PROWESS

Behind the Report—Testing Addendum:

OpenRadioss on AWS®: HPC Workload Testing

Methodology

Summary

OpenRadioss is a powerful, open-source software for analyzing dynamic events. Researchers and engineers use OpenRadioss to support the rapid development of technologies like batteries, composite materials, human body models, and autonomous driving.

Altair released OpenRadioss, an open-source version of its Altair® Radioss® finite element analysis (FEA) dynamic simulation code, in September 2022. Altair derived its models from the LS-DYNA® models originally developed by the National Crash Analysis Center (NCAC), a research and resource center established by the US Department of Transportation (DOT).

For this high-performance computing (HPC) workload testing, sponsored by Intel, Prowess Consulting tested different Amazon Web Services® (AWS®) instance configurations. We changed the following variables: instance type to vary the underlying processor, OpenRadioss model type, and the OpenMP® thread setting to configure physical cores. We aimed to identify the instances that delivered the most value for minimizing project schedules.

The following sections provide more details on the processor, model, and OpenMP variables.

Processors

AWS positions AWS Graviton® processor–based instances aggressively from a price perspective, compared to Intel® Xeon® Scalable processor–based instances. Our goal was to compare the value of the instances using the Intel Xeon and AWS Graviton processor families when running an HPC workload. We compared the results of simulations run on instances with 3rd Gen Intel Xeon Scalable processors with simulations run on instances with AWS Graviton2 processors. We also compared the results of simulations run on instances with 4th Gen Intel Xeon Scalable processors with simulations run on instances with AWS Graviton3E processors.

OpenRadioss Models

We ran two OpenRadioss tests: the Chrysler® Neon 1M element HPC benchmark and the Ford® Taurus 10M element HPC benchmark. The Neon 1M model has one million finite elements and is designed to test an HPC cluster with a low number of cluster nodes or a single compute server. The Taurus 10M model has 10 million finite elements and is designed for scalability testing of HPC clusters with large numbers of nodes. We compiled OpenRadioss with Intel® oneAPI and the Intel® Fortran Compiler (ifx).

OpenMP® Settings

We wanted to do an "apples to apples" comparison of AWS and Intel® processors. To this end, we configured a 64-vCPU C7i Intel instance with the OpenMP thread set to one thread instead of two threads. This allowed us to do a more direct comparison between the 64-vCPU C6i and 64-vCPU C7i AWS with Intel instances and the 64-physical core AWS with Graviton instances, C6g.16xlarge and C7gn.16xlarge. With this setting change, the 64-vCPU C6i and 64-vCPU C7i AWS with Intel instances were configured to provide 64 physical cores.

Test Procedure

Prowess Consulting used OpenRadioss to benchmark the HPC performance of eight compute-optimized Amazon® Elastic Compute Cloud (Amazon EC2®) instance types:

- Amazon EC2 C6i instances powered by 3rd Gen Intel Xeon Scalable processors:
  - **C6i.16xlarge**, **C6i.24xlarge**, and **C6i.32xlarge**
- Amazon EC2 C7i instances powered by 4th Gen Intel Xeon Scalable processors:
  - **C7i.16xlarge**, **C7i.24xlarge**, and **C7i.48xlarge**
- Amazon EC2 C6g instance powered by Arm®-based AWS Graviton2 processors:
  - **C6g.16xlarge**
- Amazon EC2 C7gn instance powered by Arm-based AWS Graviton3E processor:
  - **C7gn.16xlarge**

For our testing, we used the public documentation found on GitHub to run the OpenRadioss and HPC benchmark models. All instances used in this testing were located in the US East (N. Virginia) US-east-1 AWS region.

We ran the Taurus 10M model on the instance types shown in Table 1. Similarly, we ran the Neon 1M model on the instance types shown in Table 2.

Table 1. Instances for the Ford® Taurus 10M model testing

	64-vCPU C6i	64-vCPU C6g	96-vCPU C6i	64-vCPU C7gn	64-vCPU C7i	96-vCPU C7i	64-vCPU C6i	64-vCPU C7i
vCPU	64	64	96	64	64	96	64	64
Thread(s) per CPU	2	2	2	2	2	2	1	1
Instance size	C6i.16xlarge	C6g.16xlarge	C6i.24xlarge	C7gn.16xlarge	C7i.16xlarge	C7i.24xlarge	C6i.32xlarge	C7i.48xlarge
Processor	Intel® Xeon® Platinum 8375C CPU	AWS® Graviton2 ARM® v8 Neoverse-N1 CPU	Intel Xeon Platinum 8375C CPU	AWS Graviton3E CPU	Intel Xeon Platinum 8488C CPU	Intel Xeon Platinum 8488C CPU	Intel Xeon Platinum 8375C CPU	Intel Xeon Platinum 8488C CPU
Note	3rd Gen Intel Xeon Scalable processor	AWS Graviton2	3rd Gen Intel Xeon Scalable processor	AWS Graviton3	4th Gen Intel Xeon Scalable processor	4th Gen Intel Xeon Scalable processor	3rd Gen Intel Xeon Scalable processor	4th Gen Intel Xeon Scalable processor

Table 2. Instances for the Chrysler® Neon 1M model testing

	64-vCPU C7gn	64-vCPU C7i	96-vCPU C7i	64-vCPU C6i	64-vCPU C7i
vCPU	64	64	96	64	64
Thread(s) per CPU	2	2	2	1	1

Instance Size	C7gn.16xlarge	C7i.16xlarge	C7i.24xlarge	C6i.32xlarge	C7i.48xlarge
Processor	AWS® Graviton3E CPU	Intel® Xeon® Platinum 8488C CPU	Intel Xeon Platinum 8488C CPU	Intel Xeon Platinum 8375C CPU	Intel Xeon Platinum 8488C CPU
Note	AWS Graviton3	4th Gen Intel Xeon Scalable processor	4th Gen Intel Xeon Scalable processor	3rd Gen Intel Xeon Scalable processor	4th Gen Intel Xeon Scalable processor

Setup Process for AWS Instances with Intel® Processors

We deployed Amazon EC2 C6i instances using the following steps:

1. Log in to the AWS dashboard.
2. Click **EC2**.
3. From the drop-down menu, select **Launch instance**.
4. In the **Name** field, enter a chosen name.
5. From the **Quick Start** section, select **Ubuntu 22.04**.
6. In the **AMI** field, select **Ubuntu 22.04**.
7. From the **Instance Type** field, select **C6i.16xlarge**, **C6i.24xlarge**, **C7i.16xlarge**, **C7i.24xlarge**, **C6i.32xlarge**, and **C7i.48xlarge**, as appropriate.
8. In the **Key Pair (login)** section, click **Create a new key pair**.
9. In the resulting window, enter a name for the key pair.
10. Leave the default selection of **RSA**.
11. Leave the default selection of **.pem**.
12. Click **Create key pair** and make note of the downloaded file, henceforth **$ssh_key**.
13. In the **Network Settings** section, leave the default selections.
14. In the **Configure Storage** section, specify the following parameters:
    - **Quantity: 1**
    - **Size: 100 GB**
    - **Type: GP2**
15. In the **Advanced Details** section, leave the default selections for all instances except for the C6i.32xlarge and C7i.48xlarge instance types. For those two instances, complete the following steps:
    a. Select the **CPU Options** check box.
    b. Set the number of CPUs to **64**.
    c. Set the number of threads per CPU to **1**.
16. Click **Create Instance**.
17. On the resulting page, click the instance ID in the success message.
18. Note the **public IPv4 Address** as **$instance_ip**.
19. Open a command-line interface (CLI) and create a Secure Shell (SSH) connection to the instance by running the following command:
```
ssh -i $ssh_key ubuntu@$instance_ip
```
20. To update the system and install the necessary tools, run the following command:
```
sudo apt update; sudo apt install -y build-essential gfortran cmake perl python3 python-is-python3
git-lfs unzip software-properties-common pkg-config nmon
```
21. To add the oneAPI repository information, run the following commands:
```
wget https://apt.repos.intel.com/intel-gpg-keys/GPG-PUB-KEY-INTEL-SW-PRODUCTS.PUB
sudo apt-key add GPG-PUB-KEY-INTEL-SW-PRODUCTS.PUB
sudo add-apt-repository "deb https://apt.repos.intel.com/oneapi all main"
```

22. Press **Enter** to continue.
23. To install the Intel® HPC Kit, run the following command:

```
sudo apt-get install -y intel-hpckit
```

24. To initialize the Intel variables, run the following command:

```
source /opt/intel/oneapi/setvars.sh
```

25. To set an unlimited core dump, run the following command:

```
Ulimit -c unlimited
```

26. To install git lfs, run the following command:

```
git lfs install
```

27. To clone the OpenRadioss repository, run the following command:

```
git clone https://github.com/OpenRadioss/OpenRadioss.git; cd OpenRadioss/starter
```

28. To build the starter, run the following command:

```
./build_script.sh -arch=linux64_intel
```

29. To switch to the engine directory, run the following command:

```
cd ../engine
```

30. To build the engine executable and the starter executable, run the following command. Replace ***$NumThreads*** with **1** for the C6i.32xlarge and C7i.48xlarge instances and **2** for the other Intel instance types:

```
./build_script.sh -arch=linux64_intel -mpi=impi -nt=$NumThreads
```

31. To set the necessary values automatically on system startup, run the following command:

```
tee >> ~/.bashrc << EOF
export OPENRADIOSS_PATH=/home/ec2-user/OpenRadioss
export RAD_CFG_PATH=\$OPENRADIOSS_PATH/hm_cfg_files
export RAD_H3D_PATH=\$OPENRADIOSS_PATH/extlib/h3d/lib/linux64
export OMP_STACKSIZE=400m
export LD_LIBRARY_PATH=\$OPENRADIOSS_PATH/extlib/hm_reader/linux64/:\$LD_LIBRARY_PATH
export OMP_NUM_THREADS=$NumThreads
source /opt/intel/oneapi/setvars.sh
EOF
```

32. To import the variables set in step 31, run the following command:

```
Source ~/.bashrc; cd ~/
```

33. To install the Neon 1M model, run the following command:

```
wget https://openradioss.atlassian.net/wiki/download/attachments/47546369/Neon1m11_2017.zip
unzip Neon1m11_2017.zip
```

34. To install the Taurus 10M model and decrease the default time increment, run the following commands:

```
cd ~/
wget https://openradioss.atlassian.net/wiki/download/attachments/47546369/Taurus10M.zip
unzip Taurus10M.zip
sed -i 's/0.0020/0.01001/g' T10M/TAURUS_A05_FFB50_0001.rad
```

Test Process for AWS Instances with Intel Processors: OpenRadioss Neon 1M and Taurus 10M Models

1. Start the Neon 1M model test by running the following command. Replace ***$P*** with **32** for the 16xlarge instance, replace it with **48** for the 24xlarge instance, and replace it with **64** for the 32xlarge and 48xlarge instance types.

```
~/OpenRadioss/exec/starter_linux64_intel -i ~/Neon1m11_2017/NEON1M11_0000.rad -np $P; mpiexec -n $P
~/OpenRadioss/exec/engine_linux64_intel_impi -i ~/Neon1m11_2017/NEON1M11_0001.rad
```

2. Start the Taurus 10M model test by running the following command. Replace ***$P*** with **32** for the 16xlarge instance, replace it with **48** for the 24xlarge instance, and replace it with **64** for the 32xlarge and 48xlarge instance types.

```
~/OpenRadioss/exec/starter_linux64_intel -i ~/T10M/TAURUS_A05_FFB50_0000.rad -np $P;  source /opt/
intel/oneapi/setvars.sh;   mpiexec -n  $P  ~/OpenRadioss/exec/engine_linux64_intel_impi -i ~/T10M/
TAURUS_A05_FFB50_0001.rad
```

Setup Process for AWS Instances with AWS® Graviton2 and Graviton3

1. Log in to the AWS dashboard.
2. Click **EC2**.
3. Click **Launch Instance**.
4. From the drop-down menu, select **Launch instance**.
5. In the **Name** field, enter a chosen name.
6. From the **Quick Start** section, select **Red Hat**.
7. In the **AMI** field, ensure the **Ubuntu 22.04** type is selected.
8. From the **Architecture** drop-down menu, select **64-bit (Arm)**.
9. From the **Instance Type** field, select **C6g.16xlarge** or **C7gn.16xlarge**, as appropriate.
10. In the **Key Pair (login)** section, click the **Create a new key pair** link.
11. In the resulting window, enter a name for the key pair.
12. Leave the default selection of **RSA**.
13. Leave the default selection of **.pem**.
14. Click **Create key pair**, and then make note of the downloaded file, henceforth **$ssh_key**.
15. In the **Network Settings** section, leave the default selections.
16. In the **Configure Storage** section, specify **1 100 GB GP2** volume.
17. In the **Advanced Details** section, leave the default selections.
18. Click **Launch Instance**.
19. On the resulting page, click the instance ID in the success message.
20. Note the **public IPv4 Address**, henceforth **$instance_ip**.
21. Open a CLI and create an SSH connection to the instance by running the following command:
```
ssh -i $ssh_key ubuntu@$instance_ip
```
22. To install necessary dependency packages, run the following command:
```
sudo apt update; sudo apt install -y build-essential gfortran cmake perl python3 python-is-python3
git-lfs unzip software-properties-common
```
23. If prompted, select default services for restarting.
24. To download Open MPI, run the following command:
```
wget https://download.open-mpi.org/release/open-mpi/v4.1/openmpi-4.1.2.tar.gz
```
25. To extract Open MPI, run the following command:
```
tar -xvzf openmpi-4.1.2.tar.gz; cd openmpi-4.1.2
```
26. To build Open MPI, run the following command:
```
./configure --prefix=/opt/openmpi; make; sudo make install
```
27. To set the configuration for Open MPI, run the following command:
```
mkdir -p ~/.openmpi ;  echo "btl_vader_single_copy_mechanism=none" >> ~/.openmpi/mca-params.conf
```
28. To install Git large file storage (LFS), run the following command:
```
cd ~/; git lfs install
```
29. To download the OpenRadioss repository, run the following command:
```
git clone https://github.com/OpenRadioss/OpenRadioss.git
```
30. To set the variables on system startup, run the following command:
```
tee >> ~/.bashrc << EOF
export OPENRADIOSS_PATH=/home/ec2-user/OpenRadioss
export RAD_CFG_PATH=\$OPENRADIOSS_PATH/hm_cfg_files
export RAD_H3D_PATH=\$OPENRADIOSS_PATH/extlib/h3d/lib/linuxa64
export OMP_STACKSIZE=400m
export LD_LIBRARY_PATH=\$OPENRADIOSS_PATH/extlib/hm_reader/linuxa64/:\$LD_LIBRARY_PATH
export OMP_NUM_THREADS=1
export PATH="$PATH:/opt/openmpi/bin"
EOF
```

31. To import the previously set variables, run the following command:

```
source ~/.bashr; cd ~/
```

32. To get the Neon 1M model, run the following command:

```
wget https://openradioss.atlassian.net/wiki/download/attachments/47546369/Neon1m11_2017.zip
unzip Neon1m11_2017.zip
```

33. To bring in the Taurus 10M model, run the following command:

```
cd ~/
wget https://openradioss.atlassian.net/wiki/download/attachments/47546369/Taurus10M.zip
unzip Taurus10M.zip
sed -i 's/0.0020/0.01001/g' T10M/TAURUS_A05_FFB50_0001.rad
```

34. To modify the Apptainer, run the following command:

```
vim ~/OpenRadioss/Apptainer/openradioss_arm.def
```

35. Add **python** to the end of line 9.
36. Add the following at line 26:

```
export CXX_comp="armclang++"
export CPP_comp="armclang++"
export C_comp="armclang"
export Fortran_comp="armflang"
```

37. To save and quit, enter **esq :wq**.
38. To build the Apptainer, run the following command:

```
cd ~/OpenRadioss/Apptainer/
sudo apptainer build openradioss.sif openradioss_arm.def && sudo cp openradioss.sif /usr/local/bin
```

Test Process for AWS Instances with Graviton2 and Graviton3: OpenRadioss Neon 1M and Taurus 10M Models

1. To trigger a test run of the Neon 1M model, run the following command:

```
openradioss.sif starter_linuxa64 -i ~/Neon1m11_2017/NEON1M11_0000.rad -np 64;
/opt/openmpi/bin/mpiexec --map-by socket:PE=1 --bind-to core -n 64 openradioss.sif engine_linuxa64_
ompi -i ~/Neon1m11_2017/NEON1M11_0001.rad
```

2. To trigger a test run of the Taurus 10M model, run the following command:

```
openradioss.sif starter_linuxa64 -i ~/T10M/TAURUS_A05_FFB50_0000.rad -np 64;
/opt/openmpi/bin/mpiexec --map-by socket:PE=1 --bind-to core -n 64 openradioss.sif engine_linuxa64_
ompi -i ~/T10M/TAURUS_A05_FFB50_0001.radp 64;
```

PROWESS

The analysis in this document was done by Prowess Consulting and commissioned by Intel.
Results have been simulated and are provided for informational purposes only.
Any difference in system hardware or software design or configuration may affect actual performance.

1223/220205