as part of the AWS Sustainability Data Initiative (https://registry.opendata.aws/noaa-hrrr-pds/). The HRRR GRIB2 archive is publicly accessible via the AWS S3 using the unique identifier "noaa-hrrr-bdp-pds." We refer to GRIB2 objects as files since they are not a cloud-native format and generally need to be downloaded to a file system to be accessed.

HRRR output accessible from IaaS providers contains eight dimensions, listed below (dimensions contained in each GRIB2 file are listed in boldface text, with the remaining dimensions encoded in the naming conventions of separate files):

- File type: Surface, subhourly, isobaric, native
- Domain: CONUS, Alaska
- Initialization time: Hourly from 2014 to the present
- Forecast lead time: 15- or 60-min intervals out to 48 h
- **Level: Pressure, height, layer**
- **Variable: Sensible weather parameters and many model-specific fields**
- **X position**
- **Y position**

Table 1 summarizes the evolution of the HRRR model from its initial operational release in 2014 to the present. We only postprocess into Zarr format a limited amount of the output from the HRRR (Table 2). We have focused on reformatting the GRIB2 surface files as many ML use cases require surface sensible weather parameters or meteorological parameters at "standard" levels in the vertical that are stored in those GRIB2 files. At present, the volume of HRRR output in Zarr format accessible from AWS exceeds 145 TB and is growing at a rate of 120 GB per day.

We focus our description of HRRR-Zarr objects on the processing of the CONUS surface GRIB2 files, each of size ~140 MB, that contain 173 grids representing a mix of variables at levels in the vertical of highest interest for many applications (Table 1). Output from HRRR model runs initialized at 0000, 0600, 1200, and 1800 UTC are available hourly from

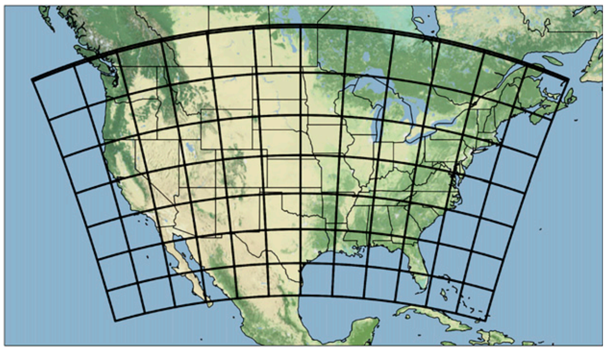

FIG. 1. HRRR domain (1799 × 1059 grid points) divided into 96 chunks of size 150 × 150 grid points, with the northernmost 12 chunks containing nine rows of "non-NaN" (null) data.

TABLE 2. Availability as of December 2021 of Zarr analysis and forecast files in AWS S3 for surface and isobaric file types.

HRRR CONUS		File type	
Version	First date	Analyses	Forecasts
2	23 Aug 2016	Surface	—
3	12 Jul 2018	Surface and isobaric	Surface
4	2 Dec 2020	Surface and isobaric	Surface

the analysis time (F00) and hourly forecast lead times out to 48 h (F48). The HRRR model runs initialized at other hours of the days are available from F00 to F18.

The subhourly files are similar to the surface files and contain variables with output available at 15-min forecast lead times. The isobaric and native files contain meteorological variables at fixed pressure or terrain-following levels, respectively, that are most relevant for users who need the HRRR output for initial and boundary conditions to initialize high-resolution forecasts or research simulations (e.g., Crosman and Horel 2017; Foster et al. 2017).

b. Zarr

Zarr is a flexible storage format for storing *N*-dimensional data arrays that are chunked (divided into subdomains) and compressed with metadata necessary for locating and interpreting data described in separate JSON-formatted files, where JSON is JavaScript Object Notation. The Zarr format provides similar functionality with some additional flexibility relative to the Hierarchical Data Format, version 5 (HDF5; Delaunay et al. 2019). Zarr stores are read and written with the Zarr Python library that depends on the widely used NumPy library (Harris et al. 2020). The Zarr format is becoming a desirable cloud-native file structure for data scientists and researchers because of its seamless ability to read and write to cloud platforms (Abernathey et al. 2021). Other benefits include its library of compression options, multithreading and multiprocessing capabilities, and its backend compatibility with format-agnostic, array-manipulation Python libraries (e.g., xarray, iris, and dask).

A Zarr store is initialized using Python and can be in memory, a directory on local disk, in distributed or cloud storage, or a zip file. Next, Zarr arrays (hereinafter zarrays) are created and filled in a similar manner to NumPy arrays by defining a data type and shape and then assigning values and defining zarray attributes (zattrs), stored in JSON files, that will serve as the key references for that zarray. These zarrays can be chunked along any specified dimension and in any shape, which allows a dataset to be manipulated and stored efficiently for use in specific applications. All chunks in a zarray are uniform in shape and stored as individual objects that are identified by their integer index (e.g., row and column) relative to other chunks.

The HRRR-Zarr archive with the unique AWS S3 bucket identifier "hrrrzarr" is available publicly as part of the AWS Sustainability Data Initiative and can be accessed from any platform (cloud, HPC, or personal machine) without incurring egress fees. The archive was designed to be relevant for users less familiar with environmental dataset formats and wanting "analysis-ready datasets" (Abernathey et al. 2021) while supporting a familiar environment for users who have used model output in netCDF-4 or GRIB2 format. The HRRR-Zarr conversion workflow follows that of the Met Office Informatics Laboratory, where they are actively using Zarr to store large datasets (Donkers 2020). Because of the challenges that surround manipulating data cubes in various file formats, the Met Office developed the Iris Python library, a format-agnostic library for processing datasets and converting between file formats (Met Office 2020). Unlike other Python libraries, Iris and its companion package, Iris-grib, were built to read data cube formats such as GRIB2 and recognize the Climate and Forecast (CF; Eaton et al. 2020) metadata conventions used in numerical model data. For this reason, the Iris libraries facilitate data conversion that maintains metadata such as units from the original GRIB2 files.

The HRRR-Zarr archive was built using the Iris and Iris-grib libraries. HRRR-Zarr data stores rely on the same self-describing metadata (keywords and CF naming scheme) as the corresponding GRIB2 files obtained from their associated index (.idx) files. As an example, consider the workflow required to process the 48 GRIB2 surface forecast files containing 173 grids from the HRRR model runs initialized at 0000 UTC. [The CF names for all 173 HRRR variables are available online (https://mesowest.utah.edu/html/hrrr/zarr_documentation/html/zarr_variables.html).] All of the grids from the 48 hourly forecast files are read into memory together and then organized into unique Iris data cubes containing data and metadata. The Iris data cubes are then converted to zarrays that are subdivided (chunked), encoded, and output into separate files identified by the parameter's CF name and atmospheric level or layer (e.g., 2-m or 500 mb).

As shown in Fig. 1, the 1799 $\times$ 1059 grid is subdivided into 96 chunks of size 150 $\times$ 150, chosen after rigorous trial and error with various sizes and considering recommendations for optimal data compression. Note that the 12 chunks along the domain's northern boundary contain data in only the southernmost nine rows. HRRR CONUS analysis (F00) files, whether for surface or isobaric files, are simply subdivided into 96 tiny 2D files each containing one 150 $\times$ 150 gridpoint array. HRRR CONUS forecast (F01–FXX) files are stored as 96 3D cubes (XX, 150, 150) where the forecast duration, FXX, depends on HRRR version and time of day (Table 1). Although the data are chunked, the entire domain can still be accessed efficiently, in terms of memory and processing time, with the zarr or xarray Python libraries.

Before sending the chunked zarrays to the Zarr store, they are encoded and compressed for optimized storage. The encoding instructions are located in the JSON metadata for the zarray and define the data type's byte order (little endian or big endian), character code (integer, floating point, Boolean, etc.), and the number of bytes. All data types in the NumPy array protocol are acceptable for zarray encoding. There exists a plethora of literature that details compression algorithm performance and benchmark test results that aid in choosing appropriate compression schemes for a particular

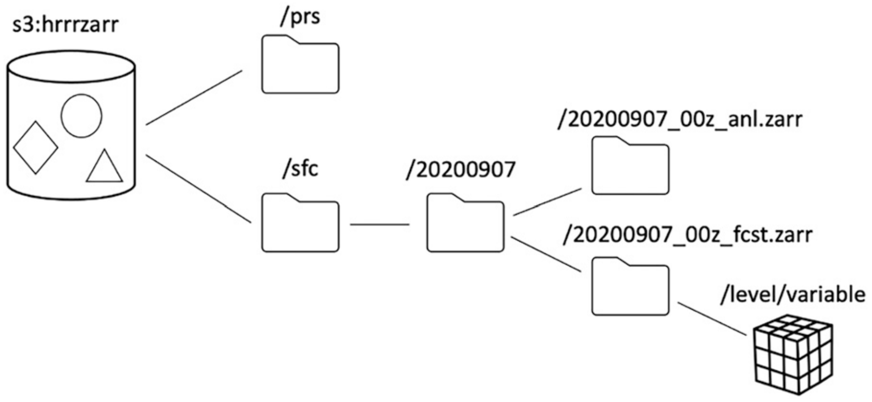

FIG. 2. Files within the AWS S3 bucket hrrrzarr are named to emulate a hierarchical data structure.

use case (Donoho 1993; Alted 2010; Almeida et al. 2014; Wang et al. 2015; Kuhn et al. 2016).

After testing, we chose the LZ4 compression codec, which is a lossless compression algorithm with the ability to quickly and efficiently compress large amounts of data (Collett 2020). The zarrays are usually encoded as 16-bit little-endian floats, although a few variables, such as surface pressure, require 32-bit little-endian floats. As part of the LZ4 compression, we use byte shuffling and a compression level of 9 within a range of 1–12 where levels 1 and 12 provide the fastest compression speed and highest compression ratio, respectively. While other compression "codecs" have been shown to produce higher data compression ratios, their decompression speeds are much slower (Collett 2020).

It is widely recognized that optimal use of cloud resources requires extensive data processing to be physically close to the data archive. Shortly after the entire model run is accessible as objects in the AWS GRIB2 S3 archive, the HRRR-Zarr stores are created using AWS Elastic Cloud Compute resources in the same region (us-west-1) as our hrrrzarr bucket. Since the HRRR-Zarr project began with GRIB2 files sourced from our University of Utah Pando archive, this is not the same region as the AWS GRIB2 archive (us-east-1). Because we must wait until all GRIB2 forecast fields are available, the most recent Zarr stores are typically available 3 h after the initialization time, e.g., 0000 UTC analysis and F01–F48 forecast files are available by 0300 UTC. HRRR-Zarr creation is dependent on the availability of the GRIB2 data in the AWS archive managed by NOAA BDP.

Within the hrrrzarr S3 bucket, all objects are contained in a flat structure where the concept of folder or directory structure is mimicked using shared name prefixes for objects with slashes to indicate a hierarchy. The zarr stores derived from the surface and isobaric sets of HRRR GRIB2 files are stored with the prefixes "sfc/" and "prs/," respectively (Fig. 2). Model runs are accessible by date, using suffixes for analysis (anl.zarr) and forecast (fcst.zarr) files for each model run; for example, files with the prefix sfc/20200907_12z_fcst.zarr/were generated from the F01–F48 HRRR GRIB2 surface files initialized at 1200 UTC 7 September 2020. Prefixes follow then based on level (e.g., 700 mb/or 10m_above_ground/) and CF naming conventions for variables a (e.g., TMP/or UGRD/) with the final part of the file name being the chunk identifier. A full list of variables (abbreviation and full name) available in the HRRR v4 output and HRRR-Zarr stores is available online (https://mesowest.utah.edu/html/hrrr/zarr_documentation/html/zarr_variables. html).

c. Accessing HRRR-Zarr objects

Answering many grand challenges that face the scientific community requires analyzing the entirety of massive datasets (Abernathey et al. 2021). Traditional analysis approaches that involve downloading individual files from data repositories become impractical for such needs. The Pangeo Project (https://pangeo.io) is an example of a geoscience-community architecture for analyzing large volumes of Earth system data (Abernathey et al. 2021). Similar to our generation of the HRRR-Zarr objects, that approach focuses on "data-proximate" cloud computing where users analyze the requisite data using cloud-based compute nodes.

Cloud-native repositories, such as HRRR-Zarr, are inherently intended to take advantage of parallelized computing using readily available Python libraries and tools such as dask. Abernathey et al. (2021) benchmarked throughput speeds for netCDF-4 versus Zarr objects as a function of the number of parallel reads. The read throughput for Zarr objects was ~10 times as fast as for the netCDF-4 objects with less indication of saturation as the number of parallel operations increased.

Blaylock et al. (2018) presented an approach to compute in parallel climatological statistics of HRRR model output at all 1.9 million grid points that required harnessing the Open Science Grid (OSG). The OSG allows users to send jobs that are repetitive in nature (statistical calculations using large datasets, data mining, etc.) to unused or idle computing resources at hundreds of locations within the OSG consortium, reducing the overall processing time for a given workflow. Although the OSG method enables large amounts of data to be simultaneously processed, its complexities can be a drawback for most users. Continually updating cumulative distributions using this approach is also difficult to sustain.

TABLE 3. Time (s) required to access CONUS-wide GRIB2 files and Zarr objects of surface pressure for F00–F48 hourly forecasts initialized at 0000 UTC 24 Nov 2021.

	Operation	AWS cloud	CHPC	Personal computer
GRIB2	Download files	53	69	563
	xarray load	35	62	48
	Total	88	131	611
Zarr	Open dataset	0.6	1	1
	xarray access	1.6	7	5
	Total	2	9	6

To help illustrate how the Zarr format and data-proximate computing can improve upon the OSG approach to compute basic statistical metrics from large amounts of model output, we first computed on a home laptop the 95th-percentile wind speed for a single Zarr subdomain from all HRRR hourly analyses during September 2017–19 (a 2160-member sample). While that operation required 143 s to download the requisite Zarr objects, only 2 s were required to process the data to obtain the 95th-percentile wind speed for this subdomain. Hence, the computation is highly IO bound, and a single machine would require 13 920 s (~4 h) to process all 96 subdomains. (Without Zarr, the 265 GB of GRIB2 files to download would make this calculation infeasible on a laptop.)

Since parallelizing IO-bound computations on a single high-performance node is ineffective, a Dask cluster using AWS Fargate was created using 48 worker nodes in addition to the scheduler. Each Fargate node runs as a Docker container in the requested region, which allows for data-proximate computing. One script is used to set up the Dask cluster, configure AWS permissions, build and host the Docker image with the required libraries, and complete the analysis. Provisioning the cluster required 150 s, and processing all 96 chunks in parallel took 345 s, for a total time of 495 s. Hence, parallelizing this computation using cloud computing resources would lead to an ~28× speedup without requiring excessive memory or local disk storage.

However, many users may need to access data from a specific environment or not have the financial resources to use cloud computing. Efficient access from their local computing environment to targeted data within cloud-based object stores is an alternative. We illustrate in Table 3 examples of the relative times to access HRRR output in GRIB2 and Zarr formats using data-proximate AWS nodes, high-performance servers managed by the Center for High Performance Computing (CHPC) at the University of Utah, or a typical laptop run over a home network. Xarray's "lazy loading" approach for multidimensional datasets improves access efficiency overall since the data are not requested until explicitly required for computations (Hoyer and Hamman 2017).

The most direct comparisons are in the leftmost column of Table 3. These are the times required using AWS data-proximate nodes to 1) download 49 separate GRIB2 forecast files (F00–F48) requiring 7.25 GB of local storage and access the CONUS-scale grids of surface pressure within those files using xarray and 2) open and access the 96 Zarr chunks from both analysis and forecast arrays totaling 356 MB. Hence, for this specific application, the data would be available for xarray operations over 40 times faster using Zarr. In fact, even reading from the 49 already-downloaded GRIB2 files is several times slower than accessing the same data from the 2 corresponding Zarr stores on S3. The times in the non-AWS columns are potentially affected by many external factors, e.g., load sharing of the CHPC high-performance compute node as a function of time of day, network architecture and distance from the different AWS data centers. However, after running these comparisons multiple times, the variations likely resulting from such factors were much smaller than the differences evident in Table 3 arising from access to GRIB2 files versus Zarr objects. For example, the CHPC high-performance computing nodes have relatively fast download speeds for either GRIB2 or Zarr due to high-speed connectivity to IaaS providers. While downloading the GRIB2 files to a local personal computer may not be practical, the direct access to AWS HRRR-Zarr objects from a laptop is comparable to that from a CHPC node.

3. HRRR-Zarr applications

The HRRR-Zarr archive was developed with the intention of expanding its utility for diverse applications that require high-velocity file throughput. While demonstrating a full ML scenario is outside the scope of our research, this section illustrates examples of situations where the Zarr storage format may be optimal in terms of efficiency and ease of use. We will use a high-impact meteorological event from September 2020 to showcase the utility of model data in Zarr format for not only research applications, but operational decision making and forecasting use cases as well. This section of the paper will be composed of subsections that detail the event that we are analyzing, followed by example use cases for HRRR model output in Zarr format.

a. Labor Day weather event (7–9 September 2020)

In the days leading up to the historic 2020 Labor Day weather event, forecasters in the western half of the United States were on high alert for the extratropical transition of Typhoon Julian as it began recurving poleward and eastward. When extratropical transitions occur, tropical cyclones may interact with the midlatitude flow such that the midlatitude ridge-wave patterns amplify, and high-impact weather occurs downstream (Bosart and Carr 1978; Cordeira et al. 2013; Feser et al. 2015; Keller et al. 2019). In this case, Typhoon Julian did modify the midlatitude wave pattern by amplifying both the anticyclone over the Gulf of Alaska and the midlatitude cyclone situated over western Canada. The rapid intensification of this ridge-trough pair ultimately produced far-reaching effects including historic windstorms and unrelenting wildfire spread (Fig. 3) in the Pacific Northwest, strong downslope winds in Utah, and a snowstorm in Colorado. We focus here on the data and model forecasts pertaining to the events that occurred in Oregon, west of the Cascade Mountains.

The Labor Day weather event was synoptically driven and well forecast several days in advance. Prior to the trough

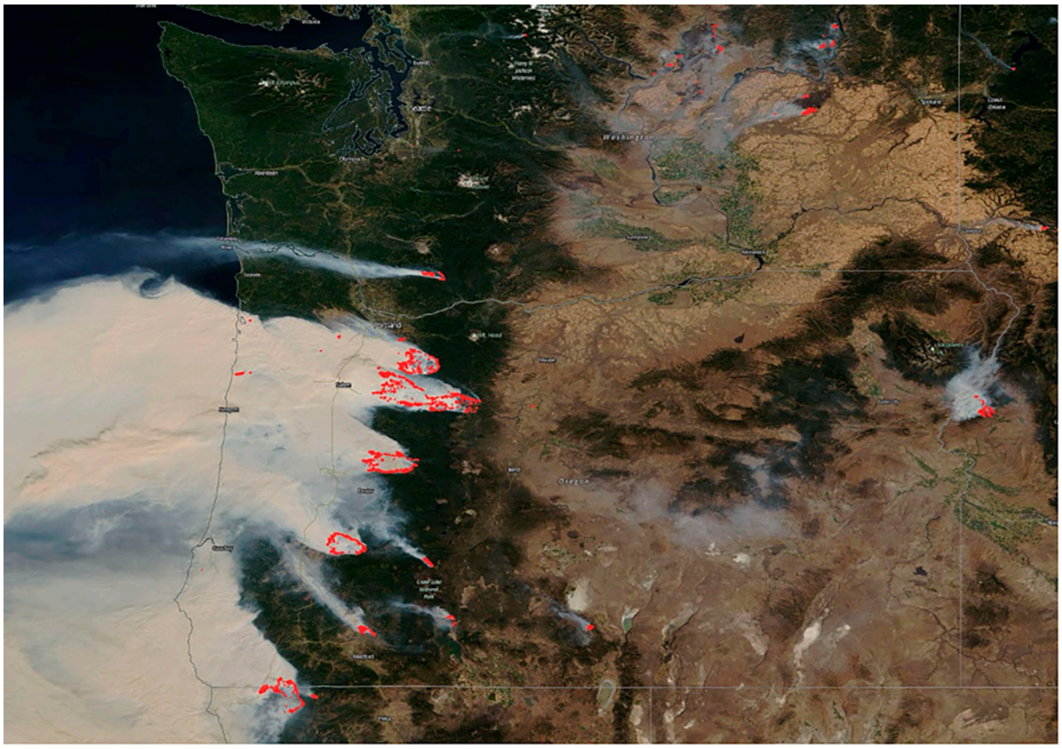

Fig. 3. Boundaries of active fires (red outlines) estimated using VIIRS 375-m thermal anomalies, and smoke from wildfires in the Pacific Northwest on 9 Sep 2020 (source: https://worldview.earthdata.nasa.gov).

arrival and onset of the downslope windstorm, the Pacific Northwest was experiencing extreme fire danger caused by warm and dry conditions, with several fires already burning in Washington and Oregon. By 1200 UTC 7 September, a thermal trough was situated over coastal Oregon with a tightening pressure gradient orthogonal to it. These conditions are indicative of impending strong northeast and easterly winds in western Oregon. As forecast, strong easterly winds arrived on the western side of the Oregon Cascades by 0000 UTC 8 September. In a near worst case scenario, wind gusts along the western slopes of the Oregon Cascades significantly intensified and spread the human-caused Riverside and Beechie Creek fires. For nearly a week after the onset of the downslope winds, persistent easterly flow propagated wildfire smoke west, resulting in historic PM2.5 measurements in excess of 500 μg m^{-3} in Portland, Salem, and Eugene, Oregon (Green 2020). Suppression efforts were challenging given the steep terrain surrounding the wildfires, making it dangerous for fire crews to safely extinguish them. Ultimately, the Riverside and Beechie Creek fires burned over 1300 km^2. The following sections use the data during this weather event to illustrate use cases for future ML applications for operational forecasting and research.

b. Forecast time series for a specific location

Despite their inherent simplicity, requiring only time and a dependent variable as input, time series can be time consuming and challenging to create when starting from data files that represent a single time in space for millions of locations, as is the case with NWP model output in GRIB2 format. Efficient access to model output as time series for specific locations was a key objective leading to the structure and organization of our HRRR-Zarr format. While identical time series can be constructed from both GRIB2 and Zarr data storage formats, the process and requirements are quite different. As discussed in section 2, the tiny two-dimensional analysis HRRR-Zarr stores can be easily accessed to estimate prior conditions at a location while the three-dimensional forecast HRRR-Zarr stores contain all forecast hours from a model run to assess how future conditions at that location may unfold.

To illustrate the utility of the Zarr format for this use case, we plot time series of forecast wind gusts from the 0000, 0600, 1200, and 1800 UTC HRRR model runs for a single point from 1200 UTC 6–1800 UTC 8 September 2020 (Fig. 3). For this case, we chose the HRRR grid point nearest to the Horse Creek (Station identifier: HSFO3) Remote Automated Weather Station (44.940 806°N, 122.400 806°W) located downwind of the Beechie Creek fire. However, since HSFO3 is located in a clearing within a densely forested region, the wind reports from this location tended to be lower than what was evident by the rapid advance of the fire line in that region. To create this visualization of forecast wind gust, 10 small chunks of data (one from each model run) totaling ~10 MB were retrieved from the hrrrzarr bucket to obtain all necessary model output. In contrast, 360 GRIB2 files totaling ~54 GB would have been needed to replicate this process or else values within byte ranges in each of those files would

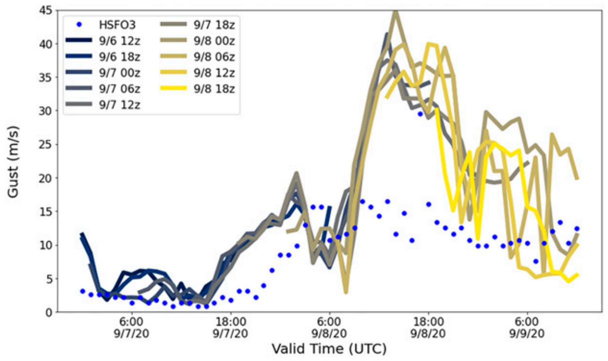

FIG. 4. Wind gusts (m s^{-1}) from HSFO3 (blue dots) and HRRR wind gust forecasts near HSFO3, colored corresponding to initialization time.

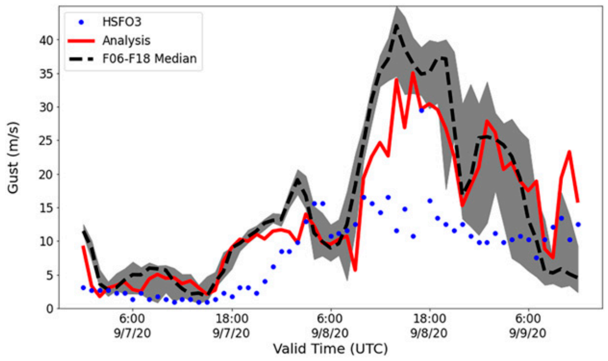

FIG. 5. Wind gusts (m s^{-1}) from HSFO3 (blue dots), HRRR analyses (red line), and median of F06–F18 forecasts (dashed black line) near HFSO3 for valid times from 0000 UTC 7 to 1200 UTC 9 Sep. The shading indicates the range between the maximum and minimum wind gusts from the F06–F18 time-lagged ensemble.

need to be determined and accessed. The single access point to all forecast hours in a model run reduces processing time and optimizes workflows for many applications.

Plotting sequentially the model runs available every hour creates a time-lagged ensemble (TLE) for a given valid time. A TLE from HRRR output can provide useful diagnostics for evaluating the uncertainty or spread in values among recent forecasts for which the most recent forecast provides only deterministic guidance (Xu et al. 2019). TLEs with sufficient lead time to be potentially useful operationally can be constructed using a set of sequential HRRR forecasts, with each model run treated as an ensemble member. In this case, we use F06–F18 forecasts from all model runs initialized from 0600 UTC 6 to 0600 UTC 9 September to calculate statistics at valid times from 0000 UTC 7 to 1200 UTC 9 September. Diagnostic values such as median, minimum, and maximum forecast wind gusts provide a simple evaluation of the model's uncertainty as the event unfolded (Fig. 4). The unrepresentativeness of the lighter HSFO3 observations relative to that analyzed and forecast by the HRRR is evident in Fig. 4.

c. Spatial analysis of forecast data

Many applications requiring HRRR model output need only a fraction of the 1.9 million grid points in the HRRR CONUS domain. Hence, users typically implement methods to subset areas of interest from the complete grids. Accessing one or more HRRR-Zarr chunks of size 450 km^2 may help simplify that process for many local applications while adjacent chunks can be stitched together to evaluate conditions for regional scales.

Model analyses are often used as proxies for observations, especially in areas of complex terrain where in situ measurements may not be available or representative of prevailing conditions (e.g., Fig. 5). As a further example, we use the HRRR-Zarr analysis stores to determine the onset time of wind gusts exceeding 10 m s^{-1} for every point within the western Oregon chunk encompassing the large fires under way on 7–8 September 2020 (Fig. 6). This wind gust threshold was chosen on the basis of criteria commonly used for red flag warnings issued by the National Weather Service. The filled contours in Fig. 6 depict the approximate onset time of the downslope windstorm event across western Oregon, with the event beginning along the highest reaches of the Cascade Range and then progressing westward later. Such diagnostics can then be related to available wind observations and damage reports to help evaluate the ability of the HRRR model to forecast the temporal evolution of the event.

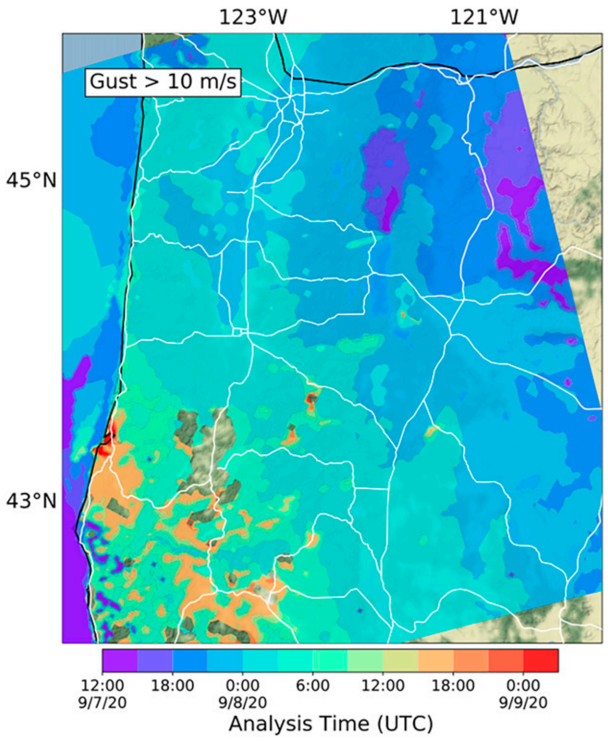

FIG. 6. Time of first HRRR analysis (F00) with a wind gust exceeding 10 m s^{-1} (shaded according to the scale) at each grid point for model runs initialized between 1200 UTC 7 and 0300 UTC 9 Sep.

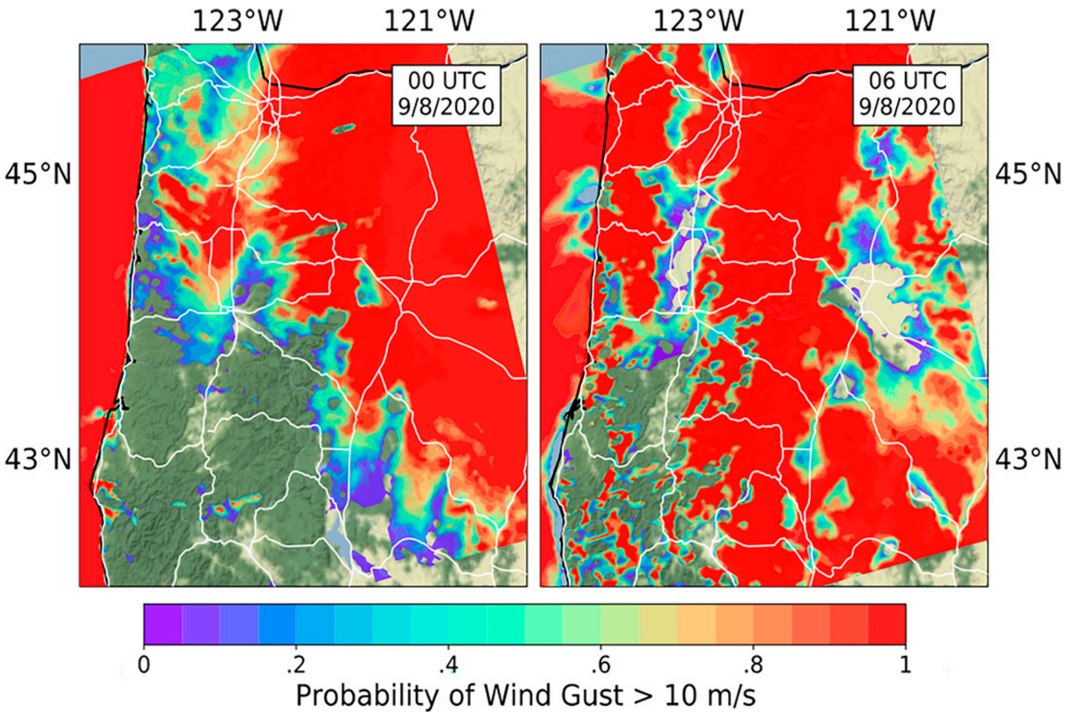

FIG. 7. Fraction of HRRR wind gust forecasts exceeding 10 m s^{-1} at valid times (left) 0000 and (right) 0600 UTC 8 Sep 2020. Contours correspond to probability values (0–1) and are shaded according to the scale.

Building on the TLE concept available from consecutive HRRR forecasts, we calculate the probability of a HRRR wind gust forecast exceeding 10 m s^{-1} at a given time during the downslope windstorm for all grid points within the western Oregon chunk. The 21 model runs (F01–F18, F24, F30, and F36) available from forecasts valid at 0000 and 0600 UTC 8 September 2020 are used to calculate the fraction of wind gusts forecasts exceeding that threshold in this subregion (Fig. 7). Using such probabilistic guidance as the event developed, forecasters might have higher confidence that the HRRR model forecasts issued earlier are being confirmed by more recent forecasts as the downslope winds continued. For a single valid time, this metric utilized wind gust values within 20 HRRR-Zarr stores, which required less than 20 MB of storage capacity, an amount easily manageable in computer memory. Actual forecast applications might limit the TLE members to those available at least 12 h in advance, for example, forecasts with lead times from F12 to F18 and those available every 6 h out to 48 h from the HRRRv4 model output now available.

d. Empirical cumulative distributions

Empirical cumulative distributions of model data and observations are often utilized to better understand the range of possible values for a given parameter as a function of time and/or location and can be used to correct for model biases (Blaylock et al. 2018; Gowan and Horel 2020). If enough data are available over an adequate period, these cumulative distributions can be thought of as a climatology and used for comparison with a parameter at an equivalent time or location to recognize conditions that are likely anomalous. Creating distributions from observations or model output typically requires data from thousands of input times and files for the information to be considered useful. This can be a daunting and time-consuming task since a large amount of storage and compute power are needed to efficiently process thousands of GRIB2 data files.

To illustrate accessing large amounts of data from the HRRR-Zarr store, we generated empirical cumulative distributions for selected atmospheric parameters using a single high-performance compute node. This use case motivated the later analysis in section 2d to evaluate the additional speedup that might result using an AWS dask cluster. To assess the anomalous nature of the downslope wind event during September 2020 in northern Oregon, we generated cumulative distributions of wind gust data for each grid point in that region by accessing all HRRR hourly analyses during the month of September during the preceding years 2016–19. Each grid point's cumulative distribution is derived then from 2880 wind gust values, one from every hourly HRRR analysis during the four calendar months. A range of percentiles can be derived from the empirical distributions to estimate normal and above normal wind gusts in this area during the month of September.

As expected, the highest wind gusts evident from the 95th-percentile values during September 2016–19 tend to occur over the Cascade Range and offshore (Fig. 8). Using this 4-yr distribution, we then compare the 95th-percentile values with the analysis and F06, F12, and F18 forecasts valid at 0600 UTC 8 September 2020 (Fig. 9). To emphasize the severity of

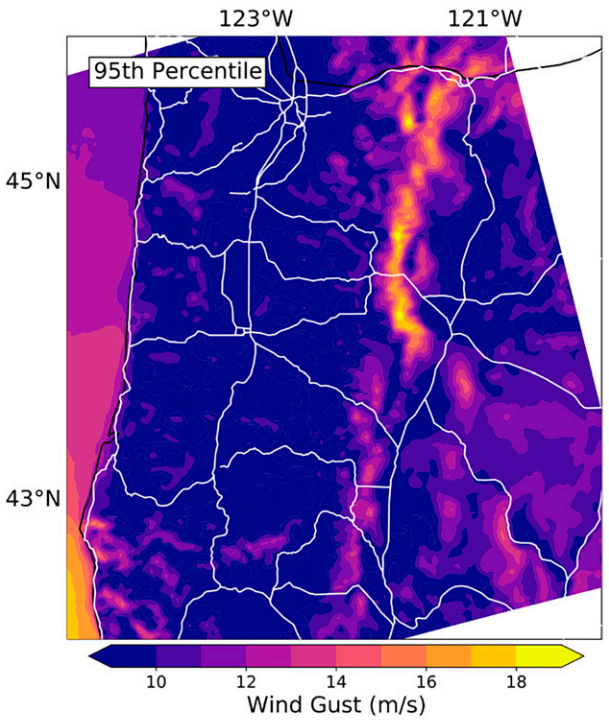

FIG. 8. The 95th-percentile wind gust values (m s^{-1}; shaded according to the scale) calculated at each grid point from empirical cumulative distributions derived from HRRR analyses during September 2016–19.

the event across the region, the excess magnitude of wind gusts values above the 95th percentile is shown. Comparing these forecasts and analysis with the cumulative distribution is a simple way to show how anomalous this event was, with wind gusts exceeding the 95th-percentile values by 15–30 m s^{-1} over the Cascade and portions of the Coast Ranges and extending into sections of the Willamette Valley.

The empirical distributions computed using 4 months of data for a single variable and chunk required less than a minute on a typical workstation, reading from Zarr stores on a local network drive. In contrast, Blaylock et al. (2018) calculated empirical cumulative distributions for all HRRR model grid points. These distributions were then used to output wind speed values at 19 percentiles at all HRRR grid points for each day of the year. As previously stated, this was a rigorous and time-intensive endeavor that required an enormous amount of model output. Ultimately, calculating these distributions resulted in the need to store 6935 additional files containing the percentiles at each of the 1.9 million HRRR grid points.

Calculating empirical cumulative distributions, as well as other large-scale statistical metrics, as needed with data in Zarr format gives the end user the ability to continually update their statistics as new information is received. This method especially benefits users who are interested in time-sensitive datasets, like those from numerical weather prediction models. Using the HRRR-Zarr method, a user will be able to efficiently compute statistics that are tailored to a specific application or workflow, without dealing with the overhead of many gigabytes of excess data.

4. Summary

Vast amounts of output produced by numerical weather prediction models are accessed and processed every day for applications ranging from operational forecasting to research and machine learning. As advancements in technology allow for finer time and spatial resolution model output, users may struggle to keep up even if they are only interested in accessing a small fraction of the data available. Much of this model output is currently available in GRIB2-formatted files containing hundreds of two-dimensional variable fields for a single valid time. Despite the highly compressible nature of GRIB2, the size of each forecast hour file is often $O(100)$ MB, making high-volume input/output applications challenging due to the memory and compute resources needed to parse them.

We present an approach that reorganizes HRRR analyses (F00) from the surface and isobaric HRRR file types into tiny two-dimensional (150, 150) files in Zarr format for each variable/vertical-level combination and 96 subdomains of the CONUS grid. HRRR forecasts from the surface files are stored as data cubes (XX, 150, 150) where the forecast dimension XX is either 48 for initialization times of 0000, 0600, 1200, and 1800 UTC or 18 for all other hours. We create the Zarr stores from the HRRR GRIB2 files provided by the NOAA BDP with support provided by the Amazon Sustainability Data Initiative. Our supplementary S3 bucket, hrrrzarr, is publicly accessible as part of the Amazon Initiative.

The structure of the HRRR-Zarr archive was designed to allow users the flexibility to access only the data they need through selecting subdomains and parameters of interest without the overhead of memory and processing requirements that comes from accessing numerous large GRIB2 files. Users may retrieve the analysis files needed to diagnose prior conditions or retrieve the forecast files in combination with the analysis files to evaluate future conditions or validate prior forecasts.

Substantial performance improvements are illustrated using Zarr in data-proximate parallelized cloud computing. However, it is also possible to access cloud-native repositories such as HRRR-Zarr very efficiently from local compute resources ranging from personal laptops to high-performance compute nodes.

Using a high-impact weather event from September 2020, we present workflow examples for analyzing large amounts of sensible weather parameters from the HRRR-Zarr data archive in a limited subdomain: assembling time series for a specific grid point of forecast conditions over a range of model runs, examining similarities and differences among samples of model forecasts for the same valid times from successive model runs, calculating empirical cumulative distributions over multiyear periods, and detecting forecasts of extreme conditions relative to conditions during other recent years. The small, compressed chunks of data are ideal for

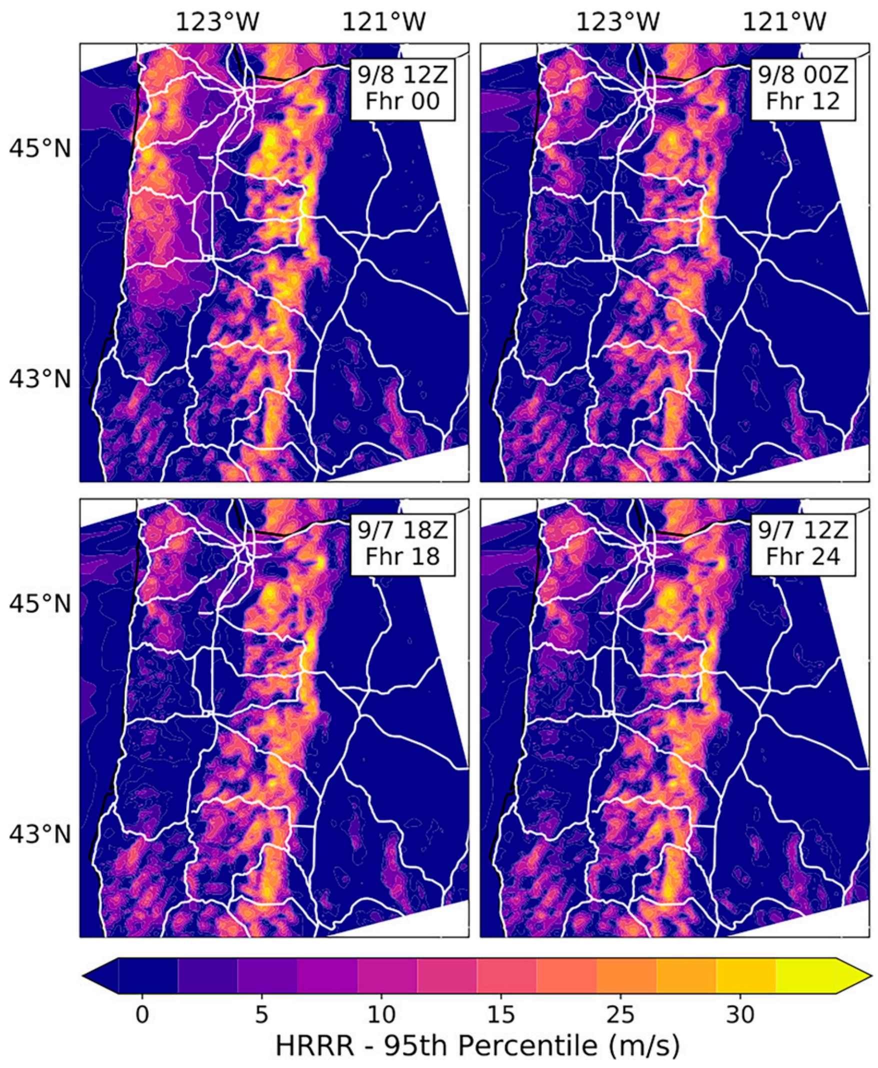

FIG. 9. Wind speed (m s^{-1}; shaded according to the scale) in excess of the 95th-percentile wind gust values at 1200 UTC 8 Sep 2020 for (top left) the verifying analysis and (top right) F12, (bottom left) F18, and (bottom right) F24 forecasts valid at that time.

high-throughput workflows where minimizing processing time or accessing files corresponding to many different valid times is critical. However, relying on the GRIB2 HRRR files accessible from AWS and Google remains the best option for initializing high-resolution model simulations that require many variables at multiple levels over a limited sample of valid times.

Our GRIB2-to-Zarr conversion of the HRRR model archive is only one of many research endeavors that aim to make model data more accessible to end users. Cloud repositories whether relying on GRIB2, netCDF-4, or Zarr data formats are increasingly being accessed using xarray library tools. While the Zarr Python library is relatively stable, xarray is continuing to undergo rapid development in order to support many different formats and protocols. Greater utilization of Zarr within the broader community will likely follow if the Open Geospatial Consortium adopts Zarr as an official community data standard. That will likely lead to diverse efforts to use Zarr in cloud-based data repositories as well as being relevant to users who want the flexibility to customize a Zarr store structure for their own purposes. Utilizing the Zarr format as an alternative file structure for the vast amount of numerical weather prediction output may help expand its already wide reach to data scientists in other disciplines while optimizing workflows for end users throughout the weather enterprise.

Acknowledgments. This research is supported by the NOAA/National Weather Service Collaborative Science, Technology, and Applied Research (CSTAR) Program Awards (55500146 and NA20NWS4680046). Creation of the HRRR-Zarr archive would not have been possible without an Amazon

Sustainability Data Initiative Promotional Credits Award. The authors thank Zac Flamig for his assistance and time in facilitating the inclusion of the HRRR-Zarr dataset in the AWS Open Data Registry. The authors also thank Brian Blaylock for his work to develop the HRRR archive at the University of Utah and recognize James Powell, who tested the dataset and provided valuable feedback. The University of Utah Center for High Performance Computing provided computational hardware and software for this work.

Data availability statement. The HRRR-Zarr archive is publicly available in the AWS Open Data Registry, made possible by credits awarded from the Amazon Sustainability Data Initiative (https://registry.opendata.aws/noaa-hrrr-pds/). Metadata documentation related to the HRRR model can be found online (https://doi.org/10.7278/S5JQ0Z5B).

REFERENCES

Abernathey, R. P., C. C. Blackmon-Luca, T. J. Crone, N. Henderson, and C. Lepore, 2021: Cloud-native repositories for big scientific data. *Comput. Sci. Eng.*, **23**, 26–35, https://doi.org/10.1109/MCSE.2021.3059437.

Adams-Selin, R. D., and C. L. Ziegler, 2016: Forecasting hail using a one-dimensional hail growth model within WRF. *Mon. Wea. Rev.*, **144**, 4919–4939, https://doi.org/10.1175/MWR-D-16-0027.1.

Almeida, S., V. Oliveira, A. Pina, and M. Melle-Franco, 2014: Two high-performance alternatives to ZLIB scientific-data compression. *Proc. 14th Int. Conf. on Computational Science and Its Applications*, Guimarães, Portugal, ICCSA, 623–638, https://doi.org/10.1007/978-3-319-09147-1_45.

Alted, F., 2010: Why modern CPUs are starving and what can be done about it. *Comput. Sci. Eng.*, **12**, 68–71, https://doi.org/10.1109/MCSE.2010.51.

Amazon, 2021: Amazon S3: Object storage built to store and retrieve any amount of data from anywhere. Accessed 2 January 2021, https://aws.amazon.com/s3/.

Ansari, S., and Coauthors, 2018: Unlocking the potential of NEXRAD data through NOAA's Big Data Partnership. *Bull. Amer. Meteor. Soc.*, **99**, 189–204, https://doi.org/10.1175/BAMS-D-16-0021.1.

Arulraj, M., and A. Barros, 2021: Automatic detection and classification of low-level orographic precipitation processes from space-borne radars using machine learning. *Remote Sensing Environ.*, **257**, 112355, https://doi.org/10.1016/j.rse.2021.112355.

Benjamin, S. G., and Coauthors, 2016: A North American hourly assimilation and model forecast cycle: The Rapid Refresh. *Mon. Wea. Rev.*, **144**, 1669–1694, https://doi.org/10.1175/MWR-D-15-0242.1.

——, J. M. Brown, G. Brunet, P. Lynch, K. Saito, and T. W. Schlatter, 2018: 100 years of progress in forecasting and NWP applications. *A Century of Progress in Atmospheric and Related Sciences: Celebrating the American Meteorological Society Centennial*, *Meteor. Monogr.*, No. 59, Amer. Meteor. Soc., https://doi.org/10.1175/AMSMONOGRAPHS-D-18-0020.1.

Blaylock, B. K., J. D. Horel, and E. T. Crosman, 2017a: Impact of lake breezes on summer ozone concentrations in the Salt Lake valley. *J. Appl. Meteor. Climatol.*, **56**, 353–370, https://doi.org/10.1175/JAMC-D-16-0216.1.

——, ——, and S. T. Liston, 2017b: Cloud archiving and data mining of High-Resolution Rapid Refresh forecast model output. *Comput. Geosci.*, **109**, 43–50, https://doi.org/10.1016/j.cageo.2017.08.005.

——, ——, and C. Galli, 2018: High-Resolution Rapid Refresh model data analytics derived on the open science grid to assist wildland fire weather assessment. *J. Atmos. Oceanic Technol.*, **35**, 2213–2227, https://doi.org/10.1175/JTECH-D-18-0073.1.

Bosart, L. F., and F. H. Carr, 1978: A case study of excessive rainfall centered around Wellsville, New York, 20–21 June 1972. *Mon. Wea. Rev.*, **106**, 348–362, https://doi.org/10.1175/1520-0493(1978)106<0348:ACSOER>2.0.CO;2.

Cintineo, J. L., M. J. Pavolonis, J. M. Sieglaff, and D. T. Lindsey, 2014: An empirical model for assessing the severe weather potential of developing convection. *Wea. Forecasting*, **29**, 639–653, https://doi.org/10.1175/WAF-D-13-00113.1.

Collett, Y., 2020: LZ4—Extremely fast compression, version 1.6.2. GitHub, accessed 1 January 2021, https://lz4.github.io/lz4/.

Cordeira, J. M., F. M. Ralph, and B. J. Moore, 2013: The development and evolution of two atmospheric rivers in proximity to western North Pacific tropical cyclones in October 2010. *Mon. Wea. Rev.*, **141**, 4234–4255, https://doi.org/10.1175/MWR-D-13-00019.1.

Crosman, E., and J. Horel, 2017: Large-eddy simulations of a Salt Lake valley cold-air pool. *Atmos. Res.*, **193**, 10–25, https://doi.org/10.1016/j.atmosres.2017.04.010.

Delaunay, X., A. Courtois, and F. Gouillon, 2019: Evaluation of lossless and lossy algorithms for the compression of scientific datasets in netCDF-4 or HDF5 files. *Geosci. Model Dev.*, **12**, 4099–4113, https://doi.org/10.5194/gmd-12-4099-2019.

Donkers, K., 2020: To the cloud and back again. Medium, accessed 28 December 2020, https://medium.com/informatics-lab/create-zarr-from-pp-files-ffa6b7972d6f.

Donoho, D. L., 1993: Unconditional bases are optimal bases for data compression and for statistical estimation. *Appl. Comput. Harmonic Anal.*, **1**, 100–115, https://doi.org/10.1006/acha.1993.1008.

Eaton, B., and Coauthors, 2020: NetCDF climate and forecast (CF) metadata conventions, version 1.8. CF Metadata Doc., 183 pp., https://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.pdf.

Elmore, K. L., Z. L. Flamig, V. Lakshmanan, B. T. Kaney, V. Farmer, H. D. Reeves, and L. P. Rothfusz, 2015: Verifying forecast precipitation type with mPING. *Wea. Forecasting*, **30**, 656–667, https://doi.org/10.1175/WAF-D-14-00068.1.

Eynard-Bontemps, G., R. Abernathey, J. Hamman, A. Ponte, and W. Rath, 2019: The Pangeo Big Data ecosystem and its use at CNES. *Proc. 2019 Conf. on Big Data from Space*, Munich, Germany, ESA, 49–52, https://doi.org/10.2760/848593.

Feser, F., M. Schubert-Frisus, H. von Storch, M. Zahn, M. Barcikowska, S. Haeseler, C. Lefebvre, and M. Stendel, 2015: Hurricane Gonzalo and its extratropical transition to a strong European storm. *Bull. Amer. Meteor. Soc.*, **96**, S51–S55, https://doi.org/10.1175/BAMS-D-15-00122.1.

Foster, C., E. Crosman, and J. Horel, 2017: Simulations of a cold-air pool in Utah's Salt Lake valley: Sensitivity to land use and snow cover. *Bound.-Layer Meteor.*, **164**, 63–87, https://doi.org/10.1007/s10546-017-0240-7.

Giuliani, G., B. Chatenoux, T. Piller, F. Moser, and P. Lacroix, 2020: Data Cube on Demand (DCoD): Generating an Earth observation data cube anywhere in the world. *Int. J. Appl. Earth Obs. Geoinf.*, **87**, 102035, https://doi.org/10.1016/j.jag.2019.102035.

Gowan, T. A., and J. Horel, 2020: Evaluation of IMERG-E precipitation estimates for fire weather applications in Alaska. *Wea. Forecasting*, **35**, 1831–1843, https://doi.org/10.1175/WAF-D-20-0023.1.

Green, A., 2020: Portland's air quality is off the charts Sunday, and parts of Oregon are just as bad due to wildfires. *The Oregonian*, accessed 15 January 2021, https://www.oregonlive.com/news/2020/09/portlands-air-quality-is-off-the-charts-on-sunday-and-much-of-oregon-is-just-as-bad-due-to-wildfires.html.

Harris, C. R., and Coauthors, 2020: Array programming with NumPy. *Nature*, **585**, 357–362, https://doi.org/10.1038/s41586-020-2649-2.

Heimbigner, D., 2021: Overview of Zarr support in netCDF-C. UCAR, accessed 31 May 2021, https://www.unidata.ucar.edu/blogs/developer/en/entry/overview-of-zarr-support-in.

Houston, J., G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, 2020: One thousand and one hours: Self driving motion prediction dataset. Lyft, accessed 9 December 2020, https://level5.lyft.com/dataset/.

Hoyer, S., and J. Hamman, 2017: xarray: N-D labeled arrays and datasets in Python. *J. Open Res. Software*, **5**, 10, https://doi.org/10.5334/jors.148.

Keller, J. H., and Coauthors, 2019: The extratropical transition of tropical cyclones. Part II: Interaction with the midlatitude flow, downstream impacts, and implications for predictability. *Mon. Wea. Rev.*, **147**, 1077–1106, https://doi.org/10.1175/MWR-D-17-0329.1.

Kuhn, M., J. M. Kunkel, and T. Ludwig, 2016: Data compression for climate data. *Supercomput. Front. Innovations*, **3**, 75–94, https://doi.org/10.14529/jsfi160105.

Lagerquist, R., 2016: Using machine learning to predict damaging straight-line convective winds. M.S. thesis, School of Meteorology, University of Oklahoma, 251 pp., http://hdl.handle.net/11244/44921.

Lazo, J. K., R. E. Morss, and J. L. Demuth, 2009: 300 billion served: Sources, perceptions, uses, and values of weather forecasts. *Bull. Amer. Meteor. Soc.*, **90**, 785–798, https://doi.org/10.1175/2008BAMS2604.1.

McCaie, T., 2019: Creating a data format for high momentum datasets. Medium, accessed 20 December 2020, https://medium.com/informatics-lab/creating-a-data-format-for-high-momentum-datasets-a394fa48b671.

McCorkle, T. A., J. D. Horel, A. A. Jacques, and T. Alcott, 2018: Evaluating the experimental High-Resolution Rapid Refresh–Alaska modeling system using USArray pressure observations. *Wea. Forecasting*, **33**, 933–953, https://doi.org/10.1175/WAF-D-17-0155.1.

McGovern, A., K. L. Elmore, D. J. Gagne, S. E. Haupt, C. D. Karstens, R. Lagerquist, T. Smith, and J. K. Williams, 2017: Using artificial intelligence to improve real-time decision-making for high-impact weather. *Bull. Amer. Meteor. Soc.*, **98**, 2073–2090, https://doi.org/10.1175/BAMS-D-16-0123.1.

Met Office, 2020: Iris: A Python library for analysing and visualising meteorological and oceanographic datasets, version 2.4.0. SciTools, accessed 22 December 2020, https://scitools.org.uk/iris.

Miles, A., and Coauthors, 2020: zarr-developers/zarr-python: v2.5.0. Zenodo, https://zenodo.org/record/4069231.

Molthan, A. L., J. L. Case, J. Venner, R. Schroeder, M. R. Checchi, B. T. Zavodsky, A. Limaye, and R. G. O'Brien, 2015: Clouds in the cloud: Weather forecasts and applications within cloud computing environments. *Bull. Amer. Meteor. Soc.*, **96**, 1369–1379, https://doi.org/10.1175/BAMS-D-14-00013.1.

Nativi, S., P. Mazzetti, and M. Craglia, 2017: A view-based model of data-cube to support big Earth data systems interoperability. *Big Earth Data*, **1**, 75–99, https://doi.org/10.1080/20964471.2017.1404232.

NOAA, 2020: Big Data Program. Accessed 8 December 2020, https://www.noaa.gov/organization/information-technology/big-data-program.

Pearson, R. D., R. Amato, and D. P. Kwiatkowski, 2019: An open dataset of *Plasmodium falciparum* genome variation in 7,000 worldwide samples. bioRxiv, https://doi.org/10.1101/824730.

Reeves, H. D., K. L. Elmore, A. Ryzhkov, T. Schuur, and J. Krause, 2014: Sources of uncertainty in precipitation-type forecasting. *Wea. Forecasting*, **29**, 936–953, https://doi.org/10.1175/WAF-D-14-00007.1.

Sharman, R., 2016: Nature of aviation turbulence. *Aviation Turbulence: Processes, Detection, Prediction*, R. Sharman and T. Lane, Eds., Springer, 3–30, https://doi.org/10.1007/978-3-319-23630-8_1.

Signell, R. P., and D. Pothina, 2019: Analysis and visualization of coastal ocean model data in the cloud. *J. Mar. Sci. Eng.*, **7**, 110, https://doi.org/10.3390/jmse7040110.

Silver, J., and C. Zender, 2017: The compression-error trade-off for large gridded data sets. *Geosci. Model Dev.*, **10**, 413–423, https://doi.org/10.5194/gmd-10-413-2017.

Siuta, D., G. West, H. Modzelewski, R. Schigas, and R. Stull, 2016: Viability of cloud computing for real-time numerical weather prediction. *Wea. Forecasting*, **31**, 1985–1996, https://doi.org/10.1175/WAF-D-16-0075.1.

Vance, T. C., and Coauthors, 2019: From the oceans to the cloud: Opportunities and challenges for data, models, computation and workflows. *Front. Mar. Sci.*, **6**, 211, https://doi.org/10.3389/fmars.2019.00211.

Vannitsem, S., and Coauthors, 2021: Statistical postprocessing for weather forecasts: Review, challenges and avenues in a big data world. *Bull. Amer. Meteor. Soc.*, **102**, E681–E699, https://doi.org/10.1175/BAMS-D-19-0308.1.

Wang, N., J. Bao, J. Lee, F. Moeng, and C. Matsumoto, 2015: Wavelet compression technique for high-resolution global model data on an icosahedral grid. *J. Atmos. Oceanic Technol.*, **32**, 1650–1667, https://doi.org/10.1175/JTECH-D-14-00217.1.

Wang, W., X. Liu, J. Bi, and Y. Liu, 2022: A machine learning model to estimate ground-level ozone concentrations in California using TROPOMI data and high-resolution meteorology. *Environ. Int.*, **158**, 106917, https://doi.org/10.1016/j.envint.2021.106917.

Wang, Y. Q., 2014: MeteoInfo: GIS software for meteorological data visualization and analysis. *Meteor. Appl.*, **21**, 360–368, https://doi.org/10.1002/met.1345.

Wilkinson, M., and Coauthors, 2016: The FAIR guiding principles for scientific data management and stewardship. *Sci. Data*, **3**, 160018, https://doi.org/10.1038/sdata.2016.18.

Xu, M., G. Thompson, D. R. Adriaansen, and S. D. Landolt, 2019: On the value of time-lag-ensemble averaging to improve numerical model predictions of aircraft icing conditions. *Wea. Forecasting*, **34**, 507–519, https://doi.org/10.1175/WAF-D-18-0087.1.

Yao, X., and Coauthors, 2020: Enabling the big Earth observation data via cloud computing and DGGS: Opportunities and challenges. *Remote Sens.*, **12**, 62, https://doi.org/10.3390/rs12010062.