aws-cmaq

Liz Adams

Feb 23, 2024

CONTENTS:

Warning: This documentation is under continuous development Previous version is available here: CMAQv5.3.3 on AWS Tutorial

CHAPTER ONE

COMMUNITY MULTISCALE AIR QUALITY MODEL

The Community Multiscale Air Quality (CMAQ) modeling system an active open-source development project of the U.S. EPA. The CMAQ system is a Linux-based suite of models that requires significant computational resources and specific system configurations to run. CMAQ combines current knowledge in atmospheric science and air quality modeling, multi-processor computing techniques, and an open-source framework to deliver fast, technically sound estimates of ozone, particulates, toxics and acid deposition.

- For additional background on CMAQ please visit the U.S. EPA CMAQ Website.
- CMAQ is a community modeling effort that is supported by the Community Modeling and Analysis System (CMAS) Center at the University of North Caroline at Chapel Hill.

CHAPTER

TWO

TUTORIAL OVERVIEW

This document provides tutorials on how to use Amazon Web Service (AWS) high performance computing services. AWS's Elastic Compute Cloud (EC2) allows you to create and run a single Linux virtual machine, or Single VM, in the cloud. Another AWS resource, ParallelCluster, is a cluster management tool that helps you to deploy and manage many VMs in the cloud. ParallelCluster is well suited for large modeling applications as it automatically sets up the required compute resources, scheduler, and shared file system.

The following tutorials will walk you through running a CMAQ benchmark case on both a Single VM and on the more advanced ParallelCluster. A benchmark case is provided with pre-installed software allowing you to jump right into running CMAQ and post-processing model output for analysis and visualization. The Developers Guide provided in Chapter 4 describes how to do the software installation process yourself in order to set up simulations tailored to your own applications.

The tutorials are aimed at users with cloud computing experience that are already familiar with AWS. For those with no cloud computing experience we recommend signing up for a free AWS Educate account, as it is open to any individual, regardless of where they are in their education, technical experience, or career journey. If you are a new user to AWS, you can sign up for a free tier account. There are also low-cost tutorials available to learn Parallel Cluster from AWS: Parallel Cluster Tutorial and AWS Workshops on HPC Computing.

CHAPTER

THREE

USER SUPPORT

Please share any issues or suggestions for running CMAQ on the Cloud to the CMAS User Forum, under the Cloud Computing Category. This forum is available for users and developers to discuss issues related to using the CMAQ system on the cloud.

To submit edits to this documentation see instructions available in Contribute to this Tutorial.

CHAPTER

FOUR

TABLE OF CONTENTS

4.1 Create Single VM

Elastic Compute Cloud (EC2) Instances can be used to create a virtual machine. This tutorial will explain how to use a public Amazon Machine Image (AMI) that is pre-loaded with software and input data to run CMAQv5.4.

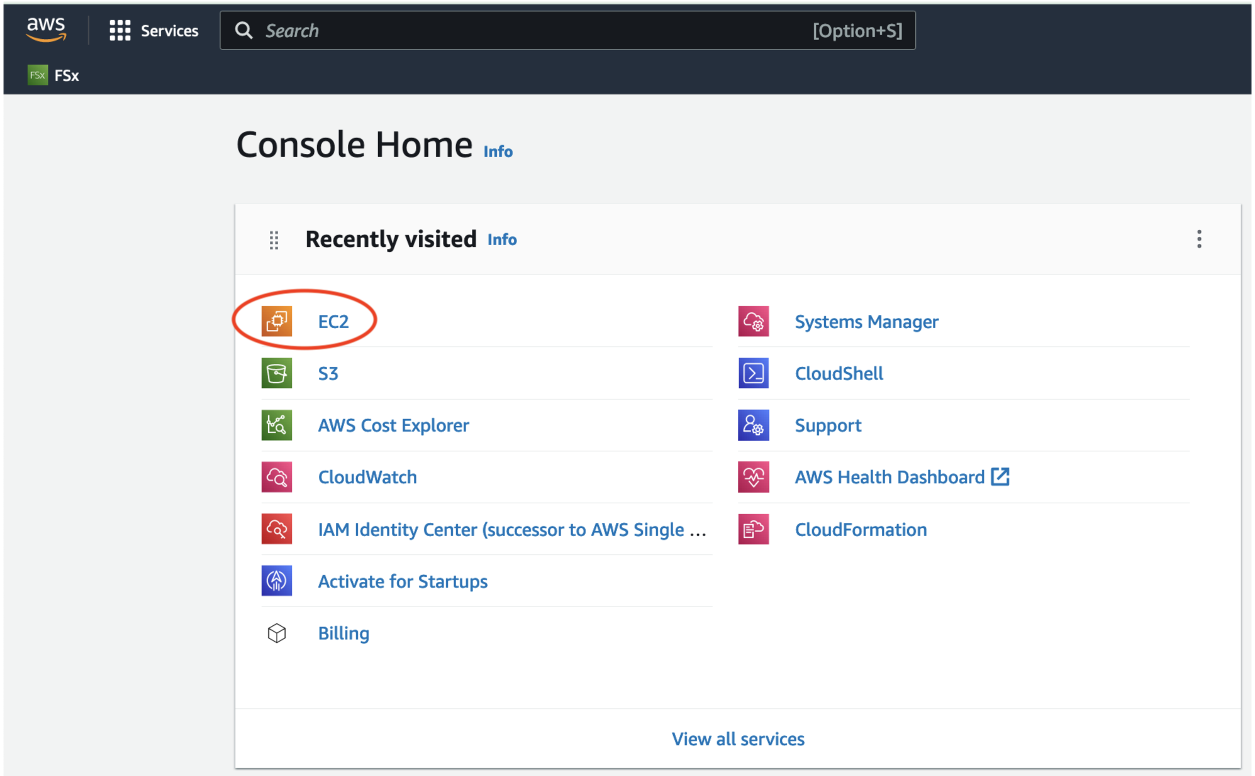
4.1.1 Create a VM from the AWS Web Console

Here we will use an Amazon Elastic Compute Cloud (EC2) C6a instance to run a small CMAQ benchmark case. The software needed to run the benchmark is pre-installed on a public Amazon Machine Image (AMI). The AMI contains all the software required to spin up your virtual server including OS, libraries (MPI, netCDF, I/O API, CMAQ) as well as input data for the benchmark case, publicly available through the AWS Open Data Program.

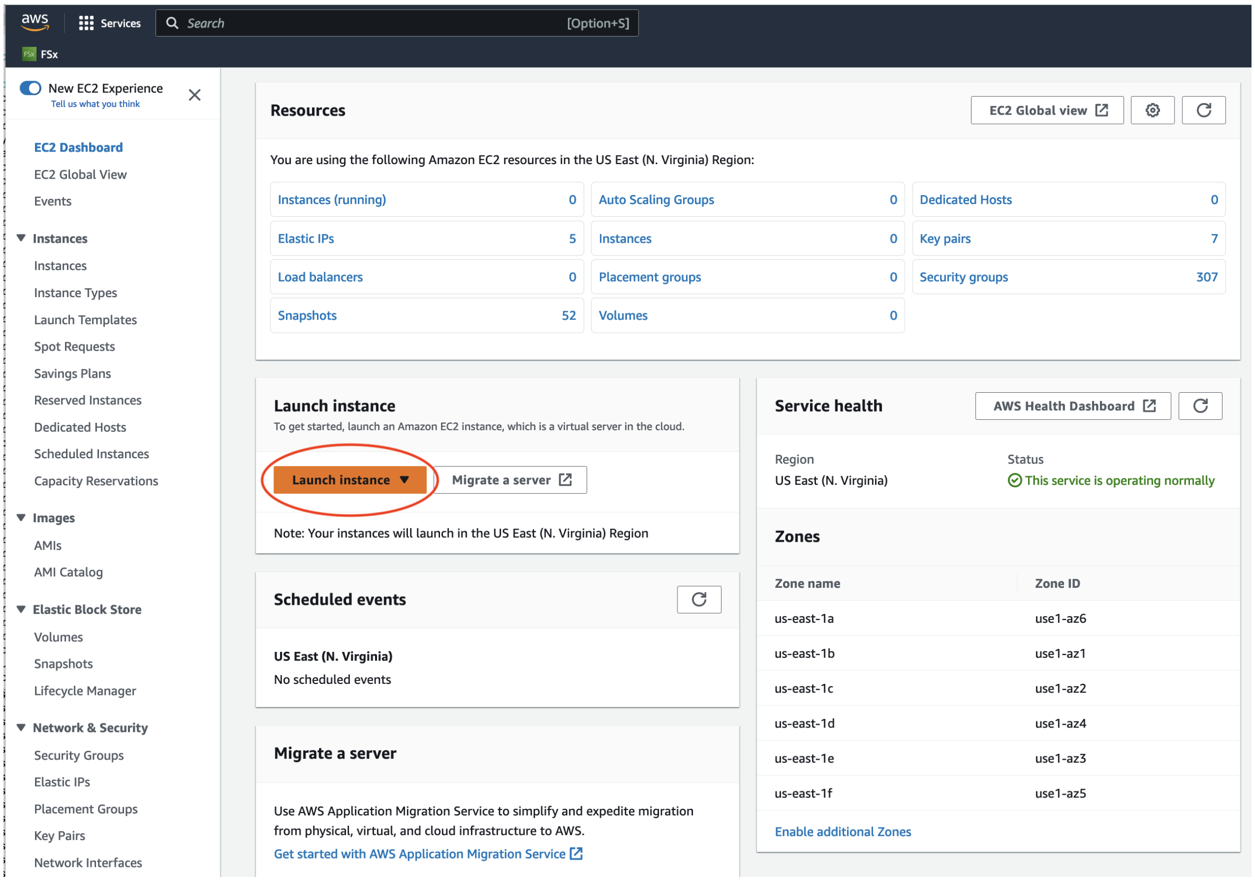
The first step is creating an Virtual Machine (VM) from the AWS Web Console. If you are not able to access the AWS Web Console, skip to section 1.2 to learn how to use the AWS Command Line Interface (CLI).

Note: When working on the AWS Cloud you will need to select a Region for your workloads. (See AWS blog on What to consider when selecting a region). The scripts used in this tutorial use the us-east-1 region, but they can be modified to use any of the supported regions listed here: CLI v3 Supported Regions

1. Login to AWS Web Console and select EC2.



2. Click on the orange “Launch Instance” button.



Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	0	Auto Scaling Groups	0	Dedicated Hosts	0
Elastic IPs	5	Instances	0	Key pairs	7
Load balancers	0	Placement groups	0	Security groups	307
Snapshots	52	Volumes	0		

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Note: Your instances will launch in the US East (N. Virginia) Region

Scheduled events

US East (N. Virginia)
No scheduled events

Migrate a server

Use AWS Application Migration Service to simplify and expedite migration from physical, virtual, and cloud infrastructure to AWS.

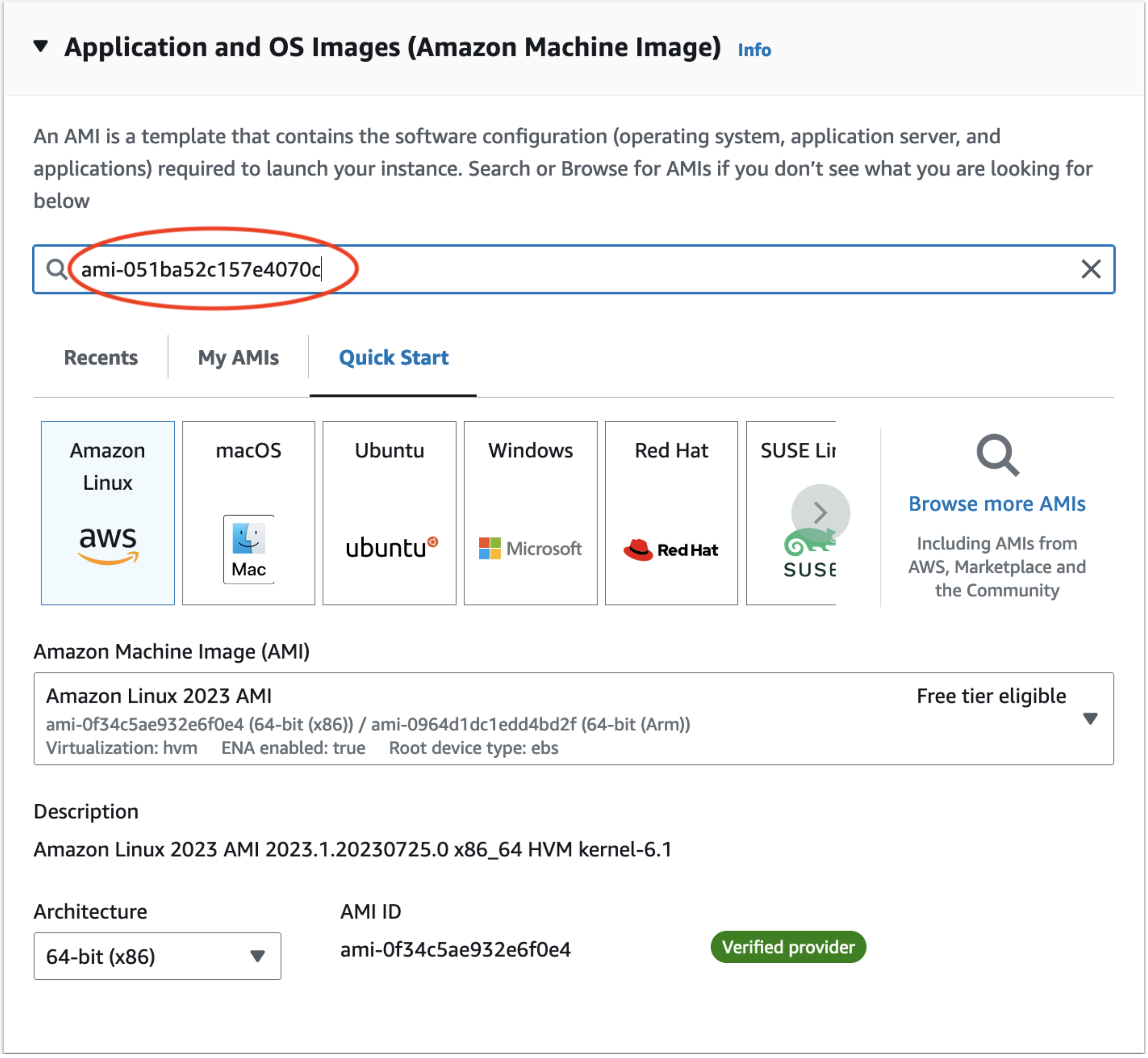
Service health

Region: US East (N. Virginia)
Status: This service is operating normally

Zones

Zone name	Zone ID
us-east-1a	use1-az6
us-east-1b	use1-az1
us-east-1c	use1-az2
us-east-1d	use1-az4
us-east-1e	use1-az3
us-east-1f	use1-az5

3. Search for AMI.
4. Enter the AMI name: ami-051ba52c157e4070c in the search box and return or enter.



▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

ami-051ba52c157e4070c

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible
ami-0f34c5ae932e6f0e4 (64-bit (x86)) / ami-0964d1dc1edd4bd2f (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

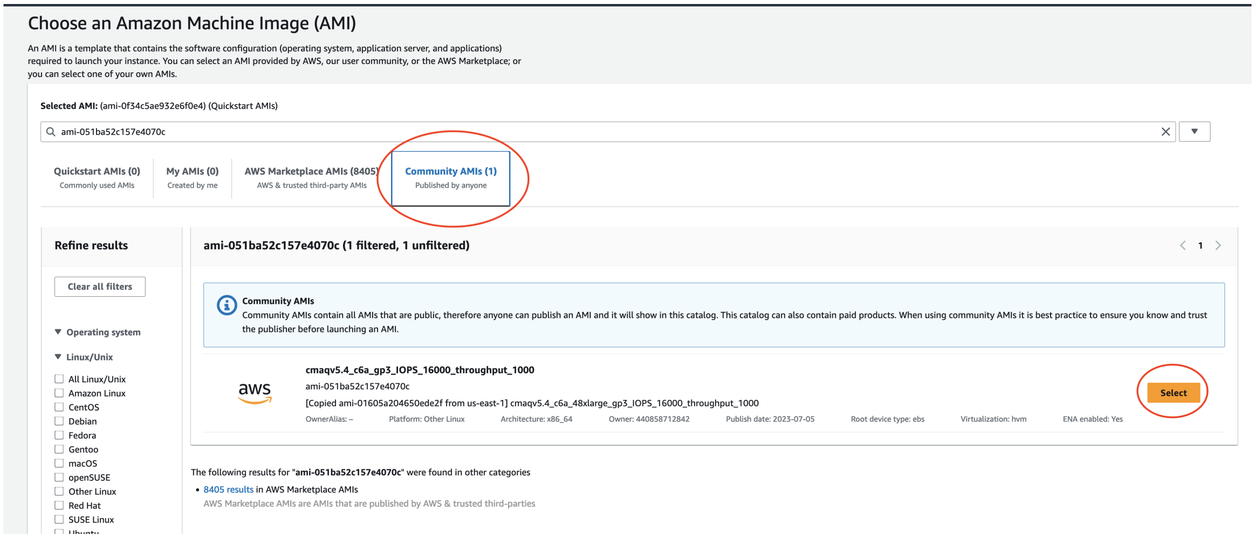
Description

Amazon Linux 2023 AMI 2023.1.20230725.0 x86_64 HVM kernel-6.1

Architecture: 64-bit (x86)

AMI ID: ami-0f34c5ae932e6f0e4

5. Click on the Community AMI tab and then and click on the orange "Select" button.



Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Selected AMI: (ami-0f34c5ae932e6f0e4) (Quickstart AMIs)

ami-051ba52c157e4070c

ami-051ba52c157e4070c (1 filtered, 1 unfiltered)

Community AMIs
Community AMIs contain all AMIs that are public, therefore anyone can publish an AMI and it will show in this catalog. This catalog can also contain paid products. When using community AMIs it is best practice to ensure you know and trust the publisher before launching an AMI.

cmaqv5.4_c6a_gp3_IOPS_16000_throughput_1000
ami-051ba52c157e4070c
[Copied ami-01605a204650ede2f from us-east-1] cmaqv5.4_c6a_48xlarge_gp3_IOPS_16000_throughput_1000
OwnerAlias: – Platform: Other Linux Architecture: x86_64 Owner: 440858712842 Publish date: 2023-07-05 Root device type: ebs Virtualization: hvm ENA enabled: Yes

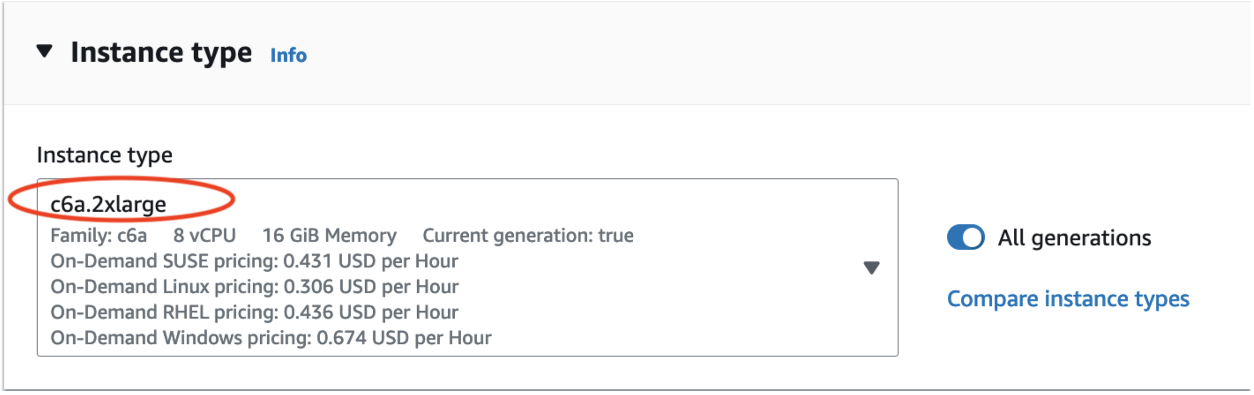
The following results for "**ami-051ba52c157e4070c**" were found in other categories

- 8405 results in AWS Marketplace AMIs
  AWS Marketplace AMIs are AMIs that are published by AWS & trusted third-parties

Note: This Amazon Machine Image (AMI) was built using a C6a EC2 Instance, with default Ubuntu OS (Ubuntu OS, 22.04 LTS, amd64 jammy), with gcc compilers, OpenMPI, netCDF, I/O API, and CMAQv5.4. Amazon EC2 C6a Instances To create a VM using a different family of EC2 instances, you would need to choose the default Ubuntu OS and follow the Developer Guide to install the compilers and software for CMAQv5.4.

6. Search for c6a.2xlarge Instance Type and select it.

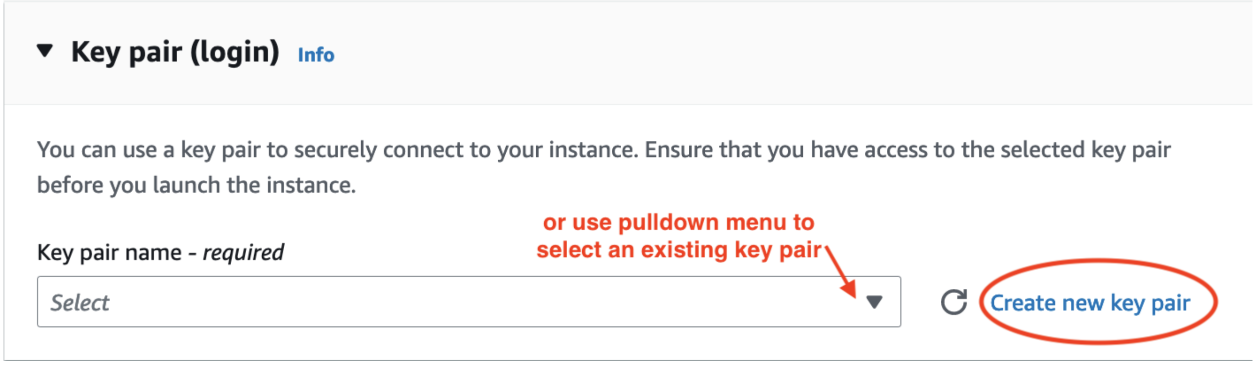
Note, the screenshots show the c6a.2xlarge instance type being selected. If you were running a larger benchmark, you would want to select a larger sized instance such as a c6a.8xlarge or c6a.48xlarge.



Instance type

c6a.2xlarge
Family: c6a 8 vCPU 16 GiB Memory Current generation: true
On-Demand SUSE pricing: 0.431 USD per Hour
On-Demand Linux pricing: 0.306 USD per Hour
On-Demand RHEL pricing: 0.436 USD per Hour
On-Demand Windows pricing: 0.674 USD per Hour

7. Select key pair name or create a new key pair.

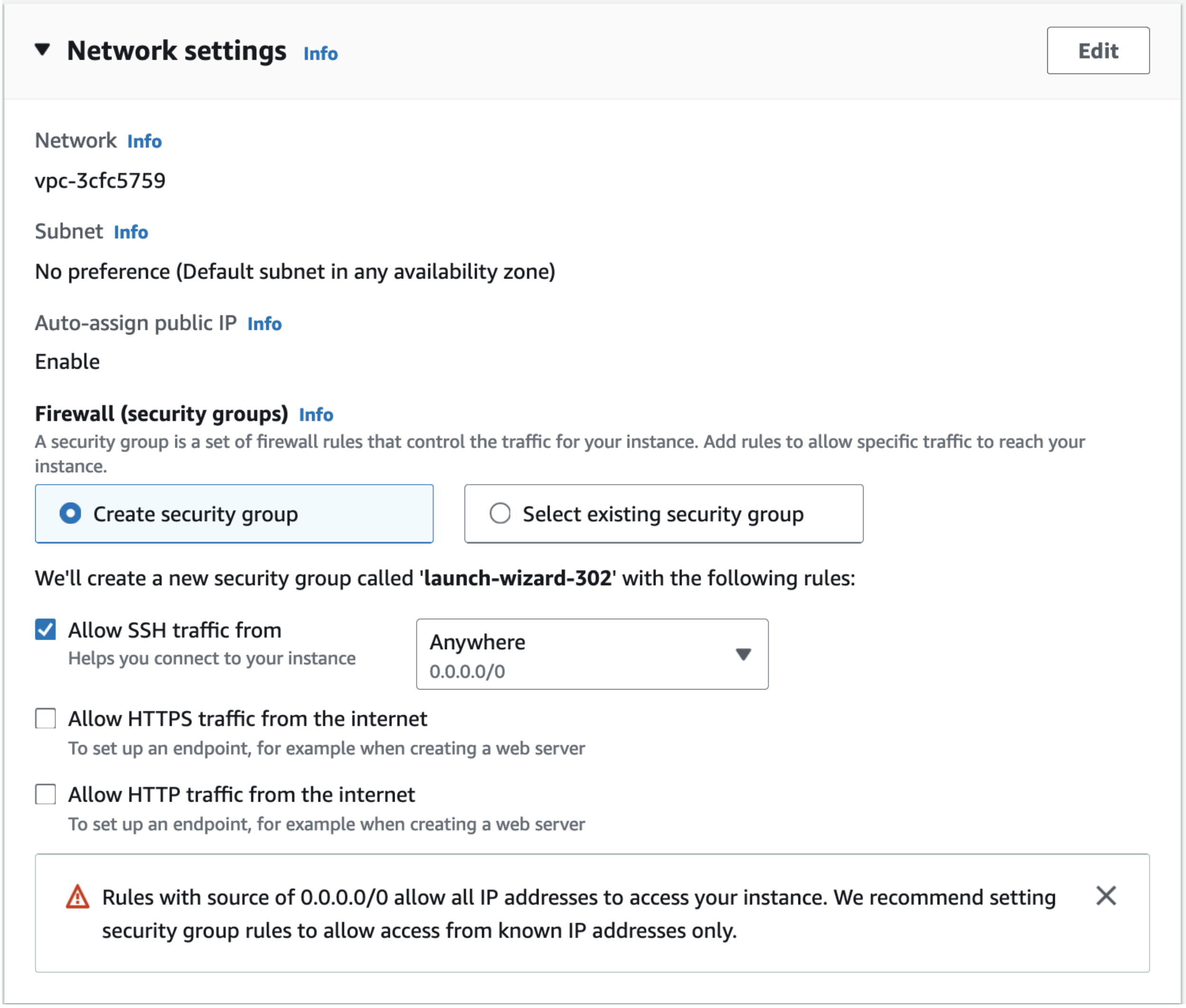


You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Select

8. Use the default network settings.



▼ **Network settings** Info

Network Info

vpc-3cfc5759

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

● Create security group ○ Select existing security group

We'll create a new security group called '**launch-wizard-302**' with the following rules:

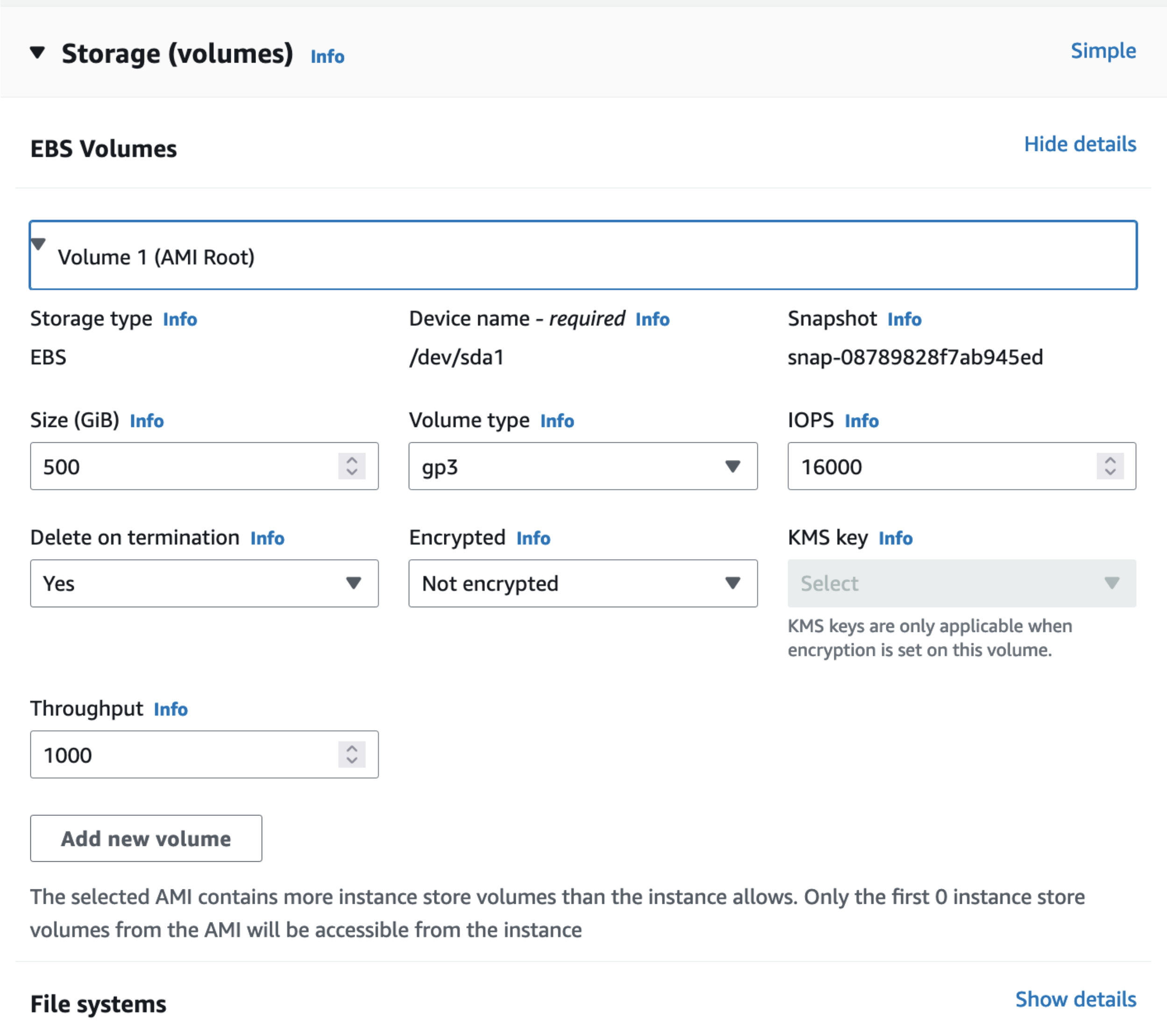
☑ Allow SSH traffic from — Anywhere 0.0.0.0/0
Helps you connect to your instance

□ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

□ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

9. Review the storage options. The AMI is preconfigured to use 500 GiB of gp3 as the root volume (not encrypted).



▼ Storage (volumes) Info — Simple

EBS Volumes — Hide details

▼ Volume 1 (AMI Root)

Storage type: EBS

Device name - *required*: /dev/sda1

Snapshot: snap-08789828f7ab945ed

Size (GiB): 500

Volume type: gp3

IOPS: 16000

Delete on termination: Yes

Encrypted: Not encrypted

KMS key: Select

KMS keys are only applicable when encryption is set on this volume.

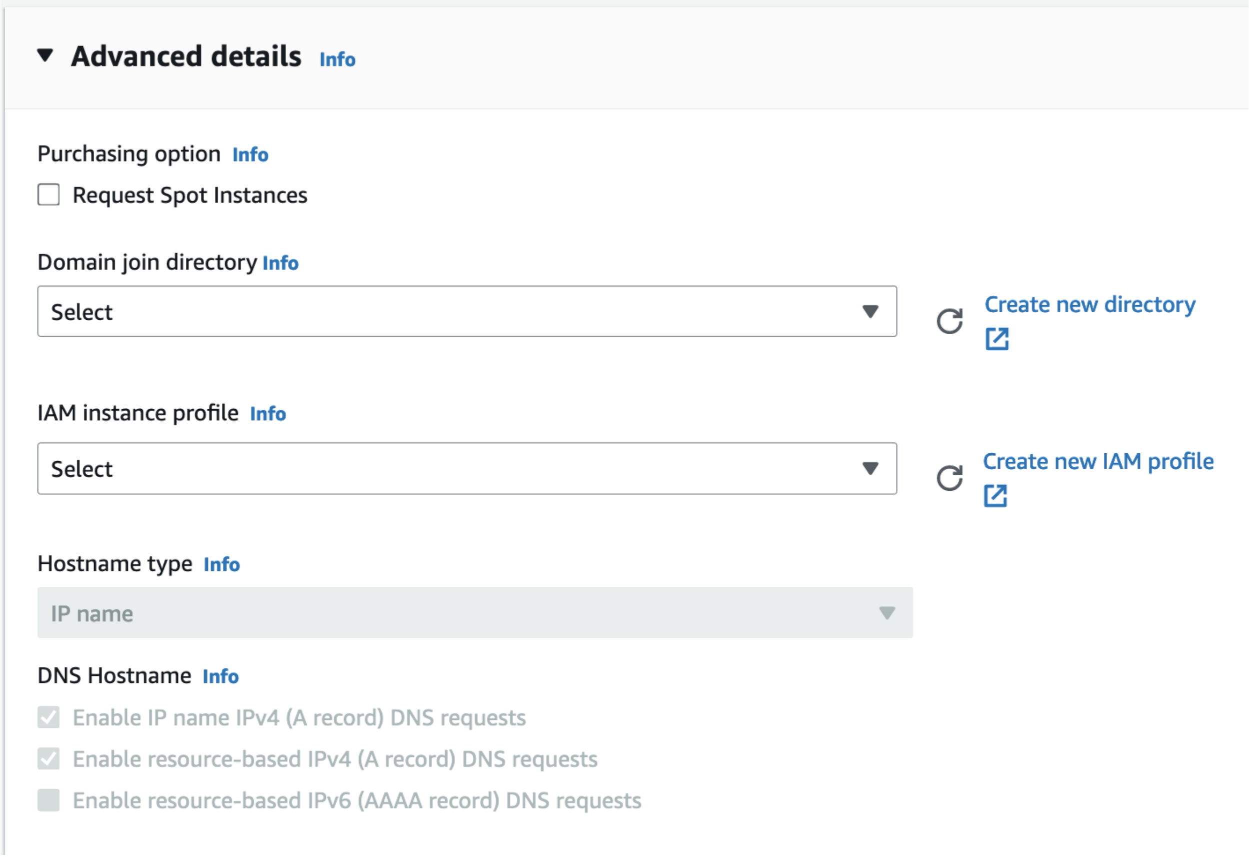
Throughput: 1000

Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

File systems — Show details

10. Select the pull-down options for Advanced details.



▼ Advanced details Info

Purchasing option Info

□ Request Spot Instances

Domain join directory Info

Select ▼

Create new directory

IAM instance profile Info

Select ▼

Create new IAM profile

Hostname type Info

IP name ▼

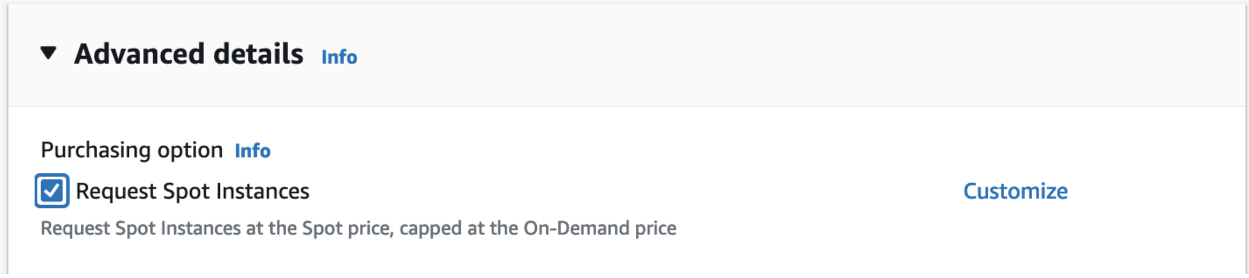
DNS Hostname Info

☑ Enable IP name IPv4 (A record) DNS requests

☑ Enable resource-based IPv4 (A record) DNS requests

□ Enable resource-based IPv6 (AAAA record) DNS requests

11. Select checkbox for Request Spot Instances.



▼ Advanced details Info

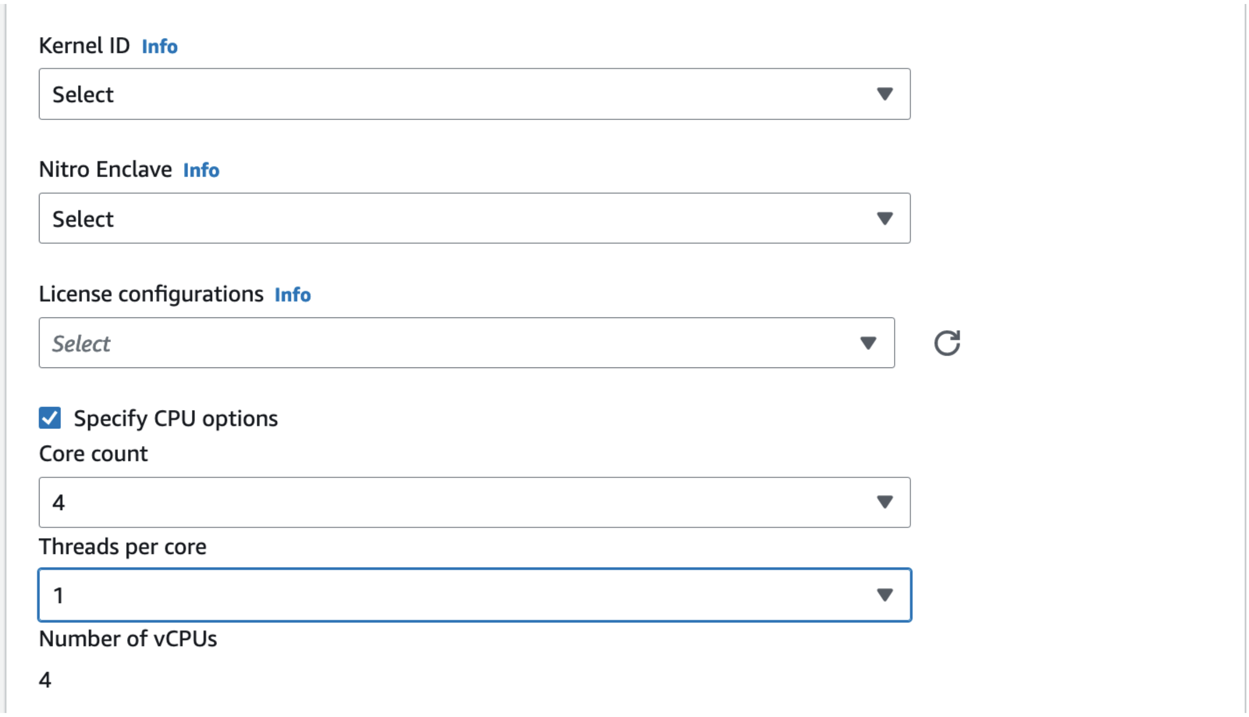
Purchasing option Info

☑ Request Spot Instances

Customize

Request Spot Instances at the Spot price, capped at the On-Demand price

12. Scroll down until you see option to Specify CPU cores.
13. Click the checkbox for "Specify CPU cores".
14. Then select 4 Cores, and 1 thread per core.



Kernel ID

Nitro Enclave

License configurations

☑ Specify CPU options

Core count

4

Threads per core

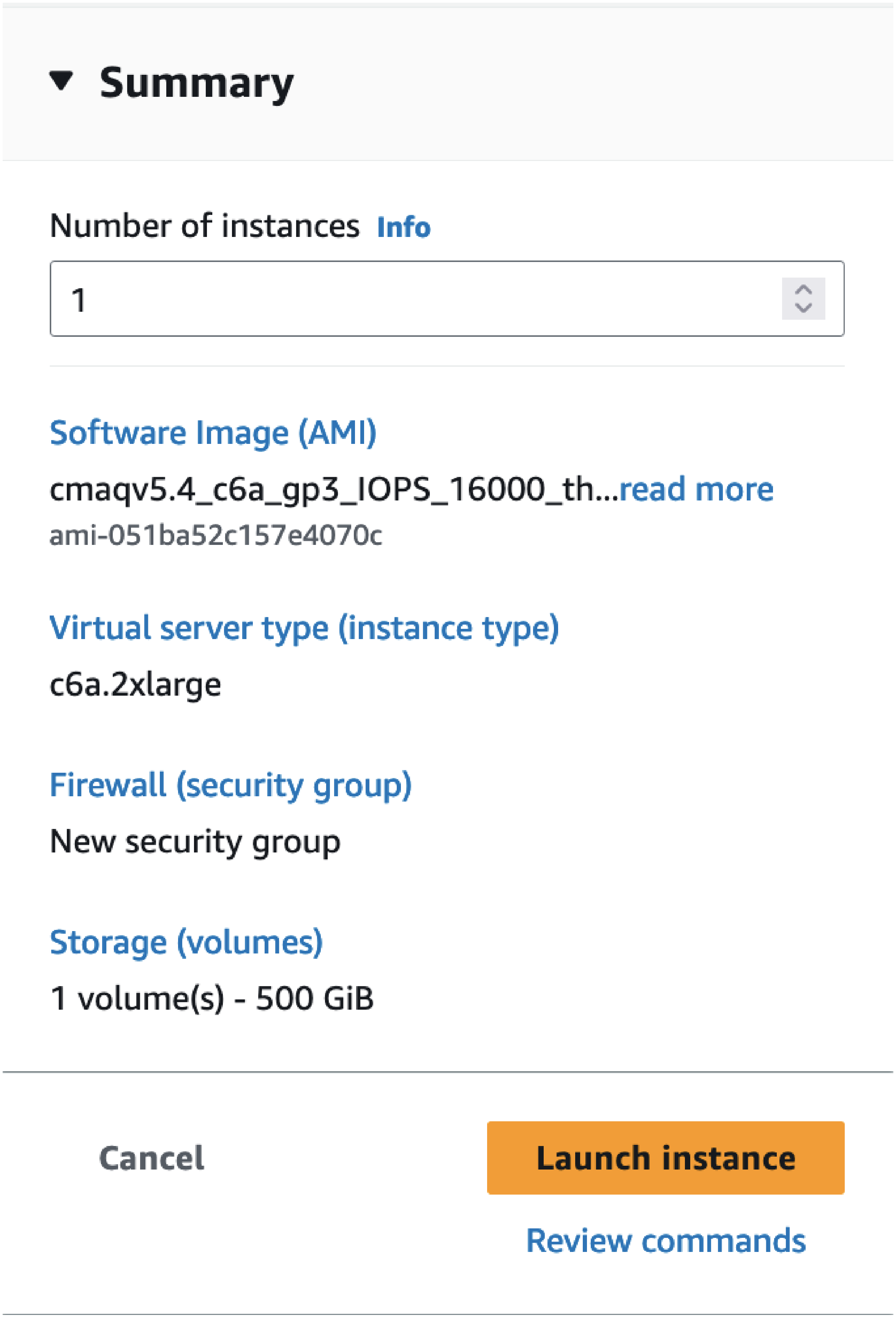
1

Number of vCPUs

4

If you are building a VM using a different instance type, just select 1 thread per core and leave the number of cores to the value that is pre-set. c6a.2xlarge (4 Cores), c6a.8xlarge (16 cores), c6a.48x large (96 cores).

15. In the Summary Menu, select Launch Instance.



▼ Summary

Number of instances Info

1

Software Image (AMI)

cmaqv5.4_c6a_gp3_IOPS_16000_th...read more

ami-051ba52c157e4070c

Virtual server type (instance type)

c6a.2xlarge

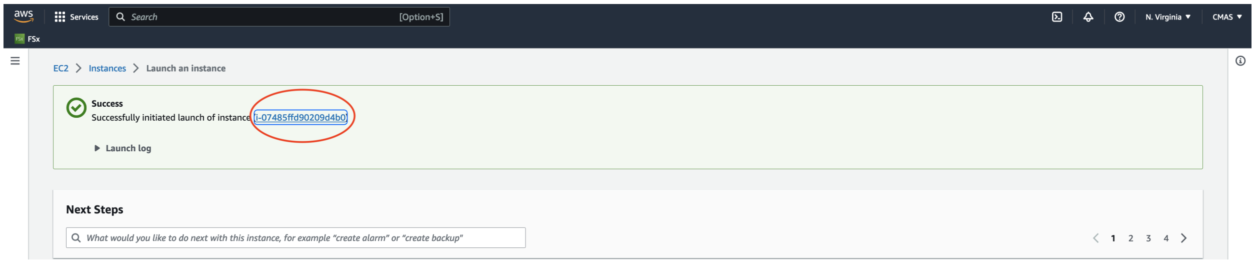
Firewall (security group)

New security group

Storage (volumes)

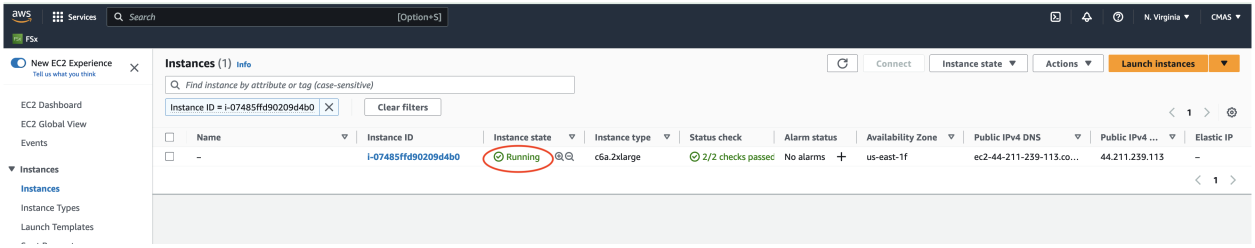
1 volume(s) - 500 GiB

16. Click on the link to the instance once it is successfully launched.


aws
Services
Search
[Option+S]
N. Virginia
CMAS
FSx
EC2 > Instances > Launch an instance
Launch log
Next Steps
What would you like to do next with this instance, for example "create alarm" or "create backup"
1 2 3 4

Success
Successfully initiated launch of instance i-07485ffd90209d4b0

17. Wait until the Status check has been completed and the Instance State is running

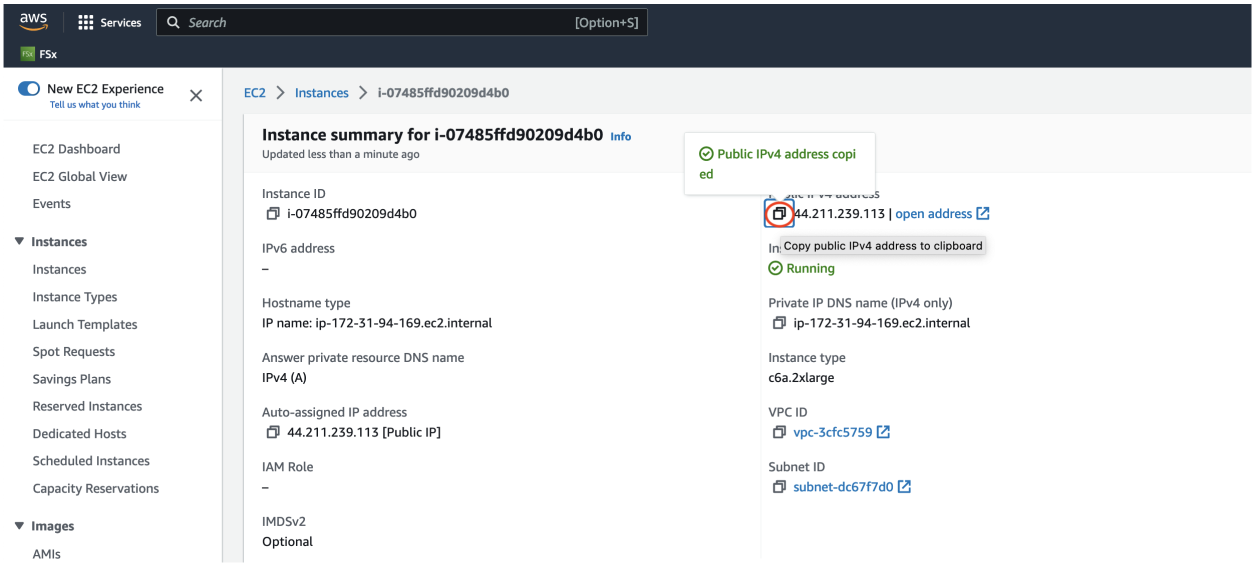


Instances (1) Info

Instance ID = i-07485ffd90209d4b0

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
–	i-07485ffd90209d4b0	Running	c6a.2xlarge	2/2 checks passed	No alarms	us-east-1f	ec2-44-211-239-113.co...	44.211.239.113	–

18. Click on the instance link and copy the Public IP address to your clipboard.



Instance summary for i-07485ffd90209d4b0 Info

Updated less than a minute ago

Public IPv4 address copied

Instance ID
i-07485ffd90209d4b0

Public IPv4 address
44.211.239.113 | open address

IPv6 address
–

Instance state
Running

Hostname type
IP name: ip-172-31-94-169.ec2.internal

Private IP DNS name (IPv4 only)
ip-172-31-94-169.ec2.internal

Answer private resource DNS name
IPv4 (A)

Instance type
c6a.2xlarge

Auto-assigned IP address
44.211.239.113 [Public IP]

VPC ID
vpc-3cfc5759

IAM Role
–

Subnet ID
subnet-dc67f7d0

IMDSv2
Optional

19. You will use this Public IP address to login into the VM that you just created (c6a.2xlarge ec2 instance).

On your local computer, you will use the following command.

```
ssh -v -Y -i ~/downloads/your-pem.pem ubuntu@xx.xxx.xxx.xxx
```

4.1.2 Create a VM using the AWS Command Line

If you are not able to use the AWS Web Interface to create the VM from the public AMI in the previous section, then you can use the AWS Command Line (CLI).

1. Install the AQS CLI on your local computer using the following instructions: Install AWS CLI
2. Create your key pair and security group Create key pair and security group
3. Verify that you can see the public AMI on the us-east-1 region.

```
aws ec2 describe-images --region us-east-1 --image-id ami-051ba52c157e4070c
```

Output:

```
{
    "Images": [
        {
            "Architecture": "x86_64",
            "CreationDate": "2023-07-05T14:10:42.000Z",
            "ImageId": "ami-051ba52c157e4070c",
            "ImageLocation": "440858712842/cmaqv5.4_c6a_gp3_IOPS_16000_throughput_1000",
            "ImageType": "machine",
            "Public": true,
            "OwnerId": "440858712842",
            "PlatformDetails": "Linux/UNIX",
            "UsageOperation": "RunInstances",
            "State": "available",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "DeleteOnTermination": true,
                        "Iops": 16000,
                        "SnapshotId": "snap-08789828f7ab945ed",
                        "VolumeSize": 500,
                        "VolumeType": "gp3",
                        "Throughput": 1000,
                        "Encrypted": false
                    }
                },
                {
                    "DeviceName": "/dev/sdb",
                    "VirtualName": "ephemeral0"
                },
                {
                    "DeviceName": "/dev/sdc",
                    "VirtualName": "ephemeral1"
                }
            ],
            "Description": "[Copied ami-01605a204650ede2f from us-east-1] cmaqv5.4_c6a_
↪48xlarge_gp3_IOPS_16000_throughput_1000",
            "EnaSupport": true,
            "Hypervisor": "xen",
            "Name": "cmaqv5.4_c6a_gp3_IOPS_16000_throughput_1000",
            "RootDeviceName": "/dev/sda1",
```

(continues on next page)

(continued from previous page)

```
            "RootDeviceType": "ebs",
            "SriovNetSupport": "simple",
            "VirtualizationType": "hvm",
            "DeprecationTime": "2025-07-05T14:10:42.000Z"
        }
    ]
}
```

3. Use q to exit out of the command line. Note, the AMI uses the default values of iops and throughput for the gp3 volume.

4. To use the AWS CLI, you will need to have a key.pair that was created on an EC2 instance.

See also:

Guide to obtaining AWS Key Pair

5. To launch a Spot Instance with RunInstances API create the configuration file as described below:

```
cat <<EoF > ./runinstances-config.gp3.json
{
    "DryRun": false,
    "MaxCount": 1,
    "MinCount": 1,
    "InstanceType": "c6a.2xlarge",
    "ImageId": "ami-051ba52c157e4070c",
    "InstanceMarketOptions": {
        "MarketType": "spot"
    },
    "TagSpecifications": [
        {
            "ResourceType": "instance",
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "EC2SpotCMAQv54"
                }
            ]
        }
    ]
}
EoF
```

6. Use the publically available AMI to launch a spot c6a.2xlarge EC2 instance using a gp3 volume with hyperthreading disabled. Specify the number of cores and set the number of threads per core to 1 to disable hyperthreading. Use the command line option to specify the number of cores to match the selected EC2 instance type, and to disable hyperthreading. Below is an example command.

```
aws ec2 run-instances --debug --key-name cmaqv5.4 --security-group-ids launch-wizard-179
--region us-east-1 --dry-run --ebs-optimized --cpu-options CoreCount=4,ThreadsPerCore=1
--cli-input-json file://runinstances-config.gp3.json
```

Note:

- The size of instance determines the number of compute cores (CoreCount). In the example above the c6a.2xlarge EC2 instance contains 4 cores with hyperthreading turned off and is sized to run the tutorial benchmark case (i.e., *–cpu-options CoreCount=XX, ThreadsPerCore=1*). If you wish to try the benchmark with a larger VM you can edit the runinstances-config.gp3.json file to select a different version of c6a and then change the CoreCount to match:

```
c6a.2xlarge, CoreCount=4
c6a.8xlarge, CoreCount=16
c6a.48xlarge, CoreCount=96
```

- You will need to obtain a security group id from your IT administrator that allows ssh login access. If this is enabled by default, then you can remove the –security-group-ids launch-wizard-with-tcp-access.
- Launch-wizard-with-tcp-access needs to be replaced by your security group ID, and your-pem key needs to be replaced by the name of your-pem.pem key.

7. Once you have verified that the command above works with the –dry-run option, rerun it after removing the –dry-run option as follows:

```
aws ec2 run-instances --debug --key-name cmaqv5.4 --security-group-ids launch-wizard-179
--region us-east-1 --ebs-optimized --cpu-options CoreCount=4,ThreadsPerCore=1
--cli-input-json file://runinstances-config.gp3.json
```

8. Use the q command to return to the cursor.
9. Use the following command to obtain the public IP address of the machine. Also use this command to verify that it has switched from an initializing state to a running state.

```
aws ec2 describe-instances --region=us-east-1 --filters "Name=image-id,
Values=ami-051ba52c157e4070c" | grep -A 3 PublicIpAddress
```

4.1.3 Run CMAQv5.4 on c6a.2xlarge

In the last sections you created and logged into a VM (c6a.2xlarge EC2 instance) based on the public AMI. Here you will use this VM to run a benchmark case for CMAQ version 5.4.

1. Obtain the IP address for the VM from AWS Web Console or from using the following AWS CLI command (same steps as the end of section 1.1 and 1.2):

```
aws ec2 describe-instances --region=us-east-1 --filters "Name=image-id,
Values=ami-051ba52c157e4070c" | grep PublicIpAddress
```

2. Use the IP address and your key pair to login to the EC2 instance.

```
ssh -v -Y -i ~/downloads/your-pem.pem ubuntu@ip.address
```

3. Login to the EC2 instance again, so that you have two windows logged into the machine.

```
ssh -Y -i ~/downloads/your-pem.pem ubuntu@your-ip-address
```

4. Load the environment modules

```
module avail
```

```
module load ioapi-3.2/gcc-11.3.0-netcdf mpi/openmpi-4.1.2 netcdf-4.8.1/gcc-11.3
```

5. Verify that the input data for the benchmark is available. The benchmark case (12US1_LISTOS) for this example is small with only 25 rows and 25 columns. The GRIDDESC file defines the modeling domain which has 12 km x 12km horizontal grid spacing and is centered over Long Island, New York and Connecticut.

```
ls -lrt /shared/data/12US1_LISTOS/*
```

```
-rw-rw-r-- 1 ubuntu ubuntu  207 Jun  6 20:05 /shared/data/12US1_LISTOS/GRIDDESC

/shared/data/12US1_LISTOS/emis:
total 28
drwxrwxr-x 2 ubuntu ubuntu 4096 Jun  6 20:19 cmv_c3
drwxrwxr-x 2 ubuntu ubuntu 4096 Jun  6 20:19 pt_oilgas
drwxrwxr-x 2 ubuntu ubuntu 4096 Jun  6 20:19 gridded
drwxrwxr-x 2 ubuntu ubuntu 4096 Jun  6 20:19 ptegu_nopfas
drwxrwxr-x 2 ubuntu ubuntu 4096 Jun  6 20:19 ptnonipm
drwxrwxr-x 2 ubuntu ubuntu 4096 Jun  6 20:19 smk_dates
drwxrwxr-x 2 ubuntu ubuntu 4096 Jun  6 20:19 ptnonipm_nopfas

/shared/data/12US1_LISTOS/met:
total 12
drwxrwxr-x 2 ubuntu ubuntu 4096 Jun  6 20:19 lightning
drwxrwxr-x 2 ubuntu ubuntu 4096 Jun  6 20:19 mcip
drwxrwxr-x 2 ubuntu ubuntu 4096 Jun  6 20:19 wrfout

/shared/data/12US1_LISTOS/icbc:
total 24976
-rw-rw-r-- 1 ubuntu ubuntu 21774044 Jun  6 20:05 ICON_v54_12km_Listos_profile_timeind.nc
-rw-rw-r-- 1 ubuntu ubuntu   109793 Jun  6 20:05 BCON_v54_12km_Listos_profile_timeind.nc.
↪CO.txt
-rw-rw-r-- 1 ubuntu ubuntu  3683924 Jun  6 20:05 BCON_v54_12km_Listos_profile_timeind.nc
drwxrwxr-x 2 ubuntu ubuntu     4096 Jun  6 20:19 cb6r3_ae7_aq

/shared/data/12US1_LISTOS/surface:
total 2668
-rw-rw-r-- 1 ubuntu ubuntu 2199208 Jun  6 20:05 OCEAN_08_L3m_MC_CHL_chlor_a_12US1_Listos.
↪nc3
-rw-rw-r-- 1 ubuntu ubuntu  363296 Jun  6 20:05 OCEAN_08_L3m_MC_CHL_chlor_a_12US1.nc
-rw-rw-r-- 1 ubuntu ubuntu  145796 Jun  6 20:05 GRIDMASK_STATES_12US1_m3clple_12listos.
↪ncf
-rw-rw-r-- 1 ubuntu ubuntu   16452 Jun  6 20:05 12US1_surf_m3clple_12listos.ncf
```

cat /shared/data/12US1_LISTOS/GRIDDESC

```
GRIDDESC

'2018_12Listos'
'LamCon_40N_97W'   1812000.000    240000.000     12000.000     12000.000   25   25    1
```

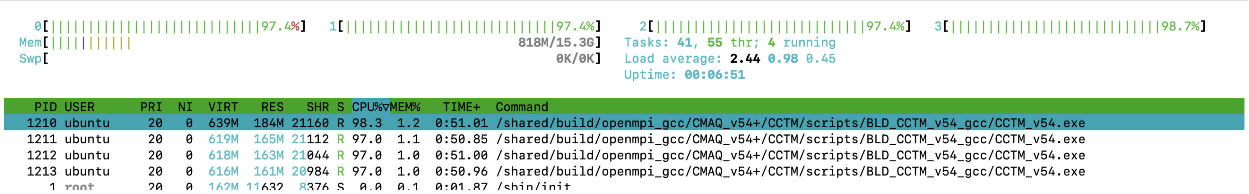
6. Run the CMAQv5.4 12US1_LISTOS benchmark case for 3 days on 4 processors. There is no job scheduler (such as SLURM) installed on the AMI. Submit the job using the command line:

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts
./run_cctm_2018_12US1_listos.csh | & tee ./run_cctm_2018_12US1_listos.c6a.2xlarge.log
```

7. Use HTOP to view performance.

htop

Output:



```
    0[||||||||||||||||||||||||||||97.4%]   1[||||||||||||||||||||||||||||97.4%]     2[||||||||||||||||||||||||||||97.4%]   3[||||||||||||||||||||||||||||98.7%]
  Mem[|||||||||||                                                 818M/15.3G]   Tasks: 41, 55 thr; 4 running
  Swp[                                                                0K/0K]   Load average: 2.44 0.98 0.45
                                                                                 Uptime: 00:06:51

    PID USER       PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
   1210 ubuntu      20   0  639M  184M 21160 R 98.3  1.2  0:51.01 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54_gcc/CCTM_v54.exe
   1211 ubuntu      20   0  619M  165M 21112 R 97.0  1.1  0:50.85 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54_gcc/CCTM_v54.exe
   1212 ubuntu      20   0  618M  163M 21044 R 97.0  1.0  0:51.00 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54_gcc/CCTM_v54.exe
   1213 ubuntu      20   0  616M  161M 20984 R 97.0  1.0  0:50.96 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54_gcc/CCTM_v54.exe
      1 root        20   0  162M 11632  8376 S  0.0  0.1  0:01.87 /sbin/init
```

8. After the benchmark is complete, use the following command to view the timing results.

```
tail -n 20 run_cctm_2018_12US1_listos.c6a.2xlarge.log
```

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2018-08-05
End Day:   2018-08-07
Number of Simulation Days: 3
Domain Name:               2018_12Listos
Number of Grid Cells:      21875  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       4
   All times are in seconds.

Num  Day        Wall Time
01   2018-08-05   165.5
02   2018-08-06   165.8
03   2018-08-07   169.5
     Total Time = 500.80
      Avg. Time = 166.93
```

9. Use lscpu to view number of cores. Confirm that there are 4 cores on the c6a.2xlarge ec2 instance that was created with hyperthreading turned off (1 thread per core). If the EC2 instance is configured to use 1 thread per core in the advanced setting, then it will have 4 cores. For MPI or parallel applications such as CMAQ it is best to turn off hyperthreading.

```
lscpu
```

Output:

```
lscpu
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         48 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  4
  On-line CPU(s) list:   0-3
Vendor ID:               AuthenticAMD
  Model name:            AMD EPYC 7R13 Processor
    CPU family:          25
    Model:               1
    Thread(s) per core:  1
    Core(s) per socket:  4
    Socket(s):           1
    Stepping:            1
    BogoMIPS:            5299.98
```

(continues on next page)

(continued from previous page)

```
Virtualization features:
  Hypervisor vendor:     KVM
  Virtualization type:   full
Caches (sum of all):
  L1d:                   128 KiB (4 instances)
  L1i:                   128 KiB (4 instances)
  L2:                    2 MiB (4 instances)
  L3:                    16 MiB (1 instance)
NUMA:
  NUMA node(s):          1
  NUMA node0 CPU(s):     0-3
```

Note: If the run time seems to take a while at the beginning of each day, then you may need to resubmit the job. There is an initial latency issue when storage blocks are initially pulled down from Amazon S3 and written to the volume. For the 12US1 or other large benchmarks with larger input file sizes, this latency or delay is longer.

You will need to use a larger EC2 instance to run the larger '12US1' benchmark, and also follow instructions available on how to initialize the volume prior to running: Initialize EBS Volume.

10. Once you have successfully run the benchmark, terminate the instance. Terminate the c6a.2xlarge either thru the Web Console or using the CLI. Find the InstanceID using the following command on your local machine.

```
aws ec2 describe-instances --region=us-east-1 | grep InstanceId
```

Output

i-xxxx

```
aws ec2 terminate-instances --region=us-east-1 --instance-ids i-xxxx
```

11. Verify that the instance is being shut down.

```
aws ec2 describe-instances --region=us-east-1
```

12. The cost to run the simulation per day is a factor of the simulation time and the cost per hour of the instance type (c6a.2xlarge on-demand cost is .306)

166.93 seconds x 1 min/60seconds x 1 hour/60 min = .0459 hours

.0459 hours x $.306/hour = $.014 per simulation day

4.2 Create a Parallel Cluster and run CMAQv5.4

Why might I need to use ParallelCluster?

The AWS ParallelCluster may be configured to be the equivalent of a High Performance Computing (HPC) environment, including using job schedulers such as Slurm, running on multiple nodes using code compiled with Message Passing Interface (MPI), and reading and writing output to a high performance, low latency shared disk. The advantage of using the AWS ParallelCluster command line interface is that the compute nodes can be easily scaled up or down to match the compute requirements of a given simulation. HPC compute nodes such as hpc6a or hpc7g are available in a limited set of regions at significantly discounted pricing (60% below on demand costs). Users can also attempt to reduce costs by using Spot instances rather than On-Demand for the compute nodes. ParallelCluster also supports submitting multiple jobs to the job submission queue.

Our goal is make this user guide to running CMAQ on a ParallelCluster as helpful and user-friendly as possible. Any feedback is both welcome and appreciated.

4.2.1 Build a Demo ParallelCluster

Step by Step Instructions to Build a Demo ParallelCluster.

Establish Identity and Permissions

AWS Identity and Access Management Roles Requires the user to have AWS Identity and Access Management roles in AWS ParallelCluster

See also:

AWS Identity and Access Management roles in AWS ParallelCluster

AWS ParallelCluster uses multiple AWS services to deploy and operate a cluster. See the complete list in the AWS Services used in AWS ParallelCluster section. It appears you can create the demo cluster, and even the intermediate or advanced cluster, but you can't submit a slurm job and have it provision compute nodes until you have the IAM Policies set for your account. This likely requires the system administrator who has permissions to access the AWS Web Interface with root access to add these policies and then to attach them to each user account.

Use the AWS Web Interface to add a policy called AWSEC2SpotServiceRolePolicy to the account prior to running a job that uses spot pricing on the ParallelCluster.

Install Parallel Cluster AWS CLI 3.0

Use Parallel Cluster AWS Command Line Interface (CLI) v3.0 to configure and launch a demo cluster

Requires the user to have a key.pair that was created on an ec2.instance

See also:

Guide to obtaining AWS Key Pair

Install AWS ParallelCluster Command Line Interface on your local machine

Create a virtual environment on a linux machine to install aws-parallel cluster

See also:

Guide to install AWS CL3 in Virtual Environment"

```
python3 -m virtualenv ~/apc-ve
source ~/apc-ve/bin/activate
python --version
python3 -m pip install --upgrade aws-parallelcluster
pcluster version
```

Install node.js

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh
chmod ug+x ~/.nvm/nvm.sh
source ~/.nvm/nvm.sh
nvm install node
node --version
```

Verify that AWS ParallelCluster is installed on local machine

Run pcluster version.

```
pcluster version
```

Output:

```
{
"version": "3.1.2"
}
```

Note: If you start a new terminal window, you need to re-activate the virtual environment using the following commands:

```
source ~/apc-ve/bin/activate
source ~/.nvm/nvm.sh
```

Verify that the parallel cluster is working using:

`pcluster version`

Configure AWS Command line credentials on your local machine

See also:

Link to Instructions for Setting up AWS Credential Instructions

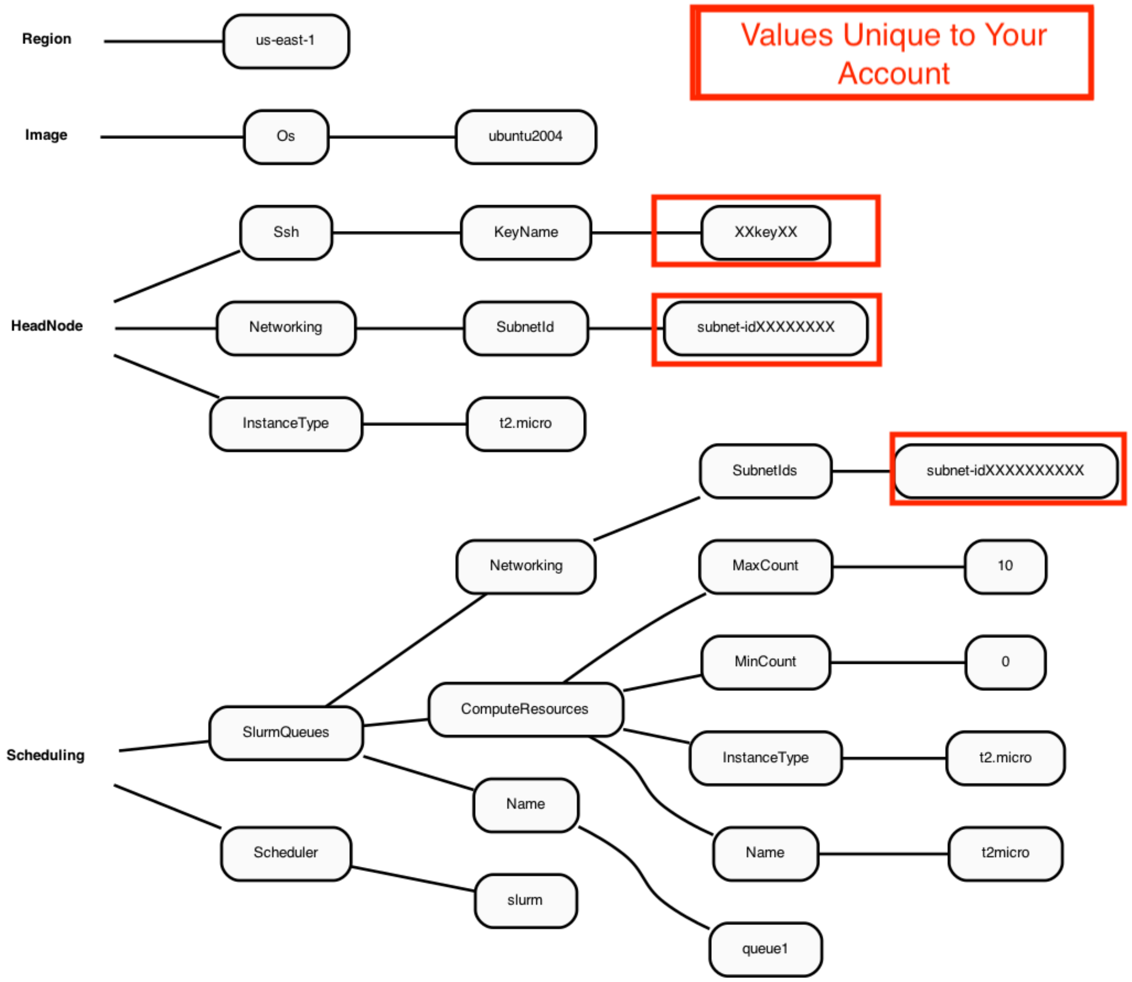
`aws configure`

Configure a Demo Cluster

To create a parallel cluster, a yaml file needs to be created that is unique to your account.

An example of the yaml file contents is described in the following Diagram:

Figure 1. Diagram of YAML file used to configure a ParallelCluster with a t2.micro head node and t2.micro compute nodes



See also:

Cluster Configuration File

Create a yaml configuration file for the cluster following these instructions

See also:

Link to ParallelCluster Configure Instructions

`pcluster configure --config new-hello-world.yaml`

Input the following answers at each prompt:

1. Allowed values for AWS Region ID: `us-east-1`
2. Allowed values for EC2 Key Pair Name: `choose your key pair`
3. Allowed values for Scheduler: `slurm`
4. Allowed values for Operating System: `ubuntu2004`
5. Head node instance type: `t2.micro`
6. Number of queues: `1`

7. Name of queue 1: `queue1`
8. Number of compute resources for queue1 [1]: `1`
9. Compute instance type for compute resource 1 in queue1: `t2.micro`
10. Maximum instance count [10]: `10`
11. Automate VPC creation?: `y`
12. Allowed values for Availability Zone: `1`
13. Allowed values for Network Configuration: `2. Head node and compute fleet in the same public subnet`

Beginning VPC creation. Please do not leave the terminal until the creation is finalized

Note: The choice of operating system (specified during the yaml creation, or in an existing yaml file) determines what modules and gcc compiler versions are available.

1. Centos7 has an older gcc version 4
2. Ubuntu2004 has gcc version 9+
3. Alinux or Amazon Linux/Red Hat Linux (haven't tried)

Examine the yaml file

`cat new-hello-world.yaml`

```
Region: us-east-1
Image:
  Os: ubuntu2004
HeadNode:
  InstanceType: t2.micro
  Networking:
    SubnetId: subnet-xx-xx-xx                  <<< unique to your account
  Ssh:
    KeyName: your-key                          <<< unique to your account
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: queue1
    ComputeResources:
    - Name: t2micro
      InstanceType: t2.micro
      MinCount: 0
      MaxCount: 10
    Networking:
      SubnetIds:
      - subnet-xx-xx-xx                        <<< unique to your account
```

Note: The above yaml file is the very simplest form available. If you upgrade the compute node to using a faster compute instance, then you will need to add additional configuration options (networking, elastic fabric adapter) to the yaml file. These modifications will be highlighted in the yaml figures provided in the tutorial.

The key pair and Subnetid in the yaml file are unique to your account. To create the AWS Intermediate ParallelCluster, the key pair and subnet ID from the new-hello-world.yaml file that you created using your account will need to be transferred to the Yaml files that will be used to create the Intermediate ParallelCluster in the next section of the tutorial. You will need to edit these yaml files to use the key pair and your Subnetid that are valid for your AWS Account.

Create a Demo Cluster

```
pcluster create-cluster --cluster-configuration new-hello-world.yaml --cluster-name
hello-pcluster --region us-east-1
```

Check on the status of the cluster

```
pcluster describe-cluster --region=us-east-1 --cluster-name hello-pcluster
```

List available clusters

```
pcluster list-clusters --region=us-east-1
```

Check on status of cluster again

```
pcluster describe-cluster --region=us-east-1 --cluster-name hello-pcluster
```

After 5-10 minutes, you see the following status: “clusterStatus”: “CREATE_COMPLETE”

While the cluster has been created, only the t2.micro head node is running. Before any jobs can be submitted to the slurm queue, the compute nodes need to be started.

Note: The compute nodes are not “provisioned” or “created” at this time (so they do not begin to incur costs). The compute nodes are only provisioned when a slurm job is scheduled. After a slurm job is completed, then the compute nodes will be terminated after 5 minutes of idletime.

Login to the ParallelCluster

Note: replace the your-key.pem key pair with your key pair you will need to change the permissions on your key pair so to be read only by owner.

```
cd ~
chmod 400 your-key.pem
```

Example: pcluster ssh -v -Y -i ~/your-key.pem –cluster-name hello-pcluster

```
pcluster ssh -v -Y -i ~/[your-key-pair] --cluster-name hello-pcluster
```

login prompt should look something like (this will depend on what OS was chosen in the yaml file).

[ip-xx-x-xx-xxx pcluster-cmaq]

Check what modules are available on the ParallelCluster

```
module avail
```

Check what version of the compiler is available

```
gcc --version
```

Need a minimum of gcc 8+ for CMAQ

Check what version of openmpi is available

```
mpirun --version
```

Need a minimum openmpi version 4.0.1 for CMAQ

Verify that Slurm is available (if slurm is not available, then you may need to try a different OS)

```
which sbatch
```

Do not install sofware on this demo cluster

the t2.micro head node is too small

Save the key pair and SubnetId from this new-hello-world.yaml to use in the yaml for the Intermediate Tutorial

Exit the cluster

```
exit
```

Delete the Demo Cluster

```
pcluster delete-cluster --cluster-name hello-pcluster --region us-east-1
```

See also:

```
pcluster --help
```

4.2.2 Use ParallelCluster with Software and Data pre-installed on hpc7g.16xlarge

Step by step instructions to configuring and running a ParallelCluster for the CMAQ 12US1 benchmark

Notice

The CMAQ libraries were installed using the gcc compiler on c6g.large.

Configure the ParallelCluster

Use an existing yaml file from the git repo to create a ParallelCluster

```
cd /your/local/machine/install/path/
```

Use a configuration file from the github repo that was cloned to your local machine

```
git clone -b main https://github.com/CMASCenter/pcluster-cmaq.git pcluster-cmaq
```

```
cd pcluster-cmaq/yaml
```

Edit the hpc7g.16xlarge.ebs_unencrypted_installed_public_ubuntu2004.fsx_import.yaml

```
vi hpc7g.16xlarge.ebs_unencrypted_installed_public_ubuntu2004.fsx_import.yaml
```

Note:

1. the hpc7g.16xlarge*.yaml is configured to use ONDEMAND instance pricing for the compute nodes.
2. the hpc7g.16xlarge*.yaml is configured to the the hpc7g.16xlarge as the compute node for the compute-resource-1 queue, with up to 12 compute nodes, specified by MaxCount: 12.

3. the hpc7g.16xlarge*.yaml is configured to the the hpc7g.8xlarge as the compute node for the compute-resource-1 queue, with up to 7 compute nodes.
4. the hpc7g.16xlarge*.yaml is configured to disable multithreading (This option restricts the computing to CPUS rather than allowing the use of all virtual CPUS. (128 virtual cpus reduced to 64 cpus)
5. the hpc7g.16xlarge*.yaml is configured to enable the setting of a placement group to allow low inter-node latency
6. the hpc7g.16xlarge*.yaml is configured to enables the elastic fabric adapter
7. given this yaml configuration, the maximum number of PEs that could be used to run CMAQ is 64 cpus x 12 = 768, the max settings for NPCOL, NPROW is NPCOL = 24, NPROW = 32 or NPCOL=32, NPROW=24 in the CMAQ run script. Note: CMAQ will need to be benchmarked using the 12US1 to determine the optimal number of compute nodes to use.

Replace the key pair and subnet ID in the hpc7g.16xlarge*.yaml file with the values created when you configured the demo cluster

```
Region: us-east-1
Image:
  Os: ubuntu2004
HeadNode:
  InstanceType: c7g.large
  Networking:
    SubnetId: subnet-xx-xx-xx           << replace
  DisableSimultaneousMultithreading: true
  Ssh:
    KeyName: your_key                     << replace
  LocalStorage:
    RootVolume:
      Encrypted: true
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue1
      CapacityType: ONDEMAND
      Networking:
        SubnetIds:
          - subnet-xx-xx-x         x    << replace
        PlacementGroup:
          Enabled: true
      ComputeResources:
        - Name: compute-resource-1
          InstanceType: hpc7g.16xlarge
          MinCount: 0
          MaxCount: 12
          DisableSimultaneousMultithreading: true
          Efa:
            Enabled: true
            GdrSupport: false
        - Name: compute-resource-2
          InstanceType: hpc7g.8xlarge
          MinCount: 0
          MaxCount: 7
          DisableSimultaneousMultithreading: true
```

(continues on next page)

(continued from previous page)

```
          Efa:
            Enabled: true
            GdrSupport: false
SharedStorage:
  - MountDir: /shared
    Name: ebs-shared
    StorageType: Ebs
    EbsSettings:
      Encrypted: true
      SnapshotId: snap-0049a7c309f238500
  - MountDir: /fsx
    Name: name2
    StorageType: FsxLustre
    FsxLustreSettings:
      StorageCapacity: 1200
      ImportPath: s3://cmas-cmaq/
```

The Yaml file for the hpc7g.16xlarge contains the settings as shown in the following diagram.

Figure 1. Diagram of YAML file used to configure a ParallelCluster with a c6g.large head node and hpc7g.16xlarge compute nodes using ONDEMAND pricing

../yml_plots/hpc7g.16xlarge.png

(to do!)

Create the hpc7g pcluster

Note, this yaml file is configured to have 12 nodes of the hpc7g.16xlarge (64 pe per node) and 7 nodes of the hpc7g.8xlarge (32 pe per node).

```
pcluster create-cluster --cluster-configuration hpc7g.16xlarge.ebs_unencrypted_installed_public_ubuntu2
fsx_import.yaml --cluster-name cmaq --region us-east-1
```

Check on status of cluster

```
pcluster describe-cluster --region=us-east-1 --cluster-name cmaq
```

After 5-10 minutes, you see the following status: “clusterStatus”: “CREATE_COMPLETE”

If the cluster fails to start, use the following command to check for an error

```
pcluster get-cluster-stack-events --cluster-name cmaq --region us-east-1 --query
'events[?resourceStatus==CREATE_FAILED]'
```

4.2.3 Run CMAQ on hpc7g.16xlarge

Login to cluster

Note: Replace the your-key.pem with your Key Pair.

```
pcluster ssh -v -Y -i ~/your-key.pem --region=us-east-1 --cluster-name cmaq
```

Check to make sure elastic network adapter (ENA) is enabled

```
modinfo ena
lspci
```

Verify the gcc compiler version is greater than 8.0

```
gcc --version
```

output:

gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0 Copyright (C) 2019 Free Software Foundation, Inc. This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Change default shell to .tcsh

```
sudo usermod -s /bin/tcsh ubuntu
```

Copy file to .cshrc

```
cp /shared/pcluster-cmaq/install/dot.cshrc.pcluster ~/.cshrc
```

Note that the .cshrc to add custom module path

```
module use --append /shared/build/Modules/modulefiles
```

Change shell to csh

logout and log back in to switch to the default shell

Use module list and then module load to load the libraries

```
module load netcdf-4.8.1/gcc-9.5  ioapi-3.2/gcc-9.5-netcdf
```

Description of the hpc7g.16xlarge instance:

Instance Size Physical Cores Memory (GiB) Instance Storage EFA Network Bandwidth (Gbps) Network Bandwidth (Gbps)* hpc7g.16xlarge 64 128 EBS-only 200 25

Verify that you have an updated set of run scripts from the pcluster-cmaq repo

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts
ls -lrt  run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.2x64.ncclassic.csh
```

If they don't exist or are not identical, then copy the run scripts from the repo

```
cd /shared/pcluster-cmaq
git pull
cp /shared/pcluster-cmaq/run_scripts/hpc7g.16xlarge/run_cctm* /shared/build/openmpi_gcc/
↪CMAQ_v54+/CCTM/scripts/
```

Verify that the input data is imported to /fsx from the S3 Bucket

```
cd /fsx/
```

Preloading the files

Amazon FSx copies data from your Amazon S3 data repository when a file is first accessed. CMAQ is sensitive to latencies, so it is best to preload contents of individual files or directories using the following command:

```
nohup find /fsx/ -type f -print0 | xargs -0 -n 1 sudo lfs hsm_restore &
```

Create a directory that specifies the full path that the run scripts are expecting.

```
mkdir -p /fsx/data/CMAQ_Modeling_Platform_2018/
```

Link the 2018_12US1 directoy

```
cd /fsx/data/CMAQ_Modeling_Platform_2018/
ln -s /fsx/CMAQv5.4_2018_12US1_Benchmark_2Day_Input/2018_12US1/ .
```

Link the 12LISTOS_Training data

```
cd /fsx/data/
ln -s /fsx/CMAQv5.4_2018_12LISTOS_Benchmark_3Day_Input/12LISTOS_Training ./12US1_LISTOS
```

Link the 2018_12NE3 Benchmark data

```
ln -s /fsx/CMAQv5.4_2018_12NE3_Benchmark_2Day_Input/2018_12NE3 .
```

netCDF-3 classic input files are used

The *.nc4 compressed netCDF4 files on /fsx input directory were converted to netCDF classic (nc3) files

Create the output directory`

```
mkdir -p /fsx/data/output
```

Note, that the 12US1 Domain will not run on 64 cores using the hpc7g.16xlarge, as it doesn't have enough memory per node. It is possible to run on 64 cores using the hpc7g.8xlarge using 2 x 32 cores per node (as there is more memory per core).

Run the 12US1 Domain on 128 cores

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.2x64.ncclassic.csh
```

Note, it will take about 3-5 minutes for the compute notes to start up. This is reflected in the Status (ST) of CF (configuring)

Check the status in the queue

```
squeue -u ubuntu`
```

Output:

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 3    queue1     CMAQ   ubuntu  CF                2 queue1-dy-compute-
↪resource-1-[1-2]
```

After 5 minutes the status will change once the compute nodes have been created and the job is running

```
squeue -u ubuntu
```

Output:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 3    queue1     CMAQ   ubuntu  R       0:58      2 queue1-dy-compute-
↪resource-1-[1-2]
```

Check on the status of the cluster using CloudWatch (optional)

Cloudwatch Dashboard

Monitoring Dashboard for ParallelCluster

Check the timings while the job is still running using the following command

```
cd /fsx/data/output/output_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_2x64_classic/
```

```
grep 'Processing completed' CTM_LOG_001*
```

Output:

```
            Processing completed...       7.4020 seconds
            Processing completed...       5.5893 seconds
            Processing completed...       5.5588 seconds
            Processing completed...       5.5470 seconds
            Processing completed...       5.5449 seconds
            Processing completed...       5.5105 seconds
            Processing completed...       5.5182 seconds
            Processing completed...       5.5343 seconds
            Processing completed...       5.5482 seconds
```

When the job has completed, use tail to view the timing from the log file.

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
```

```
tail run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.128.8x16pe.2day.20171222start.2x64.log
```

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       128
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   2074.2
02   2017-12-23   2298.9
     Total Time = 4373.10
      Avg. Time = 2186.55
```

Check whether the scheduler thinks there are cpus or vcpus

`sinfo -lN`

Output:

```
Thu Jun 29 22:31:30 2023
NODELIST                         NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_
↪DISK WEIGHT AVAIL_FE REASON
queue1-dy-compute-resource-1-1       1   queue1*   allocated 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-2       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, Scheduler health che
queue1-dy-compute-resource-1-3       1   queue1*   allocated 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-4       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-5       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-6       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-7       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-8       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-9       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-10      1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-11      1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-12      1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
```

When the jobs are both submitted to the queue they will be dispatched to different compute nodes.

```
squeue
```

output

```
Submitted batch job 4
ip-10-0-1-243:/shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts> squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 4    queue1     CMAQ   ubuntu CF       0:01      1 queue1-dy-compute-
↪resource-1-3
                 3    queue1     CMAQ   ubuntu  R      21:28      2 queue1-dy-compute-
↪resource-1-[1-2]
```

Information about the error obtained when running on 1 node using hpc7g.16xlarge

1 pe job is dying, running out of memory, which means that the 12US1 case takes more than 128 GB of memory.

user_guide_pcluster/cmaqv54-cluster/top_just_before_1x64_hpc7g.16xlarge_dies.png

2 GB Memory per core for hpc7g.16xlarge

```
tail -n 30 run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.64.8x8pe.2day.
20171222start.1x64.log
```

Output

```
a non-zero exit code. Per user-direction, the job has been aborted.
--------------------------------------------------------------------------
--------------------------------------------------------------------------
mpirun noticed that process rank 12 with PID 6866 on node queue1-dy-compute-resource-1-1␣
↪exited on signal 9 (Killed).
--------------------------------------------------------------------------
11.857u 17.117s 1:24.37 34.3%	0+0k 382640+17960io 4860pf+0w

**************************************************************
** Runscript Detected an Error: CGRID file was not written. **
**   This indicates that CMAQ was interrupted or an issue   **
**   exists with writing output. The runscript will now     **
**   abort rather than proceeding to subsequent days.       **
**************************************************************


==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 1
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       64
   All times are in seconds.
```

(continues on next page)

(continued from previous page)

```

Num  Day        Wall Time
01   2017-12-22   12
     Total Time = 12.00
      Avg. Time = 12.00
```

```
tail -n 30 CTM_LOG_012.v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_1x64_classic_20171222
```

```
     File "OMI" opened for input on unit:  92
     /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/OMI_1979_to_2019.dat

OMI Ozone column data has Lat by Lon Resolution:      17X     17
     Total column ozone will be interpolated to day 0:00:00   Dec. 22, 2017
     from data available on the OMI input file
```

Switched to running on more than one node c7g.8xlarge, and CMAQv5.4 ran successfully as it had access to more memory.

When the job has completed, use tail to view the timing from the log file.

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
```

```
tail -n 30 run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.256.16x16pe.2day.
20171222start.4x64.log
```

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       256
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   1347.3
02   2017-12-23   1501.4
     Total Time = 2848.70
      Avg. Time = 1424.35
```

Submit a job to run on 192 pes, 3x64 nodes

`sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.3x64.ncclassic.csh`

Verify that it is running on 3 nodes

`sbatch`

output:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 5    queue1     CMAQ   ubuntu  R       4:29      3 queue1-dy-compute-
↪resource-1-[1-3]
```

Check the log for how quickly the job is running

`grep ‘Processing completed’

Output:

```

```

When the job has completed, use tail to view the timing from the log file.

`cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/`

`tail -n 30 run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.192.12x16pe.2day.20171222start.3x64.log`

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       192
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   1617.1
02   2017-12-23   1755.3
     Total Time = 3372.40
      Avg. Time = 1686.20
```

Submit a job to run on 320 pes running on 5 nodes

Output

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       320
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   1177.0
02   2017-12-23   1266.6
     Total Time = 2443.60
      Avg. Time = 1221.80

```

Submit a job to run on 128 cores with 32 cores per node.

Running on 4x32 cores using the hpc7g.8xlarge instances

```
sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.4x32.ncclassic.csh -w
queue1-dy-compute-resource-2[1-4]
```

4.2.4 Run DESID CMAQ on hpc7g.16xlarge

Note: This content was transferred from the CMAQ on AWS Workshop, and has not been fully tested.

Run CMAQ for DESID

Edit the DESID Namelist

1. **Edit the CMAQ DESID Chemical Species Control File**

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc
cp CMAQ_Control_DESID_cb6r5_ae7_aq.nml CMAQ_Control_DESID_cb6r5_ae7_aq_RED_EGU_POINT_NY.
↪nml
vi CMAQ_Control_DESID_cb6r5_ae7_aq_RED_EGU_POINT_NY.nml
```

2. **Add the following lines to the bottom of the file according to the DESID Tutorial Instructions**

https://github.com/USEPA/CMAQ/blob/main/DOCS/Users_Guide/Tutorials/CMAQ_UG_tutorial_emissions.md#scale_stream (place the line before the / file marker)

```
   ! PT_EGU Emissions Scaling reduce PT_EGU emissions in New York by 25%. Note, to␣
↪reduce the emissions by 25% we use DESID to multiply what had been 100% emissions by .
↪75, so that the resulting emissions is reduced by 25%.
   'NY'  , 'PT_EGU'      ,'All'    ,'All'         ,'All' ,.75    ,'UNIT','o',
```

3. **Activate DESID Diagnostics**

Create a DESID Control File and edit it to define NY as a region, and activate DESID emissions diagnostics Define NY as a region in the DESID Region Definitions

```
cp CMAQ_Control_DESID.nml CMAQ_Control_DESID_RED_EGU_POINT_NY.nml
vi  CMAQ_Control_DESID_RED_EGU_POINT_NY.nml &
```

Modify the following section to use the NY region that is specified in the CMAQ_MASKS file, note the CMAQ_MASKS file is defined in the DESID Run script.

```
&Desid_RegionDef
 Desid_Reg_nml  =
 !            Region Label   | File_Label  | Variable on File
 !              'EVERYWHERE'  ,'N/A'        ,'N/A',
               'NY'         ,'CMAQ_MASKS' ,'NY',
 !<Example>    'ALL'         ,'ISAM_REGIONS','ALL',
/
```

4. **Create two stream family definitions, one that includes all point source emissions, and the second that only contains PT_EGU**

```
!-------------------------------------------------------------------------------!
! Emissions Scaling Family Definitions                                          !
!    This component includes definitions for families of emission streams and   !
!    region combinations.                                                       !
!-------------------------------------------------------------------------------!
&Desid_StreamFamVars
 Desid_N_Stream_Fams = 2            ! Exact number of stream family definitions
 Desid_Max_Stream_Fam_Members = 20 ! Larger than the number of streams on all
                                   ! family definitions
/

&Desid_StreamFam
! For emission streams available in several run scripts under CCTM/scripts

  StreamFamilyName(1)     = 'PT_SOURCES'
  StreamFamilyMembers(1,1:8)= 'PT_NONEGU','PT_OTHER', 'PT_AGFIRES', 'PT_FIRES', 'PT_
↪RXFIRES', 'PT_OTHFIRES', 'PT_OILGAS','PT_CMV_C1C2'

  StreamFamilyName(2)     = 'PT_EGUS'
  StreamFamilyMembers(2,1:1)= 'PT_EGU'
&Desid_Diag
```

5. **activate DESID diagnostics to report the reduction in PT_EGU emissions.**

Note, if you define only one diagnostic rule, you must comment out all other rules.

```
&Desid_DiagVars
  Desid_N_Diag_Rules = 1     ! Exact Number of Diagnostic Rules Below
```

(continues on next page)

(continued from previous page)

```
  Desid_Max_Diag_Streams=20 ! Maximum number of species variables on all rules
                            ! below (do not count expansions)
  Desid_Max_Diag_Spec = 80  ! Maximum number of species variables on all rules
                            ! below (do not count expansions)
/
```

```
! Create a diagnostic of the sum of the components of the PT_SOURCES
 ! family (defined in the stream family section). This file will be column sums
 ! and will include all the emitted species as long as they appear on at least
 ! one of the streams within PT_SOURCES.



    Desid_Diag_Streams_Nml(1,:)= 'PT_EGUS'
    Desid_Diag_Fmt_Nml(1)      = 'COLSUM'
    Desid_Diag_Spec_Nml(1,:)   = 'ALL'
```

6. **Verify that the settings are correct by comparing to the version in the github repo directory**

```
diff CMAQ_Control_DESID_RED_EGU_POINT_NY.nml /shared/pcluster-cmaq/qa_scripts/workshop/
↪CMAQ_Control_DESID_RED_EGU_POINT_NY.nml
```

7. **Copy the Run script and edit it to use the DESID namelist files**

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc
cp run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.3x64.ncclassic.csh run_cctm_2018_12US1_v54_
↪cb6r5_ae6.20171222.3x64.ncclassic_DESID_RED_NY.csh
```

Edit runscript to use DESID Namelist

1. **Copy the Run script and edit it to use the DESID namelist files**

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc
cp run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.3x64.ncclassic.csh run_cctm_2018_12US1_v54_
↪cb6r5_ae6.20171222.3x64.ncclassic_DESID_RED_NY.csh
```

2. **Change APPL to a new name**

```
set APPL      = 12US1_DESID_REDUCE        #> Application Name (e.g. Gridname)
```

3. **Verify the following emission stream names match the names used in the DESID namelist.**

```
grep STK_EMIS_LAB_00 ../run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.3x64.ncclassic.DESID_
↪RED_NY.csh
```

Output

```
setenv STK_EMIS_LAB_001 PT_NONEGU
setenv STK_EMIS_LAB_002 PT_EGU
setenv STK_EMIS_LAB_003 PT_OTHER
setenv STK_EMIS_LAB_004 PT_AGFIRES
```

(continues on next page)

(continued from previous page)

```
setenv STK_EMIS_LAB_005 PT_FIRES
setenv STK_EMIS_LAB_006 PT_RXFIRES
setenv STK_EMIS_LAB_007 PT_OTHFIRES
setenv STK_EMIS_LAB_008 PT_OILGAS
setenv STK_EMIS_LAB_009 PT_CMV_C1C2
```

4. **Compare the above settings to those used in the Emission Stream Family defined in the DESID Namelist.**

```
grep -A 2 -B 2 StreamFamilyMembers CMAQ_Control_DESID_RED_EGU_POINT_NY.nml
```

Output

```
  StreamFamilyName(1)     = 'PT_SOURCES'
  StreamFamilyMembers(1,1:4)= 'PT_NONEGU','PT_OTHER', 'PT_AGFIRES', 'PT_FIRES', 'PT_
↪RXFIRES', 'PT_OTHFIRES', 'PT_OILGAS','PT_CMV_C1C2'

  StreamFamilyName(2)     = 'PT_EGUS'
  StreamFamilyMembers(2,1:1)= 'PT_EGU'
```

Note: CMAQ won't crash if the stream name in CMAQ_Control_DESID__RED_EGU_POINT_NY.nml was set incorrectly. CMAQ just ignores the incorrect stream name and won't apply scaling.

5. **Update the DESID namelist file names in the run script to use the Reduced PT_EGU and diagnostic instructions.**

```
cd  /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc
vi run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.3x64.ncclassic_DESID_RED_NY.csh
```

Modify the namelist setting to use the DESID namelist:

```
setenv DESID_CTRL_NML ${BLD}/CMAQ_Control_DESID_RED_EGU_POINT_NY.nml
setenv DESID_CHEM_CTRL_NML ${BLD}/CMAQ_Control_DESID_${MECH}_RED_EGU_POINT_NY.nml
```

6. **Update the Spatial Masks for Emissions Scaling to use a file that contains state definitions for New York.**

```
#> Spatial Masks For Emissions Scaling
  setenv CMAQ_MASKS $SZpath/GRIDMASK_STATES_12US1_m3clple_12listos.ncf
```

7. **Verify that the file contains New York**

```
ncdump /shared/build/GRIDMASK/GRIDMASK_STATES_12US1.nc | grep NY
```

Output

```
    float NY(TSTEP, LAY, ROW, COL) ;
        NY:long_name = "NY              " ;
        NY:units = "fraction        " ;
        NY:var_desc = "NY fractional area per grid cell
```

Run CMAQ using DESID

Note: The CMAQ run script has been configured to run on 192 cores (3 compute nodes of hpc7g with 64 cores/node)

1. **Change directories to the run script location**

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts
```

2. **Submit the Run script to the SLURM queue**

```
sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.3x64.ncclassic.DESID_RED_NY.csh
```

3. **Check the status of the job**

```
squeue
```

Output

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 1   compute     CMAQ  ec2-user CF       0:30      3 compute-dy-hpc7g-
↪[1-3]
```

Wait for the status to change from CF to R

4. **Login to the compute node, install and run htop**

```
ssh -Y compute-dy-hpc7g-1
sudo yum install -y htop
htop
```

static/images/2-run-cmaq-htop.png

Htop should show that 64 processes are running and that 80.2G out of 124 G of memory is being used. ~

Review Log file from DESID run

Note: The CMAQ run script has been configured to run on 192 cores (3 compute nodes of hpc7g with 64 cores/node)

1. **Review the Emissions Scaling Report Section in the CTM_LOG File to verify that for the NY region, the EGU emissions were scaled by 75%**

```
cd output_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_3x64_classic_DESID_REDUCE
grep -A 20 'Stream Type: "Point Emissions File   2' CTM_LOG_001*
```

Output:

```
    Stream Type: "Point Emissions File   2" | Sector Label: PT_EGU (04)

       Table of Aerosol Size Distributions Available for Use Sector-Wide.
       Note that Mode 1 is reserved for gas-phase species and emission variable.
         Number  Em. Var.  Mode  Reference Mode (see AERO_DATA.F)
         ------  --------------  --------------------------------
         2       FINE                        FINE_REF
         3       COARSE                    COARSE_REF

       CMAQ Species     Phase/Mode  Em. Var.          Region            Op ScaleFac␣
↪Basis FinalFac
       ------------     ----------  ---------         ------            -- -------- ---
↪-- --------
         NO2              GAS        NO2              EVERYWHERE        a   1.000   ␣
↪UNIT   1.000
                                                      NY                o   0.750   ␣
↪UNIT   0.750
         NO               GAS        NO               EVERYWHERE        a   1.000   ␣
↪UNIT   1.000
                                                      NY                o   0.750   ␣
↪UNIT   0.750
         HONO             GAS        HONO             EVERYWHERE        a   1.000   ␣
↪UNIT   1.000
                                                      NY                o   0.750   ␣
↪UNIT   0.750
         SO2              GAS        SO2              EVERYWHERE        a   1.000   ␣
↪UNIT   1.000
                                                      NY                o   0.750   ␣
↪UNIT   0.750
         SULF             GAS        SULF             EVERYWHERE        a   0.000   ␣
↪UNIT   0.000
                                                      NY                o   0.750   ␣
↪UNIT   0.750
```

4.2.5 Use ParallelCluster with Software and Data pre-installed for c6a.48xlarge

Step by step instructions to configuring and running a ParallelCluster for the CMAQ 12US1 benchmark

Notice

The CMAQ libraries were installed using the gcc compiler on c6a.large.

Create CMAQ Cluster using SPOT pricing

Use an existing yaml file from the git repo to create a ParallelCluster

`cd /your/local/machine/install/path/`

Use a configuration file from the github repo that was cloned to your local machine

`git clone -b main https://github.com/CMASCenter/pcluster-cmaq.git pcluster-cmaq`

Edit the c6a.large-48xlarge.ebs_unencrypted_installed_public_ubuntu2004.fsx_import.yaml

`cd pcluster-cmaq/yaml`

`vi c6a.large-48xlarge.ebs_unencrypted_installed_public_ubuntu2004.fsx_import.yaml`

Note:

1. the c6a.large-48xlarge*.yaml is configured to use SPOT instance pricing for the compute nodes.
2. the c6a.large-48xlarge*.yaml is configured to the the c6a-48xlarge as the compute node, with up to 10 compute nodes, specified by MaxCount: 10.
3. the c6a.large-48xlarge*.yaml is configured to disable multithreading (This option restricts the computing to CPUS rather than allowing the use of all virtual CPUS. (192 virtual cpus reduced to 96 cpus)
4. the c6a.large-48xlarge*.yaml is configured to enable the setting of a placement group to allow low inter-node latency
5. the c6a.large-48xlarge*.yaml is configured to enables the elastic fabric adapter
6. given this yaml configuration, the maximum number of PEs that could be used to run CMAQ is 96 cpus x 10 = 960, the max settings for NPCOL, NPROW is NPCOL = 24, NPROW = 40 or NPCOL=40, NPROW=24 in the CMAQ run script. Note: CMAQ does not scale well beyond 2-3 compute nodes.

Replace the key pair and subnet ID in the c6a.large-48xlarge*.yaml file with the values created when you configured the demo cluster

```
Region: us-east-1
Image:
  Os: ubuntu2004
HeadNode:
  InstanceType: c6a.large
  Networking:
    SubnetId: subnet-xx-xx-xx           << replace
  DisableSimultaneousMultithreading: true
  Ssh:
    KeyName: your_key                     << replace
  LocalStorage:
    RootVolume:
      Encrypted: true
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue1
      CapacityType: SPOT
      Networking:
        SubnetIds:
```

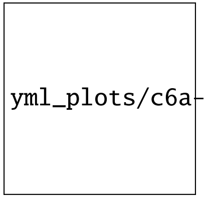
(continues on next page)

(continued from previous page)

```
          - subnet-xx-xx-x         x    << replace
        PlacementGroup:
          Enabled: true
      ComputeResources:
        - Name: compute-resource-1
          InstanceType: c6a.48xlarge
          MinCount: 0
          MaxCount: 10
          DisableSimultaneousMultithreading: true
          Efa:
            Enabled: true
            GdrSupport: false
SharedStorage:
  - MountDir: /shared
    Name: ebs-shared
    StorageType: Ebs
    EbsSettings:
      Encrypted: false
      SnapshotId: snap-05a36eeec1f5267bd
  - MountDir: /fsx
    Name: name2
    StorageType: FsxLustre
    FsxLustreSettings:
      StorageCapacity: 1200
      ImportPath: s3://cmas-cmaq/
```

The Yaml file for the c6a.large-48xlarge contains the settings as shown in the following diagram.

Figure 1. Diagram of YAML file used to configure a ParallelCluster with a c6a.large head node and c6a.48xlarge compute nodes using SPOT pricing



(to do!)

Create the c6a.48xlarge pcluster

`pcluster create-cluster --cluster-configuration c6a.large-48xlarge.ebs_unencrypted_installed_public_ubuntu2004.fsx_import.yaml --cluster-name cmaq --region us-east-1`

Check on status of cluster

`pcluster describe-cluster --region=us-east-1 --cluster-name cmaq`

After 5-10 minutes, you see the following status: “clusterStatus”: “CREATE_COMPLETE”

If the cluster fails to start, use the following command to check for an error

`pcluster get-cluster-stack-events --cluster-name cmaq --region us-east-1 --query 'events[?resourceStatus==CREATE_FAILED]'`

Proceed to the next chapter to login and run CMAQv5.4 on the Parallel Cluster.

4.2.6 Run CMAQ on c6a.48xlarge

Login to cluster

Note: Replace the your-key.pem with your Key Pair.

```
pcluster ssh -v -Y -i ~/your-key.pem --region=us-east-1 --cluster-name cmaq
```

Check to make sure elastic network adapter (ENA) is enabled

```
modinfo ena
```

```
lspci
```

Change default shell to .tcsh

```
sudo usermod -s /bin/tcsh ubuntu
```

Copy file to .cshrc

```
cp /shared/pcluster-cmaq/install/dot.cshrc.pcluster ~/.cshrc
```

logout and log back in to activate default tcsh shell

Check what modules are available on the cluster

```
module avail
```

Load the openmpi module

```
module load openmpi
```

Load the Libfabric module

```
module load libfabric-aws
```

Verify the gcc compiler version is greater than 8.0

```
gcc --version
```

output:

```
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

See also:

Link to the AWS Enhanced Networking Adapter Documentation

See also:

ParallelCluster User Manual

Verify that you have an updated set of run scripts from the pcluster-cmaq repo

```
cd /shared/pcluster-cmaq/run_scripts/cmaqv54+/
```

```
ls -lrt run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.2x96.ncclassic.csh
```

```
diff run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.2x96.ncclassic.csh /shared/pcluster-cmaq/
run_scripts/cmaqv54+/
```

If they don't exist or are not identical, then copy the run scripts from the repo

```
cp /shared/pcluster-cmaq/run_scripts/cmaqv54+/run_cctm* /shared/build/openmpi_gcc/
CMAQ_v54+/CCTM/scripts/
```

Verify that the input data is imported to /fsx from the S3 Bucket

```
cd /fsx/
ls */*
```

Preloading the files

Amazon FSx copies data from your Amazon S3 data repository when a file is first accessed. CMAQ is sensitive to latencies, so it is best to preload contents of individual files or directories using the following command:

```
nohup find /fsx/ -type f -print0 | xargs -0 -n 1 sudo lfs hsm_restore &
```

Create a directory that specifies the full path that the run scripts are expecting.

```
mkdir -p /fsx/data/CMAQ_Modeling_Platform_2018/
```

Link the 2018_12US1 directoy

```
cd /fsx/data/CMAQ_Modeling_Platform_2018/
```

```
ln -s /fsx/CMAQv5.4_2018_12US1_Benchmark_2Day_Input/2018_12US1/ .
```

Link the 12LISTOS_Training data

```
cd /fsx/data/
```

```
ln -s /fsx/CMAQv5.4_2018_12LISTOS_Benchmark_3Day_Input/12LISTOS_Training ./12US1_LISTOS
```

Link the 2018_12NE3 Benchmark data

```
ln -s /fsx/CMAQv5.4_2018_12NE3_Benchmark_2Day_Input/2018_12NE3 .
```

netCDF-3 classic input files are used

The *.nc4 compressed netCDF4 files on /fsx input directory were converted to netCDF classic (nc3) files

Create the output directory`

```
mkdir -p /fsx/data/output
```

Run the 12US1 Domain on 192 pes

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
```

```
sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.2x96.ncclassic.csh
```

Note, it will take about 3-5 minutes for the compute notes to start up. This is reflected in the Status (ST) of CF (configuring)

Check the status in the queue

```
squeue -u ubuntu
```

Output:

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 3    queue1     CMAQ   ubuntu  CF                2 queue1-dy-compute-
↪resource-1-[1-2]
```

After 5 minutes the status will change once the compute nodes have been created and the job is running

`squeue -u ubuntu`

Output:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 3    queue1     CMAQ   ubuntu  R       0:58      2 queue1-dy-compute-
↪resource-1-[1-2]
```

If you get the following message, then you likely need to upgrade the Parallel Cluster to using OnDemand Compute Nodes instead of SPOT instances.

```
ubuntu@ip-10-0-1-70:/shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 3    queue1     CMAQ   ubuntu PD       0:00      2 (Nodes required for␣
↪job are DOWN, DRAINED or reserved for jobs in higher priority partitions)
```

If you need to update cluster to use ONDEMAND instead of SPOT instances

Stop Compute Nodes

```
pcluster update-compute-fleet --region us-east-1 --cluster-name cmaq --status STOP_
↪REQUESTED
```

Upgrade compute nodes to ONDEMAND

```
 pcluster update-cluster --region us-east-1 --cluster-name cmaq --cluster-configuration␣
↪c6a.large-48xlarge.ebs_unencrypted_installed_public_ubuntu2004.fsx_import_ondemand.yaml
```

Restart the compute nodes

`pcluster update-compute-fleet --region us-east-1 --cluster-name cmaq --status START_REQUESTED`

Verify compute nodes have started:

```
pcluster describe-cluster --region=us-east-1 --cluster-name cmaq
```

Relogin to the cluster

```
pcluster ssh -v -Y -i ~/cmas.pem --region=us-east-1 --cluster-name cmaq
```

Resubmit the job to the queue

The 192 pe job should take 62 minutes to run (31 minutes per day)

Check on the status of the cluster using CloudWatch (optional)

Cloudwatch Dashboard

Monitoring Dashboard for ParallelCluster

Check the timings while the job is still running using the following command

`cd /fsx/data/output/output_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_2x96_classic/`

`grep 'Processing completed' CTM_LOG_001*`

Output:

```
            Processing completed...       6.3736 seconds
            Processing completed...       5.0755 seconds
            Processing completed...       5.1098 seconds
```

When the job has completed, use tail to view the timing from the log file.

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
```

```
tail run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.192.16x12pe.2day.20171222start.2x96.log
```

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       192
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   1853.4
02   2017-12-23   2035.1
     Total Time = 3888.50
      Avg. Time = 1944.25
```

Submit a request for a 96 pe job (1 x 96 pe) or 1 nodes instead of 2 nodes

```
sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.1x96.ncclassic.csh
```

Check on the status in the queue

```
squeue -u ubuntu
```

Note, it takes about 5 minutes for the compute nodes to be initialized, once the job is running the ST or status will change from CF (configure) to R

Output:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 4    queue1     CMAQ   ubuntu  R       7:20      1 queue1-dy-compute-
↪resource-1-3
```

Check the status of the run

```
tail run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.96.12x8pe.2day.20171222start.1x96.log
```

The 96 pe job should take 104 minutes to run (52 minutes per day) Note, this is a different domain (12US1 versus 12US2) than what was used for the HPC6a.48xlarge Benchmark runs, so the timings are not directly comparible. The 12US1 domain is larger than 12US2.

‘12US1’ ‘LAM_40N97W’ -2556000. -1728000. 12000. 12000. 459 299 1

Check whether the scheduler thinks there are cpus or vcpus

`sinfo -lN`

Output:

```
Wed Jun 14 00:49:36 2023
NODELIST                         NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_
↪DISK WEIGHT AVAIL_FE REASON
queue1-dy-compute-resource-1-1       1    queue1*   allocated 96     96:1:1 373555        ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-2       1    queue1*   allocated 96     96:1:1 373555        ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-3       1    queue1*       idle~ 96     96:1:1 373555        ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-4       1    queue1*       idle~ 96     96:1:1 373555        ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-5       1    queue1*       idle~ 96     96:1:1 373555        ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-6       1    queue1*       idle~ 96     96:1:1 373555        ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-7       1    queue1*       idle~ 96     96:1:1 373555        ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-8       1    queue1*       idle~ 96     96:1:1 373555        ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-9       1    queue1*       idle~ 96     96:1:1 373555        ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-10      1    queue1*       idle~ 96     96:1:1 373555        ␣
↪0       1 dynamic, none
```

Note: on a c6a.48xlarge, the number of virtual cpus is 192.

If the YAML contains the Compute Resources Setting of DisableSimultaneousMultithreading: false, then all 192 vcpus will be used

If DisableSimultaneousMultithreading: true, then the number of cpus is 96 and there are no virtual cpus.

Verify that the yaml file used DisableSimultaneousMultithreading: true

When the jobs are both submitted to the queue they will be dispatched to different compute nodes.

`squeue`

output

```
Submitted batch job 4
ip-10-0-1-243:/shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts> squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 4    queue1     CMAQ   ubuntu CF       0:01      1 queue1-dy-compute-
↪resource-1-3
                 3    queue1     CMAQ   ubuntu  R      21:28      2 queue1-dy-compute-
↪resource-1-[1-2]
```

When the job has completed, use tail to view the timing from the log file.

`cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/`

`tail -n 30 run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.96.12x8pe.2day.20171222start.1x96.log`

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       96
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   3153.2
02   2017-12-23   3485.9
     Total Time = 6639.10
      Avg. Time = 3319.55

```

Based on the Total Time, adding an additional node gave a speed-up of 1.7. 6639.10/3888.50 = 1.7074

Submit a job to run on 288 pes, 3x96 nodes

`sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.3x96.ncclassic.csh`

Verify that it is running on 3 nodes

`sbatch`

output:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 5    queue1     CMAQ   ubuntu  R       4:29      3 queue1-dy-compute-
↪resource-1-[1-3]
```

Check the log for how quickly the job is running

`grep 'Processing completed' run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.288.18x16pe.2day.20171222start.3x96.log`

Output:

```
 Processing completed...       4.0245 seconds
           Processing completed...       4.0263 seconds
           Processing completed...       3.9885 seconds
           Processing completed...       3.9723 seconds
           Processing completed...       3.9934 seconds
           Processing completed...       4.0075 seconds
           Processing completed...       3.9871 seconds
```

When the job has completed, use tail to view the timing from the log file.

`cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/`

```
tail -n 30 run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.288.18x16pe.2day.
20171222start.3x96.log
```

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       288
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   1475.9
02   2017-12-23   1580.7
     Total Time = 3056.60
      Avg. Time = 1528.30
```

Once you have submitted a few benchmark runs and they have completed successfully, proceed to the next chapter.

4.3 Performance and Cost Optimization

Timing information and scaling plots to assist users in optimizing the performance of their parallel cluster.

Performance Optimization

4.3.1 ParallelCluster Configuration

Selection of the compute nodes depends on the domain size and resolution for the CMAQ case, and what your model run time requirements are. Larger hardware and memory configurations may also be required for instrumented versions of CMAQ incuding CMAQ-ISAM and CMAQ-DDM3D. The ParallelCluster allows you to run the compute nodes only as long as the job requires, and you can also update the compute nodes as needed for your domain.

4.3.2 CMAQv5.4 Benchmarks

Benchmark Name	Grid Domain	Recommended EC2 Instance	vCPU	Cores	Memory	EFA Network Bandwidth	Storage (EBS Only)	On Demand Hourly Cost	Spot Hourly Cost	Region	Time (hr) per Simulation Day	Cost per Simulation Day
Training 12km Listos	(25x25x35)	c6a.2xlarge	8	4	16 GiB	Up to 12500 Megabit	gp3	0.306	0.2879	anywhere	.0459	$.014
12NE3	(100x105x35)	c6a.8xlarge	32	16	64 GiB	12500 Megabit	gp3	1.224	1.0008	anywhere	.274	$.335
12US1	(459x299x35)	c6a.48xlarge	192	96	384 GiB	50000 Megabit	gp3	7.344	5.5809	anywhere	.827	$6.07
12US1	(459x299x35)	hpc6a.48xlarge	n/a	96	384 GiB	100 Gbps	gp3	2.88	n/a	us-east-2b	.877	$2.53
12US1	(459x299x35)	hpc7g.8xlarge	n/a	64 (2x32)	256 GiB	200 Gbps	gp3	1.6832*2nodes	n/a	us-east-1	.855	$2.87

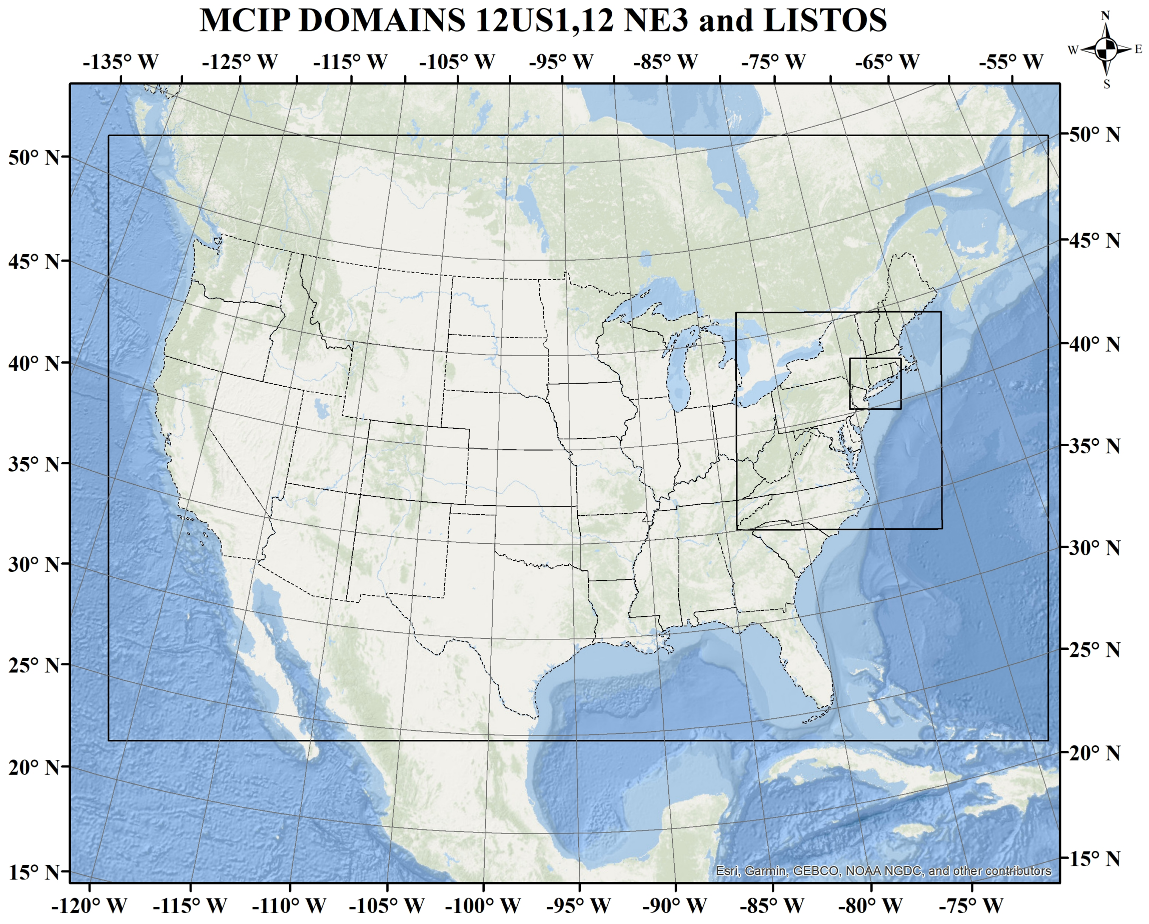
Note: *Hpc6a instances have simultaneous multi-threading disabled to optimize for HPC codes. This means that unlike other EC2 instances, Hpc6a vCPUs are physical cores, not threads. *Hpc6a instances available in US East (Ohio) and GovCloud (US-West) *HPC6a is available ondemand only (no spot pricing)

Note: *Two hpc7g.8xlarge nodes with 32 cores/node can run the 12US1 case as it has 256 GiB memory. hpc7g.16xlarge with 64cores/node only has 128 GiB memory, and can't run the 12US1 case on 1 node

Note: *hpc7g instances have simultaneous multi-threading disabled to optimize for HPC codes. The instances with fewer cores, 16, 32 pes are custom to only those instances, you are not sharing a slice of an instance (this also removes the need for pinning). hpc7g offers 16, 32 or 64 physical cpu instance size at launch

Note: If you are using SPOT pricing, ie. for the c6a.48xlarge compute nodes. Sometimes, the nodes are not available for SPOT pricing in the region you are using. If this is the case, the job will not start runnning in the queue, see AWS Troubleshooting. ParallelCluster Troubleshooting To avoid this, use the hpc EC2 instances, ie. hpc6a.48xlarge or hpc7g.16xlarge.

Data in table above is from the following: Sizing and Price Calculator from AWS



An explanation of why a scaling analysis is required for Multinode or Parallel MPI Codes

Quote from the following link.

“IMPORTANT: The optimal value of –nodes and –ntasks for a parallel code must be determined empirically by conducting a scaling analysis. As these quantities increase, the parallel efficiency tends to decrease. The parallel efficiency is the serial execution time divided by the product of the parallel execution time and the number of tasks. If multiple nodes are used then in most cases one should try to use all of the CPU-cores on each node.”

Note: For the scaling analysis that was performed with CMAQ, the parallel efficiency was determined as the runtime for the smallest number of CPUs divided by the product of the parallel execution time and the number of additional cpus used. If smallest NPCOLxNPROW configuration was 18 cpus, the run time for that case was used, and then the parallel efficiency for the case where 36 cpus were used would be parallel efficiency = runtime_18cpu/(runtime_36cpu*2)*100

See also:

Scaling Analysis - see section on Multinode or Parallel MPI Codes

Example Slurm script for Multinode Runs

4.3.3 Slurm Compute Node Provisioning

AWS ParallelCluster relies on SLURM to make the job allocation and scaling decisions. The jobs are launched, terminated, and resources maintained according to the Slurm instructions in the CMAQ run script. The YAML file for Parallel Cluster is used to set the identity of the head node and the compute node, and the maximum number of compute nodes that can be submitted to the queue. The head node can't be updated after a cluster is created. The compute nodes, and the maximum number of compute nodes can be updated after a cluster is created.

Number of compute nodes dispatched by the slurm scheduler is specified in the run script using #SBATCH –nodes=XX #SBATCH –ntasks-per-node=YY where the maximum value of tasks per node or YY limited by many CPUs are on the compute node.

Resources specified in the YAML file:

- Ubuntu2004
- Disable Simultaneous Multi-threading
- Spot Pricing
- Shared EBS filesystem to install software
- 1.2 TiB Shared Lustre file system with imported S3 Bucket (1.2 TiB is the minimum file size that you can specify for Lustre File System) mounted as /fsx or EBS volume 500 GB size mounted as /shared/data
- Slurm Placement Group enabled
- Elastic Fabric Adapter Enabled

See also:

EC2 Instance Types

Note: Pricing information in the tables below are subject to change. The links from which this pricing data was collected are listed below.

See also:

EC2 SPOT Pricing

See also:

EC2 On-Demand Pricing

See also:

Working with Spot Instances - ParallelCluster

hpc7g offers 16, 32 or 64 physical cpu instance size at launch

Note: Sometimes, the nodes are not available for SPOT pricing in the region you are using. If this is the case, the job will not start runnning in the queue, see AWS Troubleshooting. ParallelCluster Troubleshooting

4.3.4 Benchmark Timings for CMAQv5.4 12US1 Benchmark

Benchmark Timing for c6a.48xlarge

Table 2. Timing Results for CMAQv5.4 2 Day 12US1 Run on Parallel Cluster with c6a.xlarge head node and c6a.48xlarge Compute Nodes with Disable Simultaneous Multithreading turned on (using physical cores, not vcpus)

CPUs	NodesxCPU	COL-ROW	Day1 Timing (sec)	Day2 Timing (sec)	Total-Time	CPU Hours/day Input-Data	Input-Data	Equation using Spot Pricing	Spot-Cost	Equation using On Demand Pricing	On-Demand-Cost
96	1x96	12x8	3153.2	3485.9	6639.10	1.844	/fsx	$5.5809/hr * 1 node * 1.844 =	10.29	7.34/hr * 1 node * 1.844 =	13.53
192	2x96	16x12	1853.4	2035.1	3888.50	1.08	/fsx	$5.5809/hr * 2 node * 1.08 =	12.05	7.34/hr * 2 node * 1.08 =	15.85
288	3x96	16x18	1475.9	1580.7	3056.60	.849	/fsx	5.5809/hr * 3 node * .849 =	14.21	7.34/hr * 3 node * .849 =	18.6

Benchmark Timing for hpc6a.48xlarge

Table 2. Timing Results for CMAQv5.4 2 Day 12US1 Run on Parallel Cluster with c6a.xlarge head node and c6a.48xlarge Compute Nodes with Disable Simultaneous Multithreading turned on (using physical cores, not vcpus)

CPUs	NodesxCPU	COL-ROW	Day1 Timing (sec)	Day2 Timing (sec)	Total-Time	CPU Hours/day Input-Data	Input-Data	Equation using Spot Pricing	Spot-Cost	Equation using On Demand Pricing	On-Demand-Cost
96	1x96	12x8	3157.3	3493.4	6650.70	1.84	/fsx	n/a	n/a	2.88/hr * 1 node * 1.845 =	5.32
192	2x96	16x12	1850.0	2058.0	3908.00	1.085	/fsx	n/a	n/a	2.88/hr * 2 node * 1.085 =	6.25
288	3x96	16x18	1491.5	1599.8	3091.30	.859	/fsx	n/a	n/a	2.88/hr * 3 node * .859 =	7.41

Benchmark Timing for hpc7g.8xlarge with 32 processors per node

Table 3. Timing Results for CMAQv5.4 2 Day 12US1 Run on Parallel Cluster with c7g.large head node and hpc7g.8xlarge Compute Nodes with 32 processors per node.

CPUs	NodesxCPU	COLROW	Day1 Timing (sec)	Day2 Timing (sec)	TotalTime	CPU Hours/day	InputData	Equation using On Demand Pricing	OnDemandCost
32	1x32	4x8	6933.3	6830.2	13763.50	3.82	/fsx	1.6832/hr * 1 node * 3.82 =	6.42
64	2x32	8x8	3080.9	3383.5	6464.40	1.795	/fsx	1.6832/hr * 2 node * 1.795 =	6.04
96	3x32	12x8	2144.2	2361.9	4506.10	1.252	/fsx	1.6832/hr * 3 node * 1.252 =	6.32
128	4x32	16x8	1696.6	1875.7	3572.30	.992	/fsx	1.6832/hr * 4 node * .992 =	6.68

Benchmark Timing for hpc7g.16xlarge with 64 processors per node

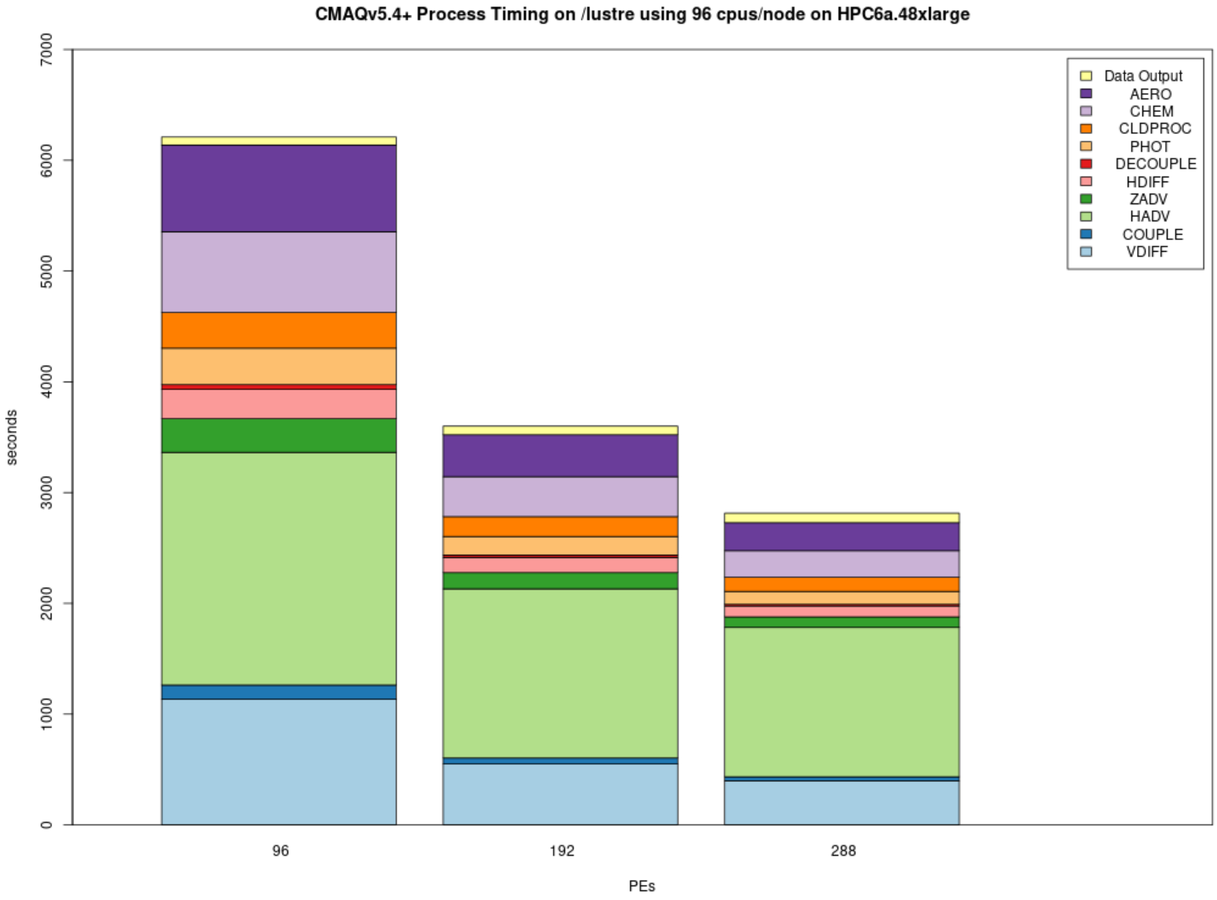
Table 4. Timing Results for CMAQv5.4 2 Day 12US1 Run on Parallel Cluster with c7g.large head node and hpc7g.16xlarge Compute Nodes with 64 processors per node.

CPUs	NodesxCPU	COLROW	Day1 Timing (sec)	Day2 Timing (sec)	TotalTime	CPU Hours/day	InputData	Equation using On Demand Pricing	OnDemandCost
64	1x64	8x8	crash	crash	crash	n/a	/fsx	1.6832/hr * 1 node * n/a =	n/a
128	2x64	8x16	2074.2	2298.9	4373.10	1.215	/fsx	1.6832/hr * 2 node * 1.214 =	4.089
192	3x64	12x16	1617.1	1755.3	3372.40	.937	/fsx/	1.6832/hr * 3 node * .937 =	4.730
256	4x64	16x16	1347.3	1501.4	2848.70	.7913	/fsx/	1.6832/hr * 4 node * .7913 =	5.327
320	5x64	16x20	1177.0	1266.6	2443.60	.6788	/fsx/	1.6832/hr * 5 node * .6788 =	5.713

4.3.5 Benchmark Scaling Plots for CMAQv5.4 12US1 Benchmark

Benchmark Scaling Plot for hpc6a.48xlarge

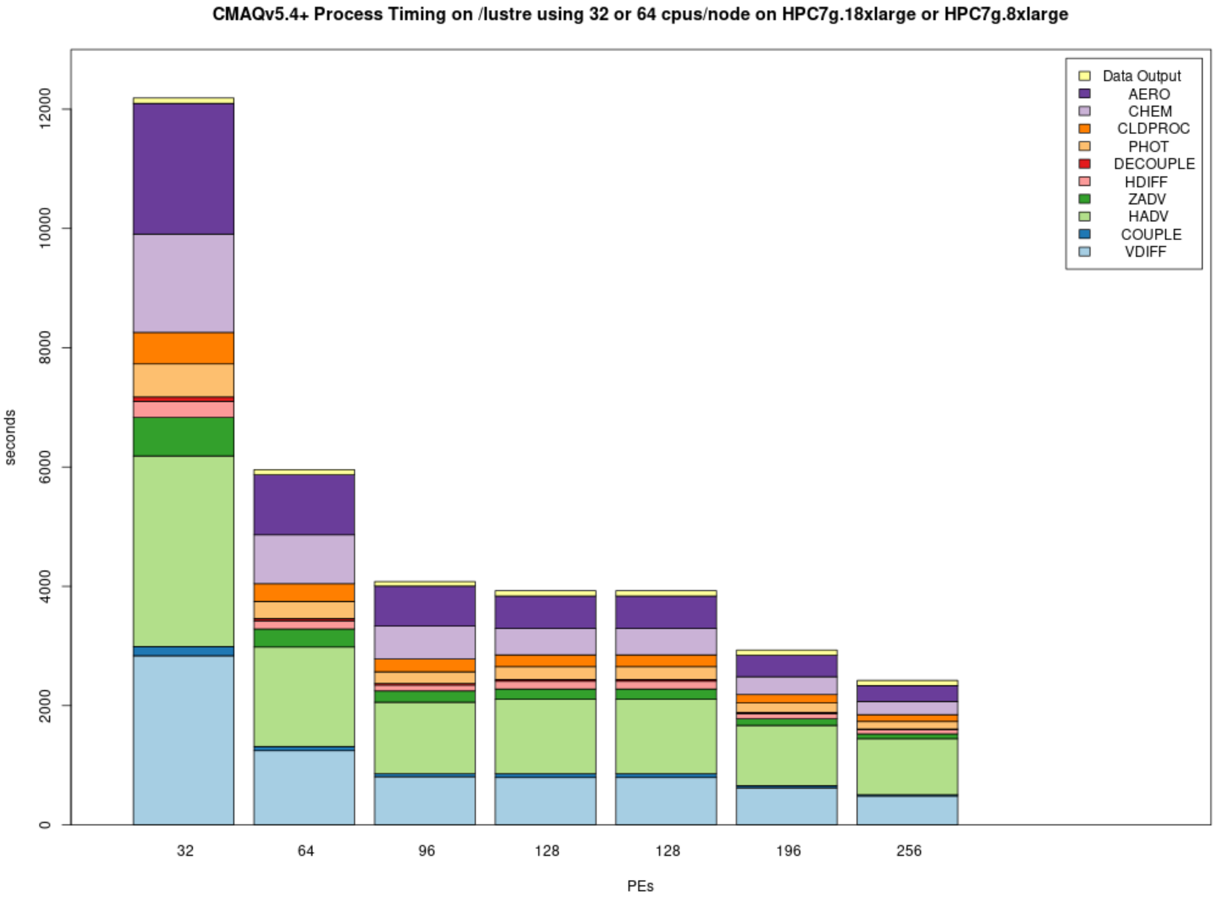
Figure 1. Scaling per Node on hpc6a.48xlarge Compute Nodes (96 cores/node)



```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=96
```

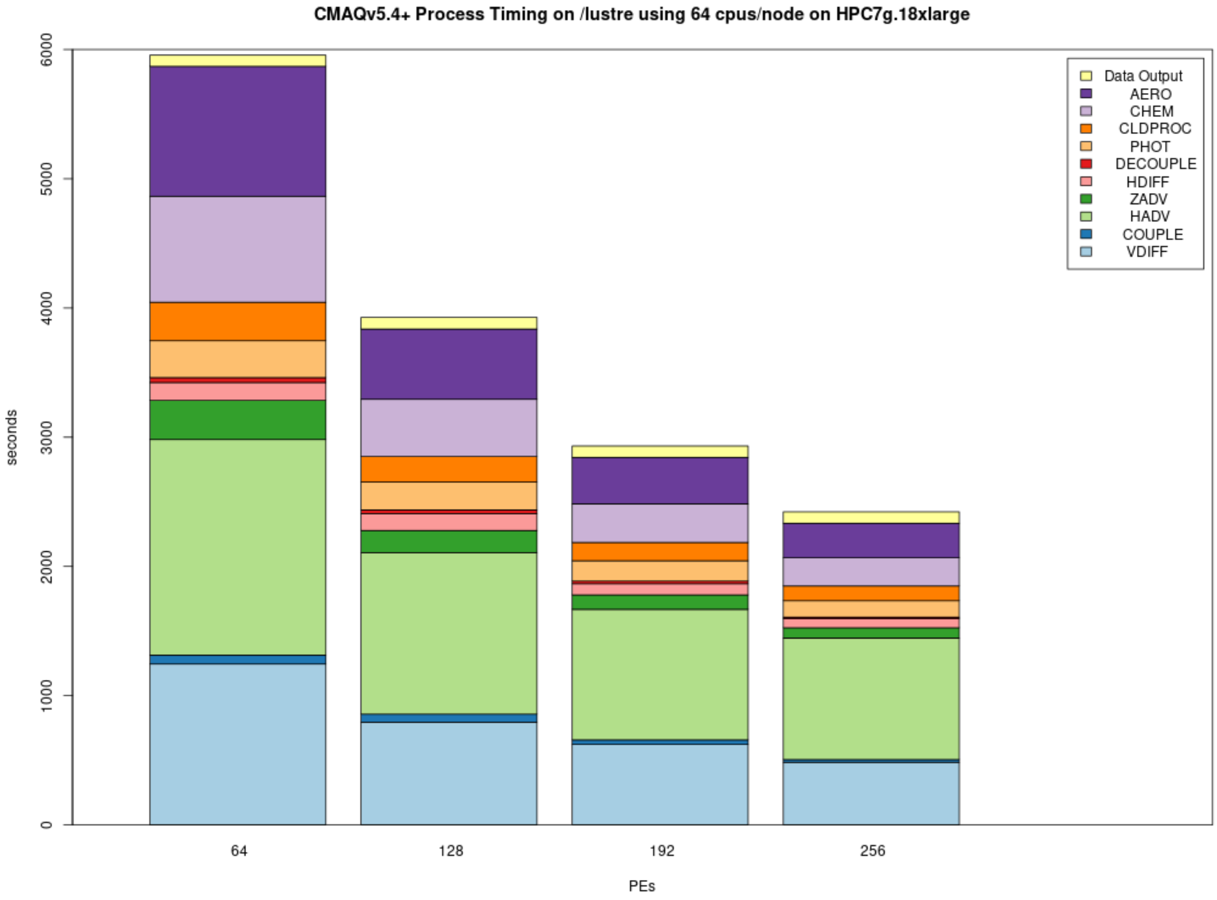
Benchmark Scaling Plot for hpc7g.8xlarge

Figure 2. Scaling per CPU on hpc7g.8xlarge compute node (32 cores/node)



Benchmark Scaling Plot for hpc7g.16xlarge

Figure 3. Scaling per Node on hpc7g.16xlarge Compute Nodes (64 cores/node)



Total Time and Cost versus CPU Plot for hpc7g.8xlarge

Figure 4. Plot of Total Time and On Demand Cost varies as additional CPUs are used. Note that the run script and yaml settings settings that were optimized for running CMAQ on the cluster.

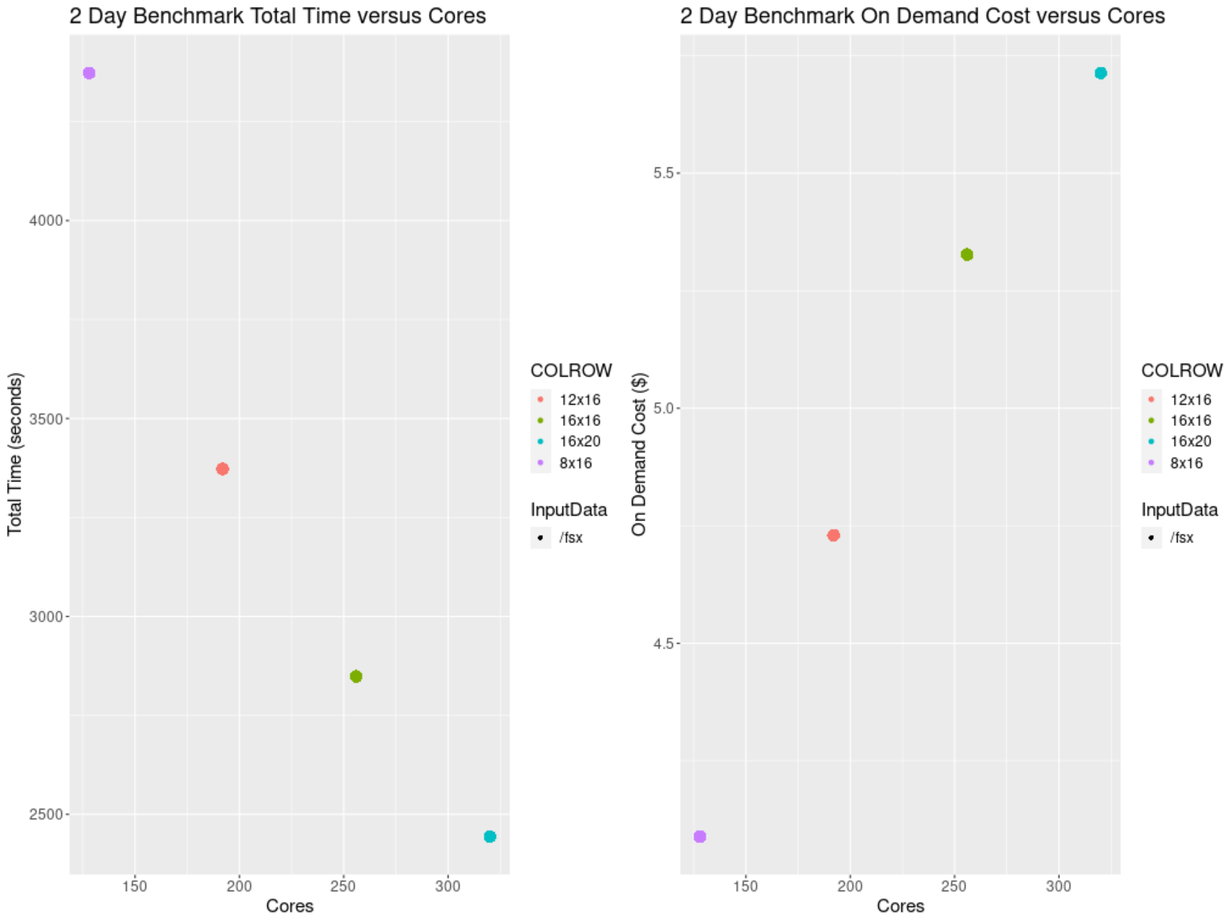
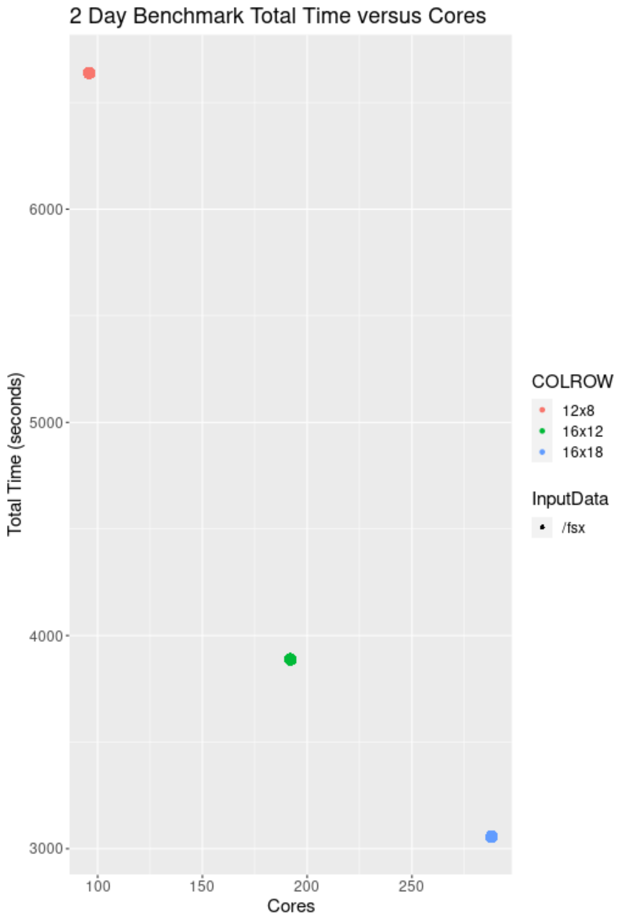
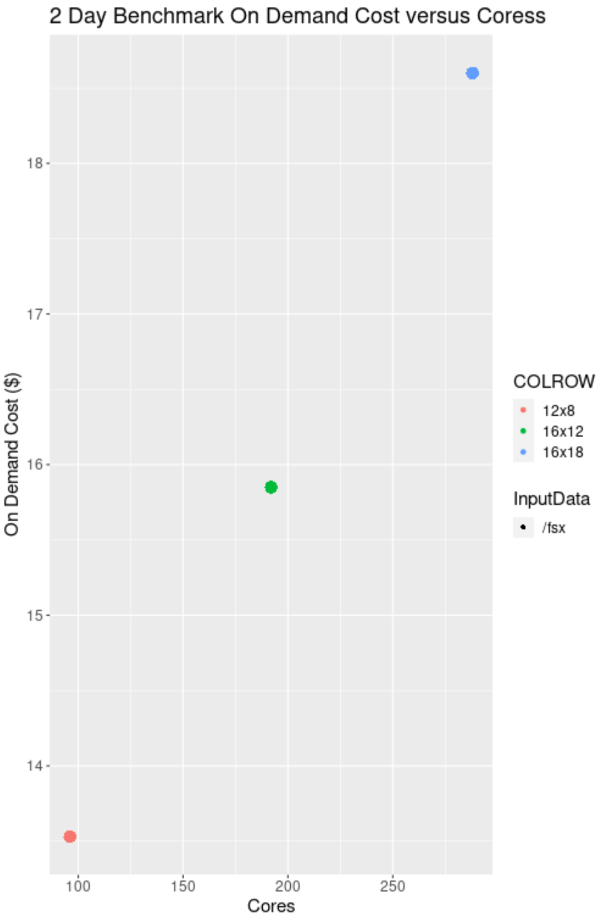


Figure 5. Plot of Total Time and On Demand Cost versus CPUs for c6a.48xlarge





4.3.6 Cost Information

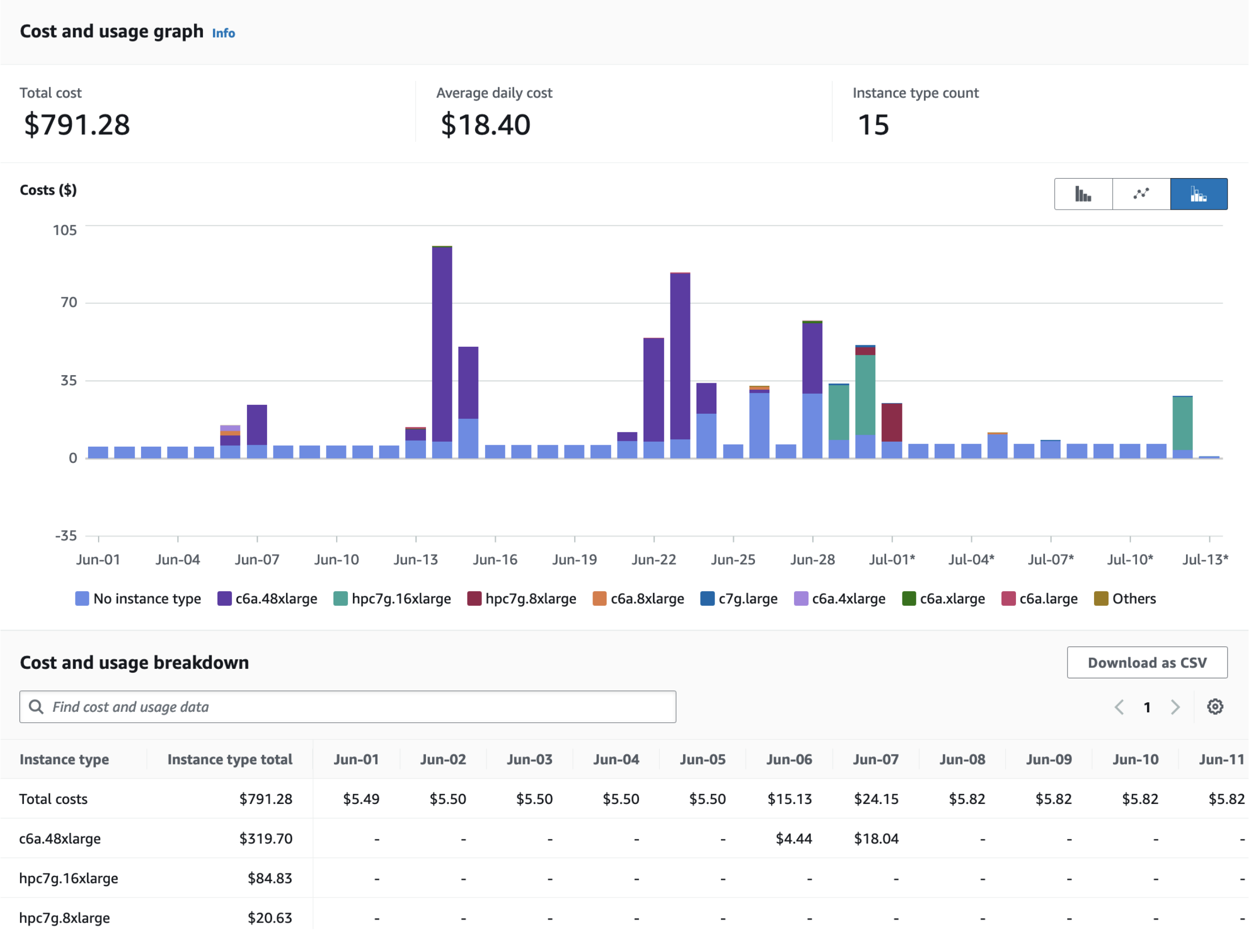
Cost information is available within the AWS Web Console for your account as you use resources, and there are also ways to forecast your costs using the pricing information available from AWS.

Cost Explorer

Example screenshots of the AWS Cost Explorer Graphs were obtained after running several of the CMAQ Benchmarks, varying # nodes and # cpus and NPCOL/NPROW. These costs are of a two day session of running CMAQ on the ParallelCluster, and should only be used to understand the relative cost of the EC2 instances (head node and compute nodes), compared to the storage, and network costs.

In Figure 6 The Cost Explorer Display shows the cost of different EC2 Instance Types: note that c6a.48xlarge (purple) is highest cost - as these the most expensive compute nodes that were used. The hpc7g.16xlarge compute nodes incurred less cost (green).

Figure 6. Cost by Instance Type - AWS Console



Cost and usage graph Info

Total cost
$791.28

Average daily cost
$18.40

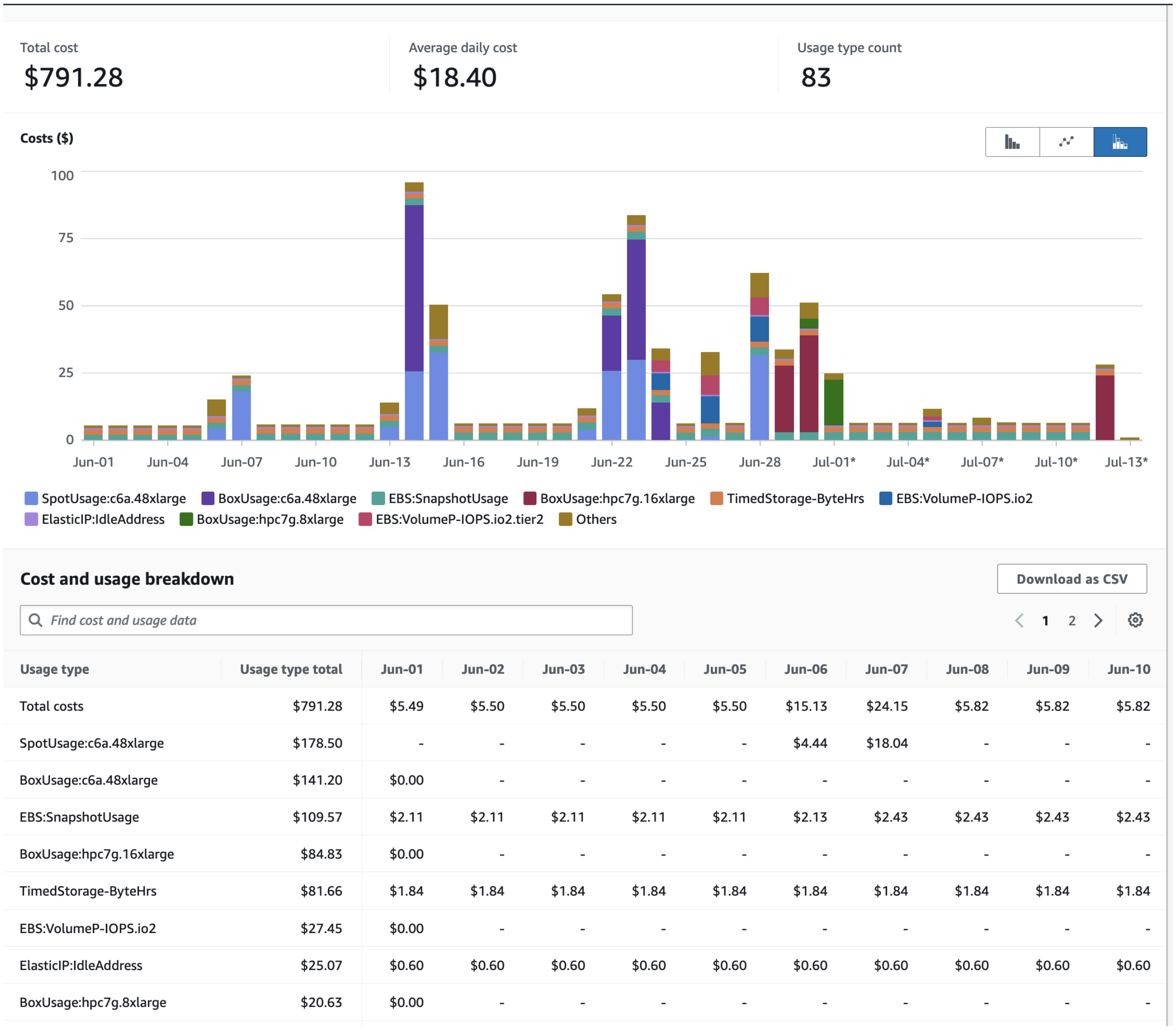
Instance type count
15

Cost and usage breakdown

Instance type	Instance type total	Jun-01	Jun-02	Jun-03	Jun-04	Jun-05	Jun-06	Jun-07	Jun-08	Jun-09	Jun-10	Jun-11
Total costs	$791.28	$5.49	$5.50	$5.50	$5.50	$5.50	$15.13	$24.15	$5.82	$5.82	$5.82	$5.82
c6a.48xlarge	$319.70	-	-	-	-	-	$4.44	$18.04	-	-	-	-
hpc7g.16xlarge	$84.83	-	-	-	-	-	-	-	-	-	-	-
hpc7g.8xlarge	$20.63	-	-	-	-	-	-	-	-	-	-	-

In Figure 7 The Cost Explorer displays a graph of the cost categorized by usage by spot or OnDemand, NatGateway, or Timed Storage. Note: c6a.48xlarge is highest generating cost resource, but other resources such as storage on the EBS volume and the network NatGatway or SubnetIDs also incur costs

Figure 7. Cost by Usage Type - AWS Console



Total cost
$791.28

Average daily cost
$18.40

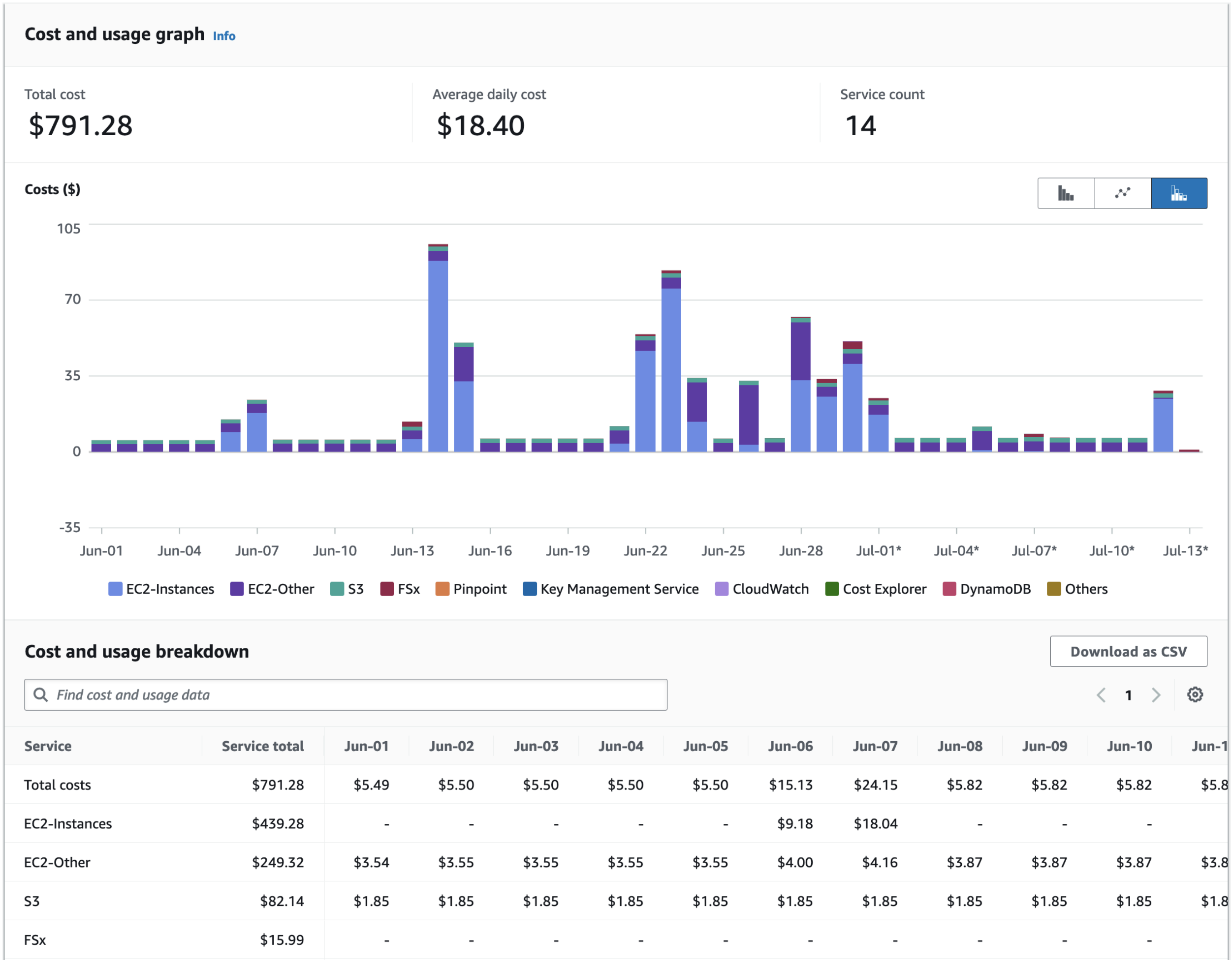
Usage type count
83

Cost and usage breakdown

Usage type	Usage type total	Jun-01	Jun-02	Jun-03	Jun-04	Jun-05	Jun-06	Jun-07	Jun-08	Jun-09	Jun-10
Total costs	$791.28	$5.49	$5.50	$5.50	$5.50	$5.50	$15.13	$24.15	$5.82	$5.82	$5.82
SpotUsage:c6a.48xlarge	$178.50	-	-	-	-	-	$4.44	$18.04	-	-	-
BoxUsage:c6a.48xlarge	$141.20	$0.00	-	-	-	-	-	-	-	-	-
EBS:SnapshotUsage	$109.57	$2.11	$2.11	$2.11	$2.11	$2.11	$2.13	$2.43	$2.43	$2.43	$2.43
BoxUsage:hpc7g.16xlarge	$84.83	$0.00	-	-	-	-	-	-	-	-	-
TimedStorage-ByteHrs	$81.66	$1.84	$1.84	$1.84	$1.84	$1.84	$1.84	$1.84	$1.84	$1.84	$1.84
EBS:VolumeP-IOPS.io2	$27.45	$0.00	-	-	-	-	-	-	-	-	-
ElasticIP:IdleAddress	$25.07	$0.60	$0.60	$0.60	$0.60	$0.60	$0.60	$0.60	$0.60	$0.60	$0.60
BoxUsage:hpc7g.8xlarge	$20.63	$0.00	-	-	-	-	-	-	-	-	-

In Figure 8. The Cost Explorer Display shows the cost by Services including EC2 Instances, S3 Buckets, and FSx Lustre File Systems

Figure 8. Cost by Service Type - AWS Console



Total cost $791.28

Average daily cost $18.40

Service count 14

Cost and usage breakdown

Service	Service total	Jun-01	Jun-02	Jun-03	Jun-04	Jun-05	Jun-06	Jun-07	Jun-08	Jun-09	Jun-10	Jun-1
Total costs	$791.28	$5.49	$5.50	$5.50	$5.50	$5.50	$15.13	$24.15	$5.82	$5.82	$5.82	$5.8
EC2-Instances	$439.28	-	-	-	-	-	$9.18	$18.04	-	-	-	
EC2-Other	$249.32	$3.54	$3.55	$3.55	$3.55	$3.55	$4.00	$4.16	$3.87	$3.87	$3.87	$3.8
S3	$82.14	$1.85	$1.85	$1.85	$1.85	$1.85	$1.85	$1.85	$1.85	$1.85	$1.85	$1.8
FSx	$15.99	-	-	-	-	-	-	-	-	-	-	

Compute Node Cost Estimate

Head node c7g.large compute cost = entire time that the parallel cluster is running (creation to deletion) = 6 hours * $0.0725/hr = $ .435 using ondemand pricing.

Table 5. Extrapolated Cost of compute nodes used for CMAQv5.4+ Annual Simulation based on 2 day 12US1 benchmark

Benchmark Case	Compute Node	Number of PES	Number of Nodes	Pricing	Cost per node	Time to completion (hour)	Equation Extrapolate Cost for Annual Simulation	Annual Cost	Days to Complete Annual Simulation
! 2 day 12US1	c6a.48xlarge	96	1	ON-DEMAND	$7.344/hour	6639.10/3600 = 1.84	1.84/2 * 365 = 336.6 hours/node * 1 node = 336.6 hr * 7.344/hr =	$2,471	14
! 2 day 12US1	hpc6a.48xlarge	96	1	ON-DEMAND	$2.88/hour	6639.10/3600 = 1.84	1.84/2 * 365 = 336.6 hours/node * 1 node = 336.6 hr * 2.88/hr =	$969.4	14
2 day 12US1	hpc7g.16xlarge	128	2	ON-DEMAND	$1.6832/hour	4574.00/3600 = 1.27	1.27/2 * 365 = 231.87 hours/node * 2 nodes = 463.75 hr * $1.6832/hr =	$780	9.6
2 day 12US1	hpc7g.16xlarge	192	3	ON-DEMAND	$1.6832/hour	3509.80/3600 = .9749	.9749/2 * 365 = 177.9 hours/node * 3 nodes = 533.75 hr * $1.6832/hr =	$898	7.4

Note: These cost estimates depend on the availability of number of nodes for the instance type. If fewer nodes are available, then it will take longer to complete the annual run, but the costs should be accurate, as the 12US1 Domain Benchmark scales well up to this number of nodes. The cost of running an annual simulation on 2 hpc7g.16xlarge nodes using OnDemand Pricing is $780, the cost of running an annual simulation on 3 hpc7g.16xlarge nodes using OnDemand pricing is $898. If you run on only 2 nodes, then you would pay less, but wait longer for the run to be completed, 9.6 days using 2 nodes versus 7.4 days using 3 nodes.

Storage Cost Estimate

See also:

AWS Lustre Pricing

Table 6. Lustre SSD File System Pricing for us-east-1 region

Storage Type	Storage options	Pricing with data compression enabled*	Pricing (monthly)
Persistent	125 MB/s/TB	$0.073	$0.145/month
Persistent	250 MB/s/TB	$0.105	$0.210/month
Persistent	500 MB/s/TB	$0.170	$0.340/month
Persistent	1,000 MB/s/TB	$0.300	$0.600/month
Scratch	200/MB/s/TiB	$0.070	$0.140/month

Note, there is a difference in the storage sizing units that were obtained from AWS.

See also:

TB vs TiB

Quote from the above website; "One tebibyte is equal to 2^40 or 1,099,511,627,776 bytes." One terabyte is equal to 1012 or 1,000,000,000,000 bytes. A tebibyte equals nearly 1.1 TB. That's about a 10% difference between the size of a tebibyte and a terabyte, which is significant when talking about storage capacity."

Lustre Scratch SSD 200 MB/s/TiB is tier of the storage pricing that we have configured in the yaml for the cmaq parallel cluster.

See also:

YAML FSxLustreSettings

Cost example: 0.14 USD per month / 730 hours in a month = 0.00019178 USD per hour

Note: 1.2 TiB is the minimum file size that you can specify for the lustre file system

```
1,200 GiB x 0.00019178 USD per hour x 24 hours x 5 days = 27.6 USD
```

Question is 1.2 TiB enough for the output of a yearly CMAQ run?

For the output data, assuming 2 day CONUS Run, all 35 layers, all 244 variables in CONC output

```
cd /fsx/data/output/output_CCTM_v532_gcc_2016_CONUS_16x8pe_full
du -sh
```

Size of output directory when CMAQ is run to output all 35 layers, all 244 variables in the CONC file, includes all other output files

```
173G .
```

So we need 86.5 GB per day

Storage requirement for an annual simulation if you assumed you would keep all data on lustre filesystem

```
 86.5 GB * 365 days = 31,572.5 GB  = 31.5 TB
```

Annual simulation local storage cost estimate

Assuming it takes 5 days to complete the annual simulation, and after the annual simulation is completed, the data is moved to archive storage.

```
 31,572.5 GB x 0.00019178 USD per hour x 24 hours x 5 days = $726.5 USD
```

To reduce storage requirements; after the CMAQ run is completed for each month, the post-processing scripts are run and completed, and then the CMAQ Output data for that month is moved from the Lustre Filesystem to the Archived Storage. Monthly data volume storage requirements to store 1 month of data on the lustre file system is approximately 86.5 x 30 days = 2,595 GB or 2.6 TB.

```
  2,595 GB x 0.00019178 USD per hour x 24 hours x 5 days = $60 USD
```

Estimate for S3 Bucket cost for storing an annual simulation

See also:

S3 Storage Pricing Tiers

S3 Standard - General purpose storage	Storage Pricing
First 50 TB / Month	$0.023 per GB
Next 450 TB / Month	$0.022 per GB
Over 500 TB / Month	$0.021 per GB

Archive Storage cost estimate for annual simulation - assuming you want to save it for 1 year

31.5 TB * 1024 GB/TB * .023 per GB * 12 months = $8,903

S3 Glacier Flexible Retrieval (Formerly S3 Glacier)	Storage Pricing
long-term archives with retrieval option from 1 minute to 12 hours	
All Storage / Month	$0.0036 per GB

S3 Glacier Flexible Retrieval Costs 6.4 times less than the S3 Standard

31.5 TB * 1024 GB/TB * $.0036 per GB * 12 months = $1393.0 USD

Lower cost option is S3 Glacier Deep Archive (accessed once or twice a year, and restored in 12 hours)

31.5 TB * 1024 GB/TB * $.00099 per GB * 12 months = $383 USD

4.3.7 Recommended Workflow for extending to annual run

Post-process monthly save output and/or post-processed outputs to S3 Bucket at the end of each month.

Still need to determine size of post-processed output (combine output, etc).

```
86.5 GB * 31 days = 2,681.5 GB * 1 TB/1024 GB =  2.62 TB
```

Cost for lustre storage of a monthly simulation

```
2,681.5 GB x 0.00019178 USD per hour x 24 hours x 5 days = $61.7 USD
```

Goal is to develop a reproducable workflow that does the post processing after every month, and then copies what is required to the S3 Bucket, so that only 1 month of output is imported at a time to the lustre scratch file system from the S3 bucket. This workflow will help with preserving the data in case the cluster or scratch file system gets pre-empted.

4.4 Developer Guide to install and run CMAQv5.4 on Single VM or Parallel Cluster

4.4.1 Install CMAQv5.4+ on Single Virtual Machine Advanced (optional)

Run CMAQv5.4+ on a single Virtual Machine (VM) using c6a.xlarge (4 CPUs) and Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-1031-aws x86_64), then upgrade to c6a.48xlarge.

Install Software and run CMAQv5.4 on c6a.xlarge for the 12km Listos Training Domain

Instructions are provided to build and install CMAQ on c6a.xlarge compute node installed from Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-1031-aws x86_64) Image that contains modules for git, openmpi and gcc. The compute node does not have a SLURM scheduler on it, so jobs are run interactively from the command line.

Instructions to install data and CMAQ libraries and model are provided along with sample run scripts to run CMAQ on 4 processors on a single c6a.xlarge instance.

This will provide users with experience using the AWS Console to create a Virtual Machine, select Operating System, select the size of the VM as c6a.xlarge vcpus, 8 GiB memory, using an SSH private key to login and install and run CMAQ.

Using this method, the user needs to be careful to start and stop the Virtual Machine and only have it run while doing the intial installation, and while running CMAQ. The full c6a.xlarge instance will incur charges as long as it is on, even if a job isn't running on it.

This is different than the Parallel Cluster, where if CMAQ is not running in the queue, then the Compute nodes are down, and not incurring costs.

Build CMAQv5.4+ on c6a.xlarge EC2 instance

Create a c6a.xlarge Virtual Machine

1. Login to AWS Console
2. Select Get Started with EC2
3. Select Launch Instance
4. Select Architechure (64-bit(x86)
5. Application and OS (Operating System) Images: Select Ubunutu 22.04 LTS(HVM), SSD Volume Type (the version of OS determines what packages are available from apt-get and that determines the version of software obtained, ie. cdo version > 2.0 for Ubuntu 22.04 LTS, or cdo version < 2.0 for Ubuntu 18.04.
6. Instance Type: Select c6a.xlarge ($0.153/hr)
7. Key pair - SSH public key, select existing key or create a new one.
8. Network settings - select default settings
9. Configure storage - select 100 GiB gp3 Root volume
10. Select Launch instance

user_guide_pcluster/aws_ec2_images/AWS_EC2_Virtual_Machine_Create.png

Login to the Virtual Machine

Change the permissions on the public key using command

```
chmod 400 [your-key-name].pem
```

Login to the Virtual Machine using ssh to the IP address using the public key.

```
ssh -Y -i ./xxxxxxx_key.pem ubuntu@xx.xx.xx.xx
```

Make the /shared directory

sudo mkdir /shared

Change the group and ownership of the shared directory

```
sudo chown ubuntu /shared
sudo chgrp ubuntu /shared
```

Change directories and verify that you see the /shared directory with Size of 100 GB

```
cd /shared
```

```
df -h
```

Output

```
df -h
Filesystem       Size  Used Avail Use% Mounted on
/dev/root         97G  1.6G   96G   2% /
tmpfs             16G     0   16G   0% /dev/shm
tmpfs            6.2G  876K  6.2G   1% /run
tmpfs            5.0M     0  5.0M   0% /run/lock
/dev/nvme0n1p15  105M  6.1M   99M   6% /boot/efi
tmpfs            3.1G  4.0K  3.1G   1% /run/user/1000
```

Create subdirectories on /shared

Create a /shared/build, /shared/data and /shared/cyclecloud-cmaq directory

```
cd /shared
mkdir build
mkdir data
```

Check operating system version

```
lsb_release -a
```

output

```
No LSB modules are available.
Distributor ID:        Ubuntu
Description:        Ubuntu 22.04.2 LTS
Release:        22.04
Codename:        jammy
```

Install Environment Modules

```
sudo apt-get upgrade
sudo apt-get install environment-modules
```

Logout and then log back in to activate modules command

Verify module command works

```
 module list
```

Output:

No Modulefiles Currently Loaded.

```
module avail
```

Output:

```
------------------------------------------------------------------------------------- /
↪usr/share/modules/modulefiles --------------------------------------------------------
↪-----------------------------
dot  module-git  module-info  modules  null  use.own
```

Set up build environment

Load the git module

```
module load module-git
```

If you do not see git available as a module, you may need to install it as follows:

```
sudo apt-get install git
```

Install Compilers and OpenMPI

```
sudo apt-get update
sudo apt-get install gcc-9
sudo apt-get  install gfortran-9
sudo apt-get install openmpi-bin openmpi-common libopenmpi-dev libgtk2.0-dev
sudo apt-get install tcsh
```

Change shell to use tcsh

```
sudo usermod -s /usr/bin/tcsh ubuntu
```

Logout and log back in, then check the shell

```
echo $SHELL
```

output

```
/usr/bin/tcsh
```

Check available versions of compiler

```
dpkg --list | grep compiler
```

Choose gcc-9 and gfortran-9 as default compilers

```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 9
sudo update-alternatives --install /usr/bin/gfortran gfortran /usr/bin/gfortran-9 9
```

Check version of gcc

```
gcc --version
```

output

```
gcc --version
gcc (Ubuntu 9.5.0-1ubuntu1~22.04) 9.5.0
```

Check version of gfortran

```
gfortran --version
```

Output

```
GNU Fortran (Ubuntu 9.5.0-1ubuntu1~22.04) 9.5.0
```

Check version of OpenMPI

```
mpirun --version
```

output

```
mpirun (Open MPI) 4.1.2
```

Install Parallel Cluster CMAQ Repo

```
cd /shared
```

```
git clone -b main https://github.com/CMASCenter/pcluster-cmaq.git
```

Install and build netcdf C, netcdf Fortran, I/O API, and CMAQ

```
cd /shared/pcluster-cmaq/install
```

Install netcdf-C and netcdf-Fortran

```
./gcc_netcdf_singlevm.csh |& tee ./gcc_netcdf_singlevm.log
```

If successful, you will see the following output, that at the bottom shows what versions of the netCDF library were installed.

```

+-------------------------------------------------------------+
| Congratulations! You have successfully installed the netCDF |
| Fortran libraries.                                          |
|                                                             |
| You can use script "nf-config" to find out the relevant     |
| compiler options to build your application. Enter           |
|                                                             |
|     nf-config --help                                        |
|                                                             |
| for additional information.                                 |
|                                                             |
| CAUTION:                                                    |
|                                                             |
| If you have not already run "make check", then we strongly  |
| recommend you do so. It does not take very long.            |
|                                                             |
| Before using netCDF to store important data, test your      |
| build with "make check".                                    |
|                                                             |
| NetCDF is tested nightly on many platforms at Unidata       |
| but your platform is probably different in some ways.       |
|                                                             |
| If any tests fail, please see the netCDF web site:          |
| https://www.unidata.ucar.edu/software/netcdf/                |
```

(continues on next page)

(continued from previous page)

```
|                                                               |
| NetCDF is developed and maintained at the Unidata Program    |
| Center. Unidata provides a broad array of data and software |
| tools for use in geoscience education and research.          |
| https://www.unidata.ucar.edu                                   |
+---------------------------------------------------------------+

make[3]: Leaving directory '/shared/build/netcdf-fortran-4.5.4'
make[2]: Leaving directory '/shared/build/netcdf-fortran-4.5.4'
make[1]: Leaving directory '/shared/build/netcdf-fortran-4.5.4'
netCDF 4.8.1
netCDF-Fortran 4.5.3
```

Install I/O API

```
./gcc_ioapi_singlevm.csh |& tee ./gcc_ioapi_singlevm.log
```

Find what operating system is on the system:

```
 cat /etc/os-release
```

Output

```
PRETTY_NAME="Ubuntu 22.04.2 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.2 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
```

Copy a file to set paths

```
cd /shared/pcluster-cmaq/install
```

```
cp dot.cshrc.singlevm ~/.cshrc
```

Exit cluster and log back in to activate the update shell, or use csh

Create Custom Environment Module for Libraries

There are two steps required to create your own custome module:

1. write a module file
2. add a line to your ~/.cshrc to update the MODULEPATH

Create a new custom module that will be loaded with:

```
module load ioapi-3.2/gcc-9.5-netcdf
```

Step 1: Create the module file for ioapi-3.2.

First, create a path to store the module file. The path must contain /Modules/modulefiles/ and should have the general form //Modules/modulefiles// where is typically numerical and is the actual module file.

```
mkdir -p /shared/build/Modules/modulefiles/ioapi-3.2
```

Next, create the module file and save it in the directory above.

```
cd /shared/build/Modules/modulefiles/ioapi-3.2
vim gcc-9.5-netcdf
```

Contents of gcc-9.5-netcdf:

```
#%Module

proc ModulesHelp { } {
   puts stderr "This module adds ioapi-3.2/gcc-9.5 to your path"
}

module-whatis "This module adds ioapi-3.2/gcc-9.5 to your path\n"

set basedir "/shared/build/ioapi-3.2/"
prepend-path PATH "${basedir}/Linux2_x86_64gfort"
prepend-path LD_LIBRARY_PATH "${basedir}/ioapi/fixed_src"
```

The example module file above sets two evironment variables.

The modules update the PATH and LD_LIBRARY_PATH.

Step 2. Create the module file for netcdf-4.8.1

```
mkdir -p /shared/build/Modules/modulefiles/netcdf-4.8.1
```

Next, create the module file and save it in the directory above.

```
cd /shared/build/Modules/modulefiles/netcdf-4.8.1
vim gcc-9.5
```

Contents of gcc-9.5

```
#%Module

proc ModulesHelp { } {
   puts stderr "This module adds netcdf-4.8.1/gcc-9.5 to your path"
}

module-whatis "This module adds netcdf-4.8.1/gcc-9.5 to your path\n"

set basedir "/shared/build/netcdf"
prepend-path PATH "${basedir}/bin"
prepend-path LD_LIBRARY_PATH "${basedir}/lib"
module load mpi/openmpi-4.1.2
```

Step 3. Create the module file for mpi

```
mkdir -p /shared/build/Modules/modulefiles/mpi
```

Next, create the module file and save it in the directory above.

```
cd /shared/build/Modules/modulefiles/mpi
vim openmpi-4.1.2
```

Contents of openmpi-4.1.2

```
#%Module

proc ModulesHelp { } {
   puts stderr "This module adds mpi/openmpi-4.1.2 to your path"
}

module-whatis "This module adds mpi/openmpi-4.1.2 to your path\n"

set basedir "/usr/lib/x86_64-linux-gnu/openmpi/"
prepend-path PATH "/usr/bin/"
prepend-path LD_LIBRARY_PATH "${basedir}/lib"
```

Step 4: Add the module path to MODULEPATH.

Now that the module file has been created, add the following line to your ~/.cshrc file so that it can be found:

```
module use --append /shared/build/Modules/modulefiles
```

Step 5: View the modules available after creation of the new module

The module avail command shows the paths to the module files on a given cluster.

```
module avail
```

Output

```
ioapi-3.2/gcc-9.5-netcdf  mpi/openmpi-4.1.2  netcdf-4.8.1/gcc-9.5
```

Step 4: Load the new modules

```
ioapi-3.2/gcc-9.5-netcdf  mpi/openmpi-4.1.2  netcdf-4.8.1/gcc-9.5
```

Find path for openmpi libraries

```
ompi_info --path libdir
```

output

```
 Libdir: /usr/lib/x86_64-linux-gnu/openmpi/lib
```

Find path for include files for openmpi

```
ompi_info --path incdir
```

output

```
Incdir: /usr/lib/x86_64-linux-gnu/openmpi/include
```

Edit the config_cmaq_singlevm.csh script to specify the paths for OpenMPI

Note, search for case gcc so that you edit the section of the file that is using the gcc compiler.

```
       setenv MPI_INCL_DIR     /usr/lib/x86_64-linux-gnu/openmpi/include              #>␣
↪MPI Include directory path
        setenv MPI_LIB_DIR     /usr/lib/x86_64-linux-gnu/openmpi/lib             #> MPI␣
↪Lib directory path
```

Install Python

```
sudo apt-get install python3 python3-pip
```

Check Version

```
python3 --version
Python 3.10.6
ip-172-31-27-148:/shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts> python3 -m pip --
↪version
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)
```

Install jupyter notebook.

```
pip install jupyterlab
```

Install and Build CMAQ

```
cd /shared/pcluster-cmaq/install
./gcc_cmaq54+_singlevm.csh |& tee ./gcc_cmaq54+_singlevm.log
```

Add compile option to makefile to get beyond a type mismatch error (note, this is only needed if you were using the gcc-11 compiler.

SKIP this step.

Add the following to the compile option: -fallow-argument-mismatch

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54_gcc
vi Makefile.gcc
```

Output:

```
 FSTD = -fallow-argument-mismatch -O3 -funroll-loops -finit-character=32 -Wtabs -
↪Wsurprising -ftree-vectorize  -ftree-loop-if-convert -finline-limit=512
```

Run make again

```
make |& tee Make.log
```

Verfify that the executable was successfully built.

```
ls /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54_gcc/*.exe
```

Output

```
/shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54_gcc/CCTM_v54.exe
```

Copy the run scripts from the repo to the run directory

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts
```

```
cp /shared/pcluster-cmaq/run_scripts/c6a/*.csh .
```

List the scripts available

```
ls -rlt *.csh*
```

Output

```
-rwxrwxr-x 1 ubuntu ubuntu  8374 Jun  6 19:06 bldit_mech.csh
-rwxrwxr-x 1 ubuntu ubuntu 37040 Jun  6 19:06 bldit_cctm.csh
-rwxrwxr-x 1 ubuntu ubuntu 39500 Jun  6 19:06 run_cctm_cracmm_2018_4LISTOS1.csh
-rwxrwxr-x 1 ubuntu ubuntu 38493 Jun  6 19:06 run_cctm_Bench_2018_12NE3.csh
-rwxrwxr-x 1 ubuntu ubuntu 51329 Jun  6 19:06 run_cctm_Bench_2018_12NE3.WRFCMAQ.csh
-rwxrwxr-x 1 ubuntu ubuntu 38158 Jun  6 19:06 run_cctm_Bench_2016_12SE1.csh
-rwxrwxr-x 1 ubuntu ubuntu 39265 Jun  6 19:06 run_cctm_2016_12US1.csh
-rwxrwxr-x 1 ubuntu ubuntu 37458 Jun  6 19:06 run_cctm_2015_HEMI.csh
-rwxrwxr-x 1 ubuntu ubuntu 37583 Jun  6 19:06 run_cctm_2010_4CALIF1.csh
-rwxrwxr-x 1 ubuntu ubuntu  38460 Jun  6 19:36 run_cctm_2018_12US1_listos.csh
```

Download the Input data from the S3 Bucket

Install aws command line

see Install AWS CLI

```
cd /shared/build
```

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

Install unzip and unzip file

```
sudo apt install zip
```

```
/usr/bin/unzip awscliv2.zip
```

```
sudo ./aws/install
```

output

```
You can now run: /usr/local/bin/aws --version
```

Note, you will need to add this path to your .cshrc

Edit .cshrc

vi ~/.cshrc

add the following to the path /usr/local/bin

Output:

```
# start .cshrc

umask 002

if ( ! $?LD_LIBRARY_PATH ) then
    setenv LD_LIBRARY_PATH /shared/build/netcdf/lib
else
    setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/shared/build/netcdf/lib
endif

set path = ($path /shared/build/netcdf/bin /shared/build/ioapi-3.2/Linux2_x86_64gfort /
↪opt/slurm/bin/ /usr/local/bin/ )

if ($?tcsh) then
   source /usr/share/modules/init/tcsh
else
   source /usr/share/modules/init/csh
endif
```

Install the input data using the s3 script

```
cd /shared/pcluster-cmaq/s3_scripts/
```

```
./s3_copy_nosign_cmaqv5.4-listos_cmas_opendata_to_shared.csh
```

Link the input data directory to the default location

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/data
ln -s /shared/data/12US1_LISTOS .
```

Note, this Virtual Machine does not have Slurm installed or configured.

Run CMAQ interactively using the following command:

First check to see how many cpus you have available on the machine.

```
lscpu
```

Output

```
CPU(s):                  4
  On-line CPU(s) list:   0-3
```

Verify that the run script is set to run on 4 cpus

```
   @ NPCOL  =  2; @ NPROW =  2
```

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts
```

```
./run_cctm_2018_12US1_listos.csh |& tee ./run_cctm_2018_12US1_listos.log
```

When the run has completed, record the timing of the two day benchmark.

```
tail -n 30 run_cctm_2018_12US1_listos.log
```

Output using a c6a.xlarge with hyperthreading on: c6a.xlarge 4vcpu, with hypthreading disabled, has 2 cores.

```

==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2018-08-05
End Day:   2018-08-07
Number of Simulation Days: 3
Domain Name:               2018_12Listos
Number of Grid Cells:      21875  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       4
   All times are in seconds.

Num  Day        Wall Time
01   2018-08-05   256.4
```

(continues on next page)

(continued from previous page)

```
02   2018-08-06   257.9
03   2018-08-07   262.8
     Total Time = 777.10
      Avg. Time = 259.03
```

The run time depends on whether the EC2 instances are using hyperthreading or not. For the following run, hyperthreading was disabled on a c6a.2xlarge. (8vcpu) but with hyperthreading disabled has 4 cores.

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2018-08-05
End Day:   2018-08-07
Number of Simulation Days: 3
Domain Name:               2018_12Listos
Number of Grid Cells:      21875  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       4
   All times are in seconds.

Num  Day        Wall Time
01   2018-08-05   166.3
02   2018-08-06   166.2
03   2018-08-07   169.5
     Total Time = 502.00
      Avg. Time = 167.33
```

If you upgrade this VM from a c6.xlarge to a c6.8xlarge, then you could run CMAQ interactively on 16 pes using the following command:

`cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts`

Edit the script to change NPCOL NPROW to 4 x 4

`cp run_cctm_2018_12US1_listos.csh run_cctm_2018_12US1_listos_16pe.csh`

`vi run_cctm_2018_12US1_listos_16pe.csh`

change to use 16 processors

```
   @ NPCOL  =  4; @ NPROW =  4
```

`./run_cctm_2018_12US1_listos_16pe.csh |& tee run_cctm_2018_12US1_listos_16pe.log`

Or if you were to upgrade to a c6.16xlarge, then you could run on 32 cores after copying the run script and editing the NPCOL and NPROW, and using the following command:

`cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts`

`./run_cctm_2018_12US1_listos_32pe.csh |& tee run_cctm_2018_12US1_listos_32pe.log`

Output

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2018-08-05
End Day:   2018-08-07
Number of Simulation Days: 3
Domain Name:               2018_12Listos
Number of Grid Cells:      21875  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       32
   All times are in seconds.

Num  Day        Wall Time
01   2018-08-05   80.6
02   2018-08-06   72.7
03   2018-08-07   76.3
     Total Time = 229.60
      Avg. Time = 76.53
```

Need to add command line options to upgrade the VM and to disable hyperthreading.

Install Software and run CMAQv5.4 on c7g-hpc7g for the 12km Listos Training Domain

Instructions are provided to build and install CMAQ on c7g.xlarge compute node installed from Canonical, Ubuntu, 22.04 LTS, arm64 jammy image build on 2023-05-16 Image that contains modules for git, openmpi and gcc. The compute node does not have a SLURM scheduler on it, so jobs are run interactively from the command line.

Instructions to install data and CMAQ libraries and model are provided along with sample run scripts to run CMAQ on 4 processors on a single c7g.xlarge instance.

This will provide users with experience using the AWS Console to create a Virtual Machine, select Operating System, select the size of the VM as c7g.xlarge vcpus, 8 GiB memory, using an SSH private key to login and install and run CMAQ.

Using this method, the user needs to be careful to start and stop the Virtual Machine and only have it run while doing the intial installation, and while running CMAQ. The full c7g.xlarge instance will incur charges as long as it is on, even if a job isn't running on it.

This is different than the Parallel Cluster, where if CMAQ is not running in the queue, then the Compute nodes are down, and not incurring costs.

Build CMAQv5.4+ on c7g.xlarge EC2 instance

Create a c7g.xlarge Virtual Machine

1. Login to AWS Console
2. Select Get Started with EC2
3. Select Launch Instance
4. Select Architecture - 64-bit (Arm)

5. Select Application and OS (Operating System) Images: Select Ubunutu 22.04 LTS(HVM), SSD Volume Type (the version of OS and the architecture selected (64-bit (Arm)) determines what packages are available from apt-get and that determines the version of software obtained.
6. Instance Type: Select c7g.xlarge ($0.145/hr)
7. Key pair - SSH public key, select existing key or create a new one.
8. Network settings - select default settings
9. Configure storage - select 100 GiB gp3 Root volume
10. Select Launch instance

user_guide_pcluster/aws_ec2_images/AWS_EC2_Virtual_Machine_Create.png

Login to the Virtual Machine

Change the permissions on the public key using command

`chmod 400 [your-key-name].pem`

Login to the Virtual Machine using ssh to the IP address using the public key.

`ssh -Y -i ./xxxxxxx_key.pem ubuntu@xx.xx.xx.xx`

Make the /shared directory

sudo mkdir /shared

Change the group and ownership of the shared directory

```
sudo chown ubuntu /shared
sudo chgrp ubuntu /shared
```

Change directories and verify that you see the /shared directory with Size of 100 GB

`cd /shared`

`df -h`

Output

```
df -h
Filesystem       Size  Used Avail Use% Mounted on
/dev/root         97G  1.6G   96G   2% /
tmpfs             16G     0   16G   0% /dev/shm
tmpfs            6.2G  876K  6.2G   1% /run
tmpfs            5.0M     0  5.0M   0% /run/lock
/dev/nvme0n1p15  105M  6.1M   99M   6% /boot/efi
```

(continues on next page)

(continued from previous page)

```
tmpfs            3.1G  4.0K  3.1G   1% /run/user/1000

```

Create subdirectories on /shared

Create a /shared/build, /shared/data and /shared/cyclecloud-cmaq directory

```
cd /shared
mkdir build
mkdir data
```

Check operating system version

```
lsb_release -a
```

output

```
No LSB modules are available.
Distributor ID:	Ubuntu
Description:	Ubuntu 22.04.2 LTS
Release:	22.04
Codename:	jammy

```

Install Environment Modules

```
sudo apt-get upgrade
sudo apt-get install environment-modules
```

Logout and then log back in to activate modules command

Verify module command works

```
 module list
```

Output:

No Modulefiles Currently Loaded.

```
module avail
```

Output:

```
------------------------------------------------------------------------------------- /
↪usr/share/modules/modulefiles -------------------------------------------------------
↪------------------------------
dot  module-git  module-info  modules  null  use.own
```

Set up build environment

Load the git module

```
module load module-git
```

If you do not see git available as a module, you may need to install it as follows:

sudo apt-get install git

Install Compilers and OpenMPI

```
sudo apt-get update
sudo apt-get install gcc-9
sudo apt-get  install gfortran-9
sudo apt-get install openmpi-bin openmpi-common libopenmpi-dev libgtk2.0-dev
sudo apt-get install tcsh
```

Compiler versions

Note, that there are more recent compilers, any CMAQ may run faster using them.

<a href="https://community.arm.com/arm-community-blogs/b/tools-software-ides-blog/posts/gcc-12"<ARM gcc-12 Performance optimization

Change shell to use tcsh

```
sudo usermod -s /usr/bin/tcsh ubuntu
```

Logout and log back in, then check the shell

```
echo $SHELL
```

output

```
/usr/bin/tcsh
```

Check available versions of compiler

```
dpkg --list | grep compiler
```

Choose gcc-9 and gfortran-9 as default compilers

```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 9
sudo update-alternatives --install /usr/bin/gfortran gfortran /usr/bin/gfortran-9 9
```

Check version of gcc

```
gcc --version
```

output

```
gcc --version
gcc (Ubuntu 9.5.0-1ubuntu1~22.04) 9.5.0
```

Check version of gfortran

```
gfortran --version
```

Output

```
GNU Fortran (Ubuntu 9.5.0-1ubuntu1~22.04) 9.5.0
```

Check version of OpenMPI

```
mpirun --version
```

output

```
mpirun (Open MPI) 4.1.2
```

Install Parallel Cluster CMAQ Repo

`cd /shared`

`git clone -b main https://github.com/CMASCenter/pcluster-cmaq.git`

Install and build netcdf C, netcdf Fortran, I/O API, and CMAQ

`cd /shared/pcluster-cmaq/install`

Install netcdf-C and netcdf-Fortran

```
./gcc_netcdf_singlevm.csh |& tee ./gcc_netcdf_singlevm.log
```

If successful, you will see the following output, that at the bottom shows what versions of the netCDF library were installed.

```

+-------------------------------------------------------------+
| Congratulations! You have successfully installed the netCDF |
| Fortran libraries.                                          |
|                                                             |
| You can use script "nf-config" to find out the relevant     |
| compiler options to build your application. Enter           |
|                                                             |
|     nf-config --help                                        |
|                                                             |
| for additional information.                                 |
|                                                             |
| CAUTION:                                                    |
|                                                             |
| If you have not already run "make check", then we strongly  |
| recommend you do so. It does not take very long.            |
|                                                             |
| Before using netCDF to store important data, test your      |
| build with "make check".                                    |
|                                                             |
| NetCDF is tested nightly on many platforms at Unidata       |
| but your platform is probably different in some ways.       |
|                                                             |
| If any tests fail, please see the netCDF web site:          |
| https://www.unidata.ucar.edu/software/netcdf/                |
|                                                             |
| NetCDF is developed and maintained at the Unidata Program   |
| Center. Unidata provides a broad array of data and software |
| tools for use in geoscience education and research.         |
| https://www.unidata.ucar.edu                                 |
+-------------------------------------------------------------+

make[3]: Leaving directory '/shared/build/netcdf-fortran-4.5.4'
make[2]: Leaving directory '/shared/build/netcdf-fortran-4.5.4'
make[1]: Leaving directory '/shared/build/netcdf-fortran-4.5.4'
netCDF 4.8.1
netCDF-Fortran 4.5.3
```

Install I/O API

```
./gcc_ioapi_singlevm.csh |& tee ./gcc_ioapi_singlevm.log
```

Find what operating system is on the system:

```
 cat /etc/os-release
```

Output

```
PRETTY_NAME="Ubuntu 22.04.2 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.2 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
```

Copy a file to set paths

cd /shared/pcluster-cmaq/install

cp dot.cshrc.singlevm ~/.cshrc

Exit cluster and log back in to activate the update shell, or use csh

Create Custom Environment Module for Libraries

There are two steps required to create your own custome module:

1. write a module file
2. add a line to your ~/.cshrc to update the MODULEPATH

Create a new custom module that will be loaded with:

```
module load ioapi-3.2/gcc-9.5-netcdf
```

Step 1: Create the module file for ioapi-3.2.

First, create a path to store the module file. The path must contain /Modules/modulefiles/ and should have the general form //Modules/modulefiles// where is typically numerical and is the actual module file.

```
mkdir -p /shared/build/Modules/modulefiles/ioapi-3.2
```

Next, create the module file and save it in the directory above.

```
cd /shared/build/Modules/modulefiles/ioapi-3.2
vim gcc-9.5-netcdf
```

Contents of gcc-9.5-netcdf:

```
#%Module

proc ModulesHelp { } {
   puts stderr "This module adds ioapi-3.2/gcc-9.5 to your path"
}

```

(continues on next page)

(continued from previous page)

```
module-whatis "This module adds ioapi-3.2/gcc-9.5 to your path\n"

set basedir "/shared/build/ioapi-3.2/"
prepend-path PATH "${basedir}/Linux2_x86_64gfort"
prepend-path LD_LIBRARY_PATH "${basedir}/ioapi/fixed_src"
```

The example module file above sets two evironment variables.

The modules update the PATH and LD_LIBRARY_PATH.

Step 2. Create the module file for netcdf-4.8.1

```
mkdir -p /shared/build/Modules/modulefiles/netcdf-4.8.1
```

Next, create the module file and save it in the directory above.

```
cd /shared/build/Modules/modulefiles/netcdf-4.8.1
vim gcc-9.5
```

Contents of gcc-9.5

```
#%Module

proc ModulesHelp { } {
   puts stderr "This module adds netcdf-4.8.1/gcc-9.5 to your path"
}

module-whatis "This module adds netcdf-4.8.1/gcc-9.5 to your path\n"

set basedir "/shared/build/netcdf"
prepend-path PATH "${basedir}/bin"
prepend-path LD_LIBRARY_PATH "${basedir}/lib"
module load mpi/openmpi-4.1.2
```

Step 3. Create the module file for mpi

```
mkdir -p /shared/build/Modules/modulefiles/mpi
```

Next, create the module file and save it in the directory above.

```
cd /shared/build/Modules/modulefiles/mpi
vim openmpi-4.1.2
```

Contents of openmpi-4.1.2

```
#%Module

proc ModulesHelp { } {
   puts stderr "This module adds mpi/openmpi-4.1.2 to your path"
}

module-whatis "This module adds mpi/openmpi-4.1.2 to your path\n"

set basedir "/usr/lib/x86_64-linux-gnu/openmpi/"
```

(continues on next page)

(continued from previous page)

```
prepend-path PATH "/usr/bin/"
prepend-path LD_LIBRARY_PATH "${basedir}/lib"
```

Step 4: Add the module path to MODULEPATH.

Now that the module file has been created, add the following line to your ~/.cshrc file so that it can be found:

```
module use --append /shared/build/Modules/modulefiles
```

Step 5: View the modules available after creation of the new module

The module avail command shows the paths to the module files on a given cluster.

```
module avail
```

Output

```
ioapi-3.2/gcc-9.5-netcdf  mpi/openmpi-4.1.2  netcdf-4.8.1/gcc-9.5
```

Step 4: Load the new modules

```
ioapi-3.2/gcc-9.5-netcdf  mpi/openmpi-4.1.2  netcdf-4.8.1/gcc-9.5
```

Find path for openmpi libraries

```
ompi_info --path libdir
```

output

```
 Libdir: /usr/lib/x86_64-linux-gnu/openmpi/lib
```

Find path for include files for openmpi

```
ompi_info --path incdir
```

output

```
Incdir: /usr/lib/x86_64-linux-gnu/openmpi/include
```

Edit the config_cmaq_singlevm.csh script to specify the paths for OpenMPI

Note, search for case gcc so that you edit the section of the file that is using the gcc compiler.

```
       setenv MPI_INCL_DIR     /usr/lib/x86_64-linux-gnu/openmpi/include             #>␣
↪MPI Include directory path
        setenv MPI_LIB_DIR     /usr/lib/x86_64-linux-gnu/openmpi/lib            #> MPI␣
↪Lib directory path
```

Install Python

```
sudo apt-get install python3 python3-pip
```

Check Version

```
python3 --version
Python 3.10.6
ip-172-31-27-148:/shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts> python3 -m pip --
↪version
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)

```

Install jupyter notebook.

```
pip install jupyterlab
```

Install and Build CMAQ

```
cd /shared/pcluster-cmaq/install
./gcc_cmaq54+_singlevm.csh |& tee ./gcc_cmaq54+_singlevm.log
```

Add compile option to makefile to get beyond a type mismatch error (note, this is only needed if you were using the gcc-11 compiler.

SKIP this step.

Add the following to the compile option: -fallow-argument-mismatch

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54_gcc
vi Makefile.gcc
```

Output:

```
 FSTD = -fallow-argument-mismatch -O3 -funroll-loops -finit-character=32 -Wtabs -
↪Wsurprising -ftree-vectorize  -ftree-loop-if-convert -finline-limit=512
```

Run make again

```
make |& tee Make.log
```

Verfify that the executable was successfully built.

`ls /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54_gcc/*.exe`

Output

```
/shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54_gcc/CCTM_v54.exe
```

Copy the run scripts from the repo to the run directory

`cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts`

`cp /shared/pcluster-cmaq/run_scripts/c6a/*.csh .`

List the scripts available

`ls -rlt *.csh*`

Output

```
-rwxrwxr-x 1 ubuntu ubuntu  8374 Jun  6 19:06 bldit_mech.csh
-rwxrwxr-x 1 ubuntu ubuntu 37040 Jun  6 19:06 bldit_cctm.csh
-rwxrwxr-x 1 ubuntu ubuntu 39500 Jun  6 19:06 run_cctm_cracmm_2018_4LISTOS1.csh
-rwxrwxr-x 1 ubuntu ubuntu 38493 Jun  6 19:06 run_cctm_Bench_2018_12NE3.csh
-rwxrwxr-x 1 ubuntu ubuntu 51329 Jun  6 19:06 run_cctm_Bench_2018_12NE3.WRFCMAQ.csh
-rwxrwxr-x 1 ubuntu ubuntu 38158 Jun  6 19:06 run_cctm_Bench_2016_12SE1.csh
-rwxrwxr-x 1 ubuntu ubuntu 39265 Jun  6 19:06 run_cctm_2016_12US1.csh
-rwxrwxr-x 1 ubuntu ubuntu 37458 Jun  6 19:06 run_cctm_2015_HEMI.csh
-rwxrwxr-x 1 ubuntu ubuntu 37583 Jun  6 19:06 run_cctm_2010_4CALIF1.csh
-rwxrwxr-x 1 ubuntu ubuntu  38460 Jun  6 19:36 run_cctm_2018_12US1_listos.csh
```

Download the Input data from the S3 Bucket

Install aws command line

see Install AWS CLI

`cd /shared/build`

`curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`

Install unzip and unzip file

`sudo apt install zip`

`/usr/bin/unzip awscliv2.zip`

`sudo ./aws/install`

output

```
You can now run: /usr/local/bin/aws --version
```

Note, you will need to add this path to your .cshrc

Edit .cshrc

vi ~/.cshrc

add the following to the path /usr/local/bin

Output:

```
# start .cshrc

umask 002

if ( ! $?LD_LIBRARY_PATH ) then
    setenv LD_LIBRARY_PATH /shared/build/netcdf/lib
else
    setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/shared/build/netcdf/lib
endif

set path = ($path /shared/build/netcdf/bin /shared/build/ioapi-3.2/Linux2_x86_64gfort /
↪opt/slurm/bin/ /usr/local/bin/ )

if ($?tcsh) then
   source /usr/share/modules/init/tcsh
else
   source /usr/share/modules/init/csh
endif
```

Install the input data using the s3 script

`cd /shared/pcluster-cmaq/s3_scripts/`

`./s3_copy_nosign_cmaqv5.4-listos_cmas_opendata_to_shared.csh`

Link the input data directory to the default location

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/data
ln -s /shared/data/12US1_LISTOS .
```

Note, this Virtual Machine does not have Slurm installed or configured.

Run CMAQ interactively using the following command:

First check to see how many cpus you have available on the machine.

```
lscpu
```

Output

```
CPU(s):                  4
  On-line CPU(s) list:   0-3
```

Verify that the run script is set to run on 4 cpus

```
   @ NPCOL  =  2; @ NPROW =  2
```

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts
```

```
./run_cctm_2018_12US1_listos.csh |& tee ./run_cctm_2018_12US1_listos.log
```

When the run has completed, record the timing of the two day benchmark.

```
tail -n 30 run_cctm_2018_12US1_listos.log
```

Output using a c6a.xlarge with hyperthreading on: c6a.xlarge 4vcpu, with hypthreading disabled, has 2 cores.

```

==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2018-08-05
End Day:   2018-08-07
Number of Simulation Days: 3
Domain Name:               2018_12Listos
Number of Grid Cells:      21875  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       4
   All times are in seconds.

Num  Day        Wall Time
01   2018-08-05   256.4
02   2018-08-06   257.9
03   2018-08-07   262.8
     Total Time = 777.10
      Avg. Time = 259.03
```

The run time depends on whether the EC2 instances are using hyperthreading or not. For the following run, hyperthreading was disabled on a c6a.2xlarge. (8vcpu) but with hyperthreading disabled has 4 cores.

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2018-08-05
End Day:   2018-08-07
Number of Simulation Days: 3
Domain Name:               2018_12Listos
Number of Grid Cells:      21875  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       4
   All times are in seconds.

Num  Day        Wall Time
01   2018-08-05   166.3
02   2018-08-06   166.2
03   2018-08-07   169.5
     Total Time = 502.00
      Avg. Time = 167.33

```

If you upgrade this VM from a c6.xlarge to a c6.8xlarge, then you could run CMAQ interactively on 16 pes using the following command:

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts
```

Edit the script to change NPCOL NPROW to 4 x 4

```
cp run_cctm_2018_12US1_listos.csh run_cctm_2018_12US1_listos_16pe.csh
```

```
vi run_cctm_2018_12US1_listos_16pe.csh
```

change to use 16 processors

```
   @ NPCOL  =  4; @ NPROW =  4
```

```
./run_cctm_2018_12US1_listos_16pe.csh |& tee run_cctm_2018_12US1_listos_16pe.log
```

Or if you were to upgrade to a c6.16xlarge, then you could run on 32 cores after copying the run script and editing the NPCOL and NPROW, and using the following command:

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts
```

```
./run_cctm_2018_12US1_listos_32pe.csh |& tee run_cctm_2018_12US1_listos_32pe.log
```

Output

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2018-08-05
End Day:   2018-08-07
Number of Simulation Days: 3
Domain Name:               2018_12Listos
Number of Grid Cells:      21875  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       32
   All times are in seconds.

Num  Day        Wall Time
01   2018-08-05   80.6
02   2018-08-06   72.7
03   2018-08-07   76.3
     Total Time = 229.60
      Avg. Time = 76.53
```

Need to add command line options to upgrade the VM and to disable hyperthreading.

Install I/O API libraries that support HDF5

This is required in order to:

1. Run CMAQ using the compressed netCDF-4 input files provided on the S3 bucket or
2. Convert the *.nc4 files to *.nc files (to uncompressed classic netCDF-3 input files)

First build HDF5 libraries, then build netCDF-C, netCDF-Fortran

```
cd /shared/pcluster-cmaq
./gcc11_install_hdf5.csh
```

Upgrade to run CMAQ on larger EC2 Instance

Save the AMI and create a new VM using a larger c6a.8xlarge (with 32 processors)

Requires access to the AWS Web Interface (I will look for insructions on how to do this from the aws command line, but I don't currently have a method for this.)

Use the AWS Console to Stop the Image

add screenshot

Use the AWS Console to Create a new AMI

add screenshot

Check to see that the AMI has been created by examining the status. Wait for the status to change from Pending to Available.

Use the newly created AMI to launch a new Single VM using a larger EC2 instance.

Launch a new instance using the AMI with the software loaded and request a spot instance for the c6a.8xlarge EC2 instance

Load the modules

Test running the listos domain on 32 processors

Output

```
    Processing Day/Time [YYYYDDD:HHMMSS]: 2017357:235600
      Which is Equivalent to (UTC): 23:56:00 Saturday,  Dec. 23, 2017
      Time-Step Length (HHMMSS): 000400
                VDIFF completed...       3.6949 seconds
               COUPLE completed...       0.3336 seconds
                 HADV completed...       1.8413 seconds
                 ZADV completed...       0.5154 seconds
                HDIFF completed...       0.4116 seconds
             DECOUPLE completed...       0.0696 seconds
                 PHOT completed...       0.7443 seconds
              CLDPROC completed...       2.4009 seconds
                 CHEM completed...       1.3362 seconds
                 AERO completed...       1.3210 seconds
           Master Time Step
           Processing completed...      12.6698 seconds

     =--> Data Output completed...       0.9872 seconds


    ==============================================
```

(continues on next page)

(continued from previous page)

```
     |>---   PROGRAM COMPLETED SUCCESSFULLY   ---<|
     ==============================================
     Date and time 0:00:00   Dec. 24, 2017  (2017358:000000)

     The elapsed time for this simulation was    3389.0 seconds.

315644.552u 1481.008s 56:29.98 9354.7%        0+0k 33221248+26871200io 9891pf+0w

CMAQ Processing of Day 20171223 Finished at Wed Jun  7 02:25:47 UTC 2023

\\\\\=====\\\\\=====\\\\\=====\\\\\=====/////=====/////=====/////=====/////



==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2018-08-05
End Day:   2018-08-07
Number of Simulation Days: 3
Domain Name:               2018_12Listos
Number of Grid Cells:      21875  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       32
   All times are in seconds.

Num  Day        Wall Time
01   2018-08-05   80.6
02   2018-08-06   72.7
03   2018-08-07   76.3
     Total Time = 229.60
      Avg. Time = 76.53
```

Run CMAQv5.4 for the full 12US1 Domain on c6a.48xlarge

Download the full 12US1 Domain that is netCDF4 compressed and convert it to classic netCDF-3 compression.

Note: I first tried running this domain on the c6a.8xlarge on 32 processors. The model failed, with a signal 9 - likely not enough memory available to run the model.

I re-saved the AMI and launched a c6a.48xlarge with 192 vcpus, running as spot instance.

Spot Pricing cost for Linux in US East Region

c6a.48xlarge $6.4733 per Hour

Run utility to uncompress hdf5 *.nc4 files and save as classic *.nc files

May need to look at disabling hyperthreading at runtime.

Disable Hyperthreading

Increased disk space on /shared to 500 GB

Ran out of disk space when trying to run the full 12US1 domain, so it is necessary to increase the size. You can do this in the AWS Web Interface without stopping the instance.

Expanded the root volume to 500 GB, and increased the throughput to 1000 MB/s and then expanded it using these instructions, and then resized it.

Recognize Expanded Volume

Rerunning the 12US1 case on 8x12 processors - for total of 96 processors.

It takes about 13 minutes of initial I/O prior to the model starting.

Successful run output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       96
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   3395.1
02   2017-12-23   3389.0
     Total Time = 6784.10
      Avg. Time = 3392.05
```

Note, this run time is slower than a single node of the Parallel Cluster using the HPC6a.48xlarge (total time = 5000 seconds). Note the 12US1 domain is larger than the 12US2 domain that was used for the HPC6a.48xlarge benchmarks. It would be good to do another benchmark for 12US1 using HPC6a.48xlarge a compute node that is configured for HPC by AWS. AWS turns off hyperthreading by default for HPC6a.48xlarge, and there may be other optimizations for HPC applications (disk/networking/cpu).

4.4.2 Install CMAQv5.4 on ParallelCluster (optional)

- Create the ParallelCluster with the base Ubutu OS using c7g.large head node and c7g.16xlarge as the compute node.
- Learn how to install CMAQ software and underlying libraries, copy input data, and run CMAQ.

Notice

Skip this tutorial if you do not want to learn how to install the CMAQv5.4 software and proceed to the post-processing and QA instructions. Note, you may wish to build the underlying libraries and CMAQ and code if you wish to create a ParallelCluster using a different family of compute nodes, such as the c6gn.16xlarge compute nodes AMD Graviton.

CMAQv5.4 on Parallel Cluster Advanced Tutorial (optional)

Configure Parallel Cluster

Use ParallelCluster with default Ubuntu OS using hpc7g.large head node and hpc7g.16xlarge compute node.

Step by step instructions to configuring and running a ParallelCluster for the CMAQ 12US1 benchmark with instructions to install the libraries and software.

Notice

Skip this tutorial if you successfully completed the CMAQv5.4 on Parallel Cluster Intermediate Tutorial. Unless you need to build the CMAQ libraries and code and run on a different family of compute nodes, such as the c6gn.16xlarge compute nodes AMD Graviton.

Activate the virtual environment to use the ParallelCluster command line

```
source ~/apc-ve/bin/activate
source ~/.nvm/nvm.sh
```

Upgrade to get the latest version of ParallelCluster

`python3 -m pip install --upgrade "aws-parallelcluster"`

Verify that the ParallelCluster AWS CLI is installed by checking the version

`pcluster version`

Output:

```
                {
  "version": "3.6.0"
}
```

Use an existing yaml file from the git repo to create a ParallelCluster

`cd /your/local/machine/install/path/`

Use a configuration file from the github repo that was cloned to your local machine

`git clone -b main https://github.com/CMASCenter/pcluster-cmaq.git pcluster-cmaq`

`cd pcluster-cmaq/yaml`

Edit the hpc7g.16xlarge.ebs_unencrypted_installed_public_ubuntu2004.fsx_import.yaml

```
vi hpc7g.16xlarge.ebs_unencrypted_installed_public_ubuntu2004.fsx_import.yaml
```

Note:

1. the hpc7g-16xlarge*.yaml is configured to use ONDEMAND instance pricing for the compute nodes.
2. the hpc7g-16xlarge*.yaml is configured to the the hpc7g.16xlarge as the compute node, with up to 10 compute nodes, specified by MaxCount: 10.
3. the hpc7g-16xlarge*.yaml is configured to disable multithreading (This option restricts the computing to CPUS rather than allowing the use of all virtual CPUS. (192 virtual cpus reduced to 96 cpus)
4. the hpc7g-16xlarge*.yaml is configured to enable the setting of a placement group to allow low inter-node latency
5. the hpc7g-16xlarge*.yaml is configured to enables the elastic fabric adapter
6. given this yaml configuration, the maximum number of PEs that could be used to run CMAQ is 64 cpus x 10 = 640, the max settings for NPCOL, NPROW is NPCOL = 32, NPROW = 20 or NPCOL=20, NPROW=32 in the CMAQ run script. Note: CMAQ does not scale well beyond 2-3 compute nodes.

Replace the key pair and subnet ID in the hpc7g.16xlarge*.yaml file with the values created when you configured the demo cluster

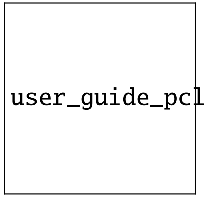
```
Region: us-east-1
Image:
  Os: ubuntu2004
HeadNode:
  InstanceType: c7g.large
  Networking:
    SubnetId: subnet-xx-xx-xx           << replace
  DisableSimultaneousMultithreading: true
  Ssh:
    KeyName: your_key                     << replace
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: queue1
      CapacityType: ONDEMAND
      Networking:
        SubnetIds:
          - subnet-xx-xx-x         x    << replace
        PlacementGroup:
          Enabled: true
      ComputeResources:
        - Name: compute-resource-1
          InstanceType: hpc7g.16xlarge
          MinCount: 0
          MaxCount: 10
          DisableSimultaneousMultithreading: true
          Efa:
            Enabled: true
            GdrSupport: false
SharedStorage:
  - MountDir: /shared
    Name: ebs-shared
```

(continues on next page)

(continued from previous page)

```
    StorageType: Ebs
  - MountDir: /fsx
    Name: name2
    StorageType: FsxLustre
    FsxLustreSettings:
      StorageCapacity: 1200
```

The Yaml file for the hpc7g.16xlarge contains the settings as shown in the following diagram.



user_guide_pcluster/yml_plots/hpc7g.16xlarge-yaml.png

Figure 1. Diagram of YAML file used to configure a ParallelCluster with a hpc7g.large head node and hpc7g.16xlarge compute nodes using ONDEMAND

Create the hpc7g.16xlarge pcluster

```
pcluster create-cluster --cluster-configuration hpc7g.16xlarge.ebs_unencrypted_installed_public_ubuntu2
fsx_import.yaml --cluster-name cmaq --region us-east-1
```

Check on status of cluster

```
pcluster describe-cluster --region=us-east-1 --cluster-name cmaq
```

After 5-10 minutes, you see the following status: “clusterStatus”: “CREATE_COMPLETE”

Login to cluster

Note: Replace the your-key.pem with your Key Pair.

```
pcluster ssh -v -Y -i ~/your-key.pem --region=us-east-1 --cluster-name cmaq
```

Note: Notice that the hpc7g.16xlarge yaml configuration file contains a setting for PlacementGroup.

```
PlacementGroup:
          Enabled: true
```

A placement group is used to get the lowest inter-node latency.

A placement group guarantees that your instances are on the same networking backbone.

Check to make sure elastic network adapter (ENA) is enabled

```
modinfo ena
```

```
lspci
```

Check what modules are available on the cluster

```
module avail
```

Output:

```
module avail
-----------------------------------------------------------------------------------------
↪------- /usr/share/modules/modulefiles ------------------------------------------------
↪-------------------------------------------------
armpl/21.0.0  dot  libfabric-aws/1.17.1  module-git  module-info  modules  null  openmpi/
↪4.1.5  use.own
```

Load the openmpi module

```
module load openmpi/4.1.5
```

Load the Libfabric module

```
module load libfabric-aws/1.17.1
```

Verify the gcc compiler version is greater than 8.0

```
gcc --version
```

output:

```
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

See also:

Link to the AWS Enhanced Networking Adapter Documentation

See also:

ParallelCluster User Manual

Install CMAQ sofware and libraries on ParallelCluster version 3.6

note, when you update the version of ParallelCluster, you often get different versions of the openmpi, libfabric, and gcc compilers and environment modules.

Login to updated cluster

Note: Replace the your-key.pem with your Key Pair.

```
pcluster ssh -v -Y -i ~/your-key.pem --cluster-name cmaq
```

Change shell to use .tcsh

Note: This command depends on what OS you have installed on the ParallelCluster

```
sudo usermod -s /bin/tcsh ubuntu
```

or

```
sudo usermod -s /bin/tcsh centos
```

Log out and log back in to have the tcsh shell be active

```
exit
```

```
pcluster ssh -v -Y -i ~/your-key.pem --cluster-name cmaq
```

Check to see the tcsh shell is default

```
echo $SHELL
```

Reload the environment modules

```
module load openmpi/4.1.5 libfabric-aws/1.17.1
```

The following instructions assume that you will be installing the software to a /shared/build directory

```
mkdir /shared/build
```

Install the pcluster-cmaq git repo to the /shared directory

```
cd /shared
```

```
git clone -b main https://github.com/CMASCenter/pcluster-cmaq.git pcluster-cmaq
```

Check to make sure elastic network adapter (ENA) is enabled

```
modinfo ena
```

```
lspci
```

Verify the gcc compiler version is greater than 8.0

```
gcc --version
```

Output:

```
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Change directories to install and build the libraries and CMAQ

```
cd /shared/pcluster-cmaq/install
```

The install process currently uses .csh scripts to install the libraries.

An alternative is to keep a copy of the source code for netcdf-C and netcdf-Fortran and all of the other underlying code on an S3 bucket and to use custom bootstrap actions to build the sofware as the ParallelCluster is provisioned.

The following link provides instructions on how to create a custom bootstrap action to pre-load software from an S3 bucket to the ParallelCluster at the time that the cluster is created.

Custom Bootstrap Actions

Build netcdf C and netcdf F libraries - these scripts work for the gcc 8+ compiler

Note, if this script fails, it is typically because NCAR has released a new version of netCDF C or Fortran, so the old version is no longer available, or if they have changed the name or location of the download file.

```
./gcc_netcdf_cluster.csh
```

A .cshrc script with LD_LIBRARY_PATH was copied to your home directory, enter the shell again and check environment variables that were set using

```
cat ~/.cshrc
```

If the .cshrc was not created use the following command to create it

```
cp dot.cshrc.pcluster.v36 ~/.cshrc
```

Execute the shell to activate it

```
csh
```

```
env
```

Verify that you see the following setting

```
env | grep LD_LIBRARY_PATH
```

Output:

```
LD_LIBRARY_PATH=/opt/amazon/openmpi/lib64:/shared/build/netcdf/lib:/shared/build/netcdf/
↪lib
```

Build I/O API library

```
./gcc_ioapi_cluster.v36.csh
```

Build CMAQ

```
./gcc_cmaq54+_pcluster.csh
```

Check to confirm that the cmaq executable has been built

```
ls /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/*.exe
```

Install netCDF libraries that use HDF5 and support nc4 compressed files

Need to have this version of the library installed to uncompress the *.nc4 data using the indexer.csh script.

```
cd /shared/pcluster-cmaq/install
```

```
./gcc_install_hdf5.csh
```

Create Custom Environment Module for Libraries

There are two steps required to create your own custome module:

1. write a module file
2. add a line to your ~/.cshrc to update the MODULEPATH

Create a new custom module that will be loaded with:

```
module load ioapi-3.2/gcc-9.5-netcdf
```

Step 1: Create the module file for ioapi-3.2.

First, create a path to store the module file. The path must contain /Modules/modulefiles/ and should have the general form //Modules/modulefiles// where is typically numerical and is the actual module file.

```
mkdir -p /shared/build/Modules/modulefiles/ioapi-3.2
```

Next, create the module file and save it in the directory above.

```
cd /shared/build/Modules/modulefiles/ioapi-3.2
vim gcc-9.5-netcdf
```

Contents of gcc-9.5-netcdf:

```
#%Module

proc ModulesHelp { } {
   puts stderr "This module adds ioapi-3.2/gcc-9.5 to your path"
}

module-whatis "This module adds ioapi-3.2/gcc-9.5 to your path\n"

set basedir "/shared/build/ioapi-3.2/"
prepend-path PATH "${basedir}/Linux2_x86_64gfort"
prepend-path LD_LIBRARY_PATH "${basedir}/ioapi/fixed_src"
```

The example module file above sets two evironment variables.

The modules update the PATH and LD_LIBRARY_PATH.

Step 2. Create the module file for netcdf-4.8.1

```
mkdir -p /shared/build/Modules/modulefiles/netcdf-4.8.1
```

Next, create the module file and save it in the directory above.

```
cd /shared/build/Modules/modulefiles/netcdf-4.8.1

vim gcc-9.5
```

Contents of gcc-9.5

```
#%Module

proc ModulesHelp { } {
   puts stderr "This module adds netcdf-4.8.1/gcc-9.5 to your path"
}

module-whatis "This module adds netcdf-4.8.1/gcc-9.5 to your path\n"

set basedir "/shared/build/netcdf"
prepend-path PATH "${basedir}/bin"
prepend-path LD_LIBRARY_PATH "${basedir}/lib"
module load openmpi
```

Step 3: Add the module path to MODULEPATH.

Now that the module file has been created, add the following line to your ~/.cshrc file so that it can be found:

```
module use --append /shared/build/Modules/modulefiles
```

Step 4. Source your shell

```
csh
```

Step 5: View the modules available after creation of the new module

The module avail command shows the paths to the module files on a given cluster.

```
module avail
```

Output

```
---------------------------------------------- /usr/share/modules/modulefiles -----------
↪-------------------------------------------------------
armpl/21.0.0  dot  libfabric-aws/1.17.1  module-git  module-info  modules  null  openmpi/
↪4.1.5  use.own

---------------------------------------------- /shared/build/Modules/modulefiles --------
↪--------------------------------------------------------
ioapi-3.2/gcc-9.5-netcdf  netcdf-4.8.1/gcc-9.5

```

Step 6: Load the modules

```
 module load ioapi-3.2/gcc-9.5-netcdf  netcdf-4.8.1/gcc-9.5
```

Step 7: List the loaded modules

```
module list
```

Output:

```
Currently Loaded Modulefiles:
 1) openmpi/4.1.5   2) libfabric-aws/1.17.1   3) ioapi-3.2/gcc-9.5-netcdf   4) netcdf-4.
↪8.1/gcc-9.5
```

The following step is not typically needed.

Install gh following these instructions

```
type -p curl >/dev/null || (sudo apt update && sudo apt install curl -y)
curl -fsSL https://cli.github.com/packages/githubcli-archive-keyring.gpg | sudo dd of=/
↪usr/share/keyrings/githubcli-archive-keyring.gpg \
&& sudo chmod go+r /usr/share/keyrings/githubcli-archive-keyring.gpg \
&& echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/githubcli-
↪archive-keyring.gpg] https://cli.github.com/packages stable main" | sudo tee /etc/apt/
↪sources.list.d/github-cli.list > /dev/null \
&& sudo apt update \
&& sudo apt install gh -y
```

Use gh authentication

Run CMAQ using hpc7g.16xlarge compute nodes

Verify that you have an updated set of run scripts from the pcluster-cmaq repo

```
cd /shared/pcluster-cmaq/run_scripts/hpc7g.16xlarge/
```

```
ls -lrt run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.1x64.ncclassic.csh
```

```
diff run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.1x64.ncclassic.csh /shared/pcluster-cmaq/run_scripts/cmaqv54+/
```

If they don't exist or are not identical, then copy the run scripts from the repo

```
cp /shared/pcluster-cmaq/run_scripts/hpc7g.16xlarge/run_cctm* /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
```

Verify that the input data is imported to /fsx from the S3 Bucket

```
ls /fsx
```

Preloading the files

Amazon FSx copies data from your Amazon S3 data repository when a file is first accessed. CMAQ is sensitive to latencies, so it is best to preload contents of individual files or directories using the following command:

```
cd /fsx
nohup find /fsx/ -type f -print0 | xargs -0 -n 1 sudo lfs hsm_restore &
```

Create a /fsx/data and /fsx/data/output directory

```
mkdir -p /fsx/data/output
```

Link the data to what is being used in the run scriptso

```
setenv INPDIR  /$DISK/data/CMAQ_Modeling_Platform_2018/2018_12US1  #Input Directory
```

```
cd /fsx/data
ln -s ../CMAQv5.4_2018_12US1_Benchmark_2Day_Input ./CMAQ_Modeling_Platform_2018
```

Run the 12US1 Domain on 32 pes

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/

sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.1x32.ncclassic.csh`
```

Note, it will take about 3-5 minutes for the compute notes to start up. This is reflected in the Status (ST) of CF (configuring)

Check the status in the queue

```
squeue -u ubuntu
```

Output:

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 2    queue1     CMAQ   ubuntu  CF                1 queue1-dy-compute-
↪resource-1-[1]
```

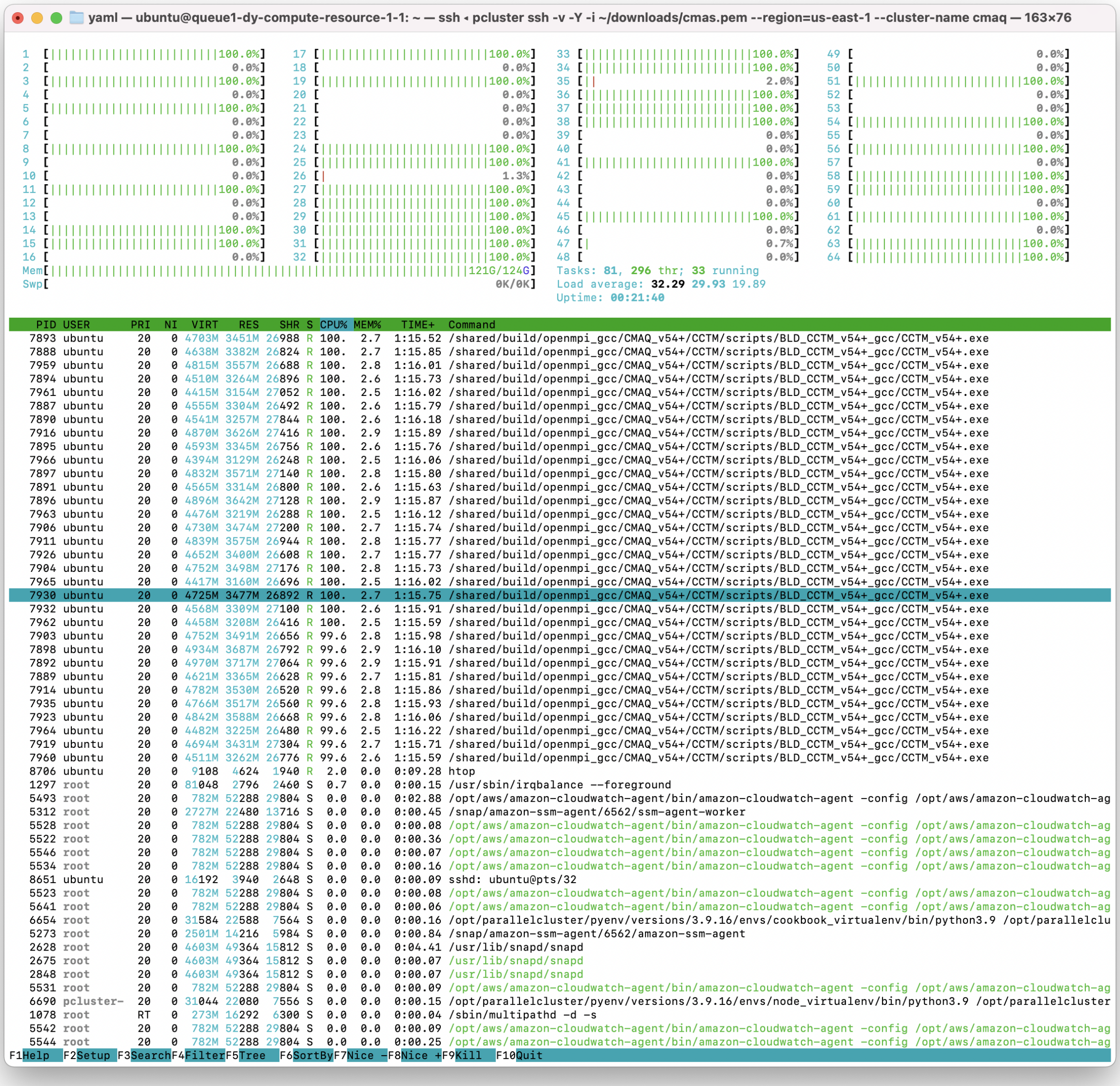
After 5 minutes the status will change once the compute nodes have been created and the job is running

```
squeue -u ubuntu
```

Output:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 2    queue1     CMAQ   ubuntu  R      19:30      1 queue1-dy-compute-
↪resource-1-1
```

The 64 pe job was crashing due to exceeding the memory available. Using 32 cores, there is more memory per core available. On 32 cores, the htop output shows the job using 121 out of 124 GB of memory.



yaml — ubuntu@queue1-dy-compute-resource-1-1: ~ — ssh ◂ pcluster ssh -v -Y -i ~/downloads/cmas.pem --region=us-east-1 --cluster-name cmaq — 163×76

```
1  [||||||||||||||||||||||||100.0%]   17 [||||||||||||||||||||||||100.0%]   33 [||||||||||||||||||||||||100.0%]   49 [                          0.0%]
2  [                          0.0%]   18 [                          0.0%]   34 [||||||||||||||||||||||||100.0%]   50 [                          0.0%]
3  [||||||||||||||||||||||||100.0%]   19 [||||||||||||||||||||||||100.0%]   35 [||                        2.0%]   51 [||||||||||||||||||||||||100.0%]
4  [                          0.0%]   20 [                          0.0%]   36 [||||||||||||||||||||||||100.0%]   52 [                          0.0%]
5  [||||||||||||||||||||||||100.0%]   21 [                          0.0%]   37 [||||||||||||||||||||||||100.0%]   53 [                          0.0%]
6  [                          0.0%]   22 [                          0.0%]   38 [||||||||||||||||||||||||100.0%]   54 [||||||||||||||||||||||||100.0%]
7  [                          0.0%]   23 [                          0.0%]   39 [                          0.0%]   55 [                          0.0%]
8  [||||||||||||||||||||||||100.0%]   24 [||||||||||||||||||||||||100.0%]   40 [                          0.0%]   56 [||||||||||||||||||||||||100.0%]
9  [                          0.0%]   25 [||||||||||||||||||||||||100.0%]   41 [||||||||||||||||||||||||100.0%]   57 [                          0.0%]
10 [                          0.0%]   26 [|                         1.3%]   42 [                          0.0%]   58 [||||||||||||||||||||||||100.0%]
11 [||||||||||||||||||||||||100.0%]   27 [||||||||||||||||||||||||100.0%]   43 [                          0.0%]   59 [||||||||||||||||||||||||100.0%]
12 [                          0.0%]   28 [||||||||||||||||||||||||100.0%]   44 [                          0.0%]   60 [                          0.0%]
13 [                          0.0%]   29 [||||||||||||||||||||||||100.0%]   45 [||||||||||||||||||||||||100.0%]   61 [||||||||||||||||||||||||100.0%]
14 [||||||||||||||||||||||||100.0%]   30 [||||||||||||||||||||||||100.0%]   46 [                          0.0%]   62 [                          0.0%]
15 [||||||||||||||||||||||||100.0%]   31 [||||||||||||||||||||||||100.0%]   47 [|                         0.7%]   63 [||||||||||||||||||||||||100.0%]
16 [                          0.0%]   32 [||||||||||||||||||||||||100.0%]   48 [                          0.0%]   64 [||||||||||||||||||||||||100.0%]
Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||121G/124G]   Tasks: 81, 296 thr; 33 running
Swp[                                                               0K/0K]   Load average: 32.29 29.93 19.89
                                                                            Uptime: 00:21:40

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 7893 ubuntu     20   0 4703M 3451M 26988 R 100.  2.7  1:15.52 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7888 ubuntu     20   0 4638M 3382M 26824 R 100.  2.7  1:15.85 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7959 ubuntu     20   0 4815M 3557M 26688 R 100.  2.8  1:16.01 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7894 ubuntu     20   0 4510M 3264M 26896 R 100.  2.6  1:15.73 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7961 ubuntu     20   0 4415M 3154M 27052 R 100.  2.5  1:16.02 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7887 ubuntu     20   0 4555M 3304M 26492 R 100.  2.6  1:15.79 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7890 ubuntu     20   0 4541M 3257M 27844 R 100.  2.6  1:16.18 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7916 ubuntu     20   0 4870M 3626M 27416 R 100.  2.9  1:15.89 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7895 ubuntu     20   0 4593M 3345M 26756 R 100.  2.6  1:15.76 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7966 ubuntu     20   0 4394M 3129M 26248 R 100.  2.5  1:16.06 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7897 ubuntu     20   0 4832M 3571M 27140 R 100.  2.8  1:15.80 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7891 ubuntu     20   0 4565M 3314M 26800 R 100.  2.6  1:15.63 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7896 ubuntu     20   0 4896M 3642M 27128 R 100.  2.9  1:15.87 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7963 ubuntu     20   0 4476M 3219M 26288 R 100.  2.5  1:16.12 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7906 ubuntu     20   0 4730M 3474M 27200 R 100.  2.7  1:15.74 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7911 ubuntu     20   0 4839M 3575M 26944 R 100.  2.8  1:15.77 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7926 ubuntu     20   0 4652M 3400M 26608 R 100.  2.7  1:15.77 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7904 ubuntu     20   0 4752M 3498M 27176 R 100.  2.8  1:15.73 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7965 ubuntu     20   0 4417M 3160M 26696 R 100.  2.5  1:16.02 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7930 ubuntu     20   0 4725M 3477M 26892 R 100.  2.7  1:15.75 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7932 ubuntu     20   0 4568M 3309M 27100 R 100.  2.6  1:15.91 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7962 ubuntu     20   0 4458M 3208M 26416 R 100.  2.5  1:15.59 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7903 ubuntu     20   0 4752M 3491M 26656 R 99.6  2.8  1:15.98 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7898 ubuntu     20   0 4934M 3687M 26792 R 99.6  2.9  1:16.10 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7892 ubuntu     20   0 4970M 3717M 27064 R 99.6  2.9  1:15.91 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7889 ubuntu     20   0 4621M 3365M 26628 R 99.6  2.7  1:15.81 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7914 ubuntu     20   0 4782M 3530M 26520 R 99.6  2.8  1:15.86 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7935 ubuntu     20   0 4766M 3517M 26560 R 99.6  2.8  1:15.93 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7923 ubuntu     20   0 4842M 3588M 26668 R 99.6  2.8  1:16.06 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7964 ubuntu     20   0 4482M 3225M 26480 R 99.6  2.5  1:16.22 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7919 ubuntu     20   0 4694M 3431M 27304 R 99.6  2.7  1:15.71 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 7960 ubuntu     20   0 4511M 3262M 26776 R 99.6  2.6  1:15.59 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
 8706 ubuntu     20   0  9108  4624  1940 R  2.0  0.0  0:09.28 htop
 1297 root       20   0 81048  2796  2460 S  0.7  0.0  0:00.15 /usr/sbin/irqbalance --foreground
 5493 root       20   0  782M 52288 29804 S  0.0  0.0  0:02.88 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-ag
 5312 root       20   0 2727M 22480 13716 S  0.0  0.0  0:00.45 /snap/amazon-ssm-agent/6562/ssm-agent-worker
 5528 root       20   0  782M 52288 29804 S  0.0  0.0  0:00.08 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-ag
 5522 root       20   0  782M 52288 29804 S  0.0  0.0  0:00.36 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-ag
 5546 root       20   0  782M 52288 29804 S  0.0  0.0  0:00.07 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-ag
 5534 root       20   0  782M 52288 29804 S  0.0  0.0  0:00.16 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-ag
 8651 ubuntu     20   0 16192  3940  2648 S  0.0  0.0  0:00.09 sshd: ubuntu@pts/32
 5523 root       20   0  782M 52288 29804 S  0.0  0.0  0:00.08 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-ag
 5641 root       20   0  782M 52288 29804 S  0.0  0.0  0:00.06 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-ag
 6654 root       20   0 31584 22588  7564 S  0.0  0.0  0:00.16 /opt/parallelcluster/pyenv/versions/3.9.16/envs/cookbook_virtualenv/bin/python3.9 /opt/parallelclu
 5273 root       20   0 2501M 14216  5984 S  0.0  0.0  0:00.84 /snap/amazon-ssm-agent/6562/amazon-ssm-agent
 2628 root       20   0 4603M 49364 15812 S  0.0  0.0  0:04.41 /usr/lib/snapd/snapd
 2675 root       20   0 4603M 49364 15812 S  0.0  0.0  0:00.07 /usr/lib/snapd/snapd
 2848 root       20   0 4603M 49364 15812 S  0.0  0.0  0:00.07 /usr/lib/snapd/snapd
 5531 root       20   0  782M 52288 29804 S  0.0  0.0  0:00.09 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-ag
 6690 pcluster-  20   0 31044 22080  7556 S  0.0  0.0  0:00.15 /opt/parallelcluster/pyenv/versions/3.9.16/envs/node_virtualenv/bin/python3.9 /opt/parallelcluster
 1078 root       RT   0  273M 16292  6300 S  0.0  0.0  0:00.04 /sbin/multipathd -d -s
 5542 root       20   0  782M 52288 29804 S  0.0  0.0  0:00.09 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-ag
 5544 root       20   0  782M 52288 29804 S  0.0  0.0  0:00.25 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-ag
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

The 32 pe job should take xx minutes to run (xx minutes per day)

check on the status of the cluster using CloudWatch

(optional)

```
<a href="https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#dashboards:
↪name=cmaq-us-east-1">Cloudwatch Dashboard</a>
<a href="https://aws.amazon.com/blogs/compute/monitoring-dashboard-for-aws-
↪parallelcluster/">Monitoring Dashboard for ParallelCluster</a>
```

check the timings while the job is still running using the following command

```
cd output_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_1x32_classic
```

```
grep 'Processing completed' CTM_LOG_001*
```

Output:

```
            Processing completed...      17.3195 seconds
            Processing completed...      17.3576 seconds
            Processing completed...      17.2984 seconds
            Processing completed...      17.2890 seconds
            Processing completed...      23.1307 seconds
            Processing completed...      19.8616 seconds
```

When the job has completed, use tail to view the timing from the log file.

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
```

```
tail run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.32.4x8pe.2day.20171222start.
1x32.log
```

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       32
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   6933.3
02   2017-12-23   6830.2
     Total Time = 13763.50
      Avg. Time = 6881.75
```

Submit a request for a 64 pe job (2 x 32 pe) using 2 nodes

```
sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.2x32.ncclassic.csh
```

Check on the status in the queue

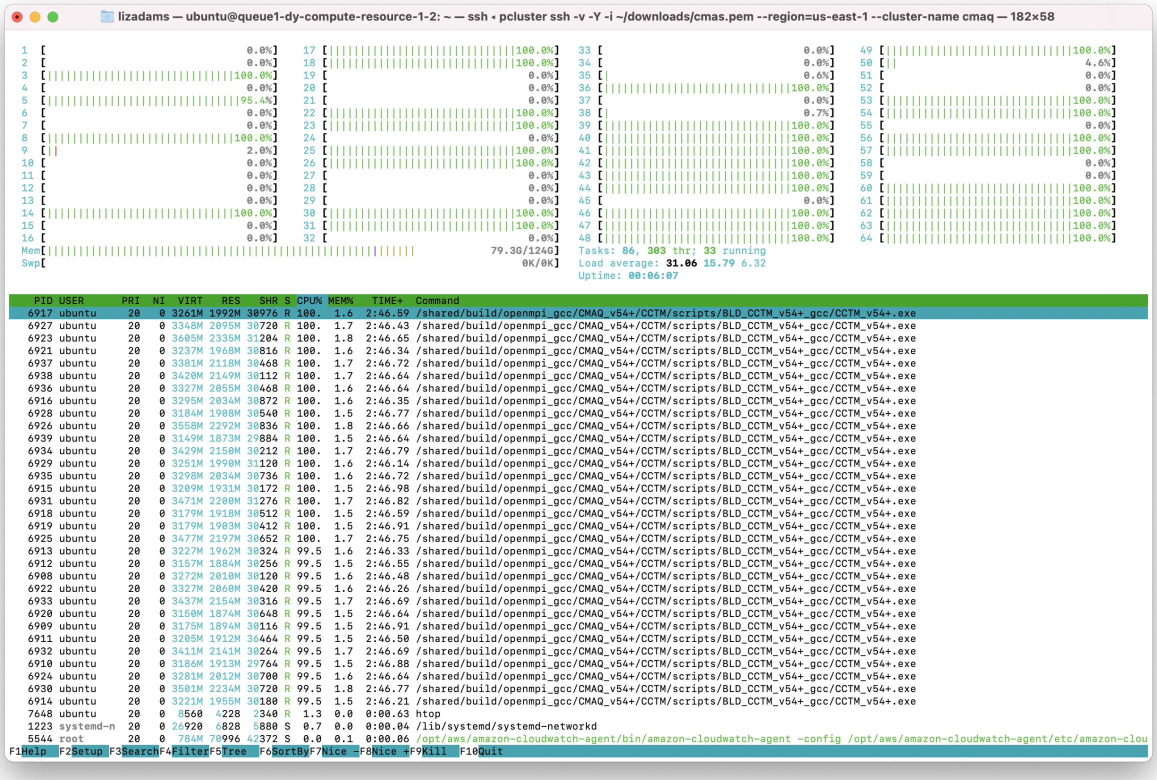
```
squeue -u ubuntu
```

Note, it takes about 5 minutes for the compute nodes to be initialized, once the job is running the ST or status will change from CF (configure) to R

Output:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 4    queue1     CMAQ   ubuntu  R       7:20      1 queue1-dy-compute-
↪resource-1-3
```

When you run 64 cores on two compute nodes, the amount of memory used per node is decreased as observed in the htop output below.



lizadams — ubuntu@queue1-dy-compute-resource-1-2: ~ — ssh ◂ pcluster ssh -v -Y -i ~/downloads/cmas.pem --region=us-east-1 --cluster-name cmaq — 182×58

```
  1  [                               0.0%]   17 [|||||||||||||||||||||||||||||100.0%]   33 [                               0.0%]   49 [|||||||||||||||||||||||||||||100.0%]
  2  [                               0.0%]   18 [|||||||||||||||||||||||||||||100.0%]   34 [                               0.0%]   50 [||                             4.6%]
  3  [|||||||||||||||||||||||||||||100.0%]   19 [                               0.0%]   35 [|                              0.6%]   51 [                               0.0%]
  4  [                               0.0%]   20 [                               0.0%]   36 [|||||||||||||||||||||||||||||100.0%]   52 [                               0.0%]
  5  [||||||||||||||||||||||||||||||95.4%]   21 [                               0.0%]   37 [                               0.0%]   53 [|||||||||||||||||||||||||||||100.0%]
  6  [                               0.0%]   22 [|||||||||||||||||||||||||||||100.0%]   38 [|                              0.7%]   54 [|||||||||||||||||||||||||||||100.0%]
  7  [                               0.0%]   23 [|||||||||||||||||||||||||||||100.0%]   39 [|||||||||||||||||||||||||||||100.0%]   55 [                               0.0%]
  8  [|||||||||||||||||||||||||||||100.0%]   24 [                               0.0%]   40 [|||||||||||||||||||||||||||||100.0%]   56 [|||||||||||||||||||||||||||||100.0%]
  9  [||                             2.0%]   25 [|||||||||||||||||||||||||||||100.0%]   41 [|||||||||||||||||||||||||||||100.0%]   57 [|||||||||||||||||||||||||||||100.0%]
  10 [                               0.0%]   26 [|||||||||||||||||||||||||||||100.0%]   42 [|||||||||||||||||||||||||||||100.0%]   58 [                               0.0%]
  11 [                               0.0%]   27 [                               0.0%]   43 [|||||||||||||||||||||||||||||100.0%]   59 [                               0.0%]
  12 [                               0.0%]   28 [                               0.0%]   44 [|||||||||||||||||||||||||||||100.0%]   60 [|||||||||||||||||||||||||||||100.0%]
  13 [                               0.0%]   29 [                               0.0%]   45 [                               0.0%]   61 [|||||||||||||||||||||||||||||100.0%]
  14 [|||||||||||||||||||||||||||||100.0%]   30 [|||||||||||||||||||||||||||||100.0%]   46 [|||||||||||||||||||||||||||||100.0%]   62 [|||||||||||||||||||||||||||||100.0%]
  15 [                               0.0%]   31 [|||||||||||||||||||||||||||||100.0%]   47 [|||||||||||||||||||||||||||||100.0%]   63 [|||||||||||||||||||||||||||||100.0%]
  16 [                               0.0%]   32 [                               0.0%]   48 [|||||||||||||||||||||||||||||100.0%]   64 [|||||||||||||||||||||||||||||100.0%]
  Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||                         79.3G/124G]   Tasks: 86, 303 thr; 33 running
  Swp[                                                                                             0K/0K]   Load average: 31.06 15.79 6.32
                                                                                                            Uptime: 00:06:07

    PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
   6917 ubuntu     20   0 3261M 1992M 30976 R 100.  1.6  2:46.59 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6927 ubuntu     20   0 3348M 2095M 30720 R 100.  1.7  2:46.43 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6923 ubuntu     20   0 3605M 2335M 31204 R 100.  1.8  2:46.65 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6921 ubuntu     20   0 3237M 1968M 30816 R 100.  1.6  2:46.34 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6937 ubuntu     20   0 3381M 2118M 30468 R 100.  1.7  2:46.72 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6938 ubuntu     20   0 3420M 2149M 30112 R 100.  1.7  2:46.64 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6936 ubuntu     20   0 3327M 2055M 30468 R 100.  1.6  2:46.64 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6916 ubuntu     20   0 3295M 2034M 30872 R 100.  1.6  2:46.35 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6928 ubuntu     20   0 3184M 1908M 30540 R 100.  1.5  2:46.77 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6926 ubuntu     20   0 3558M 2292M 30836 R 100.  1.8  2:46.66 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6939 ubuntu     20   0 3149M 1873M 29884 R 100.  1.5  2:46.64 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6934 ubuntu     20   0 3429M 2150M 30212 R 100.  1.7  2:46.79 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6929 ubuntu     20   0 3251M 1990M 31120 R 100.  1.6  2:46.14 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6935 ubuntu     20   0 3298M 2034M 30736 R 100.  1.6  2:46.72 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6915 ubuntu     20   0 3209M 1931M 30172 R 100.  1.5  2:46.98 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6931 ubuntu     20   0 3471M 2200M 31276 R 100.  1.7  2:46.82 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6918 ubuntu     20   0 3179M 1918M 30512 R 100.  1.5  2:46.59 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6919 ubuntu     20   0 3179M 1903M 30412 R 100.  1.5  2:46.91 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6925 ubuntu     20   0 3477M 2197M 30652 R 100.  1.7  2:46.75 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6913 ubuntu     20   0 3227M 1962M 30324 R 99.5  1.6  2:46.33 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6912 ubuntu     20   0 3157M 1884M 30256 R 99.5  1.5  2:46.55 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6908 ubuntu     20   0 3272M 2010M 30120 R 99.5  1.6  2:46.48 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6922 ubuntu     20   0 3327M 2060M 30420 R 99.5  1.6  2:46.26 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6933 ubuntu     20   0 3437M 2154M 30316 R 99.5  1.7  2:46.69 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6920 ubuntu     20   0 3150M 1874M 30648 R 99.5  1.5  2:46.64 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6909 ubuntu     20   0 3175M 1894M 30116 R 99.5  1.5  2:46.91 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6911 ubuntu     20   0 3205M 1912M 36464 R 99.5  1.5  2:46.50 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6932 ubuntu     20   0 3411M 2141M 30264 R 99.5  1.7  2:46.69 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6910 ubuntu     20   0 3186M 1913M 29764 R 99.5  1.5  2:46.88 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6924 ubuntu     20   0 3281M 2012M 30700 R 99.5  1.6  2:46.64 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6930 ubuntu     20   0 3501M 2234M 30720 R 99.5  1.8  2:46.77 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   6914 ubuntu     20   0 3221M 1955M 30180 R 99.5  1.5  2:46.21 /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/BLD_CCTM_v54+_gcc/CCTM_v54+.exe
   7648 ubuntu     20   0  8560  4228  2340 R  1.3  0.0  0:00.63 htop
   1223 systemd-n  20   0 26920  6828  5880 S  0.7  0.0  0:00.04 /lib/systemd/systemd-networkd
   5544 root       20   0  784M 70996 42372 S  0.0  0.1  0:00.06 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-agent/etc/amazon-clou
F1Help  F2Setup  F3SearchF4FilterF5Tree   F6SortByF7Nice -F8Nice +F9Kill   F10Quit
```

Check the status of the run

```
tail run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.64.8x8pe.2day.20171222start.
2x32.log
```

The 64 pe job should take xx minutes to run (xx minutes per day)

Note, this is a different domain (12US1 versus 12US2) than what was used for the HPC6a.48xlarge Benchmark runs, so the timings are not directly comparible. The 12US1 domain is larger than 12US2.

'12US1' 'LAM_40N97W' -2556000. -1728000. 12000. 12000. 459 299 1

Check whether the scheduler thinks there are cpus or vcpus

```
sinfo -lN
```

Output:

```
Fri Jun 30 16:39:48 2023
NODELIST                         NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_
↪DISK WEIGHT AVAIL_FE REASON
queue1-dy-compute-resource-1-1       1   queue1*   allocated 64     64:1:1 124518       ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-2       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-3       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-4       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-5       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-6       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-7       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-8       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-9       1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0       1 dynamic, none
queue1-dy-compute-resource-1-10      1   queue1*       idle~ 64     64:1:1 124518       ␣
↪0       1 dynamic, none
```

When multiple jobs are submitted to the queue they will be dispatched to different compute nodes.

```
squeue
```

output

```
ubuntu@queue1-dy-compute-resource-1-2:/shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts$␣
↪squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 2    queue1     CMAQ   ubuntu  R      59:25      1 queue1-dy-compute-
↪resource-1-1
```

(continues on next page)

(continued from previous page)

```
                 3     queue1      CMAQ    ubuntu  R       12:20      2 queue1-dy-compute-
↪resource-1-[2-3]
                 4     queue1      CMAQ    ubuntu  R        8:50      3 queue1-dy-compute-
↪resource-1-[4-6]
                 5     queue1      CMAQ    ubuntu  R        8:50      4 queue1-dy-compute-
↪resource-1-[7-10]
```

When the job has completed, use tail to view the timing from the log file.

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
```

```
tail run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.64.8x8pe.2day.20171222start.
2x32.log
```

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       64
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   3080.9
02   2017-12-23   3383.5
     Total Time = 6464.40
      Avg. Time = 3232.20
```

Based on the Total Time, adding an additional node gave a speed-up of 2.129 13763.50/6464.40 = 2.129

Submit a job to run on 96 cores, 3x32 nodes

```
sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.3x32.ncclassic.csh
```

Verify that it is running on 3 nodes

```
sbatch
```

output:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 5     queue1     CMAQ   ubuntu  R       4:29      3 queue1-dy-compute-
↪resource-1-[1-3]
```

Check the log for how quickly the job is running

```
grep 'Processing completed' run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.96.
12x8pe.2day.20171222start.3x32.log
```

Output:

```
            Processing completed...       6.6962 seconds
            Processing completed...       6.7025 seconds
            Processing completed...       6.7126 seconds
            Processing completed...       6.6939 seconds
            Processing completed...       6.6550 seconds
            Processing completed...       6.6515 seconds
            Processing completed...       9.7306 seconds
            Processing completed...       9.0629 seconds
            Processing completed...       7.0797 seconds
            Processing completed...       7.0134 seconds
```

When the job has completed, use tail to view the timing from the log file.

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
```

```
tail -n 20 run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.96.12x8pe.2day.
20171222start.3x32.log
```

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       96
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   2144.2
02   2017-12-23   2361.9
     Total Time = 4506.10
      Avg. Time = 2253.05

```

Based on the Total Time, adding 2 additional nodes gave a speed-up of 3.05 13763.50/4506.1 = 3.05

Submit a job to run on 128 cores, 4x32 nodes

```
sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.4x32.ncclassic.csh
```

Verify that it is running on 4 nodes

```
squeue
```

output:

```
                 5    queue1     CMAQ   ubuntu  R      37:14      4 queue1-dy-compute-
↪resource-1-[7-10]
```

Check the log for how quickly the job is running

`grep ‘Processing completed’ run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.128.16x8pe.2day.20171222start.4x32.lo

Output:

```
            Processing completed...       5.1118 seconds
            Processing completed...       5.0991 seconds
            Processing completed...       7.2644 seconds
            Processing completed...       8.1420 seconds
            Processing completed...       5.0802 seconds
            Processing completed...       5.0438 seconds
            Processing completed...       5.0477 seconds
```

When the job has completed, use tail to view the timing from the log file.

`cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/`

`tail -n 20 run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.128.16x8pe.2day.20171222start.4x32.log`

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       128
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   1696.6
02   2017-12-23   1875.7
     Total Time = 3572.30
      Avg. Time = 1786.15
```

Based on the Total Time, adding 3 additional nodes gave a speed-up of 3.85 out of expected 4x speedup. 13763.50/3572.30 = 3.85

Once you have submitted a few benchmark runs and they have completed successfully, proceed to the next chapter.

Run CMAQ using hpc7g.8xlarge compute nodes

Verify that you have an updated set of run scripts from the pcluster-cmaq repo

Run the 12US1 Domain on 32 pes on hpc7g.8xlarge

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/

sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.1x32.ncclassic.c7g.8xlarge.csh`
```

When the job has completed, use tail to view the timing from the log file.

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
```

```
tail run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.32.4x8pe.2day.20171222start.1x32.hpc7g.8xlarge.log
```

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       32
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   6266.1
02   2017-12-23   6868.5
     Total Time = 13134.60
      Avg. Time = 6567.30
```

Submit a request for a 64 pe job (2 x 32 pe) using 2 nodes on hpc7g.8xlarge

```
sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.2x32.ncclassic.c7g.8xlarge.csh
```

Check on the status in the queue

```
squeue -u ubuntu
```

Note, it takes about 5 minutes for the compute nodes to be initialized, once the job is running the ST or status will change from CF (configure) to R

Output:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 4    queue1     CMAQ   ubuntu  R    1:11:48      2 queue1-dy-compute-
↪resource-2-[3-4]
```

Check the status of the run

```
tail run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.64.8x8pe.2day.20171222start.
2x32.hpc7g.8xlarge.log
```

The 64 pe job should take xx minutes to run (xx minutes per day)

Check whether the scheduler thinks there are cpus or vcpus

```
sinfo -lN
```

Output:

```
Fri Jun 30 16:39:48 2023
NODELIST                         NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_
↪DISK WEIGHT AVAIL_FE REASON
NODELIST                        NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_
↪DISK WEIGHT AVAIL_FE REASON
queue1-dy-compute-resource-1-1      1    queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-2      1    queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-3      1    queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-4      1    queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-1-5      1    queue1*       idle~ 64     64:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-2-1      1    queue1*       idle~ 32     32:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-2-2      1    queue1*       idle~ 32     32:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-2-3      1    queue1*   allocated 32     32:1:1 124518       ␣
↪0      1 dynamic, none
```

(continues on next page)

(continued from previous page)

```
queue1-dy-compute-resource-2-4      1    queue1*   allocated 32     32:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-2-5      1    queue1*   allocated 32     32:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-2-6      1    queue1*   allocated 32     32:1:1 124518       ␣
↪0      1 dynamic, none
queue1-dy-compute-resource-2-7      1    queue1*   allocated 32     32:1:1 124518       ␣
↪0      1 dynamic, none
```

When multiple jobs are submitted to the queue they will be dispatched to different compute nodes.

```
squeue
```

output

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 4    queue1     CMAQ   ubuntu  R    1:13:21      2 queue1-dy-compute-
↪resource-2-[3-4]
                 7    queue1     CMAQ   ubuntu  R      57:51      3 queue1-dy-compute-
↪resource-2-[5-7]
```

When the job has completed, use tail to view the timing from the log file.

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
```

```
tail run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.64.8x8pe.2day.20171222start.
2x32.hpc7g.8xlarge.log
```

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       64
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   3122.1
02   2017-12-23   3419.1
     Total Time = 6541.20
      Avg. Time = 3270.60
```

Based on the Total Time, adding an additional node gave a speed-up of 2.008 with expected speedup of 2x 13134.60/6541.20 = 2.008

Submit a job to run on 96 cores, 3x32 nodes on hpc7g.8xlarge

```
sbatch run_cctm_2018_12US1_v54_cb6r5_ae6.20171222.3x32.ncclassic.c7g.8xlarge.csh
```

Verify that it is running on 3 nodes

```
sbatch
```

output:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 7    queue1     CMAQ   ubuntu  R      59:47      3 queue1-dy-compute-
↪resource-2-[5-7]
```

Check the log for how quickly the job is running

```
grep 'Processing completed' run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.64.12x8pe.2day.20171222start.3
```

Output:

```
            Processing completed...       5.6952 seconds
            Processing completed...       8.3384 seconds
            Processing completed...       8.2416 seconds
            Processing completed...       5.7230 seconds
            Processing completed...       5.6911 seconds
```

When the job has completed, use tail to view the timing from the log file.

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/
```

```
tail -n 20 run_cctm5.4+_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.64.12x8pe.2day.
20171222start.3x32.hpc7g.8xlarge.log
```

Output:

```
==================================
  ***** CMAQ TIMING REPORT *****
==================================
Start Day: 2017-12-22
End Day:   2017-12-23
Number of Simulation Days: 2
Domain Name:               12US1
Number of Grid Cells:      4803435  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       96
   All times are in seconds.

Num  Day        Wall Time
01   2017-12-22   2141.9
02   2017-12-23   2384.6
```

(continues on next page)

(continued from previous page)

```
     Total Time = 4526.50
      Avg. Time = 2263.25
```

Based on the Total Time, adding 2 additional nodes gave a speed-up of 2.902, close to 3x if ideal scaling 13134.60/4526.50 = 2.902

Once you have submitted a few benchmark runs and they have completed successfully, proceed to the next chapter.

Install Input Data on ParallelCluster

(note, this has already been installed if the yaml file used the s3:cmas-cmaq filesystem on lustre.

(keeping in case users need to install the data from scratch.)

Verify AWS CLI is available obtain data from AWS S3 Bucket

Check to see if the aws command line interface (CLI) is installed

`which aws`

If it is installed, skip to the next step.

If it is not available please follow these instructions to install it.

See also:

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

`cd /shared`

`curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`

`unzip awscliv2.zip`

`sudo ./aws/install`

Verify you can run the aws command

```
 aws --help
```

If not, you may need to logout and back in.

Note: If you do not have credintials, skip this. The data is on a public bucket, so you do not need credentials.

Set up your credentials for using s3 copy (you can skip this if you do not have credentials)

`aws configure`

Copy Input Data from S3 Bucket to lustre filesystem

Verify that the /fsx directory exists; this is a lustre file system where the I/O is fastest

```
ls /fsx
```

If you are unable to use the lustre file system, the data can be installed on the /shared volume, if you have resized the volume to be large enough to store the input and output data.

Use the S3 script to copy the CONUS input data from the CMAS s3 bucket

Data will be saved to the /fsx file system

```
/shared/pcluster-cmaq/s3_scripts/s3_copy_nosign_2018_12US1_conus_cmas_opendata_to_fsx_20171222_cb6r3.csh
```

check that the resulting directory structure matches the run script

Note: The CONUS 12US1 input data requires 44 GB of disk space
(if you use the yaml file to import the data to the lustre file system rather than copying the data you save this space)

```
cd /fsx/data/cmas-cmaq-modeling-platform-2018
```

```
du -sh
```

output:

```
34G     .
```

CMAQ ParallelCluster is configured to have 1.2 Terrabytes of space on /fsx filesystem (minimum size allowed for lustre /fsx), to allow multiple output runs to be stored.

For ParallelCluster: Import the Input data from a public S3 Bucket (optional)

A second method is available to import the data on the lustre file system using the yaml file to specify the s3 bucket location in the yaml file, rather than using the above aws s3 copy commands.

See also:

Example available in c5n-18xlarge.ebs_shared.fsx_import.yaml

```
cd /shared/pcluster-cmaq/yaml
vi hpc6a.48xlarge.ebs_unencrypted_installed_public_ubuntu2004.fsx_import.yaml
```

Section that of the YAML file that specifies the name of the S3 Bucket.

```
  - MountDir: /fsx
    Name: name2
    StorageType: FsxLustre
    FsxLustreSettings:
      StorageCapacity: 1200
      ImportPath: s3://cmas-cmaq-modeling-platform-2018/2018_12US1/    <<<  specify name␣
↪of S3 bucket
```

This requires that the S3 bucket specified is publically available

Convert the *.nc4 compressed netCDF4 files to netCDF classic (nc3) files

```
cd /shared/pcluster-cmaq/s3_scripts
```

```
cp indexer.csh /fsx/data/CMAQ_Modeling_Platform_2018
```

```
cd /fsx/data/CMAQ_Modeling_Platform_2018
```

```
chmod 755 indexer.csh
```

```
find . -name '*.nc4' -exec ./indexer.csh {} \;
```

4.5 Post-process and QA

4.5.1 Post-process CMAQ and Install R

Post-processing CMAQ Run, Install R and packages Instructions to install R and packages for QA of CMAQ difference in output between two runs.

Scripts to run combine and post processing

Build the POST processing routines

Instructions on how to Post-process CMAQ using the utilities under the POST directory

Note: The post-processing analysis is run on the head node.

Verify that the compute nodes are no longer running if you have completed all of the benchmark runs

```
squeue
```

You should see that no jobs are running.

Show compute nodes

```
scontrol show nodes
```

Edit, Build, and Run the POST processing routines

```
setenv DIR /shared/build/openmpi_gcc/CMAQ_v54+/

cd $DIR/POST/combine/scripts

sed -i 's/v54/v54+/g' bldit_combine.csh
./bldit_combine.csh gcc |& tee bldit_combine.log
cp run_combine.csh run_combine_12US1.csh
sed -i 's/v54/v54+/g' run_combine_12US1.csh
sed -i 's/Bench_2016_12SE1/2018_12US1_2x64_classic/g' run_combine_12US1.csh
sed -i 's/intel/gcc/g' run_combine_12US1.csh
sed -i 's/cb6r3_ae7_aq/cb6r5_ae7_aq/g' run_combine_12US1.csh
sed -i 's/2016-07-01/2017-12-22/g' run_combine_12US1.csh
```

(continues on next page)

(continued from previous page)

```
sed -i 's/2016-07-14/2017-12-23/g' run_combine_12US1.csh
sed -i 's/${VRSN}_${compilerString}_${APPL}/${VRSN}_${MECH}_${compilerString}_${APPL}/g' run_combine_12US1.csh
setenv CMAQ_DATA /fsx/data
./run_combine_12US1.csh

cp run_calc_tmetric.csh run_calc_tmetric_12US1.csh
sed -i 's/Bench_2016_12SE1/2018_12US1_2x64_classic/g' run_calc_tmetric_12US1.csh
sed -i 's/intel/gcc/g' run_calc_tmetric_12US1.csh
sed -i 's/201607/201712/g' run_calc_tmetric_12US1.csh
setenv CMAQ_DATA /fsx/data
./run_calc_tmetric_12US1.csh

cd $DIR/POST/hr2day/scripts

cp run_hr2day.csh run_hr2day_12US1.csh
sed -i 's/Bench_2016_12SE1/2018_12US1_2x64_classic/g' run_hr2day_12US1.csh
sed -i 's/intel/gcc/g' run_hr2day_12US1.csh
sed -i 's/2016182/2015356/g' run_hr2day_12US1.csh
sed -i 's/2016195/2015357/g' run_hr2day_12US1.csh
setenv CMAQ_DATA /fsx/data
./run_hr2day_12US1.csh

#cd $DIR/POST/bldoverlay/scripts

#cp run_bldoverlay.csh run_bldoverlay_12US1.csh
#sed -i 's/Bench_2016_12SE1/2018_12US1_2x64_classic/g' run_bldoverlay_12US1.csh
#sed -i 's/intel/gcc/g' run_bldoverlay_12US1.csh
#sed -i 's/2016-07-01/2015-12-22/g' run_bldoverlay_12US1.csh
#sed -i 's/2016-07-02/2015-12-23/g' run_bldoverlay_12US1.csh
#setenv CMAQ_DATA /fsx/data
#./run_bldoverlay_12US1.csh
```

Install R, Rscripts and Packages

First check to see if R is already installed.

`R --version`

If not, Install R on Ubuntu 2004 instructions available in the link below.

See also:

Install R on Ubuntu 2004

`sudo apt install build-essential`

See also:

ubuntu install

Install geospatial dependencies

be sure to have an updated system

`sudo apt-get update && sudo apt-get upgrade -y`

install PROJ

`sudo apt-get install libproj-dev proj-data proj-bin unzip -y`

optionally, install (selected) datum grid files

`sudo apt-get install proj-data`

install GEOS

`sudo apt-get install libgeos-dev -y`

install GDAL

`sudo apt-get install libgdal-dev python3-gdal gdal-bin -y`

install PDAL (optional)

`sudo apt-get install libpdal-dev pdal libpdal-plugin-python -y`

recommended to give Python3 precedence over Python2 (which is end-of-life since 2019)

`sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 1`

Install software for diagram

`sudo apt-get install graphviz`

`pip install diagrams`

Install further compilation dependencies (Ubuntu 20.04)

```
sudo apt-get install \
  build-essential \
  flex make bison gcc libgcc1 g++ ccache \
  python3 python3-dev \
  python3-opengl python3-wxgtk4.0 \
  python3-dateutil libgsl-dev python3-numpy \
  wx3.0-headers wx-common libwxgtk3.0-gtk3-dev \
  libwxbase3.0-dev   \
  libncurses5-dev \
  libbz2-dev \
  zlib1g-dev gettext \
  libtiff5-dev libpnglite-dev \
  libcairo2 libcairo2-dev \
  sqlite3 libsqlite3-dev \
  libpq-dev \
  libreadline6-dev libfreetype6-dev \
  libfftw3-3 libfftw3-dev \
  libboost-thread-dev libboost-program-options-dev  libpdal-dev\
  subversion libzstd-dev \
  checkinstall \
  libglu1-mesa-dev libxmu-dev \
  ghostscript wget -y
```

For NVIZ on Ubuntu 20.04:

```
sudo apt-get install \
  ffmpeg libavutil-dev ffmpeg2theora \
```

(continues on next page)

(continued from previous page)

```
  libffmpegthumbnailer-dev \
  libavcodec-dev \
  libxmu-dev \
  libavformat-dev libswscale-dev
```

ncdf4 package REQUIRES the netcdf library be version 4 or above, AND installed with HDF-5 support (i.e., the netcdf library must be compiled with the –enable-netcdf-4 flag). Building netcdf with HDF5 support requires curl.

```
sudo apt-get install curl
sudo apt-get install libcurl4-openssl-dev
```

```
cd /shared/pcluster-cmaq
```

Install libraries with hdf5 support

Load modules

```
module load openmpi/4.1.1
```

```
module load libfabric-aws/1.13.2amzn1.0
```

```
./gcc_install_hdf5.pcluster.csh
```

Install ncdf4 package from source:

```
cd /shared/pcluster-cmaq/qa_scripts/R_packages
```

```
sudo R CMD INSTALL ncdf4_1.13.tar.gz --configure-args="--with-nc-config=/shared/build-hdf5/install/bin/nc-config"
```

Install packages used in the R scripts

```
sudo -i R
install.packages("rgdal")
install.packages("M3")
install.packages("fields")
install.packages("mapdata")
install.packages("ggplot2")
install.packages("patchwork")
```

Install M3

```
cd /shared/pcluster-cmaq/qa_scripts/R_packages
```

`sudo R CMD INSTALL M3_0.3.tar.gz’

Install pdftoppm to convert pdf files to images

```
sudo apt install poppler-utils
```

To view the script, install imagemagick

```
sudo apt-get install imagemagick
```

Install X11

```
sudo apt install x11-apps
```

Enable X11 forwarding

```
sudo vi /etc/ssh/sshd_config
```

add line

X11Forwarding yes

Verify that it was added

```
sudo cat /etc/ssh/sshd_config | grep -i X11Forwarding
```

Restart ssh

```
sudo service ssh restart
```

Exit the cluster

```
exit
```

Be sure that you have Xquartz running if your local machine is a mac. Also that the DISPLAY environment variable is set.

For example in my .zshrc, I use the following setting

```
export DISPLAY=:0
```

If you modify your .zshrc, then resource it

```
source ~/.zshrc
```

Re-login to the cluster

```
pcluster ssh -v -Y -i ~/your-key.pem --cluster-name cmaq
```

Test display

```
display xclock
```

See also:

NICE DCV Settings in YAML

Note, it looks like the examples are using the older config or CLI 2 format, and need to convert this to a yaml format to try it out.

See also:

X11 forwarding no loner enabled on master node

The bug says that you can use a custom post installation script to re-enable X11 Forwarding.

See also:

Custom Bootstrap Actions

4.5.2 QA CMAQ

Quality Assurance: Comparing the output of two CMAQ runs.

Quality Assurance

Instructions on how to to verify a successful CMAQ Run on ParallelCluster.

Run m3diff to compare the output data for two runs that have different values for NPCOL

```
cd /fsx/data/output
ls */*ACONC*
```

```
setenv AFILE output_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_2x64_classic/CCTM_ACONC_
↪v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_2x64_classic_20171222.nc
setenv BFILE output_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_3x64_classic/CCTM_ACONC_
↪v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_3x64_classic_20171222.nc
```

```
m3diff
```

hit return several times to accept the default options

```
grep A:B REPORT
```

Should see all zeros.

Recompiled CMAQ using -march=native compiler option for gcc compiler, but still see differences in answers. The answers are the same, or the differences are all zeros if the domain decomposition uses the same NPCOL, here, NPCOL differs (10 vs 16)

This behavior is different from what was observed with removing the -march=native compiler option for gcc on the AMD Cyclecloud HBV3 processor. On cycle cloud, if CMAQ is compiled with -march=native removed from the compiler options, then the answers match if NPCOL differs.

```
@ NPCOL  =  8; @ NPROW =  16
@ NPCOL  =  12; @ NPROW =  16
```

`grep A:B REPORT`

output

```
 A:B  9.31323E-10@( 78,122, 1) -9.31323E-10@(173, 98, 1) -1.67066E-14  1.11030E-11
 A:B  3.72529E-09@(272,162, 1) -7.45058E-09@(271,162, 1) -4.96030E-14  4.09644E-11
 A:B  7.45058E-09@( 51,109, 1) -7.45058E-09@( 39,191, 1)  3.67334E-14  8.05492E-11
 A:B  9.31323E-09@(270,159, 1) -8.14907E-09@(189, 41, 1)  4.33809E-13  1.57138E-10
 A:B  1.86265E-08@( 53,108, 1) -1.30385E-08@( 54,108, 1)  8.53858E-13  2.42891E-10
 A:B  4.09782E-08@(326,165, 1) -1.90921E-08@(327,166, 1)  1.49534E-12  3.98158E-10
 A:B  1.55531E-07@(326,165, 1) -8.19564E-08@(233,101, 1)  2.42760E-12  1.00020E-09
 A:B  1.77883E-07@(325,167, 1) -1.11759E-07@(324,170, 1)  1.29720E-12  1.93561E-09
 A:B  1.94646E-07@(325,167, 1) -1.36439E-07@(331,163, 1)  1.38651E-11  2.77102E-09
 A:B  1.69501E-07@(134,119, 1) -2.81259E-07@( 50,223, 1)  2.44340E-11  3.76379E-09
```

CMAQv5.4+ does not have -march=native compile option as a default in the Makefile

Even with NPCOL the same, the answers are not matching.

`more REPORT_3x64vs4x64`

```
     FILE A:  AFILE (output_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_3x64_classic/CCTM_
↪ACONC_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_3x64_classic_20171222.nc)

     FILE B:  BFILE (output_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_4x64_classic/CCTM_
↪ACONC_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_4x64_classic_20171222.nc)


     -----------------------------------------------------------




Date and time  2017356:000000 (0:00:00   Dec. 22, 2017)
A:AFILE/NO2  vs  B:BFILE/NO2  vs  (A - B)
     MAX        @(  C,  R, L)  Min        @(  C,  R, L)  Mean         Sigma
A    6.07607E-02@(185, 12, 1)  2.09545E-06@(313,285, 1)  1.37065E-03  2.83485E-03
B    6.07607E-02@(185, 12, 1)  2.09545E-06@(313,285, 1)  1.37065E-03  2.83485E-03
A:B  3.72529E-09@(374,170, 1) -3.72529E-09@(377,170, 1) -3.22105E-14  2.88679E-11


Date and time  2017356:010000 (1:00:00   Dec. 22, 2017)
A:AFILE/NO2  vs  B:BFILE/NO2  vs  (A - B)
     MAX        @(  C,  R, L)  Min        @(  C,  R, L)  Mean         Sigma
A    6.00488E-02@(185, 13, 1)  2.17908E-06@(303,251, 1)  1.39298E-03  2.87296E-03
B    6.00488E-02@(185, 13, 1)  2.17908E-06@(303,251, 1)  1.39298E-03  2.87296E-03
A:B  7.45058E-09@(271,162, 1) -9.31323E-09@(374,169, 1) -3.20415E-15  6.70979E-11


Date and time  2017356:020000 (2:00:00   Dec. 22, 2017)
A:AFILE/NO2  vs  B:BFILE/NO2  vs  (A - B)
     MAX        @(  C,  R, L)  Min        @(  C,  R, L)  Mean         Sigma
A    5.68695E-02@(185, 13, 1)  1.81416E-06@(302,252, 1)  1.36102E-03  2.79681E-03
B    5.68695E-02@(185, 13, 1)  1.81416E-06@(302,252, 1)  1.36102E-03  2.79681E-03
A:B  7.45058E-09@( 39,191, 1) -4.65661E-09@(378,170, 1)  4.00549E-13  9.75052E-11
```

Run an R script to create the box plots and spatial plots comparing the output of two runs

Examine the script to create the box plots and spatial plots and edit to use the output that you have generated in your runs.

First check what output is available on your ParallelCluster

If your I/O directory is /fsx

`ls -rlt /fsx/data/output/*/*ACONC*`

If your I/O directory is /shared/data

`ls -lrt /shared/data/output/*/*ACONC*`

Then edit the script to use the output filenames available.

```
cd /shared/pcluster-cmaq/qa_scripts
cp compare_EQUATES_benchmark_output_CMAS_pcluster.r compare_EQUATES_benchmark_output_
```

(continues on next page)

(continued from previous page)

```
↪CMAS_pcluster_hpc7g.18xlarge.r
vi compare_EQUATES_benchmark_output_CMAS_pcluster_hpc7g.18xlarge.r
```

```
#Directory, file name, and label for first model simulation (sim1)
sim1.label <- "CMAQv54+ 16x16 cores"
sim1.dir <- "/fsx/data/output/output_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_4x64_
↪classic/"
sim1.file <- paste0(sim1.dir,"CCTM_ACONC_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_4x64_
↪classic_20171222.nc")


#Directory, file name, and label for second model simulation (sim2)
sim2.label <- "CMAQv54+ 12x16 cores"
sim2.dir <- "/fsx/data/output/output_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_3x64_
↪classic/"
sim2.file <- paste0(sim2.dir,"CCTM_ACONC_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_3x64_
↪classic_20171222.nc")
```

Run the R script

```
cd /shared/pcluster-cmaq/qa_scripts
Rscript compare_EQUATES_benchmark_output_CMAS_pcluster_hpc7g.18xlarge.r
```

Note: your plots will be created based on the setting of the output directory in the script

An example set of scripts are available, but these instructions can be modified to use the output generated in the script above.

To view the PDF plots use the command:

```
cd /shared/pcluster-cmaq/qa_scripts/qa_plots
gio open O3_MAPS_CMAQ*.pdf
```

To convert the PDF to a jpeg image use the script convert.csh.

```
cd /shared/pcluster-cmaq/qa_scripts/qa_plots
```

First examine what the convert.csh script is doing

```
more convert.csh
```

output:

```
#!/bin/csh

foreach name (`ls *.pdf`)
  set name2=`basename $name .pdf`
  echo $name
  echo $name2
  pdftoppm -jpeg -r 600 $name $name2
end
```

Run the convert script.

```
./convert.csh
```

When NPCOL is fixed, we are seeing no difference in the answers.

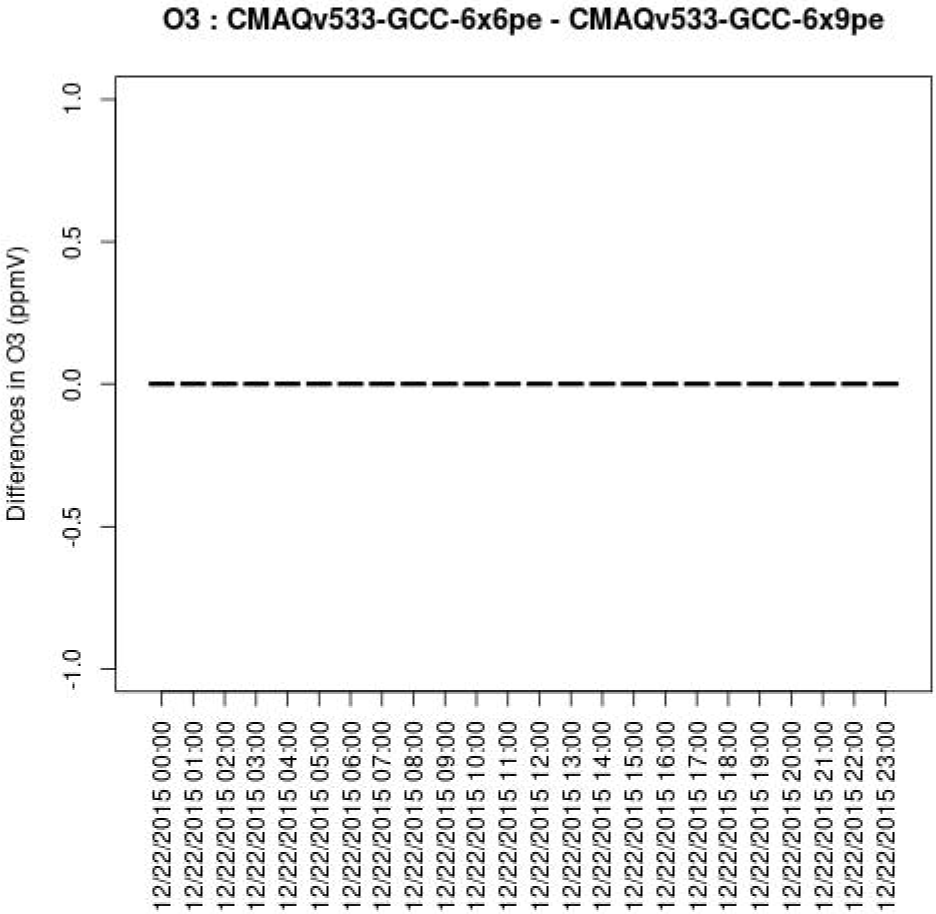
Example comparison using: 16x16 compared to 12x16

Use display to view the plots

```
display O3_BOXPLOT_CMAQv54+16x16cores_vs_CMAQv54+12x16cores.jpeg
```

They are also displayed in the following plots:

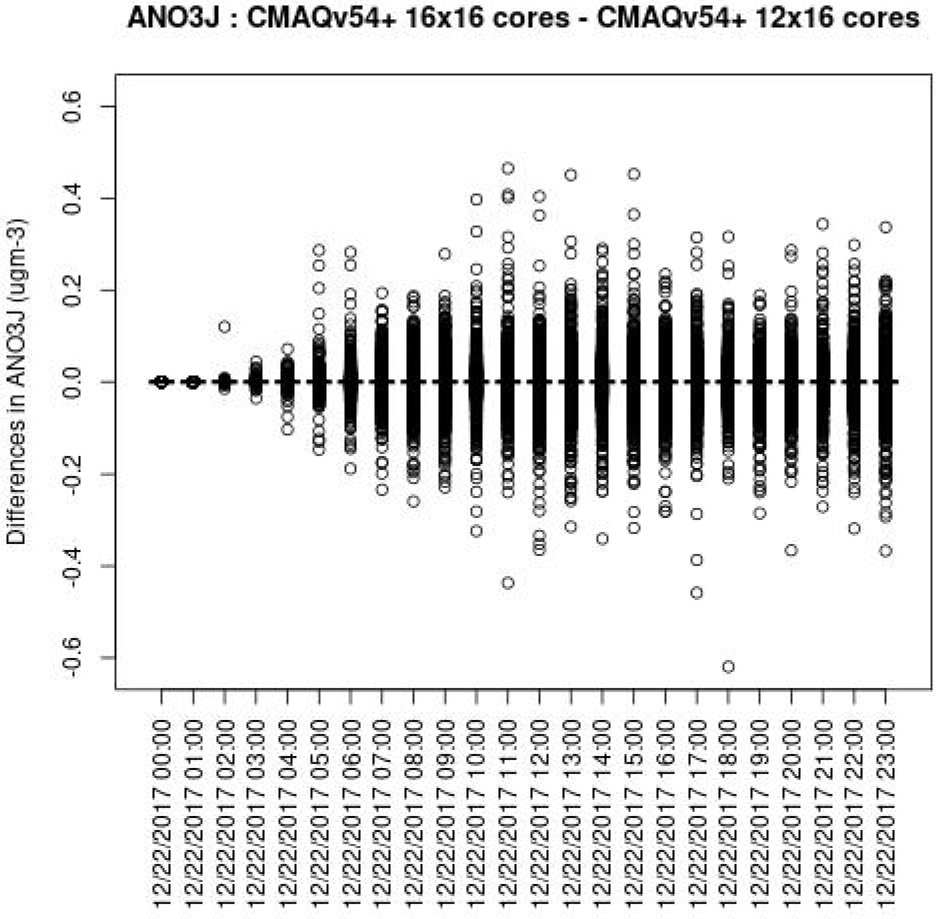
Box Plot for ANO3J when NPCOL is identical for CMAQv5.3.3 (I didn't see an identical plot for CMAQv5.4)



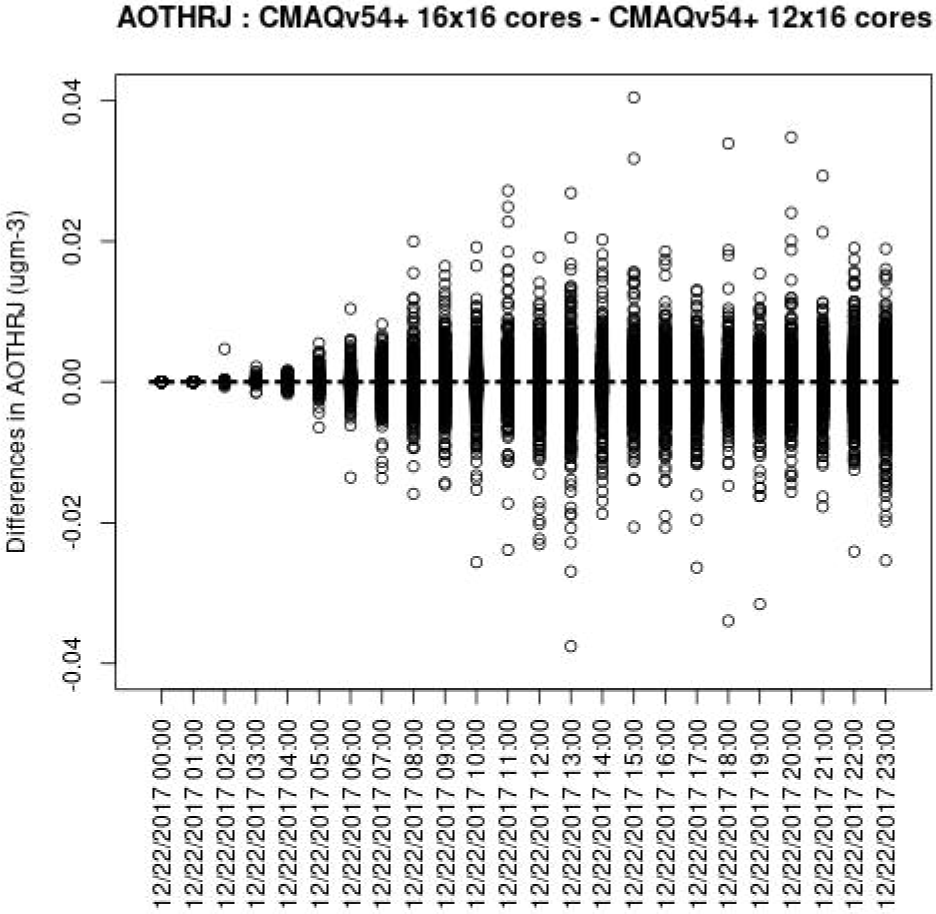
Box plot shows no difference between ACONC output for a CMAQv5.3.3 run using different PE configurations as long as NPCOL is fixed (this is true for all species that were plotted (AOTHRJ, CO, NH3, NO2, O3, OH, SO2)

Box plot shows a difference betweeen ACONC output for a CMAQv5.4+ run using different PE configurations when NPCOL is different

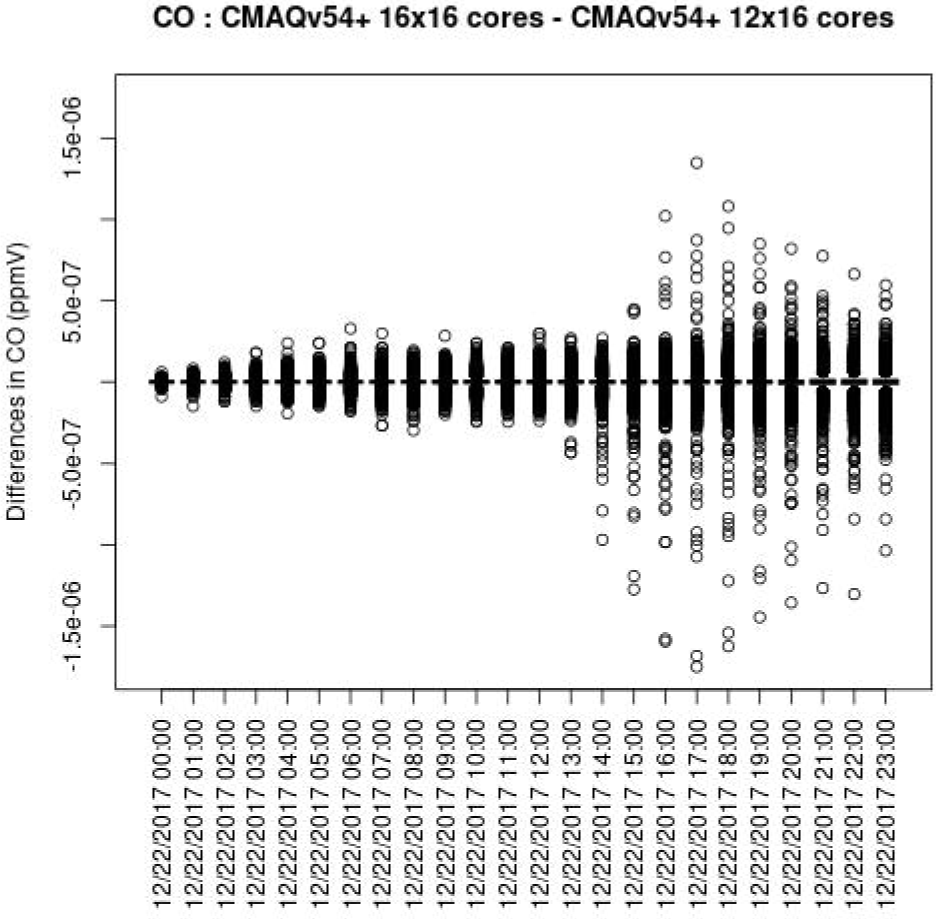
ANO3J


ANO3J : CMAQv54+ 16x16 cores - CMAQv54+ 12x16 cores
Differences in ANO3J (ugm-3)
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
12/22/2017 00:00
12/22/2017 01:00
12/22/2017 02:00
12/22/2017 03:00
12/22/2017 04:00
12/22/2017 05:00
12/22/2017 06:00
12/22/2017 07:00
12/22/2017 08:00
12/22/2017 09:00
12/22/2017 10:00
12/22/2017 11:00
12/22/2017 12:00
12/22/2017 13:00
12/22/2017 14:00
12/22/2017 15:00
12/22/2017 16:00
12/22/2017 17:00
12/22/2017 18:00
12/22/2017 19:00
12/22/2017 20:00
12/22/2017 21:00
12/22/2017 22:00
12/22/2017 23:00

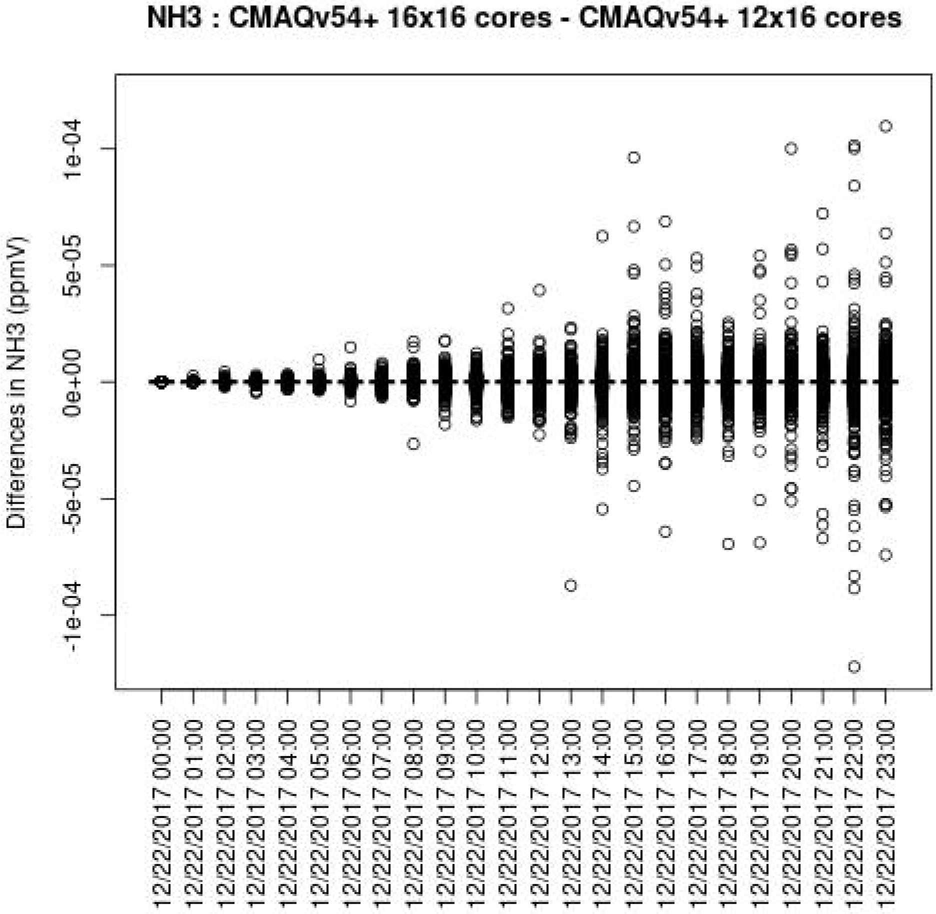
AOTHRJ


AOTHRJ : CMAQv54+ 16x16 cores - CMAQv54+ 12x16 cores
Differences in AOTHRJ (ugm-3)
0.04
0.02
0.00
-0.02
-0.04
12/22/2017 00:00
12/22/2017 01:00
12/22/2017 02:00
12/22/2017 03:00
12/22/2017 04:00
12/22/2017 05:00
12/22/2017 06:00
12/22/2017 07:00
12/22/2017 08:00
12/22/2017 09:00
12/22/2017 10:00
12/22/2017 11:00
12/22/2017 12:00
12/22/2017 13:00
12/22/2017 14:00
12/22/2017 15:00
12/22/2017 16:00
12/22/2017 17:00
12/22/2017 18:00
12/22/2017 19:00
12/22/2017 20:00
12/22/2017 21:00
12/22/2017 22:00
12/22/2017 23:00

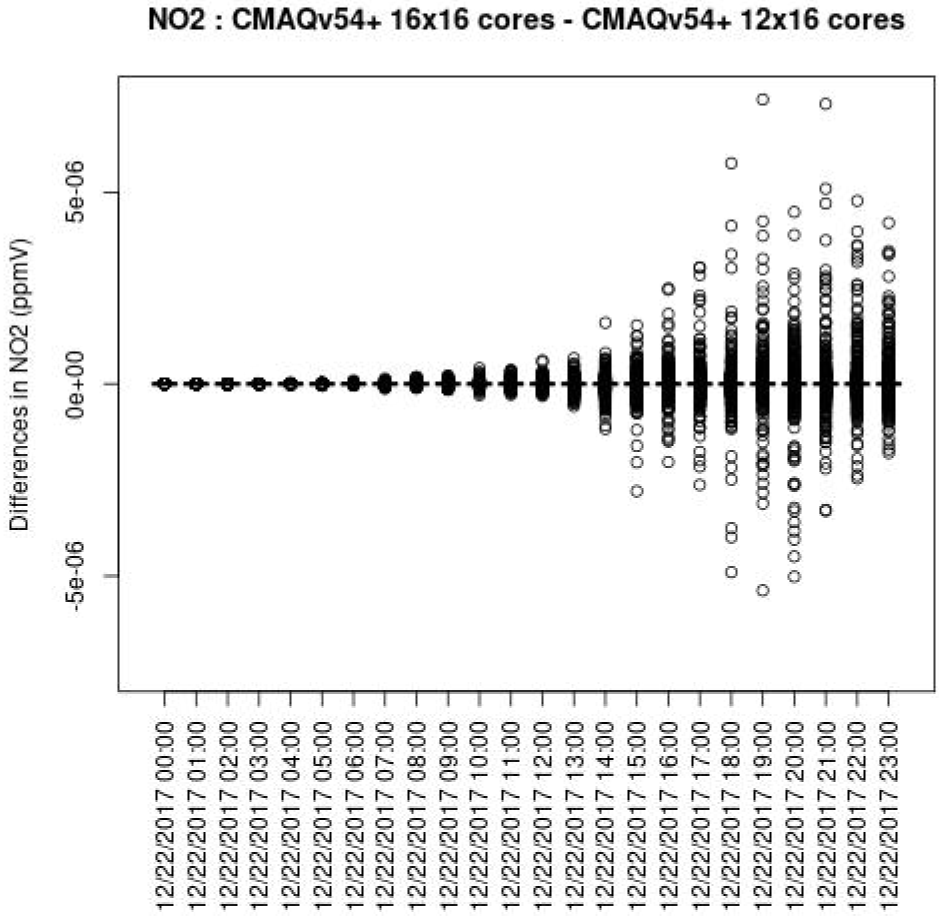
CO


CO : CMAQv54+ 16x16 cores - CMAQv54+ 12x16 cores
Differences in CO (ppmV)
1.5e-06
5.0e-07
-5.0e-07
-1.5e-06
12/22/2017 00:00
12/22/2017 01:00
12/22/2017 02:00
12/22/2017 03:00
12/22/2017 04:00
12/22/2017 05:00
12/22/2017 06:00
12/22/2017 07:00
12/22/2017 08:00
12/22/2017 09:00
12/22/2017 10:00
12/22/2017 11:00
12/22/2017 12:00
12/22/2017 13:00
12/22/2017 14:00
12/22/2017 15:00
12/22/2017 16:00
12/22/2017 17:00
12/22/2017 18:00
12/22/2017 19:00
12/22/2017 20:00
12/22/2017 21:00
12/22/2017 22:00
12/22/2017 23:00

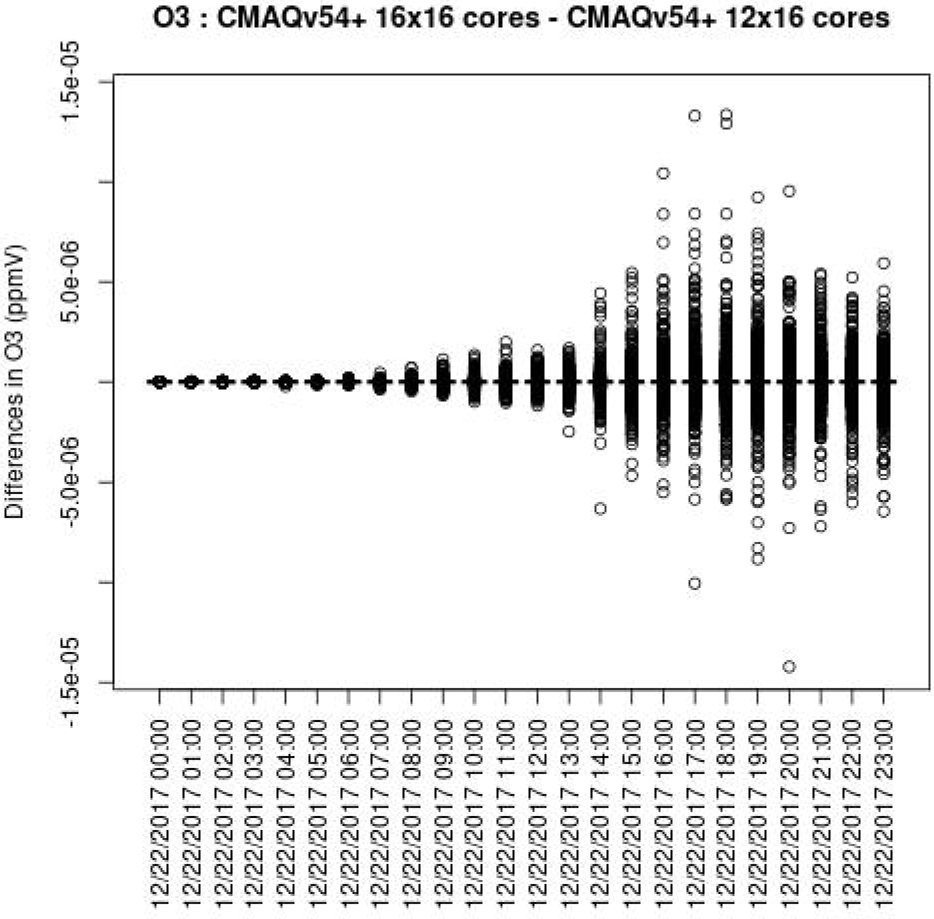
NH3


NH3 : CMAQv54+ 16x16 cores - CMAQv54+ 12x16 cores
Differences in NH3 (ppmV)
1e-04
5e-05
0e+00
-5e-05
-1e-04
12/22/2017 00:00
12/22/2017 01:00
12/22/2017 02:00
12/22/2017 03:00
12/22/2017 04:00
12/22/2017 05:00
12/22/2017 06:00
12/22/2017 07:00
12/22/2017 08:00
12/22/2017 09:00
12/22/2017 10:00
12/22/2017 11:00
12/22/2017 12:00
12/22/2017 13:00
12/22/2017 14:00
12/22/2017 15:00
12/22/2017 16:00
12/22/2017 17:00
12/22/2017 18:00
12/22/2017 19:00
12/22/2017 20:00
12/22/2017 21:00
12/22/2017 22:00
12/22/2017 23:00

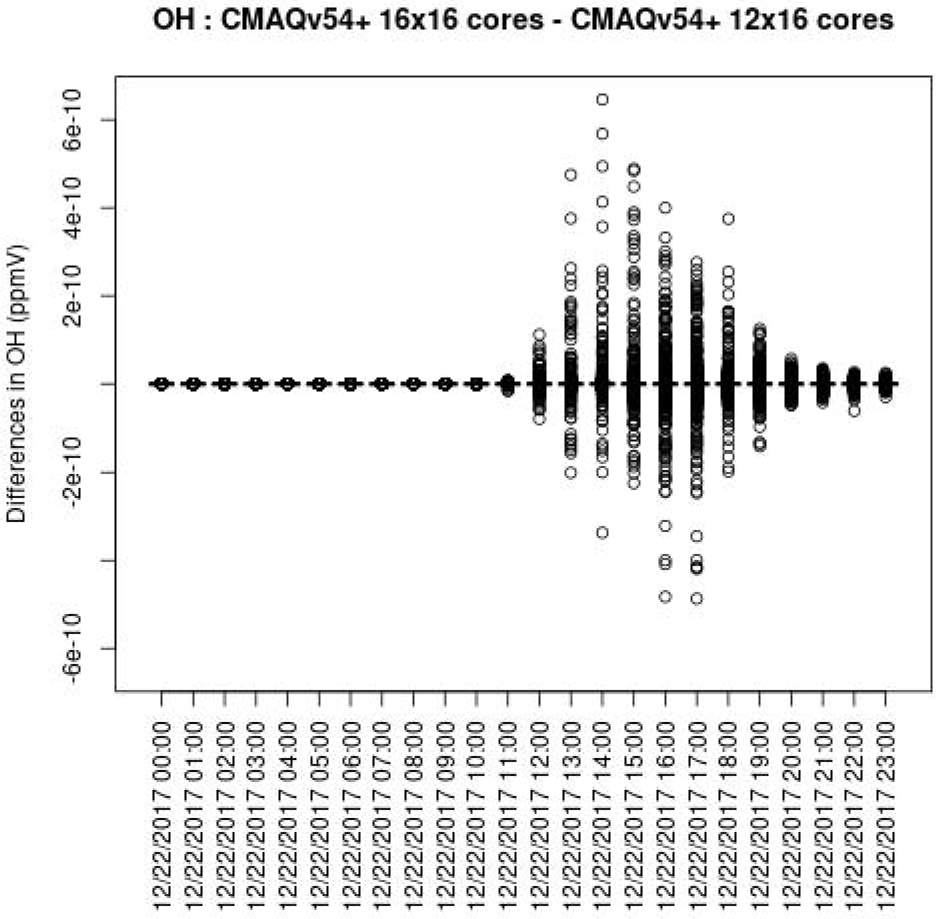
NO2


NO2 : CMAQv54+ 16x16 cores - CMAQv54+ 12x16 cores
Differences in NO2 (ppmV)
5e-06
0e+00
-5e-06
12/22/2017 00:00
12/22/2017 01:00
12/22/2017 02:00
12/22/2017 03:00
12/22/2017 04:00
12/22/2017 05:00
12/22/2017 06:00
12/22/2017 07:00
12/22/2017 08:00
12/22/2017 09:00
12/22/2017 10:00
12/22/2017 11:00
12/22/2017 12:00
12/22/2017 13:00
12/22/2017 14:00
12/22/2017 15:00
12/22/2017 16:00
12/22/2017 17:00
12/22/2017 18:00
12/22/2017 19:00
12/22/2017 20:00
12/22/2017 21:00
12/22/2017 22:00
12/22/2017 23:00

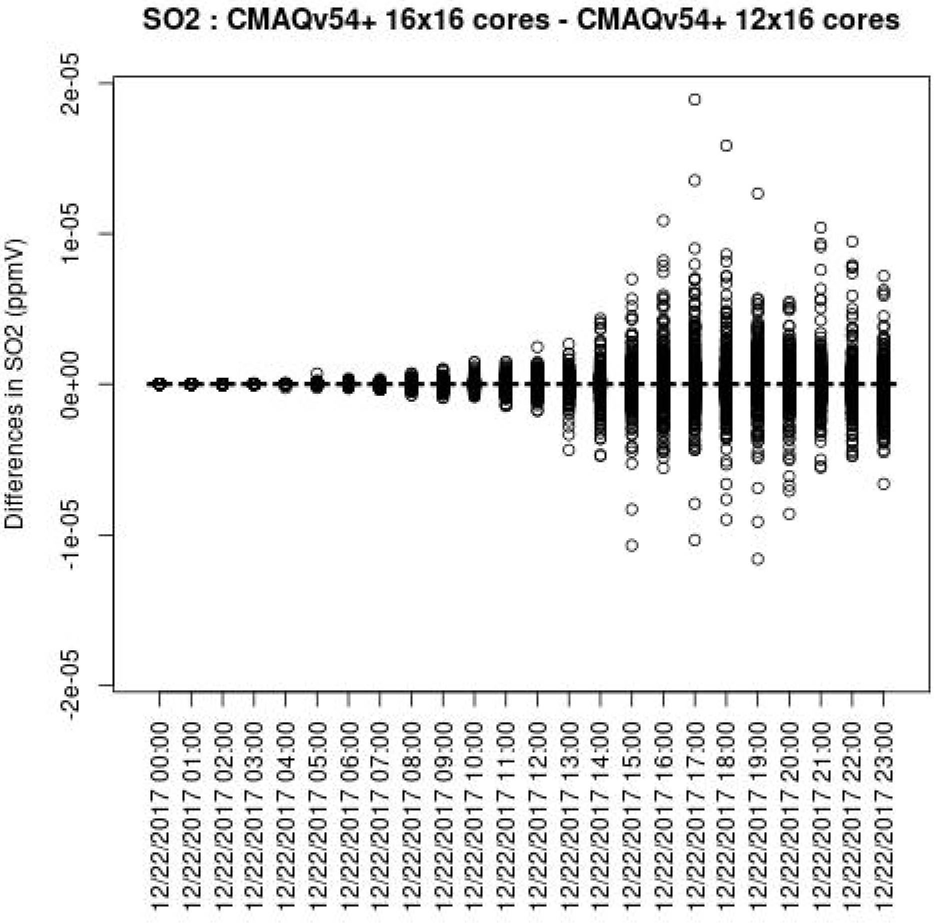
O3


O3 : CMAQv54+ 16x16 cores - CMAQv54+ 12x16 cores
Differences in O3 (ppmV)
1.5e-05
5.0e-06
-5.0e-06
-1.5e-05
12/22/2017 00:00
12/22/2017 01:00
12/22/2017 02:00
12/22/2017 03:00
12/22/2017 04:00
12/22/2017 05:00
12/22/2017 06:00
12/22/2017 07:00
12/22/2017 08:00
12/22/2017 09:00
12/22/2017 10:00
12/22/2017 11:00
12/22/2017 12:00
12/22/2017 13:00
12/22/2017 14:00
12/22/2017 15:00
12/22/2017 16:00
12/22/2017 17:00
12/22/2017 18:00
12/22/2017 19:00
12/22/2017 20:00
12/22/2017 21:00
12/22/2017 22:00
12/22/2017 23:00

OH


OH : CMAQv54+ 16x16 cores - CMAQv54+ 12x16 cores
Differences in OH (ppmV)
6e-10
4e-10
2e-10
-2e-10
-6e-10
12/22/2017 00:00
12/22/2017 01:00
12/22/2017 02:00
12/22/2017 03:00
12/22/2017 04:00
12/22/2017 05:00
12/22/2017 06:00
12/22/2017 07:00
12/22/2017 08:00
12/22/2017 09:00
12/22/2017 10:00
12/22/2017 11:00
12/22/2017 12:00
12/22/2017 13:00
12/22/2017 14:00
12/22/2017 15:00
12/22/2017 16:00
12/22/2017 17:00
12/22/2017 18:00
12/22/2017 19:00
12/22/2017 20:00
12/22/2017 21:00
12/22/2017 22:00
12/22/2017 23:00

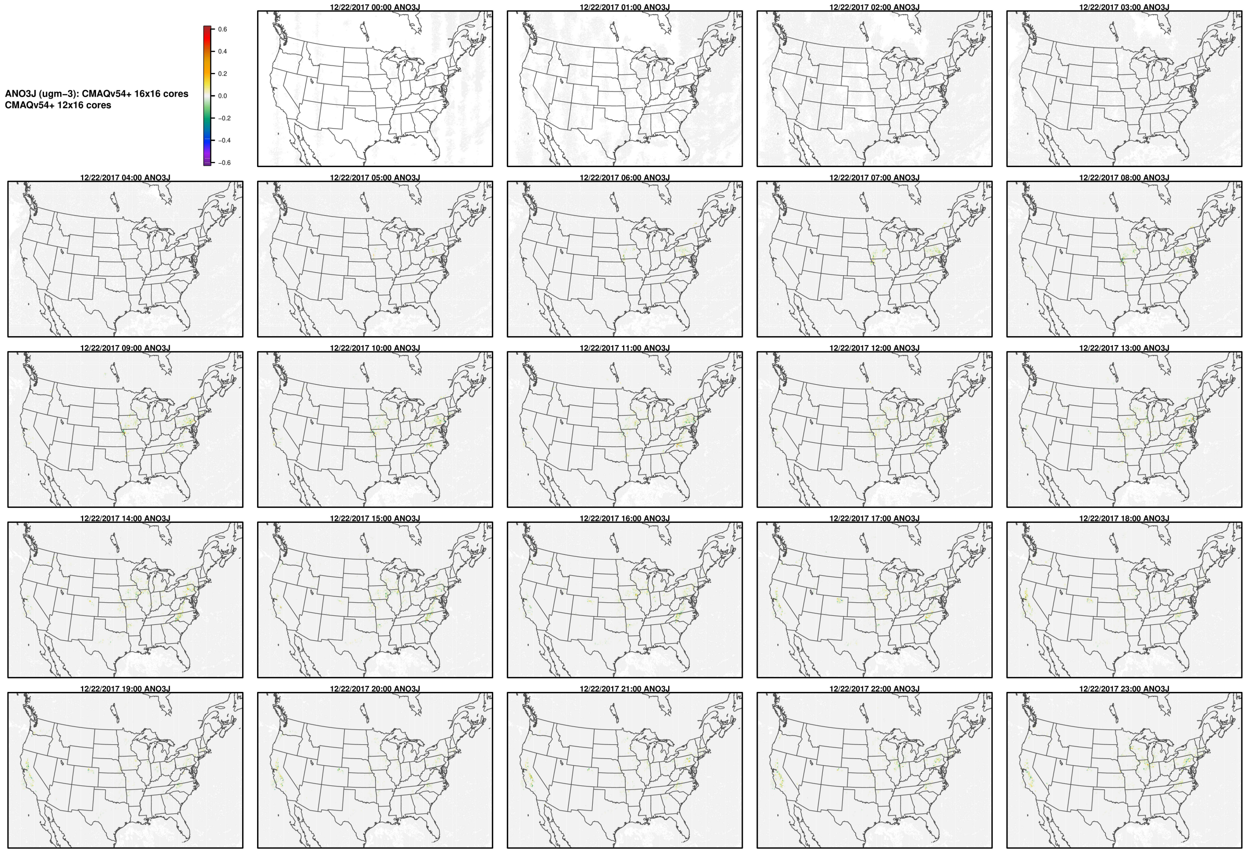
SO2



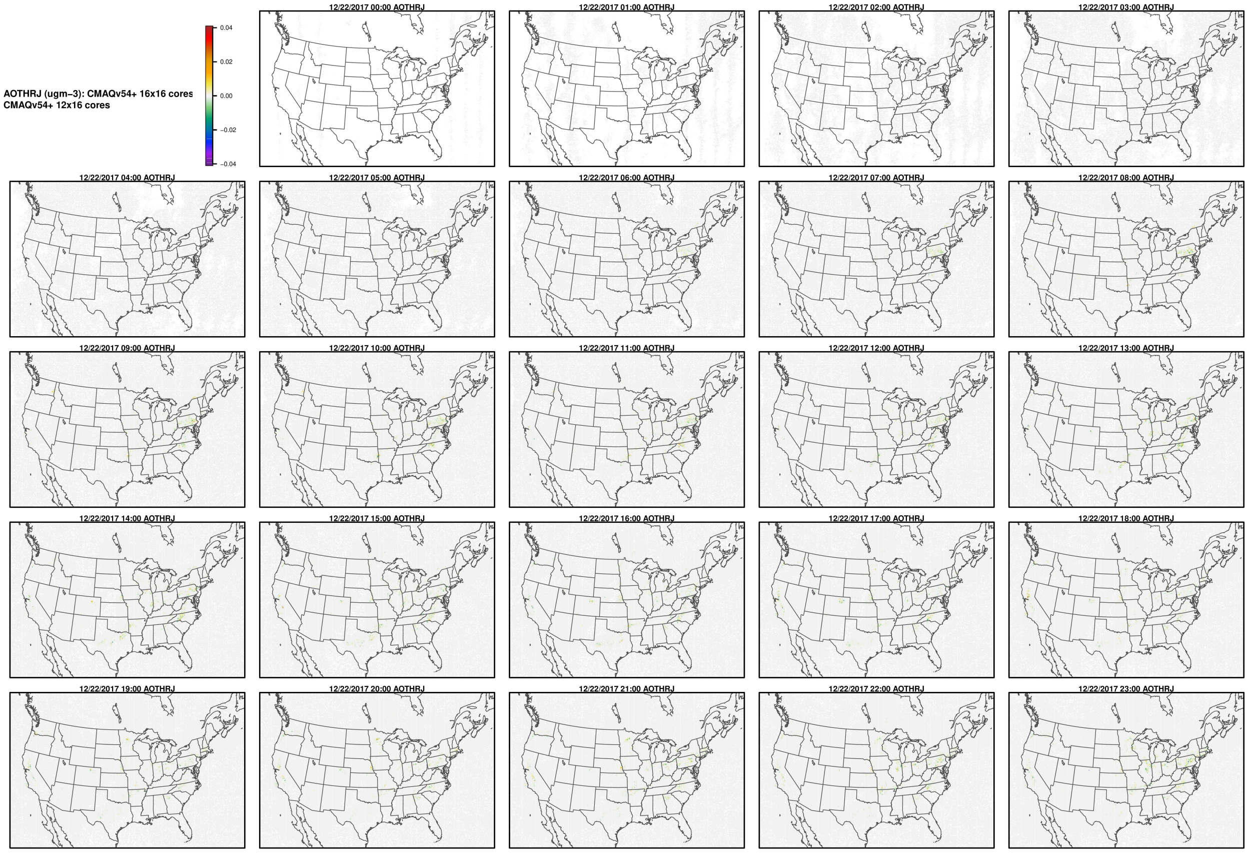
Note, the differences are small, but they grow with time. There is one plot for each of the 24 hours. The plot that contains the most differences will be in the bottom right of the panel for each species. You will need to zoom in to see the differences, as most of the grid cells do not have any difference, and they are displayed as grey. For the NO2 plot, you can see the most differences over the state of Pennsylvania at hour 12/22/2015 at hour 23:00, with the the magnitude of the maximum difference of +/- 4. E-6.

```
cd ../spatial_plots/16x16_vs_12x16
display O3_MAPS_CMAQv54+16x16cores_vs_CMAQv54+12x16cores-1.jpg
```

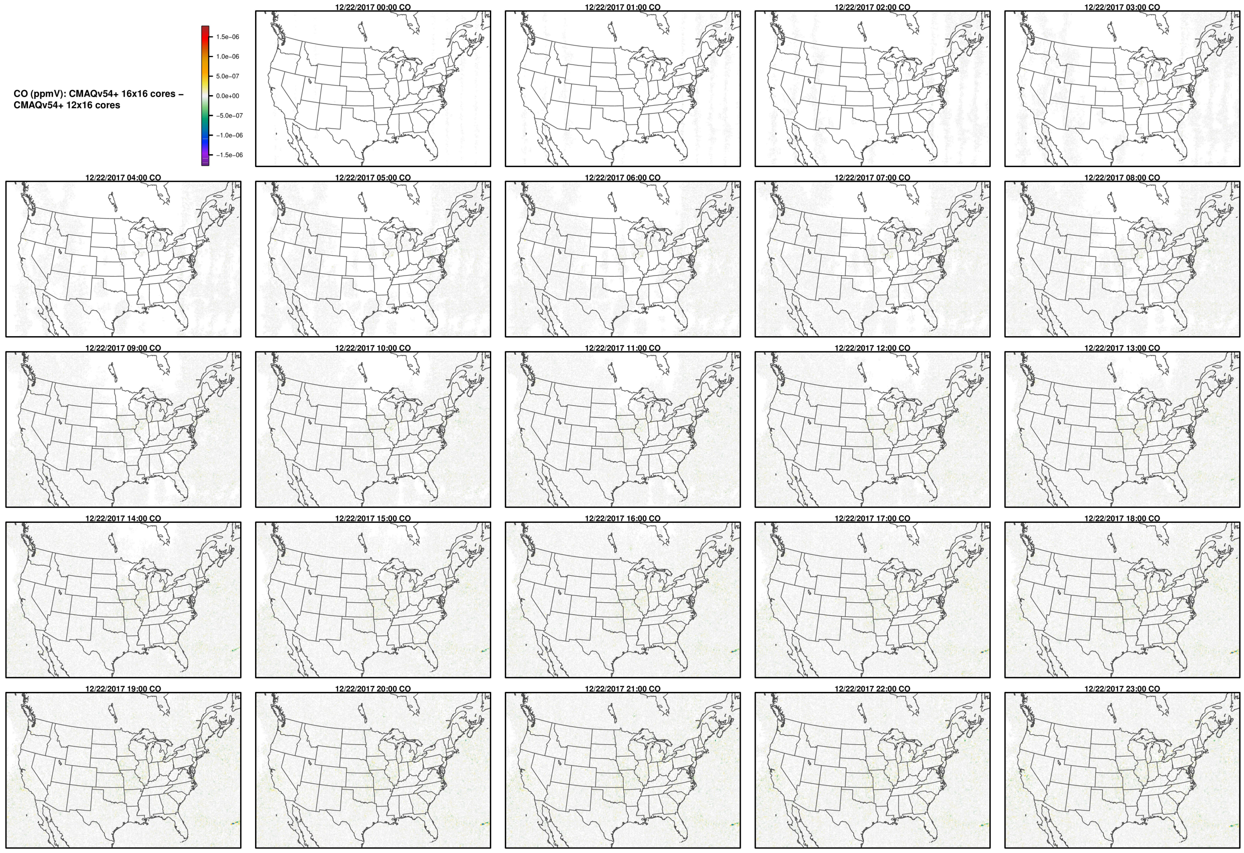
ANO3J


ANO3J (ugm−3): CMAQv54+ 16x16 cores
CMAQv54+ 12x16 cores
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
12/22/2017 00:00 ANO3J
12/22/2017 01:00 ANO3J
12/22/2017 02:00 ANO3J
12/22/2017 03:00 ANO3J
12/22/2017 04:00 ANO3J
12/22/2017 05:00 ANO3J
12/22/2017 06:00 ANO3J
12/22/2017 07:00 ANO3J
12/22/2017 08:00 ANO3J
12/22/2017 09:00 ANO3J
12/22/2017 10:00 ANO3J
12/22/2017 11:00 ANO3J
12/22/2017 12:00 ANO3J
12/22/2017 13:00 ANO3J
12/22/2017 14:00 ANO3J
12/22/2017 15:00 ANO3J
12/22/2017 16:00 ANO3J
12/22/2017 17:00 ANO3J
12/22/2017 18:00 ANO3J
12/22/2017 19:00 ANO3J
12/22/2017 20:00 ANO3J
12/22/2017 21:00 ANO3J
12/22/2017 22:00 ANO3J
12/22/2017 23:00 ANO3J

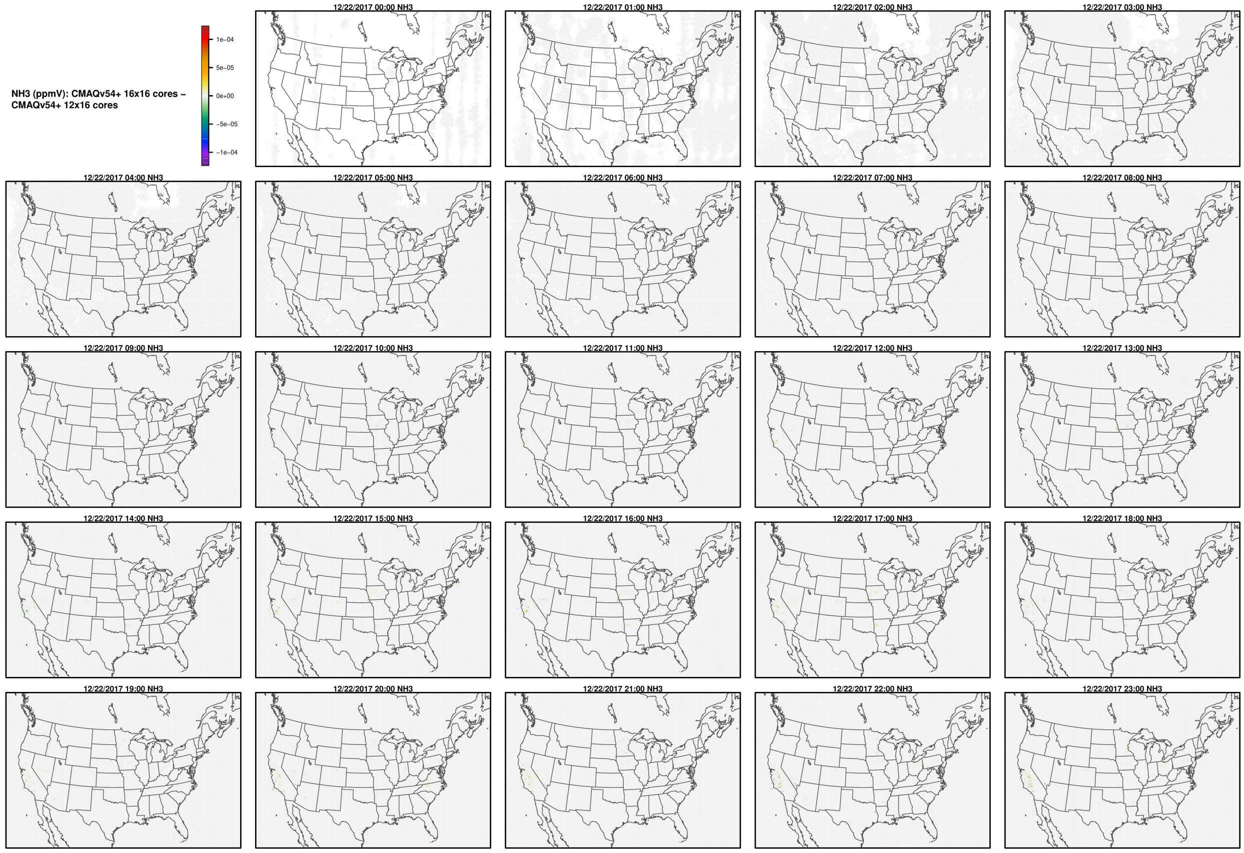
AOTHRJ


AOTHRJ (ugm−3): CMAQv54+ 16x16 cores
CMAQv54+ 12x16 cores
0.04
0.02
0.00
−0.02
−0.04
12/22/2017 00:00 AOTHRJ
12/22/2017 01:00 AOTHRJ
12/22/2017 02:00 AOTHRJ
12/22/2017 03:00 AOTHRJ
12/22/2017 04:00 AOTHRJ
12/22/2017 05:00 AOTHRJ
12/22/2017 06:00 AOTHRJ
12/22/2017 07:00 AOTHRJ
12/22/2017 08:00 AOTHRJ
12/22/2017 09:00 AOTHRJ
12/22/2017 10:00 AOTHRJ
12/22/2017 11:00 AOTHRJ
12/22/2017 12:00 AOTHRJ
12/22/2017 13:00 AOTHRJ
12/22/2017 14:00 AOTHRJ
12/22/2017 15:00 AOTHRJ
12/22/2017 16:00 AOTHRJ
12/22/2017 17:00 AOTHRJ
12/22/2017 18:00 AOTHRJ
12/22/2017 19:00 AOTHRJ
12/22/2017 20:00 AOTHRJ
12/22/2017 21:00 AOTHRJ
12/22/2017 22:00 AOTHRJ
12/22/2017 23:00 AOTHRJ

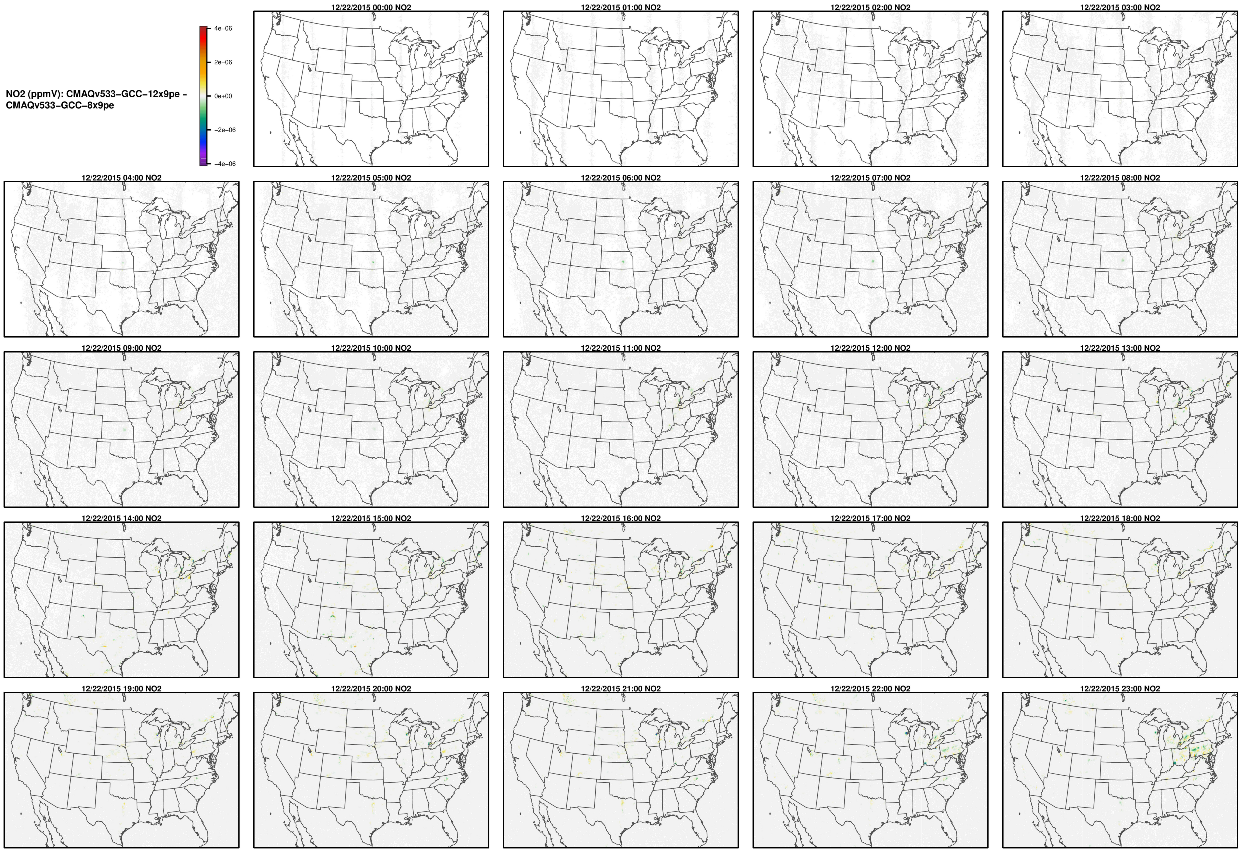
CO


CO (ppmV): CMAQv54+ 16x16 cores –
CMAQv54+ 12x16 cores
1.5e−06
1.0e−06
5.0e−07
0.0e+00
−5.0e−07
−1.0e−06
−1.5e−06
12/22/2017 00:00 CO
12/22/2017 01:00 CO
12/22/2017 02:00 CO
12/22/2017 03:00 CO
12/22/2017 04:00 CO
12/22/2017 05:00 CO
12/22/2017 06:00 CO
12/22/2017 07:00 CO
12/22/2017 08:00 CO
12/22/2017 09:00 CO
12/22/2017 10:00 CO
12/22/2017 11:00 CO
12/22/2017 12:00 CO
12/22/2017 13:00 CO
12/22/2017 14:00 CO
12/22/2017 15:00 CO
12/22/2017 16:00 CO
12/22/2017 17:00 CO
12/22/2017 18:00 CO
12/22/2017 19:00 CO
12/22/2017 20:00 CO
12/22/2017 21:00 CO
12/22/2017 22:00 CO
12/22/2017 23:00 CO

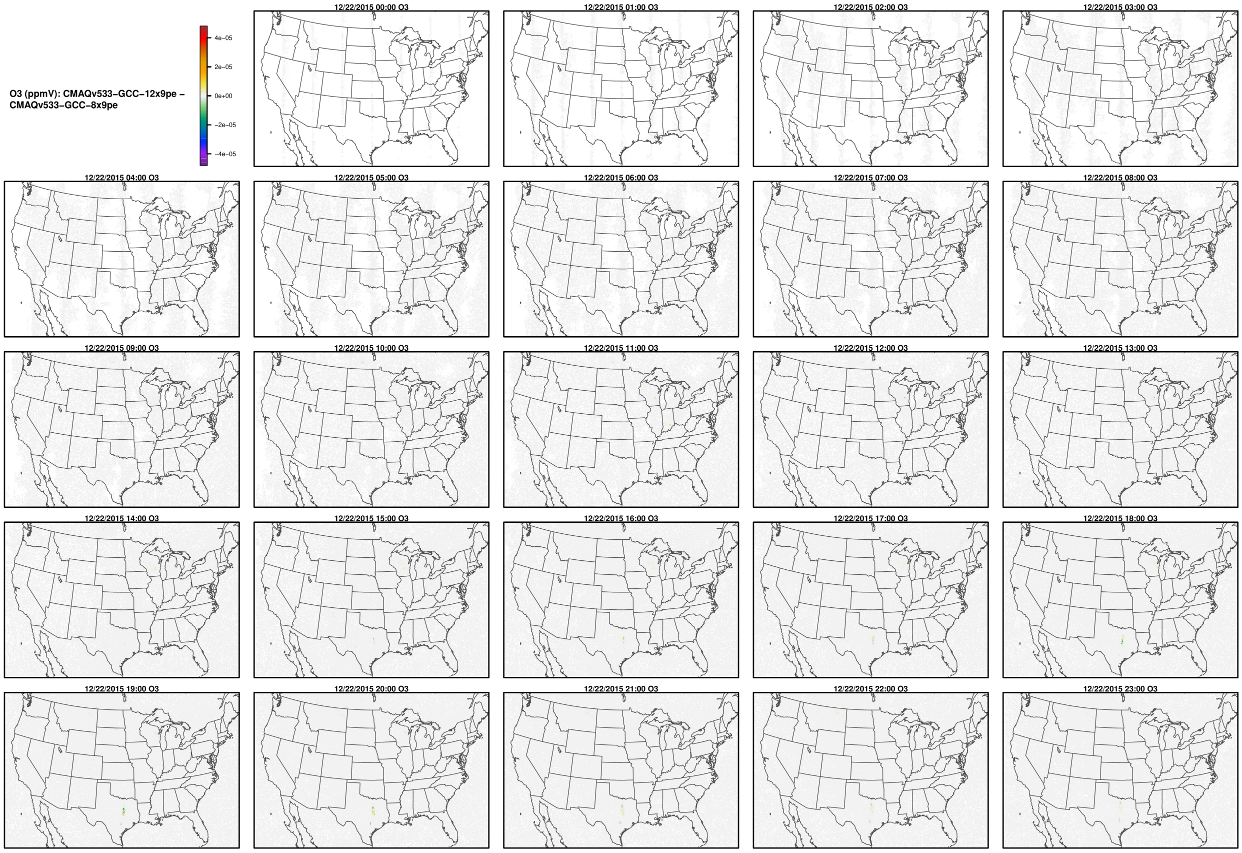
NH3


NH3 (ppmV): CMAQv54+ 16x16 cores – CMAQv54+ 12x16 cores
1e−04
5e−05
0e+00
−5e−05
−1e−04
12/22/2017 00:00 NH3
12/22/2017 01:00 NH3
12/22/2017 02:00 NH3
12/22/2017 03:00 NH3
12/22/2017 04:00 NH3
12/22/2017 05:00 NH3
12/22/2017 06:00 NH3
12/22/2017 07:00 NH3
12/22/2017 08:00 NH3
12/22/2017 09:00 NH3
12/22/2017 10:00 NH3
12/22/2017 11:00 NH3
12/22/2017 12:00 NH3
12/22/2017 13:00 NH3
12/22/2017 14:00 NH3
12/22/2017 15:00 NH3
12/22/2017 16:00 NH3
12/22/2017 17:00 NH3
12/22/2017 18:00 NH3
12/22/2017 19:00 NH3
12/22/2017 20:00 NH3
12/22/2017 21:00 NH3
12/22/2017 22:00 NH3
12/22/2017 23:00 NH3

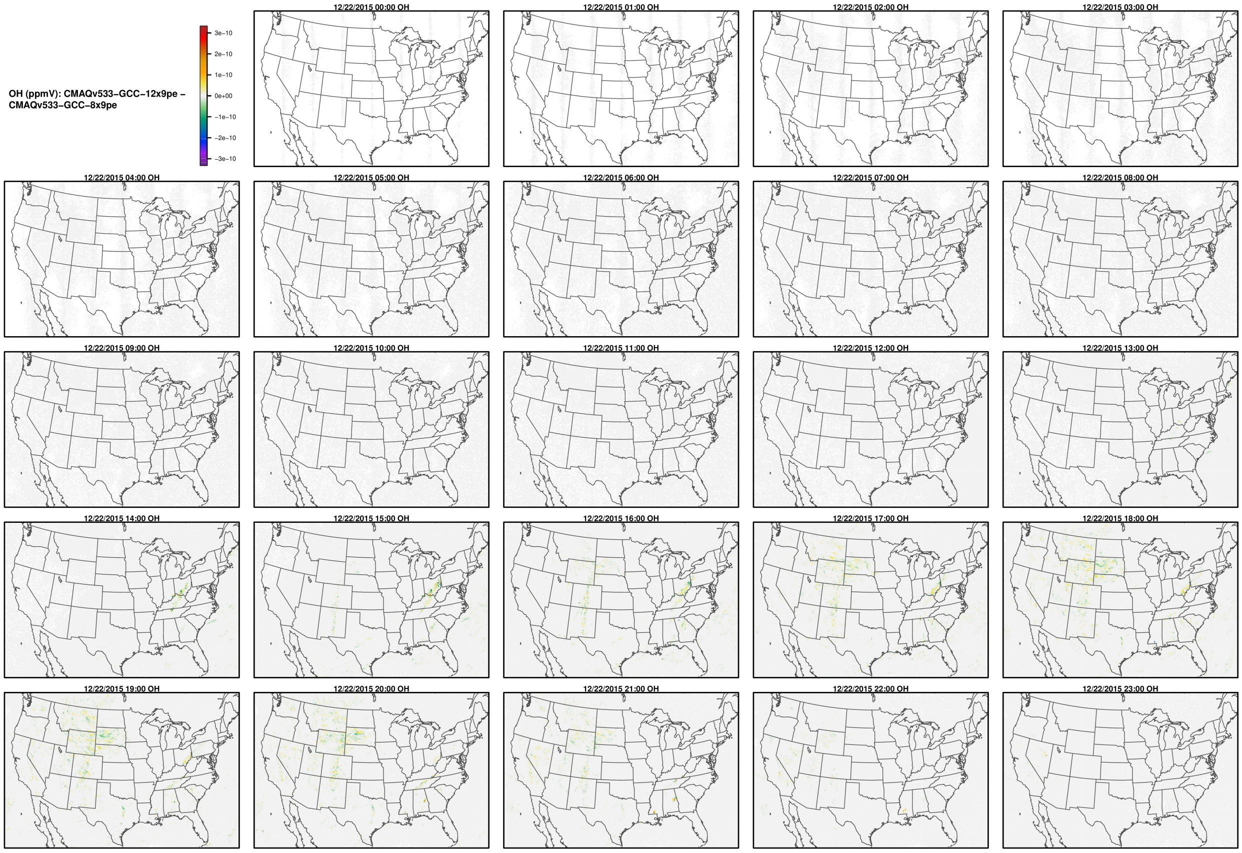
NO2

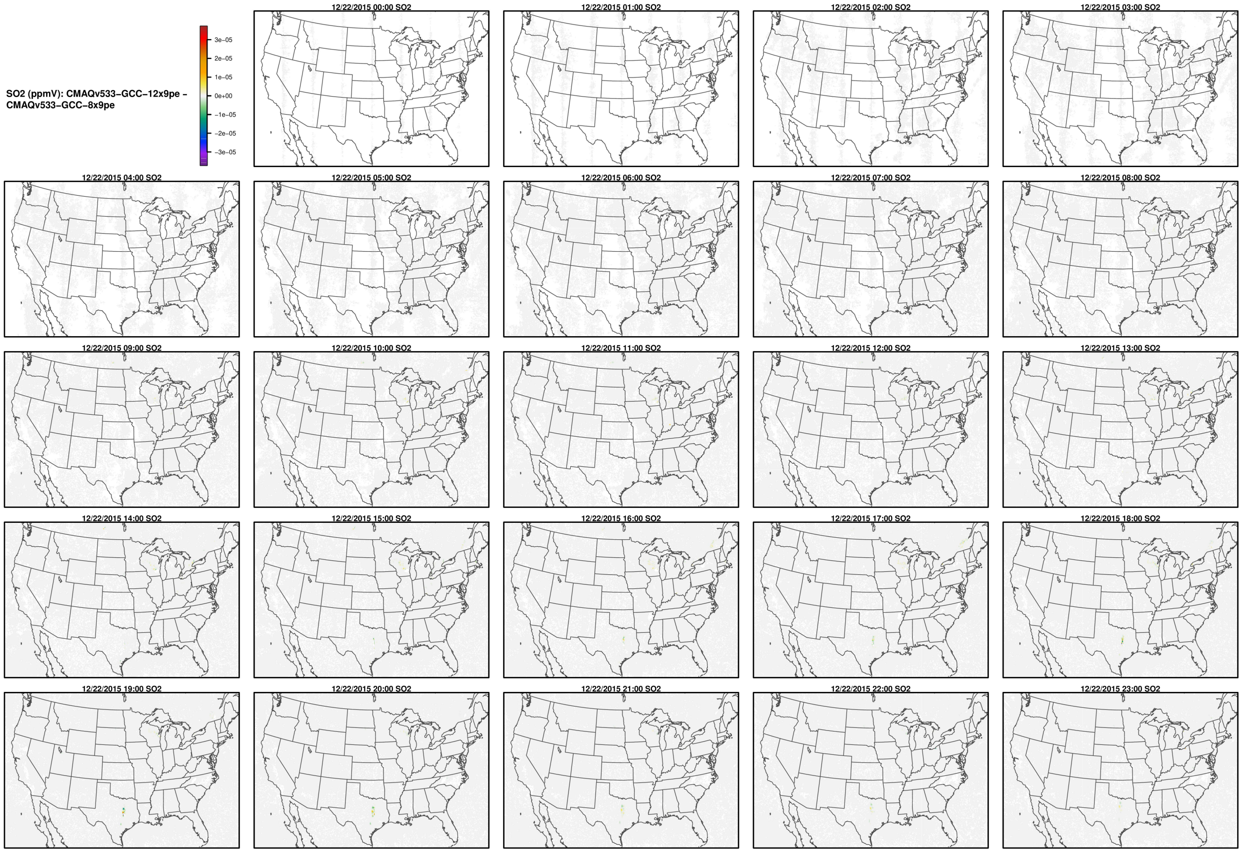

NO2 (ppmV): CMAQv533−GCC−12x9pe − CMAQv533−GCC−8x9pe
4e−06
2e−06
0e+00
−2e−06
−4e−06
12/22/2015 00:00 NO2
12/22/2015 01:00 NO2
12/22/2015 02:00 NO2
12/22/2015 03:00 NO2
12/22/2015 04:00 NO2
12/22/2015 05:00 NO2
12/22/2015 06:00 NO2
12/22/2015 07:00 NO2
12/22/2015 08:00 NO2
12/22/2015 09:00 NO2
12/22/2015 10:00 NO2
12/22/2015 11:00 NO2
12/22/2015 12:00 NO2
12/22/2015 13:00 NO2
12/22/2015 14:00 NO2
12/22/2015 15:00 NO2
12/22/2015 16:00 NO2
12/22/2015 17:00 NO2
12/22/2015 18:00 NO2
12/22/2015 19:00 NO2
12/22/2015 20:00 NO2
12/22/2015 21:00 NO2
12/22/2015 22:00 NO2
12/22/2015 23:00 NO2

O3


O3 (ppmV): CMAQv533−GCC−12x9pe −
CMAQv533−GCC−8x9pe
4e−05
2e−05
0e+00
−2e−05
−4e−05
12/22/2015 00:00 O3
12/22/2015 01:00 O3
12/22/2015 02:00 O3
12/22/2015 03:00 O3
12/22/2015 04:00 O3
12/22/2015 05:00 O3
12/22/2015 06:00 O3
12/22/2015 07:00 O3
12/22/2015 08:00 O3
12/22/2015 09:00 O3
12/22/2015 10:00 O3
12/22/2015 11:00 O3
12/22/2015 12:00 O3
12/22/2015 13:00 O3
12/22/2015 14:00 O3
12/22/2015 15:00 O3
12/22/2015 16:00 O3
12/22/2015 17:00 O3
12/22/2015 18:00 O3
12/22/2015 19:00 O3
12/22/2015 20:00 O3
12/22/2015 21:00 O3
12/22/2015 22:00 O3
12/22/2015 23:00 O3

OH


OH (ppmV): CMAQv533-GCC-12x9pe - CMAQv533-GCC-8x9pe
3e-10
2e-10
1e-10
0e+00
-1e-10
-2e-10
-3e-10
12/22/2015 00:00 OH
12/22/2015 01:00 OH
12/22/2015 02:00 OH
12/22/2015 03:00 OH
12/22/2015 04:00 OH
12/22/2015 05:00 OH
12/22/2015 06:00 OH
12/22/2015 07:00 OH
12/22/2015 08:00 OH
12/22/2015 09:00 OH
12/22/2015 10:00 OH
12/22/2015 11:00 OH
12/22/2015 12:00 OH
12/22/2015 13:00 OH
12/22/2015 14:00 OH
12/22/2015 15:00 OH
12/22/2015 16:00 OH
12/22/2015 17:00 OH
12/22/2015 18:00 OH
12/22/2015 19:00 OH
12/22/2015 20:00 OH
12/22/2015 21:00 OH
12/22/2015 22:00 OH
12/22/2015 23:00 OH


SO2 (ppmV): CMAQv533-GCC-12x9pe - CMAQv533-GCC-8x9pe
3e-05
2e-05
1e-05
0e+00
-1e-05
-2e-05
-3e-05
12/22/2015 00:00 SO2
12/22/2015 01:00 SO2
12/22/2015 02:00 SO2
12/22/2015 03:00 SO2
12/22/2015 04:00 SO2
12/22/2015 05:00 SO2
12/22/2015 06:00 SO2
12/22/2015 07:00 SO2
12/22/2015 08:00 SO2
12/22/2015 09:00 SO2
12/22/2015 10:00 SO2
12/22/2015 11:00 SO2
12/22/2015 12:00 SO2
12/22/2015 13:00 SO2
12/22/2015 14:00 SO2
12/22/2015 15:00 SO2
12/22/2015 16:00 SO2
12/22/2015 17:00 SO2
12/22/2015 18:00 SO2
12/22/2015 19:00 SO2
12/22/2015 20:00 SO2
12/22/2015 21:00 SO2
12/22/2015 22:00 SO2
12/22/2015 23:00 SO2

SO2

4.5.3 Compare Timing of CMAQ Routines

Compare the timing of CMAQ Routines for two different run configurations.

Parse timings from the log file

Compare the timings for the CONUS ParallelCluster Runs

Note: ParallelCluster Configurations can impact the model run times.

It is up the the user, as to what model run configurations are used to run CMAQ on the ParallelCluster. The following configurations may impact the run time of the model.

- Using different PE configurations, using DisableSimultaneousMultithreading: true in yaml file, using 36 cpus - no virtual cpus

  ```
  NPCOL x NPROW  , CPU   , SBATCH Command
  ```

  - [ ] 10x18 , 180 , #SBATCH –nodes=5, #SBATCH –ntasks-per-node=36
  - [ ] 16x16, 256 , #SBATCH –nodes=8, #SBATCH –ntasks-per-node=32
  - [ ] 16x18, 288 , #SBATCH –nodes=8, #SBATCH –ntasks-per-node=36
- Using different compute nodes
  - [ ] c5n.18xlarge (72 virtual cpus, 36 cpus) - with Elastic Fabric Adapter
  - [ ] c5n.9xlarge (36 virtual cpus, 18 cpus) - with Eleastic Fabric Adapter
  - [ ] c5n.4xlarge (16 virtual cpus, 4 cpus) - without Elastic Fabric Adapter
- With and without SBATCH –exclusive option
- With and without Elastic Fabric and Elastic Network Adapter turned on
- With and without network placement turned on
- Using different local storage options and copying versus importing data to lustre
  - [ ] input data imported from S3 bucket to lustre
  - [ ] input data copied from S3 bucket to lustre
  - [ ] input data copied from S3 bucket to an EBS volume
- Using different yaml settings for slurm
  - [ ] DisableSimultaneousMultithreading= true
  - [ ] DisableSimultaneousMultithreading= false

Edit the R script

First check to see what log files are available:

```
ls -lrt /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/*.log
```

Copy the log files to the repo, to save them, as once you log out and delete the cluster you won't have them. Note, they would need to be saved to your local fork of the repo.

```
cp /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts/*.log /shared/pcluster-cmaq/run_scripts/hpc7g.16xlarge/logs/
```

```
ls -lrt /shared/pcluster-cmaq/run_scripts/hpc7g.16xlarge/logs/
```

Modify the name of the log file to match what is avaible on your system.

```
cd /shared/pcluster-cmaq/qa_scripts
vi parse_timing.pes.lustre.cmaqv5.4.r
```

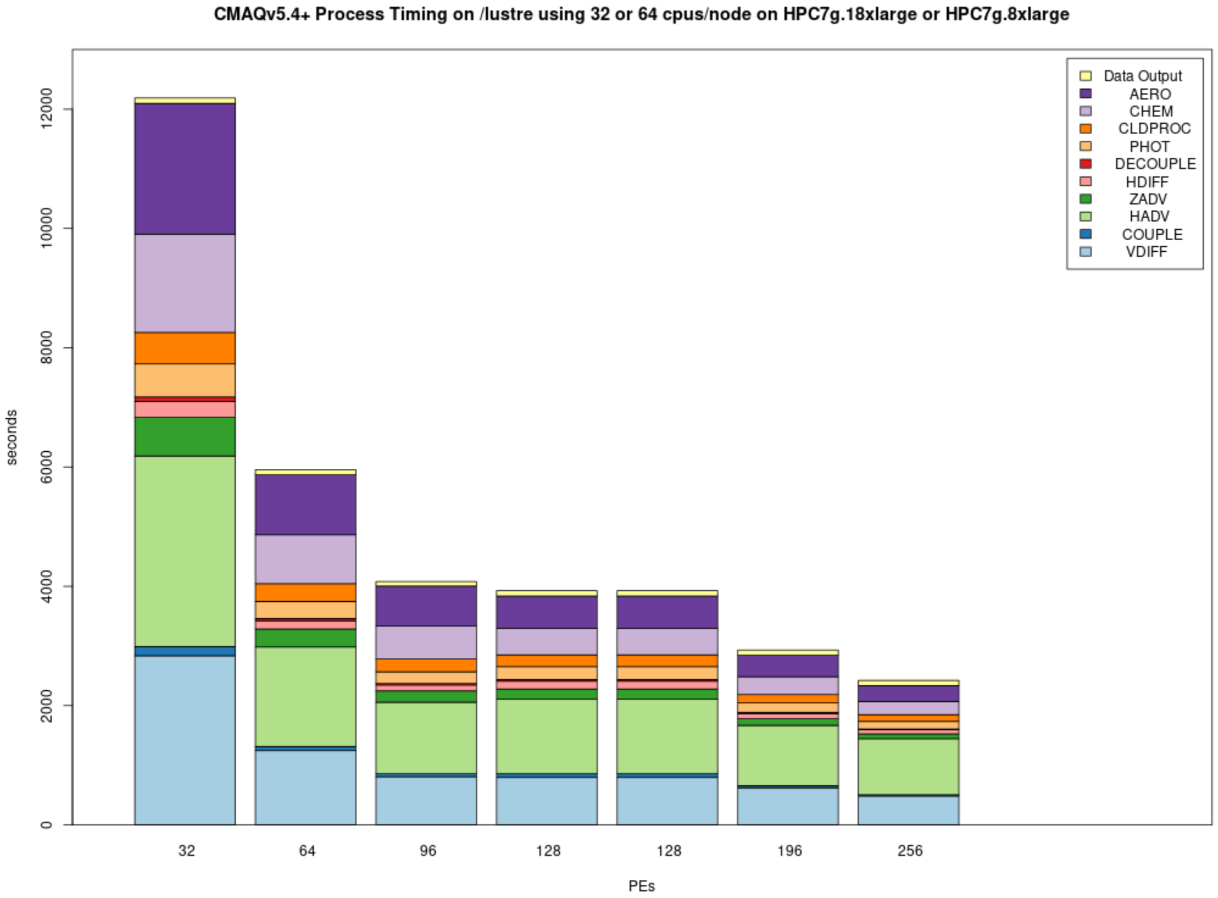
Edit the following section of the script to specify the log file names available on your ParallelCluster

```
sens.dir  <- '/shared/pcluster-cmaq/run_scripts/hpc7g.16xlarge/logs/'
base.dir  <- '/shared/pcluster-cmaq/run_scripts/hpc7g.16xlarge/logs/'
files     <- dir(sens.dir, pattern ='run_cctm5.4p_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.32.4x8pe.2day.20171222start.1x32.log' )
b.files <- dir(base.dir,pattern='run_cctm5.4p_Bench_2018_12US1_cb6r5_ae6_20200131_MYR.64.8x8pe.2day.20171222start.2x32.log')
#Compilers <- c('intel','gcc','pgi')
Compilers <- c('gcc')
# name of the base case timing. I am using the current master branch from the CMAQ_Dev repository.
# The project directory name is used for the sensitivity case.
base.name <- '12x9pe'
sens.name <- '6x18pe'
```

Run parse_timing.pes.lustre.cmaqv5.4.r script to examine timings of each science process in CMAQ

```
Rscript parse_timing.pes.lustre.cmaqv5.4.r
```

Timing Plot Comparing GCC run on 32, 64, 96, 128, 192, 256



4.5.4 Copy Output to S3 Bucket

Copy output from ParallelCluster to an S3 Bucket

Copy Output Data and Run script logs to S3 Bucket

Note: You need permissions to copy to a S3 Bucket.

See also:

S3 Access Control

Be sure you enter your access credentials on the parallel cluster by running:

`aws configure`

Currently, the bucket listed below has ACL turned off

See also:

S3 disable ACL

See example of sharing bucket across accounts.

See also:

Bucket owner granting cross-account permissions

Copy scripts and logs to /fsx

The CTM_LOG files don't contain any information about the compute nodes that the jobs were run on. Note, it is important to keep a record of the NPCOL, NPROW setting and the number of nodes and tasks used as specified in the run script: #SBATCH –nodes=16 #SBATCH –ntasks-per-node=8 It is also important to know what volume was used to read and write the input and output data, so it is recommended to save a copy of the standard out and error logs, and a copy of the run scripts to the OUTPUT directory for each benchmark.

```
cd /shared/build/openmpi_gcc/CMAQ_v54+/CCTM/scripts
mkdir -p /fsx/data/output/logs/
mkdir -p /fsx/data/output/scripts/
cp run*.log /fsx/data/output/logs/
cp run*.csh /fsx/data/output/scripts/
```

Examine the output files

Note: The following commands will vary depending on what APPL or domain decomposition was run

```
cd /fsx/data/output/output_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_12US1_3x64_classic
ls -lht
```

output:

```
total 26G
drwxrwxr-x 2 ubuntu ubuntu 153K Jul 12 20:59 LOGS
-rw-rw-r-- 1 ubuntu ubuntu 2.3M Jul 12 20:59 CCTM_BUDGET_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
↪2018_12US1_3x64_classic_20171223.txt
-rw-rw-r-- 1 ubuntu ubuntu 4.1G Jul 12 20:59 CCTM_CGRID_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
↪2018_12US1_3x64_classic_20171223.nc
-rw-rw-r-- 1 ubuntu ubuntu 1.9G Jul 12 20:58 CCTM_AELMO_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
↪2018_12US1_3x64_classic_20171223.nc
-rw-rw-r-- 1 ubuntu ubuntu 2.8G Jul 12 20:58 CCTM_ACONC_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
↪2018_12US1_3x64_classic_20171223.nc
-rw-rw-r-- 1 ubuntu ubuntu 171M Jul 12 20:58 CCTM_CONC_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
↪2018_12US1_3x64_classic_20171223.nc
-rw-rw-r-- 1 ubuntu ubuntu 1.8G Jul 12 20:58 CCTM_WETDEP1_v54+_cb6r5_ae7_aq_WR413_MYR_
↪gcc_2018_12US1_3x64_classic_20171223.nc
-rw-rw-r-- 1 ubuntu ubuntu 2.1G Jul 12 20:58 CCTM_DRYDEP_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
↪2018_12US1_3x64_classic_20171223.nc
-rw-rw-r-- 1 ubuntu ubuntu  51M Jul 12 20:58 CCTM_MEDIA_CONC_v54+_cb6r5_ae7_aq_WR413_MYR_
↪gcc_2018_12US1_3x64_classic_20171223.nc
-rw-rw-r-- 1 ubuntu ubuntu  15M Jul 12 20:58 CCTM_BSOILOUT_v54+_cb6r5_ae7_aq_WR413_MYR_
↪gcc_2018_12US1_3x64_classic_20171223.nc
-rw-rw-r-- 1 ubuntu ubuntu 3.7K Jul 12 20:30 CCTM_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_
↪12US1_3x64_classic_20171223.cfg
-rw-rw-r-- 1 ubuntu ubuntu 2.3M Jul 12 20:30 CCTM_BUDGET_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
```

(continues on next page)

(continued from previous page)

```
↪2018_12US1_3x64_classic_20171222.txt
-rw-rw-r-- 1 ubuntu ubuntu 4.1G Jul 12 20:30 CCTM_CGRID_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
↪2018_12US1_3x64_classic_20171222.nc
-rw-rw-r-- 1 ubuntu ubuntu 2.8G Jul 12 20:29 CCTM_ACONC_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
↪2018_12US1_3x64_classic_20171222.nc
-rw-rw-r-- 1 ubuntu ubuntu 1.9G Jul 12 20:29 CCTM_AELMO_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
↪2018_12US1_3x64_classic_20171222.nc
-rw-rw-r-- 1 ubuntu ubuntu 171M Jul 12 20:29 CCTM_CONC_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
↪2018_12US1_3x64_classic_20171222.nc
-rw-rw-r-- 1 ubuntu ubuntu 1.8G Jul 12 20:29 CCTM_WETDEP1_v54+_cb6r5_ae7_aq_WR413_MYR_
↪gcc_2018_12US1_3x64_classic_20171222.nc
-rw-rw-r-- 1 ubuntu ubuntu  51M Jul 12 20:29 CCTM_MEDIA_CONC_v54+_cb6r5_ae7_aq_WR413_MYR_
↪gcc_2018_12US1_3x64_classic_20171222.nc
-rw-rw-r-- 1 ubuntu ubuntu  15M Jul 12 20:29 CCTM_BSOILOUT_v54+_cb6r5_ae7_aq_WR413_MYR_
↪gcc_2018_12US1_3x64_classic_20171222.nc
-rw-rw-r-- 1 ubuntu ubuntu 2.1G Jul 12 20:29 CCTM_DRYDEP_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_
↪2018_12US1_3x64_classic_20171222.nc
-rw-rw-r-- 1 ubuntu ubuntu 3.7K Jul 12 20:00 CCTM_v54+_cb6r5_ae7_aq_WR413_MYR_gcc_2018_
↪12US1_3x64_classic_20171222.cfg
```

Check disk space

```
 du -sh
26G     .
```

Copy the output to an S3 Bucket

Examine the example script

```
cd /shared/pcluster-cmaq/s3_scripts
cat s3_upload.c7g.16xlarge.csh

```

output:

```
#!/bin/csh -f
# Script to upload output data to S3 bucket
# NOTE: a new bucket needs to be created to store each set of cluster runs

#cd /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts
#cp run*.log /fsx/data/output
#cp run*.csh /fsx/data/output

aws s3 mb s3://c7g-head-hpc7g.16xlarge-cmaqv5.4plus.12us1-output
aws s3 cp --recursive /fsx/data/output/ s3://c7g-head-hpc7g.16xlarge-cmaqv5.4plus.12us1-
↪output/jul-14-2023/fsx/data/output/
```

If you do not have permissions to write to the s3 bucket, you may need to ask the administrator of your account to add S3 Bucket writing permissions.

Run the script to copy all of the CMAQ output and logs to the S3 bucket.

```
./s3_upload.c7g.16xlarge.csh
```

4.6 Logout and Delete ParallelCluster

Logout and delete the ParallelCluster when you are done to avoid incurring costs.

4.6.1 Logout of cluster when you are done

To avoid incurring costs for the lustre file system and the c5n.xlarge compute node, it is best to delete the cluster after you have copied the output data to the S3 Bucket.

If you are logged into the Parallel Cluster then use the following command

```
exit
```

4.6.2 Delete Cluster

Run the following command on your local computer.

```
pcluster delete-cluster --region=us-east-1 --cluster-name cmaq
```

4.6.3 Verify that the cluster was deleted

```
pcluster describe-cluster --region=us-east-1 --cluster-name cmaq
```

Output:

```
"lastUpdatedTime": "2022-02-25T20:17:19.263Z",
  "region": "us-east-1",
  "clusterStatus": "DELETE_IN_PROGRESS"
```

Verify that you see the following output

```
pcluster describe-cluster --region=us-east-1 --cluster-name cmaq
```

Output:

```
pcluster describe-cluster --region=us-east-1 --cluster-name cmaq
{
  "message": "Cluster 'cmaq' does not exist or belongs to an incompatible␣
↪ParallelCluster major version."
}
```

4.7 Additional Resources

For a tutorial that explains cloud terminology as well as how to obtain single EC2 instances for running GEOS-CHEM on a single node, please see the Beginner Tutorial provided by GEOS-Chem as well as the resources in this chapter.

4.7.1 FAQ

Q. Can you update a cluster with a Snapshot ID, ie. update a cluster to use the /shared/build pre-installed software?

A. No. An existing cluster can not be updated with a Snapshot ID, solution is to delete the cluster and re-create it. see error:

`pcluster update-cluster --region us-east-1 --cluster-name cmaq --cluster-configuration c5n-18xlarge.ebs_unencrypted.fsx_import.yaml`

Output:

```
{
  "message": "Update failure",
  "updateValidationErrors": [
    {
      "parameter": "SharedStorage[ebs-shared].EbsSettings.SnapshotId",
      "requestedValue": "snap-065979e115804972e",
      "message": "Update actions are not currently supported for the 'SnapshotId'␣
↪parameter. Remove the parameter 'SnapshotId'. If you need this change, please consider␣
↪creating a new cluster instead of updating the existing one."
    }
  ],
  "changeSet": [
    {
      "parameter": "SharedStorage[ebs-shared].EbsSettings.SnapshotId",
      "requestedValue": "snap-065979e115804972e"
    }
  ]
}
```

Q. How do you figure out why a job isn't successfully running in the slurm queue?

A. Check the logs available in the following link

Pcluster Troubleshooting

`vi /var/log/parallelcluster/slurm_resume.log`

Output:

```
2022-03-23 21:04:23,600 - [slurm_plugin.instance_manager:_launch_ec2_instances] - ERROR -
↪ Failed RunInstances request: 0c6422af-c300-4fe6-b942-2b7923f7b362
2022-03-23 21:04:23,600 - [slurm_plugin.instance_manager:add_instances_for_nodes] -␣
↪ERROR - Encountered exception when launching instances for nodes (x3) ['queue1-dy-
↪compute-resource-1-4', 'queue1-dy-compute-resource-1-5', 'queue1-dy-compute-resource-1-
↪6']: An error occurred (InsufficientInstanceCapacity) when calling the RunInstances␣
↪operation (reached max retries: 1): We currently do not have sufficient c5n.18xlarge␣
↪capacity in the Availability Zone you requested (us-east-1a). Our system will be␣
↪working on provisioning additional capacity. You can currently get c5n.18xlarge␣
```

(continues on next page)

(continued from previous page)

```
↪capacity by not specifying an Availability Zone in your request or choosing us-east-1b,
↪ us-east-1c, us-east-1d, us-east-1f.
```

Q. How do I determine what node(s) the job is running on?

A. echo $SLURM_JOB_NODELIST

Slurm Environment Variables

Q. I see other tutorials that use a configure file instead of a yaml file to create the cluster. Can I use this instead?

A. No, you must convert the text based config file to a yaml file to use with the Parallel Cluster CLI 3.+ version used in this tutorial. An example of this type of tutorial is < a href="https://aws.amazon.com/blogs/compute/fire-dynamics-simulation-cfd-workflow-using-aws-parallelcluster-elastic-fabric-adapter-amazon-fsx-for-lustre-and-nice-dcv/"> Fire Dynamics Simulation CFD workflow using AWS ParallelCluster, Elastic Fabric Adapter, Amazon FSx for Lustre and NICE DCV You can try to use the v2 to v3 converter, see more: moving from v2 to v3

Q. If I find an issue, or need help with this CMAQ ParallelCluster Tutorial what do I do?

A. Please file an issue using github.

Submit Github Issue for help with documentation

Please indicate the issue you are having, and include a link from the read the doc section that you are referring to. The tutorial documentation has an edit icon in the upper right corner of each page. You can click on that, and github will fork the repo and allow you to edit the page. After you have made the edits, you can submit a pull request, and then include the link to the pull request in the github issue.

4.7.2 Free Training

Numerical Weather Prediction HPC Workship

AWS Free Training Courses

4.7.3 Another workshop to learn the AWS CLI 3.0

Workshop on learning AWS CLI 3.0

4.7.4 Youtube video

Youtube video on AWS CLI 3.0

4.7.5 Intro to AWS for HPC People - HPC Tech Shorts

Intro to AWS for HPC People - Tech Short Foundations Level 1

4.7.6 Benchmarking

Benchmarks optimized for HPC high memory

AWS Graviton WRF Performance Comparison

Deep Dive hpc7g

AWS EPYC and WRF Performance Comparison

4.7.7 Help Resources for CMAQ

1. CMAS Center Forum
2. EPA CMAQ Website
3. UNC CMAS Center Website

4.7.8 Computing on the Cloud References

WRF Cloud Computing Paper

AWS High Performance Computing (HPC) Lens for the AWS Well-Architected Framework

AWS High Performance Computing (HPC) Lens for the AWS Well-Architected Framework

HPC on AWS - WRF (uses cfnCluster - older version of Parallel Cluster

HPC on AWS

WRF on Parallel Cluster

A Scientist Guide to Cloud-HPC: Example with AWS ParallelCluster, Slurm, Spack, and WRF

Advancing Large Scale Weather and Climate Modeling Data in the Cloud

AWS and Intel Research Webinar Series: Advancing the large scale weather and climate modeling data in the cloud

AWS Well-Architected Framework

AWS Well-Architected Framework

Cost Comparison on-premisis and cloud

WRF Performance on Google Cloud

<a href=”https://journal.fluidnumerics.com/comparing-on-premise-and-cloud-costs-for-high-performance-computing>Comparing on-premise and cloud costs for hpc

4.7.9 AWS Resources for the aws cli method to launch ec2 instances.

aws cli examples

aws cli run instances command

Tutorial Launch Spot Instances

Launching EC2 Spot Instances using Run Instances API

Additional resources for spot instance provisioning.

Spot Instance Requests

4.7.10 Resources from AWS for diagnosing issues with running the Parallel Cluster

1. Github for AWS Parallel Cluster
2. User Guide
3. Getting Started Guide
4. Guide to obtaining AWS Key Pair
5. Lustre FAQ
6. Parallel Cluster FAQ (somewhat outdated..)
7. Tool to convert v2 config files to v3 yaml files for Parallel Cluster
8. Instructions for creating a fault tolerance parallel cluster using lustre filesystem
9. AWS HPC discussion forum

Issues

For AWS Parallel Cluster you can create a GitHub issue for feedback or issues: Github Issues There is also an active community driven Q&A site that may be helpful: AWS re:Post a community-driven Q&A site

Tips to managing the parallel cluster

1. The head node can be stopped from the AWS Console after stopping compute nodes of the cluster, as long as it is restarted before issuing the command to restart the cluster.
2. The pcluster slurm queue system will create and delete the compute nodes, so that helps reduce manual cleanup for the cluster.
3. The compute nodes are terminated after they have been idle for a period of time. The yaml setting used for this is as follows: SlurmSettings: ScaledownIdletime: 5
4. The default idle time is 10 minutes, and can be reduced by specifing a shorter idle time in the YAML file. It is important to verify that the are deleted after a job is finished, to avoid incurring unexpected costs.

5. copy/backup the outputs and logs to an s3 bucket for follow-up analysis
6. After copying output and log files to the s3 bucket the cluster can be deleted
7. Once the pcluster is deleted all of the volumes, head node, and compute node will be terminated, and costs will only be incurred by the S3 Bucket storage.

4.7.11 Instructions on how to create Parallel Cluster Amazon Machine Image (AMI) from the command line

Tutorial How-to Create AMI from Command Line

We also need to have additional protections if we make these AMI's public.

Building Shared AMIs

Securing Access to AMIs for AWS Marketplace

Building Pcluster from Existing AMI

4.7.12 ParallelCluster Update

1. not all settings in the yaml file can be updated
2. it is important to know what the policy is for each setting

Example Update policy:

If this setting is changed, the update is not allowed. After changing this setting, the cluster can't be updated. Either the change must be reverted or the cluster must be deleted (using pcluster delete-cluster), and then a new cluster created (using pcluster create-cluster) in the old cluster's place.

see more information

ParallelCluster Update Policy

4.7.13 Use Elastic Fabric Adapter/Elastic Network Adapter for better performance

"In order to make the most of the available network bandwidth, you need to be using the latest Elastic Network Adapter (ENA) drivers (available in the latest Amazon Linux, Red Hat 7.6, and Ubuntu AMIs, and in the upstream Linux kernel) and you need to make use of multiple traffic flows. Flows within a Placement Group can reach 10 Gbps; the rest can reach 5 Gbps. When using multiple flows on the high-end instances, you can transfer 100 Gbps between EC2 instances in the same region (within or across AZs), S3 buckets, and AWS services such as Amazon Relational Database Service (RDS), Amazon ElastiCache, and Amazon EMR."

The above was quoted from the following link:

C5n Instances

Elastic Fabric Adapter for HPC systems

"EFA is currently available on c5n.18xlarge, c5n.9xlarge, c5n.metal, i3en.24xlarge, i3en.metal, inf1.24xlarge, m5dn.24xlarge, m5n.24xlarge, r5dn.24xlarge, r5n.24xlarge, p3dn.24xlarge, p4d, m6i.32xlarge, m6i.metal, c6i.32xlarge, c6i.metal, r6i.32xlarge, and r6i.metal instances."

What are the differences between an EFA ENI and an ENA ENI?

"An ENA ENI provides traditional IP networking features necessary to support VPC networking. An EFA ENI provides all the functionality of an ENA ENI, plus hardware support for applications to communicate directly with the EFA ENI without involving the instance kernel (OS-bypass communication) using an extended programming interface. Due to the advanced capabilities of the EFA ENI, EFA ENIs can only be attached at launch or to stopped instances."

Q: What are the pre-requisites to enabling EFA on an instance?

"EFA support can be enabled either at the launch of the instance or added to a stopped instance. EFA devices cannot be attached to a running instance."

Elastic Fabric Adapter for Tightly Coupled Workloads

Quoted from the above link.

"An EFA can still handle IP traffic, but also supports an important access model commonly called OS bypass. This model allows the application (most commonly through some user-space middleware) access the network interface without having to get the operating system involved with each message. Doing so reduces overhead and allows the application to run more efficiently. Here's what this looks like (source):"

"The MPI Implementation and libfabric layers of this cake play crucial roles:"

"MPI – Short for Message Passing Interface, MPI is a long-established communication protocol that is designed to support parallel programming. It provides functions that allow processes running on a tightly-coupled set of computers to communicate in a language-independent way."

"libfabric – This library fits in between several different types of network fabric providers (including EFA) and higher-level libraries such as MPI. EFA supports the standard RDM (reliable datagram) and DGRM (unreliable datagram) endpoint types; to learn more, check out the libfabric Programmer's Manual. EFA also supports a new protocol that we call Scalable Reliable Datagram; this protocol was designed to work within the AWS network and is implemented as part of our Nitro chip."

"Working together, these two layers (and others that can be slotted in instead of MPI), allow you to bring your existing HPC code to AWS and run it with little or no change.

"You can use EFA today on c5n.18xlarge and p3dn.24xlarge instances in all AWS regions where those instances are available. The instances can use EFA to communicate within a VPC subnet, and the security group must have ingress and egress rules that allow all traffic within the security group to flow. Each instance can have a single EFA, which can be attached when an instance is started or while it is stopped."

"You will also need the following software components:"

"EFA Kernel Module – The EFA Driver is in the Amazon GitHub repo; read Getting Started with EFA to learn how to create an EFA-enabled AMI for Amazon Linux, Amazon Linux 2, and other popular Linux distributions."

"Libfabric Network Stack – You will need to use an AWS-custom version for now; again, the Getting Started document contains installation information. We are working to get our changes into the next release (1.8) of libfabric."

"Note the parallel cluster deplopyment takes care of setting this up for you."

4.7.14 VPC Management

There is a limit on the number of VPCs that are allowed per account - limit is 5.

What is the difference between a private and a public vpc? (what setting is used in the yaml file, and why is one preferred over the other?)

Note, there is a default VPC, that is used to create EC2 instances, that should not be deleted.

Q1. is there a separate default VPC for each region?

Q2. Each time you run a configure cluster command, does the ParallelCluster create a new VPC?

Q3. Why don't the VPC and subnet IDs get deleted when the ParallelClusters are deleted?

Deleting VPCs

If pcluster configure created a new VPC, you can delete that VPC by deleting the AWS CloudFormation stack it created. The name will start with "parallelclusternetworking-" and contain the creation time in a "YYYYMMDDHHMMSS" format. You can list the stacks using the list-stacks command. The following instructions are available here:

Instructions for Cleaning Up VPCs

```
$ aws --region us-east-2 cloudformation list-stacks \
   --stack-status-filter "CREATE_COMPLETE" \
   --query "StackSummaries[].StackName" | \
   grep -e "parallelclusternetworking-""parallelclusternetworking-pubpriv-20191029205804"
```

The stack can be deleted using the delete-stack command.

```
$ aws --region us-west-2 cloudformation delete-stack \
   --stack-name parallelclusternetworking-pubpriv-20191029205804
```

If pcluster configure created a new VPC, you can delete that VPC by deleting the AWS CloudFormation stack it created. The name will start with "parallelclusternetworking-" and contain the creation time in a "YYYYMMDDHHMMSS" format. You can list the stacks using the list-stacks command.

Pcluster Configure

Note: I can see why you wouldn't want to delete the VPC, if you want to reuse the yaml file that contains the SubnetID that is tied to that VPC.

I was able to use the Amazon Website to find the SubnetID, and then identify the VPC that it is part of.

I currently have the following VPCs

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR (Network border group)	IPv6 pool	DHCP options set	Main route table	Main network ACL	Tenancy	Default VPC	Owner ID
ParallelClusterVPC-20211210200003	vpc-0345c3fa089b004d8	Available	10.0.0.0/16	–	–	dopt-eaeaf888	rtb-048c503f3[illegible]	acl-[illegible]f04ead	Default	No	xxxx
ParallelClusterVPC-20211021183810	vpc-0e3f4e34a[illegible]f80f06	Available	10.0.0.0/16	–	–	dopt-eaeaf888	rtb-0a5b7ac98[illegible]	acl-[illegible]db68c	Default	No	xxxx
-	vpc-3cfc5759	Available	172.31.0.0/16	–	–	dopt-eaeaf888	rtb-99cd64fc	acl-bb9b39de	Default	Yes	440858712842
ParallelClusterVPC-2021041917455[illegible]	vpc-[illegible]ab948b66[illegible]c71ea	Available	10.0.0.0/16	–	–	dopt-eaeaf888	rtb-03fd47f05[illegible]	acl-[illegible]2858	Default	No	xxxx
ParallelClusterVPC-20211021174405	vpc-0f34a572da[illegible]15e49	Available	10.0.0.0/16	–	–	dopt-eaeaf888	rtb-0b6310d9e[illegible]	acl-[illegible]45e91	Default	No	xxxx

This is the subnet id that I am currently using in the yaml files: subnet-018cfea3edf3c4765

I currently have 11 subnet IDs - how many are no longer being used?

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Available IPv4 addresses	Availability Zone	Availability Zone ID	Network border group	Route table	Network ACL	Default subnet	Auto-assign public IPv4 address	Auto-assign customer-owned IPv4 address	Customer-owned IPv4 pool	Auto-assign IPv6 address	Owner ID
parallelcluster:public-subnet	subnet-018cfea3edf3	Available	vpc-0447653fa089b004d8-ParallelClusterVPC-20211210200003	10.0.0.0/20	-	4091	us-east-1a	use1-az6	us-east-1	rtb-034bca0be | parallelcluster:route-table-public	acl-0f9ef178f4023ead	No	Yes	No	-	No	xx

4.7.15 Using Cost Allocation Tags with ParallelCluster

This blog post uses the v2 command line Using Cost Allocation Tags

Need to update instructions for AWS v3 CLI - using yaml files.

4.8 Future Work

4.8.1 Future Work

AWS ParallelCluster

- Create yaml and software install scripts for intel compiler
- Benchmark 2 day case using intel compiler version of CMAQ and compare to GCC timings
- Repeat Benchmark Runs using c6gn.16xlarge compute nodes AMD Graviton and compare to Azure Cycle Cloud HBV3 compute nodes.
- Create script for installing all software and R packages as a custom bootstrap as the ParallelCluster is created.
- Create method to automatically checkpoint and save a job prior to it being bumped from the schedule if running on spot instances.
- Set up an additional slurm queue that uses a smaller compute node to do the post-processing and learn how to submit the post processing jobs to this queue, rather than running them on the head node.
- Install software using SPACK
- Install netCDF-4 compressed version of I/O API Library and set up environment module to compile and run CMAQ for 2018_12US1 data that is nc4 compressed

Documentation

- Create instructions on how to create a ParallelCluster using encrypted ebs volume and snapshot.

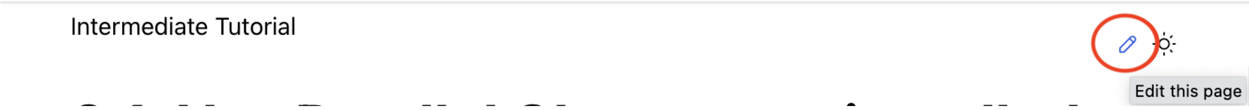
4.9 Contribute to this Tutorial

The community is encouraged to contribute to this documentation. It is open source, created by the CMAS Center, under contract to EPA, for the benefit of the CMAS Community.

4.9.1 Contribute to Pcluster-cmaq Documentation

Please take note of any issues and submit to Github Issue

Note: At the top of each page of the documentation, there is also an pencil icon, that you can click. It will create a fork of the project on your github account that you can make edits and then submit a pull request.



If you are able to create a pull request, please include the following in your issue:

- pull request number

If you are not able to create a pull request, please include the following in your issue:

- section number
- description of the issue encountered
- recommended fix, if available