Running climate model on a commercial cloud computing environment: A case study using Community Earth System Model (CESM) on Amazon AWS

Xiuhong Chen^{1,*}, Xianglei Huang¹, Chaoyi Jiao¹, Mark G. Flanner¹, Todd Raeker² and Brock Palen²

1. Department of Climate and Space Sciences and Engineering, the University of Michigan, Ann Arbor, MI 48109-2143
2. Advanced Research Computing-Technology Services, the University of Michigan, Ann Arbor, Michigan, USA

*Corresponding author: Dr. Xiuhong Chen, Dept. of Climate and Space Sciences and Engineering, the University of Michigan, Ann Arbor, Michigan, 48109-2143, USA. (Tel) +1-734-647-7687, email: xiuchen@umich.edu

Submitted to *Computers & Geosciences* on February 14, 2016

Revision on August 9, 2016

Abstract

The suites of numerical models used for simulating climate of our planet are usually run on dedicated high-performance computing (HPC) resources. This study investigates an alternative to the usual approach, i.e. carrying out climate model simulations on commercially available cloud computing environment. We test the performance and reliability of running the CESM (Community Earth System Model), a flagship climate model in the United States developed by the National Center for Atmospheric Research (NCAR), on Amazon Web Service (AWS) EC2, the cloud computing environment by Amazon.com, Inc. StarCluster is used to create virtual computing cluster on the AWS EC2 for the CESM simulations. The wall-clock time for one year of CESM simulation on the AWS EC2 virtual cluster is comparable to the time spent for the same simulation on a local dedicated high-performance computing cluster with InfiniBand connections. The CESM simulation can be efficiently scaled with the number of CPU cores on the AWS EC2 virtual cluster environment up to 64 cores. For the standard configuration of the CESM at a spatial resolution of 1.9° latitude by 2.5° longitude, increasing the number of cores from 16 to 64 reduces the wall-clock running time by more than 50% and the scaling is nearly linear. Beyond 64 cores, the communication latency starts to outweigh the benefit of distributed computing and the parallel speedup becomes nearly unchanged.

Keywords: Cloud computing; Amazon Web Service; CESM; climate models; parallelization efficiency

1. Introduction

Numerical models are the major tools for studying climate changes in response to anthropogenic emissions of greenhouse gases (Weart, 2010). Such models numerically solve a system of equations of fluid dynamics, physics, and sometimes even chemistry to simulate the time-dependent state of our climate system at appropriate spatial and temporal scales affordable by current computing resources. Due to the computationally demanding nature of such numerical simulations, they are usually run on dedicated supercomputing resources (Nyberg, 2010). Each year millions of dollars are spent on computer hardware and IT supports by climate modeling centers around the globe such that simulation results can be produced in a timely manner for the Intergovernmental Panel on Climate Change report (IPCC, 2014) as well as other scientific studies.

The emergence of commercially available cloud-based computing resources, such as Elastic Compute Cloud (EC2) at Amazon Web Services (AWS) (Amazon EC2, 2016), provides an alternative for carrying out such computationally expensive simulations. Cloud computing such as EC2 at AWS allows users to increase or decrease computing capacity within minutes instead of hours or days. Users can easily create, launch, and terminate one or multiple instances (i.e. virtual computing environments) for a variety of operating systems as well as applications. StarCluster (2011), an open source cluster-computing toolkit for Amazon cloud service, can help users set up a virtual cluster by combining several instances together, making a virtual cluster similar to the traditional high-performance computing clusters or supercomputing facilities that researchers used to running the climate models. Moreover, comparing to traditional supercomputing, a considerable advantage of commercial cloud computing environment is its

capacity of information integration. Due to the size of commercial cloud computing such as the AWS and vastly different backgrounds of its users, such cloud computing service can host tremendous amount of data from different public and private sectors. For example, NASA NEX (2016) is one of AWS centralize repository of public data sets. It includes high-resolution climate data primarily for agriculture, NASA MODIS and Landsat satellite imageries, as well as downscaled global daily forecast. Such AWS public datasets can be seamlessly integrated into AWS cloud-based applications for free. Such information integration capability is unique for cloud computing. Given the fact that the emphasis on coping with climate change has gradually shifted to adaptation and mitigation, open infrastructure that can accommodate different kinds of data sets and numerical models and seamlessly integrate them under the same computing environment is obviously an attractive feature. Such features make cloud computing a promising platform for future climate research and studies.

There are challenges for running climate models in a cloud computing environment. Traditional supercomputing or high-performance computing facilities (HPC; hereafter; both referred as HPC for brevity) usually have dedicated staff team specialized in the maintenance and monitoring of the HPC operations, for both hardware and software. Researchers only need to learn how to carry out simulation correctly. Cloud computing, however, needs the full practice from installing operating system, software, libraries, and other tools required for running climate models to transferring simulation results from remote cloud environment back to local storage. Quarati et al. (2015) discussed running WRF (weather research forecasting) model on hybrid clouds with a focus on analyzing its performance behavior of a brokering algorithm. WRF is a model primarily used in the study of short-term weather forecast. To our best knowledge, there

has been little literature discussing feasibility of running climate models on mainstream commercial cloud computing environment such as the AWS.

The community earth system model (CESM) is a flagship global climate model in the United States developed by the National Center of Atmospheric Research (Gent et al., 2011). This study uses the CESM as a case study to investigate the feasibility and performance of running a climate model on the AWS EC2 cloud services. We use the performance of a local HPC cluster in the University of Michigan as benchmark to evaluate the performance on running the CESM on the AWS EC2. Section 2 describes the CESM and explains how to create the virtual computing clusters on AWS EC2 suitable for the CESM simulation. Section 3 summarizes the findings about the performance of running the CESM on the AWS EC2 with respect to that on the local HPC. Conclusions and further discussions are presented in Section 4.

2. Running the CESM on the AWS EC2

The CESM version 1.1.1 is used in this study with its standard configuration. The CESM contains six modular components and one Coupler (cpl) running in parallel. The six modular components are Atmosphere (atm), Sea ice (ice), Land (lnd), River runoff (rof), Ocean (ocn), Land ice (glc). Figure 1 shows exchanges between the six modules and the coupler. Spatial resolution is 1.9° latitude by 2.5° longitude for the atmosphere and land modules and 1° by 1° grid for the ocean and sea ice. During a CESM run, all modules periodically exchange information with the coupler. The coupler receives results from individual modules, and then computes, maps, and merges the results and sends relevant information back to each module. The CESM uses the Message-Passing Interface (MPI, Barney, 2015) for communications between the modules and between the module and the coupler. We use openMPI threads for the

parallel implementations of the model within the MPI tasks. The CESM components are written primarily in FORTRAN language and adjunctively in C language. The compiled CESM model is ~1.7 Gigabytes (GB) in size. The input data needed for running the CESM in its standard configuration are about 21GB in total and can be downloaded online¹. The standard output of one-year CESM simulation is about ~16 GB.

Figure 1. Illustration of the exchange between modules and the coupler in the CESM 1.1.1.

Table 1. Technical descriptions of the AWS EC2 instances used in this study (ECU stands for EC2 computing unit, vCPU for virtual CPU, and SSD for Solid-State Drive. Memory refers to the total available memory on all the vCPUs).

Node	instances	ECU	vCPU	memory	storage	Network	processor
Master node	r3.8xlarge	104	32	224 GB	640 GB (SSD)	10 Gigabit Ethernet	2.5 GHz Intel Xeon E5-2670
Compute node	c3.8xlarge	108	32	60 GB	640 GB (SSD)	10 Gigabit Ethernet	2.8 GHz Intel Xeon E5-2680

¹ CESM input data is available online at <https://svn-ccsm-inputdata.cgd.ucar.edu/trunk/inputdata/>.

The virtual computing cluster is created using StarCluster following the instruction provided by AWS tutorial (Starcluster, 2016). Specifically, StarCluster needs to be installed first with creation of a folder named “.starcluster” under home directory. Then the detailed realization of the virtual cluster needs to be configured in a file named “config” under “.starcluster”. Three key parameters need to be specified. (1) The size of the cluster, i.e. number of nodes included in the cluster. One is the master node and the left are compute node/nodes, so the minimum size is 2. (2) The instance type of each node. We set r3.8xlarge for the master node instance type and c3.8xlarge for the compute node instance type. r3.8xlarge and c3.8xlarge are two instances named by Amazon Web Service. The technical specifications of both instances can be found in Table 1. Note all nodes of a virtual cluster must be chosen from the same Amazon data center (in our case, the Amazon US west data center in Oregon was chosen), and one instance is launched for one node. (3) Content in the virtual computer image, also known as Amazon Machine Image (AMI) for the AWS. The AMI contains operating system, applications, and relevant databases. Users have flexibility of building their own AMI from scratch or choosing a predefined AMI from the Amazon community public AMI list. We choose to use a predefined AMI with image ID of ami-80bedfb0. This image is installed with 64-bit ubuntu-13.04 HVM (hardware virtual machine) EBS-backed (backed by Elastic Block Store) operating system. GNU Fortran and C compilers are included in the operation system, so does the openMPI package. After the virtual cluster is set up, we copy the original CESM module codes, input data sets, and the necessary libraries (such as NETCDF libraries and HDF libraries) onto the master node/instance. For comparison, we also carry out simulation using the identical CESM code on a local HPC cluster in the University of Michigan (ARC-TS, 2011). It is a Linux-based HPC cluster using 40 Gigabit InfiniBand connection, each node having 16 cores (Intel Xeon 2.60 GHz E5-2670 processor)

with 4 GB RAM on each core (64 GB RAM in total for each node, almost the same memory as what the c3.8xlarge instance has).

Figure 2 summarizes the workflow of running the CESM on the AWS EC2 virtual computing cluster. There are three main components in the system: a single master node, compute nodes, and the storage. The batch job for running the CESM is submitted via the master node of the virtual computer cluster and the actual model simulation is executed on compute nodes. The storage is attached to the master node, which could be either a SSD storage provided by the master instance itself or specifically requested SSD or S3 storage. For this study we requested a 500 GB of SSD so that the data produced by the model runs can be conveniently attached to other AWS instance and conveniently transferred back to our local computers. Compute nodes share the compiled and executable CESM file, the storage, and the software libraries on the master node. Parallel computing on the AWS EC2 are done through SecureShell protocol.

Figure 2. Flowchart of running the CESM on the AWS EC2 virtual computing cluster

3. Performance on the AWS EC2 and benchmark with respect to the local HPC cluster

Figure 3. (a) Wall-clock running time as a function of number of cores on a single node, black line for the instance c3.8xlarge and red line for the instance r3.8xlarge. (b) Same as (a) but for speedup. The blue line shows the ideal speedup line.

We first test the computational efficiency of the AWS EC2 for one node (one instance). We use different numbers of cores (MPI tasks) to run the CESM model on the single node. The results are shown in Figure 3. Wall-clock running time is defined as the elapsed time between the actual start and the end of the simulation. The wall-clock running time is reduced linearly from 4 cores to 16 cores and then it remains nearly unchanged from 16 cores and 32 cores. Such computing efficiency for a single node holds for both the c3.8xlarge and r3.8xlarge configurations. Thus we conclude that, for a node with no more than 32 vCPUs, the optimal number of MPI tasks is 16. This is consistent with the AWS technical manual, which deems the

vCPU as a counterpart of the hyperthreading of Intel Xeon core² (two virtual cores on the AWS share the workload between them for hyperthreading)³. Based on this conclusion, 16 cores (16 MPI tasks) per node are used in the subsequent tests to optimize the parallelization efficiency within each node.

To further evaluate the parallelization efficiency of the CESM on the AWS EC2 virtual cluster with multiple cores, we change the number of cores from 16 to 160 with an increment of 16 cores (i.e. change the number of compute nodes from 1 to 10 with an increment of 1 node, 16 cores per node as mentioned above) and redo the simulation for each increment. In order to take the actual network loading into account, the simulation at each configuration has been run three times to obtain an estimate of the average wall-clock running time as well as an estimate of the standard deviation. The identical simulation was also carry out on the aforementioned local HPC cluster at the University of Michigan. Figure 4a summarizes the results, showing the wall-clock running time in hours for one year of the simulation with respect to the number of cores used in the experiments. On the AWS EC2, increasing the number of cores from 16 to 64 (i.e. from 1 node to 4 nodes) leads to a 55.6% reduction in running time and such reduction has a good linear dependence with the number of cores, as shown in Figure 4b. When more than 64 cores are used, the communication latency starts to overweight the benefit of distributed computing and the running time changes little from 64 to 160 cores. As for the local HPC cluster (black line in Figure 4a), the running time still shows a significant reduction when number of cores increases to be 112. Such contrast suggests that the inter-core communication latency impacts the CESM performance on the AWS EC2 virtual cluster more than that on the local cluster. This contrast is consistent with the network performance difference between 10 Gigabit Ethernet on AWS and

² <https://aws.amazon.com/ec2/instance-types/>.

³ <https://en.wikipedia.org/wiki/Hyper-threading>.

40 Gigabit InfiniBand on the local HPC cluster. We tested MPI bandwidths and latencies on the AWS EC2 and on the HPC cluster using NetPIPE (Network Protocol Independent Performance Evaluator)⁴ for small messages (<64 KBytes). The results are summarized in Figure 5. The bandwidths on the AWS EC2 are nearly 40 times smaller than those on local HPC, and latencies on AWS EC2 are nearly 40 times larger than those on the local HPC. A study from HPC Advisory Council (HPC AC, 2014) also found that the scalability of CAM-SE model (an atmospheric model in CESM model) using FDR InfiniBand expects to grow at larger CPU core counts than using 10 Gigabit Ethernet. This contrast is also consistent with the fact that compared to traditional HPC clusters, the AWS EC2 virtual cluster needs additional virtualization software to make a homogenous computing environment.

Figure 4 (a) Wall-clock running time for one-year CESM simulation as a function of the number of cores on the local HPC cluster (black line) and on the AWS EC2 virtual cluster (red line).

⁴ <http://bitstampj.org/netpipe/>.

Three simulations are done for each configuration to obtain the estimates of mean and standard deviation (denoted with ticked vertical lines). (b) Same as (a) but for speedup. 112 cores are the maximum cores available to us on the local HPC cluster.

Figure 5 Bandwidth (a) and latency (b) as a function of message size on local HPC and AWS EC2 tested using NetPIPE.

4. Conclusions and discussions

This study investigates the feasibility and performance of running global climate models on a cloud computing environment. We create an AWS EC2 virtual cluster using StarCluster software package and carry out the CESM simulations to benchmark the running time and parallelization efficiency. The CESM model can be run in parallel mode on such cloud virtual cluster with minimum amount of efforts of packaging and compiling the code as well as transferring input data sets to the cloud environment. We test the parallelization efficiency of the CESM model on the AWS EC2 virtual cluster and find that, up to 64 cores, the AWS EC2 can

render a parallelization efficiency comparable to (or even better than) that of the traditional Linux cluster with InfiniBand connection. For the case that we test on the AWS virtual cluster, the communication between virtual EC2 nodes outweighs the saving from distributed computing when number of cores exceeds 64 (i.e. 4 nodes). This is different from the case on a local HPC cluster, where the running time is still decreasing when number of cores increases from 64 to 112. Such difference is due to the network on the AWS virtual cluster (10 Gigabit Ethernet with latency of $\sim 80 \mu\text{s}$ for message size of 1 byte) is higher than that on local HPC cluster (40 Gigabit InfiniBand with latency of $\sim 1.7 \mu\text{s}$ for message size of 1 byte). Because 112 cores are the maximum cores available to us on the local HPC cluster, we don't know what is the maximum scalability of CESM model when using 40 Gigabit InfiniBand network on the local HPC cluster with Intel Xeon system. However, Worley et al. (2011) showed the maximum scalability of simulation performance of CESM model can be 8 times when using CPU cores from 64 to 2048 on Cray XT5 system (57.6 Gigabit three-dimensional torus network with latency of $\sim 1 \mu\text{s}$). All these confirm that CESM model is latency sensitive, likely due to the extensive exchange of information among modules and inside each module at each time step of numerical integration. This brings up a question worthy for future investigation: current ways of parallelization in the CESM (likely in other climate models as well) are optimized for the traditional supercomputing facilities, can the code and parallelization be optimized for cloud computing environment?

With the cloud computing gaining more and more popularity in both academia and private sectors, it is conceivable that in the future more and more computationally intensive work such as climate model simulations will be migrated onto cloud computing platform. Our experience of running the CESM on the AWS EC2 indicates that a high-speed network is key for the scalability of the parallelized climate models. Due to the nature and complexity of cloud

computing, the network performance of the instances that we test, i.e., c3.8xlarge and r3.8xlarge, is not always the same. Software configuration (such as compilers) for an AWS EC2 instance could be another major challenge in the way of using cloud computing. One solution is to use predefined computer image (AMI) available on AWS, to extend the original AMI by installing necessary programs and libraries for scientific computing, and then to make a new AMI. The new AMI can then be easily reused by researchers in much less effort of grasping the detailed knowledge about the implementation of cloud computing environment, making it more like the traditional way of running the climate models on the supercomputers.

Acknowledgements

We wish to thank three anonymous reviewers for the thoughtful comments, which helped improve the clarity of the presentation. The authors wish to thank Amazon, Inc. for their generous support through an award made to the University of Michigan under the Amazon Climate Change Initiative. The CESM simulation at HPC cluster in the University of Michigan is made possible through a study supported by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research, Climate and Environmental Science Division under Award Number(s) DE-SC0012969.

References

- Amazon EC2, 2016. Online:<<http://aws.amazon.com/ec2/>>.
- ARC-TS, 2011. Online:< <http://arc-ts.umich.edu/resources/compute-resources/>>.
- Barney, B., 2015. Message Passing Interface (MPI). In Lawrence Livermore National Laboratory. Online:<<https://computing.llnl.gov/tutorials/mpi/>>.

- Gent PR, et al., 2011. The Community Climate System Model Version 4. *J. Clim.* 24(19),4973–4991.
- HPC AC, 2014. CAM-SE Performance Benchmark and Profiling. Online:<http://www.hpcadvisorycouncil.com/pdf/CAM-SE_Analysis_and_Profiling.pdf>.
- IPCC, 2014. Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, R.K. Pachauri and L.A. Meyer (eds.)]. IPCC, Geneva, Switzerland, 151 pp.
- NASA NEX, 2016. Online:< <https://aws.amazon.com/datasets/nasa-nex/>>.
- Nyberg P., 2010. Cray supercomputers in climate, weather and ocean modeling, technique report, WP-XT02-1010. Online:<<http://www.cray.com/sites/default/files/resources/WP-XT02-1010.pdf>>.
- Quarati et al., 2015. Scheduling strategies for enabling meteorological simulation on hybrid clouds. *Journal of Computational and Applied Mathematics*, 273, 438–451.
- StarCluster 2011. Online:<<http://star.mit.edu/cluster/docs/latest/>>.
- Weart, S., 2010. The development of general circulation models of climate. *Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics*, 41(3), 208-217.
- Worley, P. H., Craig, A. P., Dennis, J. M., Mirin, A. A., Taylor, M. A., and Vertenstein, M., 2011. Performance of the Community Earth System Model. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC11)*, Seattle, WA, November 14-18, p. 1-54.