ENABLING IMMEDIATE ACCESS TO EARTH SCIENCE MODELS THROUGH CLOUD COMPUTING

Application to the GEOS-Chem Model

Jiawei Zhuang, Daniel J. Jacob, Judith Flo Gaya, Robert M. Yantosca, Elizabeth W. Lundgren, Melissa P. Sulprizio, and Sebastian D. Eastham

Cloud computing provides fast and easy access to the comprehensive GEOS-Chem atmospheric chemistry model and its large input datasets for the international user community.

Cloud computing involves on-demand access to a large remote pool of computing and data resources, typically through a commercial vendor.[1] It has considerable potential for Earth science modeling (Vance et al. 2016). Cloud computing addresses three common problems researchers face when performing complex computational tasks: computing, software, and data. Commercial cloud computing platforms like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform allow users to request computing resources on demand and only pay for the computing time they consume, without having to invest in local computing infrastructure (computing problem). Due to the use of virtual machines (VMs) on cloud platforms, it is very easy to replicate an existing software environment, so researchers can avoid configuring software from scratch, which can often be difficult and time-consuming (software problem). Large volumes of data can be quickly shared and processed in the cloud, saving researchers the time of downloading data to local machines and the cost of storing redundant copies of data (data problem). Yet, cloud computing has made little penetration in Earth science modeling so far because of several roadblocks. Here we show how these roadblocks can be removed, and we demonstrate practical user-oriented application with the GEOS-Chem atmospheric chemistry model, which is now fully functional and user accessible on the AWS cloud.

AFFILIATIONS: Zhuang, Jacob, Flo Gaya, Yantosca, Lundgren, and Sulprizio—John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts; Eastham—Laboratory for Aviation and the Environment, Massachusetts Institute of Technology, Cambridge, Massachusetts
CORRESPONDING AUTHOR: Jiawei Zhuang, jiaweizhuang@g.harvard.edu

The abstract for this article can be found in this issue, following the table of contents.
DOI:10.1175/BAMS-D-18-0243.1

In final form 1 May 2019

[1] See chapter 1 of Foster and Gannon (2017) for a scientist-friendly overview.

A number of Earth science models have been tested in the cloud environment, including the MITgcm (Evangelinos and Hill 2008), the Weather Research and Forecasting (WRF) Model (Withana et al. 2011; Molthan et al. 2015; Duran-Limon et al. 2016; Siuta et al. 2016), the Community Earth System Model (CESM; Chen et al. 2017), the NASA GISS ModelE (Li et al. 2017), the Regional Ocean Modeling System (ROMS; Jung et al. 2017), and the Model for Prediction Across Scales–Ocean (MPAS-O; Coffrin et al. 2019). Extensive studies have benchmarked the computational performance of cloud platforms against traditional clusters (Walker 2008; Hill and Humphrey 2009; Jackson et al. 2010; Gupta and Milojicic 2011; Yelick et al. 2011; Zhai et al. 2011; Roloff et al. 2012, 2017; Strazdins et al. 2012; Gupta et al. 2013; Mehrotra et al. 2016; Salaria et al. 2017). Cloud platforms are found to be efficient for small-to-medium-sized simulations with less than 100 CPU cores, but the typically slower internode communication on the cloud can affect the parallel efficiency of larger simulations (e.g., chapter 9.1 of Yelick et al. 2011). Cost comparisons between cloud platforms and traditional clusters show inconsistent results, either in favor of the cloud (Roloff et al. 2012; Huang et al. 2013; Oesterle et al. 2015; Thackston and Fortenberry 2015; Dodson et al. 2016) or local clusters (Carlyle et al. 2010; Freniere et al. 2016; Emeras et al. 2017; Chang et al. 2018), depending on assumptions regarding resource utilization, parallelization efficiency, storage requirement, and billing model. The cloud is particularly cost effective for occasional or intermittent workloads.

For complex model code, cloud platforms can considerably simplify the software configuration process. On traditional machines, researchers need to properly configure library dependencies like HDF5, NetCDF, and MPI, and this configuring process is becoming more and more difficult due to the growing use of complicated software frameworks like ESMF (Hill et al. 2004) and NUOPC (Carman et al. 2017). On cloud platforms with proper permissions, users can simply copy the exact software environment from an existing system (virtual machine). Once a model is built, configured, and made available on the cloud by the developer, it can be immediately shared with anyone who has access to the cloud. An example is the OpenFOAM Computational Fluid Dynamics software officially distributed through the AWS cloud (https://cfd.direct/cloud/).

Cloud computing also greatly enhances the accessibility of Earth science datasets. Earth observing systems and model simulations can produce terabytes (TBs) or petabytes (PBs) of data, and downloading these data to local machines is often impractical. Instead of "moving data to compute," the new paradigm should be "moving compute to data," that is, performing data analysis in the cloud computing environment where the data are already available (Yang et al. 2017). For example, NOAA's Next Generation Weather Radar (NEXRAD) product is shared through the AWS cloud (Ansari et al. 2018), and "data access that previously took 3+ years to complete now requires only a few days" (NAS 2018). NASA's Earth Observing System Data and Information System (EOSDIS) plans to move PBs of Earth observation data to the AWS cloud to enhance data accessibility (Lynnes et al. 2017). Other Earth science datasets such as NASA's Earth Exchange (NEX) data, NOAA's *GOES-16* data, and ESA's *Sentinel-2* data are publicly available on the AWS cloud (https://aws.amazon.com/earth/). The Google Earth Engine (Gorelick et al. 2017) and Climate Engine (Huntington et al. 2017) are other examples of cloud platforms that provide easy access to various Earth science data collections as well as the computing power to process the data.

Computing on the cloud further facilitates reproducibility in research (Howe 2012; de Oliveira et al. 2017). Scientific journals increasingly require that model source code and data be made available online (Irving 2016). However, due to complicated software dependencies of Earth science models, the configuration scripts for one system would usually require significant modifications to work on other systems, and sometimes the platform differences can lead to differences in simulation results (Hong et al. 2013; Li et al. 2016). It is difficult to reproduce a model simulation even if the source code is published online. Cloud platforms can solve this problem by guaranteeing a consistent system environment for different research groups, providing massive computing power to rerun a model, and sharing large volumes of input/output data.

Our guiding example in this article is the GEOS-Chem global 3D model of atmospheric chemistry (www.geos-chem.org), which we have made available for users on the AWS cloud. GEOS-Chem was originally described by Bey et al. (2001). It is at present used by over 150 registered research groups worldwide (GEOS-Chem 2018a) for a wide range of applications in air quality, global tropospheric–stratospheric chemistry, biogeochemical cycling of persistent pollutants, budgets of greenhouse gases, and radiative forcing of climate. For some recent applications see, for example, Christian et al. (2018), Jeong and Park (2018), Jing et al. (2018), Tian et al. (2018), Yu et al.

(2018), and Zhu et al. (2018). GEOS-Chem is driven by assimilated meteorological data from the Goddard Earth Observing System (GEOS) of the NASA Global Modeling and Assimilation Office (GMAO). It can operate in global or nested mode at resolutions as fine as 25 km in latitude–longitude or cubed-sphere grids (Eastham et al. 2018). The GEOS-Chem code (https://github.com/geoschem) is open access and undergoes continual development by its users, with updates to the standard model overseen by an international GEOS-Chem Steering Committee (GEOS-Chem 2018b). New versions are released and benchmarked every few months by the GEOS-Chem Support Team of scientific programmers based at Harvard University.

Our porting of GEOS-Chem to the cloud was motivated by the need to serve a diverse and growing base of model users, many with little access to high-performance computing (HPC) resources and software engineering support. This includes novice users needing easy access to the model for occasional computations, such as interpreting data from a field campaign, determining the atmospheric implications of laboratory measurements, or specifying boundary conditions for urban/regional models. Despite the intent for GEOS-Chem to be an easy-to-use facility, the model is becoming increasingly complicated because of more comprehensive scientific schemes, higher grid resolution, larger input datasets, and more extensive software infrastructure to support these advances. Data transfer is an increasing problem as the model resolution increases. GEOS-Chem users at present have access to 30 TB of model input data on FTP servers at Harvard University. With a typical bandwidth of 1 MB s^{-1}, it takes two weeks to download a 1-TB subset and a year to download the full 30 TB. This is a significant bottleneck for research progress.

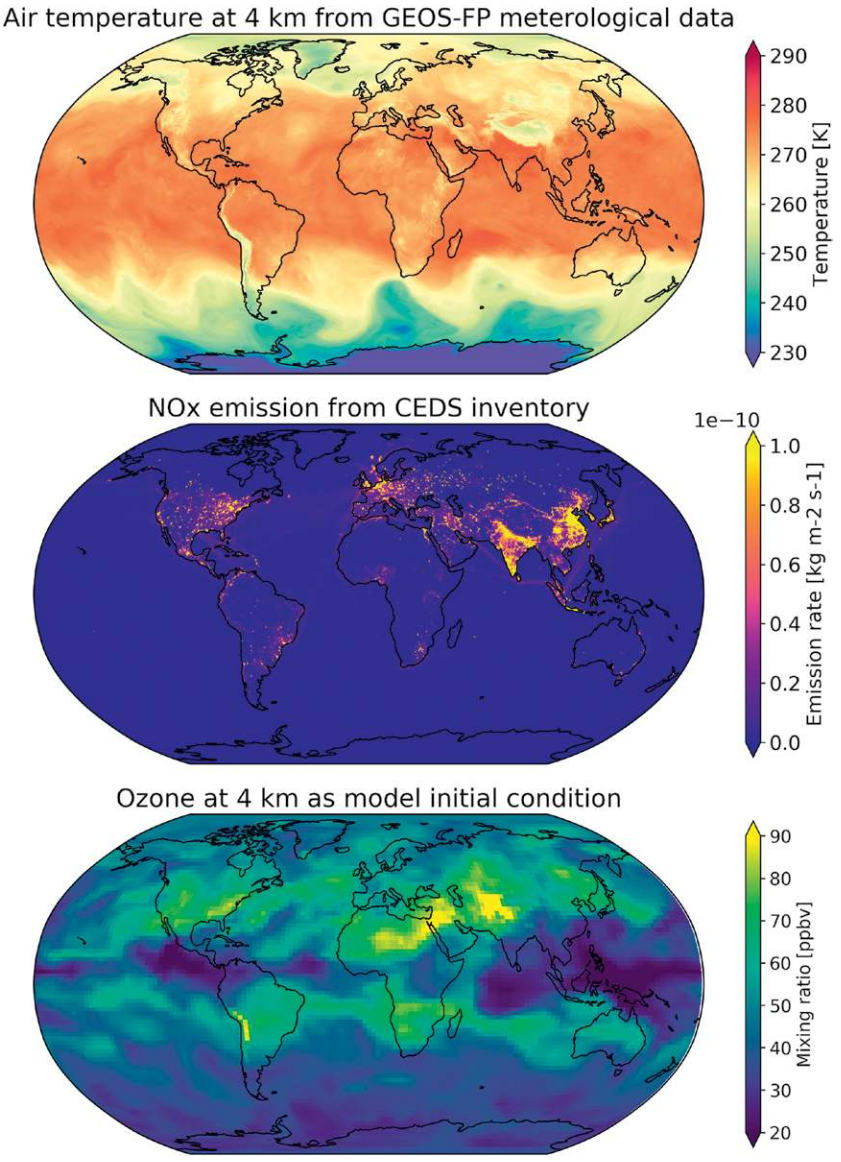

Fig. 1. Examples of GEOS-Chem input data hosted on the AWS cloud. (top) Global 25-km resolution air temperature data at 4-km altitude on 1 Jul 2016 (0000 UTC), from the GEOS-FP meteorological product. (middle) Global 50-km resolution anthropogenic nitrogen oxides (NO_x) emission data averaged over Jan 2014, from the Community Emissions Data System (CEDS; Hoesly et al. 2018). (bottom) Global 2° × 2.5° resolution ozone concentration data at 4-km altitude on 1 Jul 2016 (0000 UTC), from model initial condition files.

To solve this problem, we have made GEOS-Chem available through the AWS cloud, with the exact same code and software environment as the standard benchmarked version managed at Harvard University. All GEOS-Chem input data are now hosted on the AWS cloud under the AWS public

dataset program (https://registry.opendata.aws/geoschem-input-data/), and can be quickly accessed by users in the cloud with no additional charge. The data repository includes the GEOS-Forward Processing (GEOS-FP) meteorological product at global 0.25° × 0.3125° resolution (2012–present), the Modern-Era Retrospective Analysis for Research and Applications version 2 (MERRA-2) meteorological product at global 0.5° × 0.625° resolution (2000–present), and coarser-resolution and nested-domain versions. The repository also contains a large collection of emission inventories managed through the Harvard–NASA Emission Component (HEMCO; Keller et al. 2014) and other files needed for driving GEOS-Chem simulations such as the model initial conditions. Figure 1 shows examples of these datasets. With already-available datasets as well as a preconfigured software environment on the AWS cloud, a beginning user can start GEOS-Chem simulations immediately and with confidence that the results will replicate those of the latest standard benchmarked version. To facilitate cloud adoption by users, we provide a detailed user manual with step-by-step instructions for a complete research workflow (http://cloud.geos-chem.org/), assuming no prior knowledge of cloud computing. Because the software requirements and workflows tend to be similar between Earth science models, our work provides general guidance beyond GEOS-Chem for porting models to cloud computing platforms in a user-accessible way.

CLOUD COMPUTING FOR RESEARCH: REMOVING THE ROADBLOCKS. Four practical roadblocks have hindered scientists from exploiting cloud computing: licensing for commercial software, potential vendor lock-in, lack of science-oriented documentation and tooling, and concerns over performance and cost. Here we show how the first two issues have been effectively addressed by the availability of open-source software and HPC containers. We address the third issue by our own development of documentation targeted at Earth scientists. Performance and cost are discussed in a dedicated section.

Open-source software. The licensing of proprietary software has been a major roadblock to cloud adoption (section 3.2.2 of Netto et al. 2017). Commercial software programs such as Intel compilers and MATLAB are often preinstalled on HPC clusters at supercomputing centers and at institutions, but using the same software on cloud platforms requires researchers to bring their own licenses, whose cost can be prohibitive. Further, although an advantage of the cloud is the ability to share the entire software environment with users, it is harder to share a system containing proprietary software.[2]

The increasing availability and capability of open-source software is rapidly obviating the need for proprietary software in Earth science. In particular, GEOS-Chem has recently freed itself of the need for proprietary software by adopting Python libraries for data analysis, and by making code changes to ensure compatibility with the GNU FORTRAN compiler. Data analysis in GEOS-Chem had historically relied on GAMAP (http://acmg.seas.harvard.edu/gamap/), a data analysis package written in the proprietary Interactive Data Language (IDL) that cannot be easily used and distributed on the cloud. With the maturity of the open-source scientific Python stack in recent years (VanderPlas 2016), GEOS-Chem data analysis has migrated to Python. Existing Python libraries can easily replicate, and often surpass, the functionalities in MATLAB and IDL. Commonly used Python libraries for GEOS-Chem users include Jupyter notebooks (Shen 2014; Perkel 2018) for user interface, Xarray (Hoyer and Hamman 2017) for conveniently manipulating NetCDF files, Dask (Rocklin 2015) for parallel computation, Matplotlib (Hunter 2007) and Cartopy (Met Office 2016) for data visualization, and xESMF (Zhuang 2018) for transforming data between different grids. We provide a Python tutorial for the GEOS-Chem user community at https://github.com/geoschem/GEOSChem-python-tutorial; the tutorial code can be executed in a preconfigured Python environment on the cloud platform provided freely by the Binder project (https://mybinder.org). Most contents in the tutorial are general enough to be applied to other Earth data analysis problems.

Atmospheric models are typically written in FORTRAN, which can, in principle, accommodate different FORTRAN compilers. In practice, compilers have different syntax requirements. Earlier versions of GEOS-Chem were intended for the proprietary Intel compiler, but failed to compile with the open-source GNU compiler due to invalid syntaxes in legacy modules. With a recent refactoring of legacy code, GEOS-Chem now compiles with all major versions of the GNU FORTRAN compiler (GEOS-Chem 2018c). Although some models like WRF are found to be

[2] Strictly speaking, the Intel compiler license is only required for compiling source code, not for running precompiled applications. However, it is common for GEOS-Chem users to modify the model source code, for purposes like debugging, saving custom data fields, and implementing new schemes, so the ability to recompile source code is important.

significantly slower with GNU compilers than with Intel compilers (Langkamp and Böhner 2011; Siuta et al. 2016), for GEOS-Chem we find that switching to the GNU compiler decreases performance by only 5% (GEOS-Chem 2018d), in part due to a sustained effort to only bring hardware-independent and compiler-independent optimizations into the GEOS-Chem main code branch.

HPC containers. Being locked-in by a particular cloud vendor is a major concern for researchers (section 5 of Bottum et al. 2017), who may then be hostage to rising costs and unable to take advantage of cheap computing elsewhere. It is indeed highly desirable for researchers to be able to switch smoothly between different cloud platforms, supercomputing centers, and their own clusters, depending on the costs, the available funding, the location of data repositories, and their collaborators (Almes 2015). The container technology (Boettiger 2015) enables this by allowing immediate replication of the same software environment on different systems. Containers also ensure long-term reproducibility by freezing a legacy software environment and replicating it on new systems.

From a user's perspective, containers behave like VMs that can encapsulate software libraries, model code, and small data files into a single "image." VMs and containers both allow platform-independent deployment of software. While VMs often incur performance penalties due to running an additional guest operating system (OS) inside the original host OS, containers run on the native OS and can achieve near-native performance (Hale et al. 2017). Docker (www.docker.com) is the most widely used container and is readily available on the cloud, but it cannot be used on HPC clusters shared between many users due to security risks (Jacobsen and Canon 2015). To address Docker's limitations, HPC containers such as Shifter (Gerhardt et al. 2017), CharlieCloud (Priedhorsky et al. 2017), and Singularity (Kurtzer et al. 2017) have been recently developed to allow secure execution of Docker images on shared HPC clusters. It is now increasingly standard for large HPC clusters to be equipped with software containers. For example, Harvard's Odyssey cluster supports the Singularity container (Research Computing 2018), and NASA's Pleaides cluster supports the CharlieCloud container (NASA HECC 2018). All these different containers are compatible as they can all execute Docker images.

A GEOS-Chem user might want to perform initial pilot simulations on the AWS cloud, and then switch to their own clusters for more intensive simulations. We provide container images on Docker Hub (https://hub.docker.com) with preconfigured GEOS-Chem for users to download to their own clusters. The container provides exactly the same software environment as used on the cloud, so the model is guaranteed to compile and execute correctly when ported to the local cluster, and even achieve bit-wise reproducibility (Hacker et al. 2017).

Science-oriented documentation. Cloud-computing was initially developed for business information technology (IT) applications, and interest in the cloud for scientific computing occurred many years later (Fox 2011). According to the U.S. National Academy of Sciences (NAS 2018), "the majority of U.S. Earth science students and researchers do not have the training that they need to use cloud computing and big data." Standard cloud platform documentations (Amazon 2018a) are largely written for system administrators and web developers, and can be difficult for scientists to understand. Some research-oriented cloud materials (Amazon 2018b; Microsoft 2018a) and textbooks (Foster and Gannon 2017) have eased the learning curve but are not specific enough to guide Earth science applications. The Serverless Computing paradigm (Jonas et al. 2019) has the potential to greatly simplify the use of cloud computing and reduce the system administration burden on users, but its use for scientific computing is still at infancy.

To make cloud computing accessible to scientists, several high-level, black-box services have been proposed to hide the technical details of cloud platforms (Tsaftaris 2014; Hossain et al. 2017; Li et al. 2017; section 2.3 of Netto et al. 2017). However, they tend to make strict assumptions on the workflow and lack customizability and generalizability. We take the opposite tack—to teach low-level AWS cloud concepts to users and make them manage their own cloud infrastructures, including servers, storage, and network. Users have full flexibility in how they use GEOS-Chem, including changes to the FORTRAN source code if they wish. In cloud computing terminology, we stick to the Infrastructure as a Service (IaaS) framework, not the higher-level Software as a Service (SaaS) framework.

Our research-oriented tutorial and documentation (http://cloud.geos-chem.org/) provide the necessary training for GEOS-Chem users, with practical focus on model simulation and data management. The documentation includes "beginner tutorials" for beginning and occasional users, "advanced tutorials" for heavy users, and a "developer guide" for model developers to install software

libraries and configure environment from scratch. By following the documentation, GEOS-Chem users can successfully finish a demo project in half an hour for their first experience on the AWS cloud, and within minutes for subsequent uses. This is detailed in the next section.

Exposing low-level infrastructures makes our work highly generalizable to other Earth science applications and cloud platforms. First, users have the freedom to install and run any model by using our GEOS-Chem workflow as a reference. Second, once users get familiar with AWS concepts, it becomes relatively easy for them to learn other cloud platforms like Microsoft Azure and Google cloud, because these different platforms have similar services and technical concepts (Google 2018a; Microsoft 2018b). Third, the same set of cloud computing skills can be applied to a wide range of problems beyond model simulations, such as analyzing large public Earth science data or using GPUs on the cloud to accelerate machine learning workloads (appendix B of Chollet 2017).

RESEARCH WORKFLOW USING BASIC AWS FUNCTIONALITIES. Here we describe the workflow for GEOS-Chem users in two scenarios.

1) *A demo project for initiation to the cloud.* The users run our preconfigured "demo simulation" for a short period. The demo is configured with GEOS-Chem's standard troposphere–stratosphere oxidant–aerosol chemical mechanism and a global horizontal resolution of 4° × 5°. This same configuration is used for the official GEOS-Chem benchmark (GEOS-Chem 2018e). The configuration runs fast due to the coarse spatial resolution and is a good starting point for new users. The users perform quick analysis and visualization of the model output data, and then exit the system discarding all data files.
2) *An actual research project for scientific analysis.* The users set up their desired custom configuration of the model, conduct their simulation, archive their output, and preserve the model configuration for future runs. For illustrative and cost evaluation purposes, we assume here a 1-yr global simulation at 2° × 2.5° resolution using the standard troposphere–stratosphere oxidant–aerosol chemical mechanism. For output, the simulation stores 3D daily averaged fields for 168 transported chemical species, resulting in 150 GB of total data. This is typical of a research project using GEOS-Chem. The same workflow applies to any other configuration of the model.

The workflow uses the most basic AWS functionalities to keep the learning curve to a minimum. AWS offers over a hundred services (Amazon 2018c), leading to a complicated web portal that can be daunting for new users. However, Earth science modeling tasks can be achieved with only two core services: Elastic Compute Cloud (EC2) for computation and Simple Storage Service (S3) for data storage. Many other services target various IT/business applications and are not necessary for most scientific users. The steps for a complete modeling research workflow are illustrated in Fig. 2 and are explained in what follows.

Step 1: Launch virtual server. Under the IaaS framework, users request their own servers ("EC2 instances" in AWS terminology) with customizable hardware capacity and software environment. Requests are done through the AWS console in the user's web browser. The software environment is determined by a virtual machine image (the Amazon Machine Image, or AMI), which defines the operating system, preinstalled software libraries, and data files that a newly launched EC2 instance will contain. Here, users can select a publicly available AMI with preconfigured GEOS-Chem environment. The hardware capacity is defined by the choice of EC2 instance type (Amazon 2018d) with different capacities in CPUs, memory, disk storage, and network bandwidth. Numerical models run most efficiently on "compute-optimized" types, which prioritize CPU performance over other aspects (Montes et al. 2017). Launching a new EC2 instance only takes seconds. Repeated launches of EC2 instances can be automated by the AWS Command Line Interface (AWSCLI; https://aws.amazon.com/cli/) to avoid having to browse through the web console.

Step 2: Log into server and perform computation. Once the EC2 instance is created, it can be used as a normal server, that is, via the Secure Shell (SSH) in the command line terminal. Since most Earth scientists have experience with local servers and the command line, using EC2 instances is straightforward for them; if not, they can learn the command line basics easily from online materials like Software Carpentry (http://swcarpentry.github.io/shell-novice/). There are no cloud-computing-specific skills required at this stage.

Following standard practice for Earth science models, a GEOS-Chem simulation is controlled by a "run directory" containing run-time configurations and the model executable. A preconfigured run directory with a precompiled executable and a sample of meteorological and emission input data are provided by our GEOS-Chem AMI for the demo project, so

users can execute a GEOS-Chem simulation immediately after log-in, without having to set up the run directory and compile model code on their own. This short demo simulation finishes in minutes.

For an actual research project, users should recompile the model (using the open-source GNU compiler), with the desired model version, grid resolution, and chemical mechanism. Recompilation takes about 2 min. Because the software libraries and environment variables are already properly configured, the model is guaranteed to compile without error. Additional operations such as modifying model source code or installing new software libraries can be done as usual, just like on local servers.

GEOS-Chem users also need to access additional meteorological and emission input data for their desired simulation configurations, and they can do so from the public GEOS-Chem input data repository residing in the AWS Simple Storage Service (S3) (named "s3://gcgrid"). The data transfer from S3 to EC2 is simply done by AWSCLI command "aws s3 cp," analogous to the "cp" command for copying data on local file systems. For the actual research project scenario described here, the user should retrieve 1 year of global 2° × 2.5° meteorological input data with size of 112 GB. With a typical ~250 MB s^{-1} network bandwidth between S3 and EC2, the retrieval will finish in 8 min.

Users of shared HPC clusters are accustomed to using job schedulers to manage long model simulations. However, an EC2 instance completely belongs to a single user, so a scheduler for queuing jobs is not necessary. Instead, users just execute the program interactively. A simple way to keep the program running after logging out of the server is to execute the model inside terminal multiplexers such as GNU Screen (www.gnu.org/software/screen/) or tmux (https://github.com/tmux/tmux), which are standard tools for managing long-running programs on the cloud (Shaikh 2018).

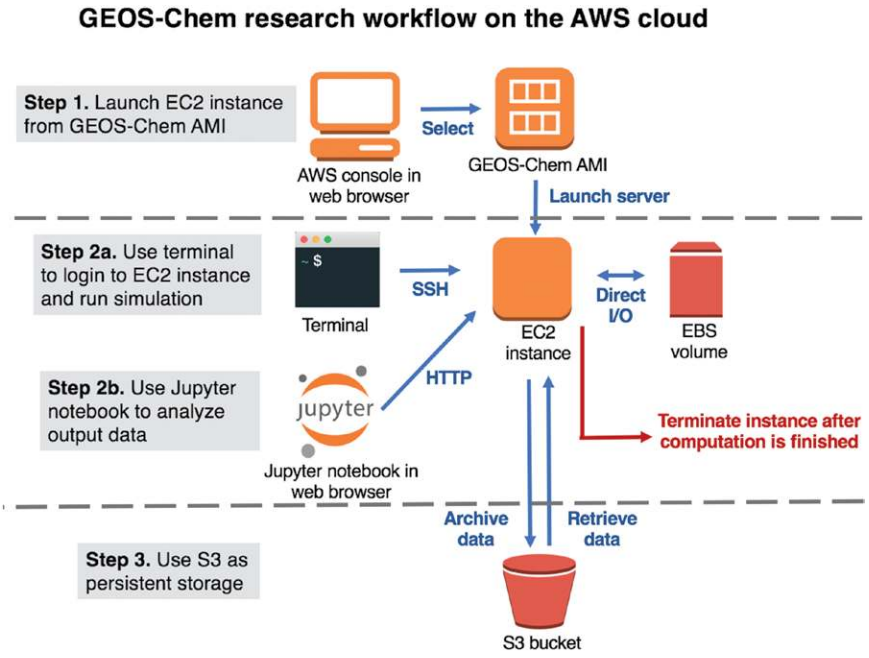

Fig. 2. A complete research workflow for using the GEOS-Chem model on the AWS cloud. From the AWS console accessed through web browsers, users select the GEOS-Chem AMI, which includes the fully configured model, and launch a Linux server (EC2 instance) from that AMI. Users then log into that EC2 instance through their local terminal, customize the model configuration as desired, and execute the model simulation. Model output can be analyzed on the cloud using Python in Jupyter notebooks, or can be downloaded to users' local computers. Local I/O is managed through the EBS volume on the EC2 instance. The complete meteorological and emission data needed to drive GEOS-Chem can be retrieved from a public bucket (named "s3://gcgrid") residing permanently in the S3 of the AWS cloud. Users can also create their own S3 buckets to archive model output data and customized run directories. After finishing the computation, users terminate the EC2 instance; data in the EBS volume will then be deleted, but data in the S3 buckets will persist.

GEOS-Chem simulations generate potentially large amounts of output data in NetCDF format (GEOS-Chem 2018f). Analysis of output data can be done within the cloud using Python in web-based Jupyter notebooks.[3] Jupyter always displays the user interface, code, and graphics in the web browser, no matter whether the program is running on the local computer or a remote cloud platform. Thus, users have a convenient data analysis environment on the cloud, as if they

[3] The Jupyter program runs on an EC2 instance and thus involves EC2 charges. It is often sufficient to use a single-core, small instance (e.g., "m5.large") for data analysis tasks with Jupyter, and the cost of such a small instance is only a few cents per hour.

are using Jupyter locally. Alternatively, the data can be downloaded to local computers for analysis, although this involves data transfer time and a data egress fee (see next section).

For a demo project, the user can simply terminate the EC2 instance after the model simulation and output data analysis are finished. The output data are deleted upon termination. For an actual research project, the output data will generally need to be archived before EC2 termination, as described next.

Step 3: Working with persistent storage. The lack of persistent disk storage is the major difference between cloud platforms and local computers. The Elastic Block Store (EBS) volume is the temporary disk that backs up an EC2 instance. However, when the EC2 instance is terminated, its EBS volume containing the user's data will typically be deleted. Instead of leaving files on EBS, important files such as output data and customized run directories should be transferred to the S3 persistent storage service. S3 is independent of EC2 and guarantees the persistence of data.

S3 offers many other advantages over EBS besides data persistence. While an EBS volume can only be attached to a single EC2 instance, data in S3 can be accessed simultaneously by any number of EC2 instances, as well as by computers outside of the cloud. Files in an EBS volume can only be seen from an EC2 instance, but files in S3 can be viewed directly in the AWS web portal. While an EBS volume has a limited storage capacity just like traditional disks (16 TB; Amazon 2018e), there is effectively no size limit on S3. Furthermore, S3 can be integrated with advanced AWS services (e.g., Amazon Elastic MapReduce) and community big-data platforms (e.g., Pangeo, https://pangeo.io) for large-scale data processing.

Despite the advantages of S3, new users might still feel uncomfortable with managing two types of storage (S3 and EBS) at the same time. This can be simplified by mounting S3 via Filesystem in Userspace (FUSE, https://github.com/s3fs-fuse/s3fs-fuse), so that data in S3 appear like normal files on the server's disk. The data transfer between EC2 and S3 happens automatically when a file is accessed, without requiring the user to explicitly transfer the data. However, this approach is currently very inefficient for NetCDF data, due to incompatibilities between the HDF5 library and the object-based storage model that S3 uses (Rocklin 2018). This problem might be improved by developing a more cloud-friendly backend, such as Zarr (https://github.com/zarr-developers/zarr), for the NetCDF library (Unidata 2019). For now, we recommend explicitly transferring data between EC2 and S3.

Once important files are transferred to S3 (or downloaded to local storage), users can safely terminate the EC2 instance. Modification to the software environment can also be saved, by snapshotting the EC2 instance into an AMI and then using the new AMI to launch subsequent EC2 instances.

Advantages and limitations of the workflow. The workflow presented here depends on very few AWS functionalities, has a light learning curve, and is flexible enough to match different usage patterns. For example, it is common to repeatedly analyze output data after a model simulation. Whenever users need to access the data in S3, they can simply launch a new EC2 instance, pull data from S3 to perform analysis, and terminate the EC2 instance after the analysis is done. Simultaneous model runs can be done in parallel by launching multiple EC2 instances, with input data replicated in each instance's EBS volume.

We tested the demo project on 15 graduate students and postdoctoral fellows of the Atmospheric Chemistry Modeling Group at Harvard University and recorded the time they spent on each step (Fig. 3). All were GEOS-Chem users, but none had any prior experience with

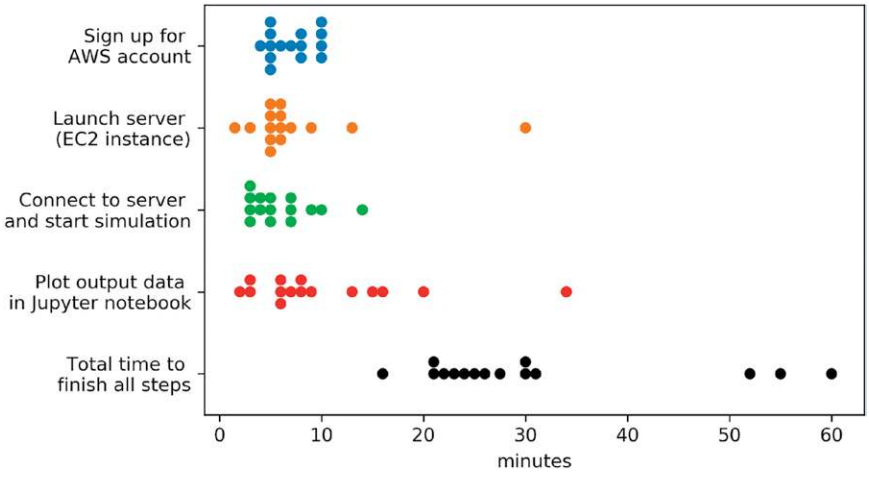

Fig. 3. Wall-clock time required for 15 GEOS-Chem users with no prior cloud experience to complete a demo GEOS-Chem simulation on the AWS cloud and visualize the output. The actual simulation runtime is not included and is a few minutes for this demo case.

cloud computing. Most finished the demo in less than 30 min, and all in less than 60 min. The time represents the initial learning curve for first-time AWS users; an experienced AWS user could finish the demo in 1 min. An actual research project requires additional steps such as configuring model details and working with S3, but the workflow is not fundamentally different. Overall, it is very fast to get the model running on the cloud, in sharp contrast to the often slow and complicated procedure of setting up the model on a local server.

The workflow as described above has several limitations for heavy users. Although GEOS-Chem is able to parallelize across multiple compute nodes using MPI (Eastham et al. 2018), the workflow presented here only allows the model to run on a single instance (with up to 64 physical cores on the "x1.32large" instance type), equivalent to a single compute node on an HPC cluster. Also, users need to manually manage the start and termination of each EC2 instance, which can be an administrative burden if there are a large number of simultaneous simulations. These issues may be addressed by creating an HPC cluster environment on the cloud, using software tools such as AWS ParallelCluster (https://docs.aws.amazon.com/parallelcluster/, supersedes the previous CfnCluster), StarCluster (http://star.mit.edu/cluster, unmaintained), ElastiCluster (http://elasticluster.readthedocs.io), AlcesFlight (http://docs.alces-flight.com), and EnginFrame (www.nice-software.com/products/enginframe). A cluster consists of a "master node" (typically a small EC2 instance) and multiple "compute nodes" (typically high-performance EC2 instances). Such a cluster allows inter-node MPI communication, provides a shared disk (EBS volume) for all nodes via the Network File System (NFS), and often supports an "auto-scaling" capability (Amazon 2018f) that automatically launches or terminates compute nodes according to the number of pending jobs. However, we find that cluster tools have a much steeper learning curve for scientists and are generally overkill for moderate computing workloads. While managing individual EC2 instances is straightforward, configuring and customizing a cluster involve heavier system administration tasks. To avoid those complications, we only conduct single-node simulations in this work. Readers who are interested in running multinode MPI applications can refer to an online tutorial provided by the lead author (Zhuang 2019). That involves more complicated setup steps and is intended for more experienced AWS users.

Although cloud computing technologies are evolving rapidly and many currently popular services might become obsolete in the future, we expect the concepts, techniques, and workflows presented in this section to stay relevant in the long term. EC2 and S3 have always been the most essential AWS services, and their basic usages have not changed since their initial release in 2006 (Barr 2006a,b), despite various

Table 1. Hardware specification and cost. AWS costs are from https://aws.amazon.com/ec2/pricing/. NASA HECC costs are from www.hec.nasa.gov/user/policies/sbus.html. Costs are in USD as of December 2018. The price can vary between countries and regions. Shown here are for the U.S. East (northern Virginia) region. Same for Tables 2 and 3.

Instance/node type[a]	Processor information[b]	Hourly cost[c]	
		On-demand	Spot[d]
AWS			
EC2 c4.8xlarge	Intel Xeon CPU E5–2666v3, 2.9 GHz, 32 vCPUs	$1.59	$0.57
EC2 c5.9xlarge	Intel Xeon Platinum 8124M, 3.0 GHz, 32 vCPUs	$1.53	$0.58
EC2 c4.4xlarge	Intel Xeon CPU E5–2666v3, 2.9 GHz, 16 vCPUs	$0.80	$0.25
EC2 c5.4xlarge	Intel Xeon Platinum 8124M, 3.0 GHz, 16 vCPUs	$0.68	$0.27
NASA HECC			
Pleiades Sandy Bridge	Intel Xeon E5–2680v2, 2.8 GHz, 16 CPU cores	$0.29	—
Pleiades Haswell	Intel Xeon E5–2680v3, 2.5 GHz, 24 CPU cores	$0.53	—

[a] The naming of an EC2 instance follows "family, generation, size." For example, "c4" refers to the "compute-optimized" family, generation four; "8xlarge" indicates the instance size, which has twice as many cores and memory as "4xlarge."

[b] EC2 instances are virtual machines and the processors are described by "virtual CPUs" (vCPUs). A vCPU is a hyperthread and corresponds to half of a physical core (Amazon 2018j).

[c] EC2 actually uses per-second billing, so that short computations are very cheap (Amazon 2017b).

[d] The spot price can fluctuate with time. We show here the price when the simulations were conducted. Price fluctuation is typically within 20% in a month.

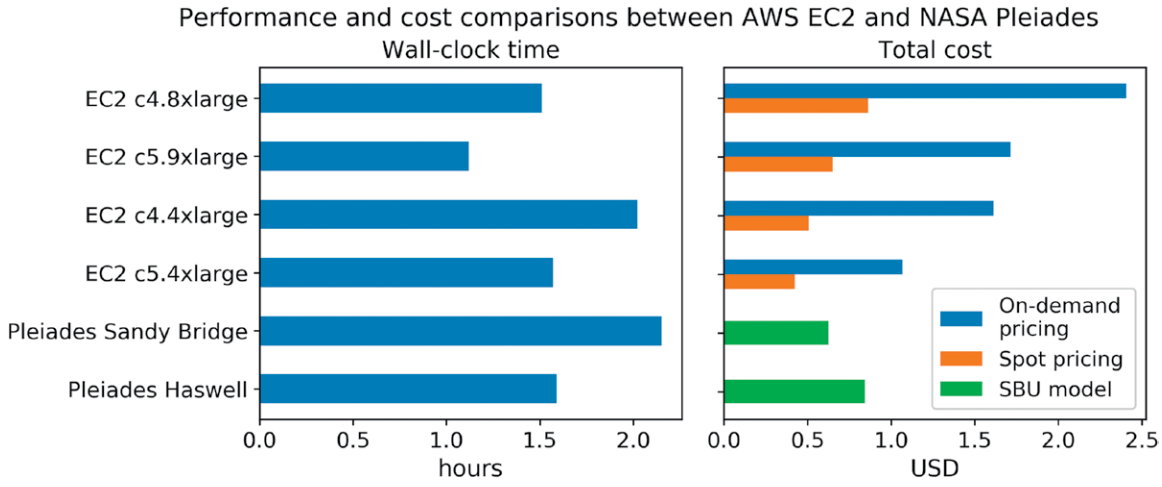

Fig. 4. Performance and cost comparisons between the AWS EC2 instances and the NASA Pleiades supercomputing cluster for a 7-day GEOS-Chem simulation with tropospheric and stratospheric chemistry at global 4° × 5° resolution, running on a single compute node. See Table 1 for description of the different hardware types. The SBU model is used by NASA to estimate the true cost of Pleiades. Cost is in USD. The AWS cloud and container environments used GNU FORTRAN compiler 7.3.0; the NASA Pleiades native environment used GNU FORTRAN compiler 6.2.0 as it is the latest available version.

performance improvements and new pricing options. Such longevity may not be true for other tools and techniques: for example, StarCluster was once a popular tool for managing HPC clusters on AWS, but has now become obsolete, superseded by modern HPC frameworks like AWS ParallelCluster. Further, users of higher-level AWS services (e.g., distributed clusters, databases, container services, etc.) will still benefit from the lower-level EC2 and S3 knowledge, because EC2 and S3 often serve as a major dependency for other higher-level services.

PERFORMANCE AND COST. *Computational performance and cost compared to local clusters.* EC2 instances incur hourly cost to users, with different pricing models. Commonly used are the standard "on-demand" pricing and the discounted "spot" pricing. Spot instances are cheaper than on-demand instances, usually by 60%–70%, with the caveat that they may be reclaimed by AWS to serve the demand from standard EC2 users. Although such spot interruption can cause model simulations to crash, a newly added "Spot Hibernation" option (Amazon 2017a) allows the instance to recover the previous state so that previous simulations can continue when capacity becomes available again. A recent update to the spot pricing model further reduces the chance of interruption, so an instance is rarely reclaimed and can generally keep uninterrupted for a month (Pary 2018). We recommend using spot pricing for computationally expensive tasks.

It is challenging to compare the pay-as-you-go pricing model on the cloud with the cost of local HPC clusters that vary in billing model (Abbo 2015). To simplify such estimation, we use the NASA Pleaides cluster that provides a simple, convenient billing model called Standard Billing Unit (SBU). NASA's High-End Computing Capability (HECC) Project uses this billing model to evaluate the cost of AWS against the Pleaides cluster (Chang et al. 2018). Jobs on Pleaides are charged by CPU hours, with the cost rate "calculated as the total annual HECC costs divided by the total number of [CPU hours used]" and thus is able to represent "the costs of operating the HECC facility including hardware and software costs, maintenance, support staff, facility maintenance, and electrical power costs." Pleiades costs are not borne directly by the users in that allocation of SBUs is through NASA research grants, but the costs are still borne by NASA. We consider several types of AWS EC2 instances and compute nodes on the Pleiades cluster with comparable performance, as summarized in Table 1.

For performance and cost comparisons we used GEOS-Chem version 12.1.1 to conduct a 7-day simulation with tropospheric and stratospheric chemistry (Eastham et al. 2014) at global 4° × 5° resolution, using OpenMP parallelization. We tested the model performance both on the native machine and inside the software container, on both the AWS cloud (using the Singularity container) and Pleiades (using the CharlieCloud container). The performance difference introduced by running the model inside the container is less than 1%, so there is no significant penalty to using containers.

The timing results and total costs are shown in Fig. 4. The newer "c5" generation has better performance and lower cost than the older "c4" generation. The smaller instance "c5.4xlarge" is more cost efficient than the bigger "c5.9xlarge" instance, due to the sublinear scaling of GEOS-Chem OpenMP parallelization and a fixed input/output (I/O) time. With spot pricing, the cost of EC2 is close to Pleiades. It is important to point out that Pleaides has a very high utilization rate of over 80% (Chang et al. 2018); for other local clusters with lower utilization rates, the pay-as-you-go pricing on the cloud becomes even more attractive. From a user standpoint, any comparative cost decision is complicated by the varying levels of research subsidy and grant-based allocation for time on local or supercomputing clusters. A better decision basis is the cost of an actual research project on the cloud, and we discuss this below.

Table 2. Description and cost of major AWS services (from https://aws.amazon.com/pricing/services/ as of December 2018).

Service	Description and purpose	Cost
EC2	Server used for computing on AWS. Users request an EC2 instance with a chosen number of vCPUs for their computing need.	~$0.05 per vCPU hour, depending on instance type. 60%–70% cheaper with spot pricing. See Table 1 for examples.
EBS	Disk storage for temporary data. It hosts input and output data during model simulations.	$0.1 per GB per month, for the standard solid-state drive (SSD). Cheaper options with different I/O characteristics are available.
S3	Major storage service. User transfer data from EC2 to S3 for persistent storage, and later retrieve data from S3 to EC2 for continued work.	$0.023 per GB per month. Cheaper options for infrequent access patterns are available.
Data egress	Downloading data to local machines.	$0.09 per GB

Total cost of an example project. Here we calculate the total cost of the actual research simulation example in the previous section (1-yr 2° × 2.5° global GEOS-Chem simulation of tropospheric and stratospheric chemistry with daily chemical fields for 168 species archived, representing 150 GB of output data). Besides the major charge with EC2, minor charges include the storage cost with EBS and S3. Users also incur a "data egress fee" if they download data to their local clusters. While AWS and the other major cloud vendors usually do not charge for transferring data into the cloud or between cloud services, they charge for transferring data out of the cloud. AWS allows academic users to waive the data egress fee, up to 15% of the total charge to the user (Amazon 2016).

Table 2 summarizes the major AWS services used in the GEOS-Chem simulation and their unit costs. The same services would be used for any other Earth science model. Table 3 summarizes the total cost of our example GEOS-Chem research simulation conducted on the most cost-efficient "c5.4xlarge" EC2 instance type. The compute cost for the 1-yr simulation with EC2 is $224 with on-demand pricing, and can be reduced to $90 with spot pricing (which we recommend). The temporary storage cost with EBS ($14) is relatively small. The long-term storage cost with S3 or the data egress fee depends on users' needs. One-year daily concentration fields for 168 chemical species on the global 3D domain represent 150 GB of data, only contributing a small amount to the total cost ($3.50 per month). However, if the user needs to produce TBs of data (e.g., hourly fields for a year), the storage cost can become important. The cost of a long-term archive can be greatly reduced by using Amazon S3 Glacier (https://aws.amazon.com/glacier/), a cheaper version of S3 with 80% price reduction but longer data retrieval time. Glacier can be a good option for long-term archiving of data after the research project has been completed.

Table 3. Cost of a 1-yr GEOS-Chem global simulation on the AWS cloud. Global simulation of tropospheric–stratospheric chemistry conducted at 2° × 2.5° horizontal resolution with 72 vertical levels, storing 3D daily output concentration fields for 168 species.

Service	Amount of resources needed	Total cost (USD)
EC2	330 h on c5.4xlarge	$224 with on-demand pricing, $90 with spot pricing
EBS	300 GB disk to host input and output data files during simulation	$14 with standard SSD
S3 or download	150 GB output data files	$3.50 per month on S3, or $13.50 to download

In summary, the total cost of a 1-yr global GEOS-Chem simulation with 2° × 2.5° resolution and full chemistry on the AWS cloud is about $120, assuming EC2 spot pricing and output data downloading. A 1-yr 4° × 5° simulation costs 4 times less ($30). The cost

is mostly from the computing itself. Downloading or archiving output data are generally cheap by comparison. Transferring GEOS-Chem input data from S3 to EBS is free and fast, as these data are already resident on the cloud.

The cost of computing on the cloud should make it highly attractive to occasional users, particularly because of the time savings in model configuration and data transfer, as well as the assurance that the simulation replicates the current standard benchmarked version of the model. AWS currently offers a $100 credit per year to university students (Amazon 2018g) to encourage experimentation on the cloud. Heavy users may still prefer local or supercomputing clusters, particularly if grant-funded. The cloud will become more attractive with a funding model comparable to the research subsidies and grant allocations for local clusters. Despite special funding programs such as NSF BIGDATA (NSF 2018a), E-CAS (NSF 2018b), AWS Cloud Credits for Research (Amazon 2018h), Azure AI for Earth (Microsoft 2018c), and Google Cloud Platform research credits (Google 2018b) that directly give credits on commercial clouds, the funding is still far lagging that of traditional supercomputing centers (section 12 of Bottum et al. 2017). Chapter 6.3.3 of NAS (2016) suggests several ways for NSF to better fund commercial cloud resources.

Heavy users running GEOS-Chem on local or supercomputing clusters can still benefit from the cloud for downloading model input data, because AWS S3 has a very high outgoing bandwidth which can often fully utilize the ingoing bandwidth of a local cluster. We achieve a data transfer rate of 100 MB s^{-1} from S3 to the Pleiades cluster, an order of magnitude faster than from the FTP server at Harvard. Transferring 1 year of global 2° × 2.5° meteorological input data (112 GB) for input to GEOS-Chem finishes in 20 min, which would have taken 3 h from the Harvard FTP server. The egress fee is only $10. The cloud, together with software containers, can thus accelerate the deployment of GEOS-Chem on local clusters as well.

CONCLUSIONS AND FUTURE WORK.

Cloud computing is becoming increasingly attractive for Earth science modeling. It provides easy access to complex models and large datasets and allows straightforward and error-free computing. We have described how the GEOS-Chem global model of atmospheric chemistry is now fully accessible for users on the AWS cloud. The cloud gives users immediate access to the computational resources, software environment, and large input data needed to perform GEOS-Chem simulations for any year and for any grid resolution supported by the standard model. Single-node GEOS-Chem simulations on the AWS cloud compare favorably in performance and true cost with local clusters. Scientists can learn the single-node workflow on the cloud very quickly by following our project documentation and tutorial. Our specific application is for GEOS-Chem, but the general principles can be adapted to other Earth science models, which tend to follow the same structure and requirements.

We find the cloud to be particularly attractive for beginning or occasional users, who otherwise may need to spend significant personnel time configuring a local computing environment. Heavy users with their own local clusters should still find the cloud useful for getting the latest model and data updates, benchmarking their local simulations against the standard model resident on the cloud, carrying out collaborative research in a reproducible environment, and temporarily expanding their computing capacity.

The cloud is also a promising vehicle for massively parallel simulations emulating local HPC clusters, but several issues need to be addressed. A proper research funding model becomes particularly important for compute-intensive simulations, as the costs can be significant. Although it is widely perceived that cloud platforms are not efficient for large-scale MPI programs due to slow communication across nodes, the network performance of cloud platforms has been improving rapidly over the past several years. The Microsoft Azure cloud now offers InfiniBand internode connection and achieves similar scalability as traditional supercomputers (Mohammadi and Bazhirov 2017). Recent updates on the AWS cloud, including a new instance type with 100 Gb s^{-1} bandwidth (which exceeds the typical ~50 Gb s^{-1} bandwidth on local HPC clusters) (Barr 2018), a low-latency network interface to improve the scaling of MPI programs (Barr 2019), and a parallel file system service for efficient I/O (Amazon 2018i), have greatly increased the potential of cloud computing for HPC applications. There remain other technical challenges. Although many HPC cluster management tools have been developed for the cloud, the learning curve of these tools is steeper than for the single-node applications described here. Also, although containers are highly portable in a single-node environment, they become less portable when internode MPI communication is needed. Docker is found to have significant overhead for multinode MPI applications (Younge et al. 2017; Zhang et al. 2017); Singularity achieves better performance by directly using the MPI installed on the host machine, but requires compatibility between

the MPI libraries inside and outside of the container. These issues will be addressed in future work.

CODE AVAILABILITY. The source code for our project documentation, the scripts for building AWS images, the Python notebooks for generating all plots in this article, are available at our project repository https://github.com/geoschem/geos-chem-cloud. Scripts for building GEOS-Chem Docker images are at https://github.com/geoschem/geos-chem-docker. The survey results producing Fig. 3 are available at https://github.com/geoschem/geos-chem-cloud/issues/15.

ACKNOWLEDGMENTS. This work was funded by the NASA Earth Science Division (Grant NASA-NNX17AI67G). Jiawei Zhuang was supported by the NASA Earth and Space Science Fellowship. The authors thank Kevin Jorissen, Joe Flasher, and Jed Sundwall at Amazon Web Services for supporting this work.

REFERENCES

Abbo, T. and Coauthors, 2015: TCO for cloud services: A framework. ECAR, 24 April, https://library.educause.edu/resources/2015/4/tco-for-cloud-services-a-framework.

Almes, G., and Coauthors, 2015: Research computing in the cloud: Functional considerations for research. ECAR, 14 July, https://library.educause.edu/resources/2015/7/research-computing-in-the-cloud-functional-considerations-for-research.

Amazon, 2016: AWS offers data egress discount to researchers. AWS Government, Education, & Nonprofits Blog, 1 March, accessed 20 December 20 2018, https://aws.amazon.com/blogs/publicsector/aws-offers-data-egress-discount-to-researchers/.

——, 2017a: Amazon EC2 spot lets you pause and resume your workloads. Amazon Web Services, 28 November, accessed 20 December 2018, https://aws.amazon.com/about-aws/whats-new/2017/11/amazon-ec2-spot-lets-you-pause-and-resume-your-workloads/.

——, 2017b: Announcing Amazon EC2 per second billing. Amazon Web Services, 2 October, accessed 20 December 2018, https://aws.amazon.com/about-aws/whats-new/2017/10/announcing-amazon-ec2-per-second-billing/.

——, 2018a: AWS official documentation. Amazon Web Services, accessed 20 December 2018, https://docs.aws.amazon.com/.

——, 2018b: AWS Research Cloud Program: Researcher's Handbook. Amazon Web Services, accessed 20 December 2018, https://aws.amazon.com/government-education/research-and-technical-computing/research-cloud-program/.

——, 2018c: AWS Cloud Products. Amazon Web Services, accessed 20 December 2018, https://aws.amazon.com/products/.

——, 2018d: Amazon EC2 Instance Types. Amazon Web Services, accessed 20 December 2018, https://aws.amazon.com/ec2/instance-types/.

——, 2018e: Amazon EBS features. Amazon Web Services, accessed 20 December 2018, https://aws.amazon.com/ebs/features/.

——, 2018f: AWS Auto Scaling. Amazon Web Services, accessed 20 December 2018, https://aws.amazon.com/autoscaling/.

——, 2018g: AWS Educate Program. Amazon Web Services, accessed 20 December 2018, https://aws.amazon.com/education/awseducate/.

——, 2018h: AWS Cloud Credits for Research. Amazon Web Services, accessed 20 December 2018, https://aws.amazon.com/research-credits/.

——, 2018i: Amazon FSx for Lustre. Amazon Web Services, accessed 20 December 2018, https://aws.amazon.com/fsx/lustre/.

——, 2018j: Physical Cores by Amazon EC2 and RDS DB Instance Type. Amazon Web Services, accessed 20 December 2018, https://aws.amazon.com/ec2/physicalcores/.

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.

Barr, J., 2006a: Amazon EC2 Beta. AWS News Blog, 25 August, accessed 30 April 2019, https://aws.amazon.com/blogs/aws/amazon_ec2_beta/.

——, 2006b: Amazon S3. AWS News Blog, 14 March, accessed 30 April 2019, https://aws.amazon.com/blogs/aws/amazon_s3/.

——, 2018: New C5n instances with 100 Gbps networking. AWS News Blog, 26 November, accessed 20 December 2018, https://aws.amazon.com/blogs/aws/new-c5n-instances-with-100-gbps-networking/.

——, 2019: Elastic Fabric Adapter (EFA) for tightly-coupled HPC workloads. AWS News Blog, 29 April, accessed 30 April 2019, https://aws.amazon.com/blogs/aws/now-available-elastic-fabric-adapter-efa-for-tightly-coupled-hpc-workloads/.

Bey, I., and Coauthors, 2001: Global modeling of tropospheric chemistry with assimilated meteorology: Model description and evaluation. *J. Geophys. Res.*, **106**, 23 073–23 095, https://doi.org/10.1029/2001JD000807.

Boettiger, C., 2015: An introduction to Docker for reproducible research. *ACM SIGOPS Oper. Syst. Rev.*, **49**, 71–79, https://doi.org/10.1145/2723872.2723882.

Bottum, J., and Coauthors, 2017: The future of cloud for academic research computing.

Microsoft, 21 pp., www.microsoft.com/en-us/research/publication/the-future-of-cloud-for-academic-research-computing/.

Carlyle, A. G., S. L. Harrell, and P. M. Smith, 2010: Cost-effective HPC: The community or the cloud? *Second Int. Conf. on Cloud Computing Technology and Science*, Indianapolis, IN, IEEE, 169–176, https://doi.org/10.1109/CloudCom.2010.115.

Carman, J. C., and Coauthors, 2017: The national earth system prediction capability: Coordinating the giant. *Bull. Amer. Meteor. Soc.*, **98**, 239–252, https://doi.org/10.1175/BAMS-D-16-0002.1.

Chang, S., and Coauthors, 2018: Evaluating the suitability of commercial clouds for NASA's high performance computing applications: A trade study. NAS Tech. Rep. NAS-2018-01, 46 pp., www.nas.nasa.gov/assets/pdf/papers/NAS_Technical_Report_NAS-2018-01.pdf.

Chen, X., X. Huang, C. Jiao, M. G. Flanner, T. Raeker, and B. Palen, 2017: Running climate model on a commercial cloud computing environment: A case study using Community Earth System Model (CESM) on Amazon AWS. *Comput. Geosci.*, **98**, 21–25, https://doi.org/10.1016/j.cageo.2016.09.014.

Chollet, F., 2017: *Deep Learning with Python*. Manning, 384 pp.

Christian, K. E., W. H. Brune, J. Mao, and X. Ren, 2018: Global sensitivity analysis of GEOS-Chem modeled ozone and hydrogen oxides during the INTEX campaigns. *Atmos. Chem. Phys.*, **18**, 2443–2460, https://doi.org/10.5194/acp-18-2443-2018.

Coffrin, C., and Coauthors, 2019: The ISTI rapid response on exploring cloud computing 2018. arXiv, 72 pp., http://arxiv.org/abs/1901.01331.

de Oliveira, A. H. M., D. de Oliveira, and M. Mattoso, 2017: Clouds and reproducibility: A way to go to scientific experiments? *Cloud Computing*, N. Antonopoulos and L. Gillam, Eds., Springer, 127–151 https://doi.org/10.1007/978-3-319-54645-2_5.

Dodson, R., and Coauthors, 2016: Imaging SKA-scale data in three different computing environments. *Astron. Comput.*, **14**, 8–22, https://doi.org/10.1016/j.ascom.2015.10.007.

Duran-Limon, H. A., J. Flores-Contreras, N. Parlavantzas, M. Zhao, and A. Meulenert-Peña, 2016: Efficient execution of the WRF model and other HPC applications in the cloud. *Earth Sci. Inform.*, **9**, 365–382, https://doi.org/10.1007/s12145-016-0253-7.

Eastham, S. D., D. K. Weisenstein, and S. R. H. Barrett, 2014: Development and evaluation of the unified tropospheric–stratospheric chemistry extension (UCX) for the global chemistry-transport model GEOS-Chem. *Atmos. Environ.*, **89**, 52–63, https://doi.org/10.1016/j.atmosenv.2014.02.001.

Eastham, S. D., and Coauthors, 2018: GEOS-Chem high performance (GCHP v11-02c): A next-generation implementation of the GEOS-Chem chemical transport model for massively parallel applications. *Geosci. Model Dev.*, **11**, 2941–2953, https://doi.org/10.5194/gmd-11-2941-2018.

Emeras, J., S. Varrette, and P. Bouvry, 2017: Amazon elastic compute cloud (EC2) vs. in-house HPC platform: A cost analysis. *Ninth IEEE Int. Conf. on Cloud Computing*, San Francisco, CA, IEEE, 284–293, https://doi.org/10.1109/CLOUD.2016.0046.

Evangelinos, C., and C. N. Hill, 2008: Cloud computing for parallel scientific HPC Applications: feasibility of running coupled atmosphere-ocean climate models on Amazon's EC2. *First Workshop on Cloud Computing and Its Applications*, Chicago, IL, University of Chicago, 2–34.

Foster, I., and D. B. Gannon, 2017: *Cloud Computing for Science and Engineering*. MIT Press, 392 pp.

Fox, A., 2011: Cloud computing—What's in it for me as a scientist? *Science*, **331**, 406–407, https://doi.org/10.1126/science.1198981.

Freniere, C., A. Pathak, M. Raessi, and G. Khanna, 2016: The feasibility of Amazon's cloud computing platform for parallel, GPU-accelerated, multiphase-flow simulations. *Comput. Sci. Eng.*, **18**, 68–77, https://doi.org/10.1109/MCSE.2016.94.

GEOS-Chem, 2018a: GEOS-Chem People and Projects. Accessed 20 December 2018, http://acmg.seas.harvard.edu/geos/geos_people.html.

——, 2018b: GEOS-Chem Steering Committee. Accessed 20 December 2018, http://acmg.seas.harvard.edu/geos/geos_steering_cmte.html.

——, 2018c: GNU Fortran compiler. Accessed 20 December 2018, http://wiki.seas.harvard.edu/geos-chem/index.php/GNU_Fortran_compiler.

——, 2018d: Timing tests with GEOS-Chem 12.0.0. Accessed 20 December 2018, http://wiki.seas.harvard.edu/geos-chem/index.php/Timing_tests_with_GEOS-Chem_12.0.0.

——, 2018e: GEOS-Chem benchmarking. Accessed 20 December 2018, http://wiki.seas.harvard.edu/geos-chem/index.php/GEOS-Chem_benchmarking.

——, 2018f: List of diagnostics archived to netCDF format. Accessed 20 December 2018, http://wiki.seas.harvard.edu/geos-chem/index.php/List_of_diagnostics_archived_to_netCDF_format.

Gerhardt, L., W. Bhimji, S. Canon, M. Fasel, D. Jacobsen, M. Mustafa, J. Porter, and V. Tsulaia, 2017: Shifter: Containers for HPC. *J. Phys. Conf. Ser.*, **898**, 082021, https://doi.org/10.1088/1742-6596/898/8/082021.

Google, 2018a: Google Cloud Platform for AWS Professionals. Accessed 20 December 2018, https://cloud.google.com/docs/compare/aws/.

——, 2018b: Google Cloud Platform announces new credits program for researchers. Accessed 20 December 2018, www.blog.google/products/google-cloud/google-cloud-platform-announces-new-credits-program-researchers/.

Gorelick, N., M. Hancher, M. Dixon, S. Ilyushchenko, D. Thau, and R. Moore, 2017: Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sens. Environ.*, **202**, 18–27, https://doi.org/10.1016/j.rse.2017.06.031.

Gupta, A., and D. Milojicic, 2011: Evaluation of HPC applications on cloud. *Sixth Open Cirrus Summit*, Atlanta, GA, IEEE, 22–26, https://doi.org/10.1109/OCS.2011.10.

——, and Coauthors, 2013: The who, what, why, and how of high performance computing applications in the cloud. *Fifth IEE Int. Conf. on Cloud Computing Technology and Science*, Bristol, United Kingdom, IEEE, https://doi.org/10.1109/CloudCom.2013.47.

Hacker, J. P., J. Exby, D. Gill, I. Jimenez, C. Maltzahn, T. See, G. Mullendore, and K. Fossell, 2017: A containerized mesoscale model and analysis toolkit to accelerate classroom learning, collaborative research, and uncertainty quantification. *Bull. Amer. Meteor. Soc.*, **98**, 1129–1138, https://doi.org/10.1175/BAMS-D-15-00255.1.

Hale, J. S., L. Li, C. N. Richardson, and G. N. Wells, 2017: Containers for portable, productive, and performant scientific computing. *Comput. Sci. Eng.*, **19**, 40–50, https://doi.org/10.1109/MCSE.2017.2421459.

Hill, C., C. DeLuca, Balaji, M. Suarez, and A. Da Silva, 2004: The architecture of the Earth system modeling framework. *Comput. Sci. Eng.*, **6**, 18–28, https://doi.org/10.1109/MCISE.2004.1255817.

Hill, Z., and M. Humphrey, 2009: A quantitative analysis of high performance computing with Amazon's EC2 infrastructure: The death of the local cluster? *Tenth IEEE/ACM International Conference on Grid Computing*, Banff, AB, Canada, IEEE, 26–33, https://doi.org/10.1109/GRID.2009.5353067.

Hoesly, R. M., and Coauthors, 2018: Historical (1750–2014) anthropogenic emissions of reactive gases and aerosols from the Community Emissions Data System (CEDS). *Geosci. Model Dev.*, **11**, 369–408, https://doi.org/10.5194/gmd-11-369-2018.

Hong, S.-Y., M.-S. Koo, J. Jang, J.-E. Esther Kim, H. Park, M.-S. Joh, J.-H. Kang, and T.-J. Oh, 2013: An evaluation of the software system dependency of a global atmospheric model. *Mon. Wea. Rev.*, **141**, 4165–4172, https://doi.org/10.1175/MWR-D-12-00352.1.

Hossain, M. M., R. Wu, J. T. Painumkal, M. Kettouch, C. Luca, S. M. Dascalu, and F. C. Harris, 2017: Web-service framework for environmental models. *2017 Internet Technologies and Applications (ITA)*, Wrexham, United Kingdom, IEEE, 104–109, https://doi.org/10.1109/ITECHA.2017.8101919.

Howe, B., 2012: Virtual appliances, cloud computing, and reproducible research. *Comput. Sci. Eng.*, **14**, 36–41, https://doi.org/10.1109/MCSE.2012.62.

Hoyer, S., and J. J. Hamman, 2017: xarray: N-D labeled arrays and datasets in Python. *J. Open Res. Software*, **5**, 1–6, https://doi.org/10.5334/jors.148.

Huang, Q., C. Yang, K. Benedict, S. Chen, A. Rezgui, and J. Xie, 2013: Utilize cloud computing to support dust storm forecasting. *Int. J. Digit. Earth*, **6**, 338–355, https://doi.org/10.1080/17538947.2012.749949.

Hunter, J. D., 2007: Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.*, **9**, 90–95, https://doi.org/10.1109/MCSE.2007.55.

Huntington, J. L., K. C. Hegewisch, B. Daudert, C. . Morton, J. T. Abatzoglou, D.J. McEvoy, and T. Erickson, 2017: Climate engine: Cloud computing and visualization of climate and remote sensing data for advanced natural resource monitoring and process understanding. *Bull. Amer. Meteor. Soc.*, **98**, 2397–2410, https://doi.org/10.1175/BAMS-D-15-00324.1.

Irving, D., 2016: A minimum standard for publishing computational results in the weather and climate sciences. *Bull. Amer. Meteor. Soc.*, **97**, 1149–1158, https://doi.org/10.1175/BAMS-D-15-00010.1.

Jackson, K. R., L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, 2010: Performance analysis of high performance computing applications on the Amazon Web Services Cloud. *Second Int. Conf. on Cloud Computing Technology and Science*, Indianapolis, IN, IEEE, 159–168, https://doi.org/10.1109/CloudCom.2010.69.

Jacobsen, D. M., and R. S. Canon, 2015: Contain this, unleashing Docker for HPC. *Cray User Group 2015*, Chicago, IL, Cray User Group, 8 pp. www.nersc.gov/assets/Uploads/cug2015udi.pdf.

Jeong, J. I., and R. J. Park, 2018: Efficacy of dust aerosol forecasts for East Asia using the adjoint of GEOS-Chem with ground-based observations. *Environ. Pollut.*, **234**, 885–893, https://doi.org/10.1016/j.envpol.2017.12.025.

Jing, Y., T. Wang, P. Zhang, L. Chen, N. Xu, and Y. Ma, 2018: Global atmospheric CO_2 concentrations simulated by GEOS-Chem: Comparison with GOSAT, carbon tracker and ground-based measurements. *Atmosphere*, **9**, 175, https://doi.org/10.3390/atmos9050175.

Jonas, E., and Coauthors, 2019: Cloud programming simplified: A Berkeley view on serverless computing. arXiv, 33 pp., http://arxiv.org/abs/1902.03383.

Jung, K., Y. Cho, and Y. Tak, 2017: Performance evaluation of ROMS v3.6 on a commercial cloud system.

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2017-270.

Keller, C. A., M. S. Long, R. M. Yantosca, A. M. Da Silva, S. Pawson, and D. J. Jacob, 2014: HEMCO v1.0: A versatile, ESMF-compliant component for calculating emissions in atmospheric models. *Geosci. Model Dev.*, **7**, 1409–1417, https://doi.org/10.5194/gmd-7-1409-2014.

Kurtzer, G. M., V. Sochat, and M. W. Bauer, 2017: Singularity: Scientific containers for mobility of compute. *PLOS ONE*, **12**, e0177459, https://doi.org/10.1371/journal.pone.0177459.

Langkamp, T., and J. Böhner, 2011: Influence of the compiler on multi-CPU performance of WRFv3. *Geosci. Model Dev.*, **4**, 611–623, https://doi.org/10.5194/gmd-4-611-2011.

Li, R., L. Liu, G. Yang, C. Zhang, and B. Wang, 2016: Bitwise identical compiling setup: Prospective for reproducibility and reliability of Earth system modeling. *Geosci. Model Dev.*, **9**, 731–748, https://doi.org/10.5194/gmd-9-731-2016.

Li, Z., C. Yang, Q. Huang, K. Liu, M. Sun, and J. Xia, 2017: Building Model as a Service to support geosciences. *Comput. Environ. Urban Syst.*, **61**, 141–152, https://doi.org/10.1016/j.compenvurbsys.2014.06.004.

Lynnes, C., K. Baynes, and M. A. McInerney, 2017: Archive management of NASA Earth observation data to support cloud analysis. NASA Doc., 4 pp., https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20170011455.pdf.

Mehrotra, P., J. Djomehri, S. Heistand, R. Hood, H. Jin, A. Lazanoff, S. Saini, and R. Biswas, 2016: Performance evaluation of Amazon Elastic Compute Cloud for NASA high-performance computing applications. *Concurr. Comput.*, **28**, 1041–1055, https://doi.org/10.1002/cpe.3029.

Met Office, 2016: Cartopy: A cartographic python library with a matplotlib interface. http://scitools.org.uk/cartopy.

Microsoft, 2018a: Microsoft Azure for research online. Accessed 20 December 2018, https://a4ronline.azurewebsites.net.

——, 2018b: Azure for AWS professionals. Accessed 20 December 2018, https://docs.microsoft.com/en-us/azure/architecture/aws-professional/.

——, 2018c: Azure AI for Earth grants. Accessed 20 December 2018, www.microsoft.com/en-us/ai-for-earth/grants.

Mohammadi, M., and T. Bazhirov, 2017: Comparative benchmarking of cloud computing vendors with high performance Linpack. *Proc. Second Int. Conf. on High Performance Compilation, Computing and Communications*, Hong Kong, China, ACM, 5 pp., https://doi.org/10.1145/3195612.3195613.

Molthan, A., J. 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.

Montes, D., J. A. Añel, T. F. Pena, P. Uhe, and D. C. H. Wallom, 2017: Enabling BOINC in infrastructure as a service cloud system. *Geosci. Model Dev.*, **10**, 811–826, https://doi.org/10.5194/gmd-10-811-2017.

NAS, 2016: *Future Directions for NSF Advanced Computing Infrastructure to Support U.S. Science and Engineering in 2017–2020*. National Academies Press, 156 pp., https://doi.org/10.17226/21886.

——, 2018: *Thriving on Our Changing Planet: A Decadal Strategy for Earth Observation from Space*. National Academies Press, 716 pp., https://doi.org/10.17226/24938.

NASA HECC, 2018: Enabling user-defined software stacks with Charliecloud. Accessed 20 December 2018, www.nas.nasa.gov/hecc/support/kb/enabling-user-defined-software-stacks-with-charliecloud_558.html.

Netto, M. A. S., R. N. Calheiros, E. R. Rodrigues, R. L. F. Cunha, and R. Buyya, 2018: HPC cloud for scientific and business applications: Taxonomy, vision, and research challenges. *ACM Comput. Surv.*, **51**, 8, https://doi.org/10.1145/3150224.

NSF, 2018a: Critical Techniques, Technologies and Methodologies for Advancing Foundations and Applications of Big Data Sciences and Engineering (BIGDATA). Accessed 20 December 2018, www.nsf.gov/funding/pgm_summ.jsp?pims_id=504767.

——, 2018b: NSF and Internet2 to explore cloud computing to accelerate science frontiers. Accessed 20 December 2018, https://nsf.gov/news/news_summ.jsp?cntn_id=297193.

Oesterle, F., S. Ostermann, R. Prodan, and G. J. Mayr, 2015: Experiences with distributed computing for meteorological applications: Grid computing and cloud computing. *Geosci. Model Dev.*, **8**, 2067–2078, https://doi.org/10.5194/gmd-8-2067-2015.

Pary, R., 2018: New Amazon EC2 Spot pricing model: Simplified purchasing without bidding and fewer interruptions. AWS Compute Blog, 13 March, accessed 20 December 2018, https://aws.amazon.com/blogs/compute/new-amazon-ec2-spot-pricing/.

Perkel, J. M., 2018: Why Jupyter is data scientists' computational notebook of choice. *Nature*, **563**, 145–146, https://doi.org/10.1038/d41586-018-07196-1.

Priedhorsky, R., T. C. Randles, and T. Randles, 2017: Charliecloud: Unprivileged containers for user-defined software stacks in HPC. *Proc. Int. Conf. High Performance Computing, Networking, Storage and*

Analysis, Denver, CO, ACM, Article 36, https://doi.org/10.1145/3126908.3126925.

Research Computing, 2018: Singularity on Odyssey. Harvard, accessed 20 December 2018, www.rc.fas.harvard.edu/resources/documentation/software/singularity-on-odyssey/.

Rocklin, M., 2015: Dask: Parallel computation with blocked algorithms and task scheduling. *Proc. 14th Python in Science Conf.*, Austin, TX, SciPy, 126–132, http://conference.scipy.org/proceedings/scipy2015/pdfs/matthew_rocklin.pdf.

——, 2018: HDF in the Cloud: Challenges and solutions for scientific data. Accessed 20 December 2018, http://matthewrocklin.com/blog/work/2018/02/06/hdf-in-the-cloud.

Roloff, E., M. Diener, A. Carissimi, and P. O. A. Navaux, 2012: High Performance Computing in the cloud: Deployment, performance and cost efficiency. *Proc. Fourth IEEE Int. Conf. on Cloud Computing Technology and Science Proceedings*, Taipei, Taiwan, IEEE, 371–378, https://doi.org/10.1109/CloudCom.2012.6427549.

——, ——, L. P. Gaspary, and P. O. A. Navaux, 2017: HPC application performance and cost efficiency in the cloud. *Proc. 2017 25th Euromicro Int. Conf. Parallel, Distribruted Network-Based Processing*, St. Petersburg, Russia, 473–477, https://doi.org/10.1109/PDP.2017.59.

Salaria, S., K. Brown, H. Jitsumoto, and S. Matsuoka, 2017: Evaluation of HPC-Big Data applications using cloud platforms. *Proc. 17th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing*, Madrid, Spain, IEEE, 1053–1061, https://doi.org/10.1109/CCGRID.2017.143.

Shaikh, R., 2018: tmux on AWS. GitHub, accessed 20 December 2018, https://github.com/reshamas/fastai_deeplearn_part1/blob/master/tools/tmux.md.

Shen, H., 2014: Interactive notebooks: Sharing the code. *Nature*, **515**, 151–152, https://doi.org/10.1038/515151a.

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.

Strazdins, P. E., J. Cai, M. Atif, and J. Antony, 2012: Scientific application performance on HPC, private and public cloud resources: A case study using climate, cardiac model codes and the NPB benchmark suite. *Proc. IEEE 26th Int. Parallel and Distributed Processing Symp. Workshops and PhD Forum*, Shanghai, China, IEEE, 1416–1424, https://doi.org/10.1109/IPDPSW.2012.186.

Thackston, R., and R. C. Fortenberry, 2015: The performance of low-cost commercial cloud computing as an alternative in computational chemistry. *J. Comput. Chem.*, **36**, 926–933, https://doi.org/10.1002/jcc.23882.

Tian, Y., Y. Sun, C. Liu, P. Xie, K. Chan, J. Xu, W. Wang, and J. Liu, 2018: Characterization of urban CO2 column abundance with a portable low resolution spectrometer (PLRS): Comparisons with GOSAT and GEOS-Chem model data. *Sci. Total Environ.*, **612**, 1593–1609, https://doi.org/10.1016/j.scitotenv.2016.12.005.

Tsaftaris, S. A., 2014: A scientist's guide to cloud computing. *Comput. Sci. Eng.*, **16**, 70–76, https://doi.org/10.1109/MCSE.2014.12.

Unidata, 2019: NetCDF and native cloud storage access via Zarr. www.unidata.ucar.edu/blogs/news/entry/netcdf-and-native-cloud-storage.

Vance, T. C., N. Merati, C. P. Yang, and M. Yuan, Eds., 2016: *Cloud Computing in Ocean and Atmospheric Sciences*. Elsevier, 454 pp.

VanderPlas, J., 2016: *Python Data Science Handbook*. O'Reilly Media, 548 pp.

Walker, E., 2008: Benchmarking Amazon EC2 for high-performance scientific computing. *;login: The USENIX Magazine*, Vol. 33, No. 5, 18–23, www.usenix.org/publications/login/october-2008-volume-33-number-5/benchmarking-amazon-ec2-high-performance.

Withana, E. C., B. Plale, and C. Mattocks, 2011: Towards enabling mid-scale geoscience experiments through Microsoft Trident and Windows Azure. *Cloud Futures 2011 Workshop*, Redmond, WA, Microsoft, 5 pp.

Yang, C., Q. Huang, Z. Li, K. Liu, and F. Hu, 2017: Big Data and cloud computing: innovation opportunities and challenges. *Int. J. Digit. Earth*, **10**, 13–53, https://doi.org/10.1080/17538947.2016.1239771.

Yelick, K., and Coauthors, 2011: The Magellan Report on Cloud Computing for Science. DOE Rep., 170 pp., https://doi.org/10.2172/1076794.

Younge, A. J., K. Pedretti, R. E. Grant, and R. Brightwell, 2017: A tale of two systems: Using containers to deploy HPC applications on supercomputers and clouds. *2017 IEEE Int. Conf. on Cloud Computing Technology and Science*, Hong Kong, China, IEEE, 74–81, https://doi.org/10.1109/CloudCom.2017.40.

Yu, K., C. A. Keller, D. J. Jacob, A. M. Molod, S. D. Eastham, and M. S. Long, 2018: Errors and improvements in the use of archived meteorological data for chemical transport modeling: An analysis using GEOS-Chem v11-01 driven by GEOS-5 meteorology. *Geosci. Model Dev.*, **11**, 305–319, https://doi.org/10.5194/gmd-11-305-2018.

Zhai, Y., M. Liu, J. Zhai, X. Ma, and W. Chen, 2011: Cloud versus in-house cluster: Evaluating Amazon

cluster compute instances for running MPI applications. *Proc. State of the Practice Reports*, Seattle, WA, ACM, Article 11, https://doi.org/10.1145/2063348.2063363.

Zhang, J., X. Lu, and D. K. Panda, 2017: Is singularity-based container technology ready for running MPI applications on HPC clouds? *Proc. 10th Int. Conf. on Utility and Cloud Computing*, Austin, TX, ACM, 151–160, https://doi.org/10.1145/3147213.3147231.

Zhu, L., M. Val Martin, L. V. Gatti, R. Kahn, A. Hecobian, and E. V. Fischer, 2018: Development and implementation of a new biomass burning emissions injection height scheme (BBEIH v1.0) for the GEOS-Chem model (v9-01-01). *Geosci. Model Dev.*, **11**, 4103–4116, https://doi.org/10.5194/gmd-11-4103-2018.

Zhuang, J., 2018: JiaweiZhuang/xESMF: v0.1.1. Zenodo, https://doi.org/10.5281/ZENODO.1134366.

——, 2019: A scientist's guide to Cloud-HPC: Example with AWS ParallelCluster, Slurm, Spack, and WRF. GitHub, accessed 20 April 2019, https://jiaweizhuang.github.io/blog/aws-hpc-guide/.