



AHUG WORKSHOP @ ISC'23

AWS Graviton 3E

Updates from the field

Brendan Bouffler ('boof')

HPC Engineering @ AWS

Amazon EC2 | The compute platform for every workload

Workload types



Machine Learning



High-Performance Computing



Media Rendering



Containers



Web-based Apps



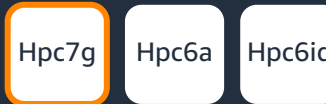
Batch Processing



Big Data

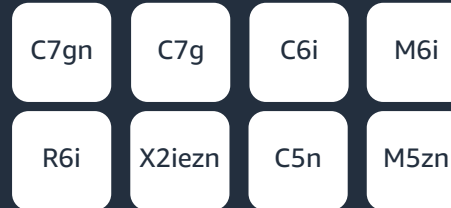
Instance types for HPC workloads

HPC Optimized



Coming soon

Compute, Memory, and Networking



Accelerators

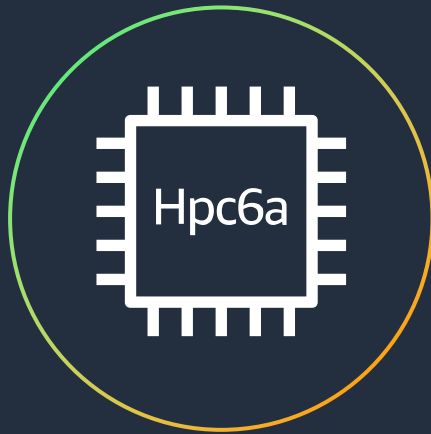


Scale tightly and loosely-coupled HPC applications

- Choice of processor (e.g., Graviton, Intel, AMD)
- Scale tightly-coupled HPC and ML workloads
- Up to 400 Gbps network bandwidth
- < 15 micro-seconds network latencies
- Accelerators use hardware to perform functions more efficiently than is possible in software running in CPUs

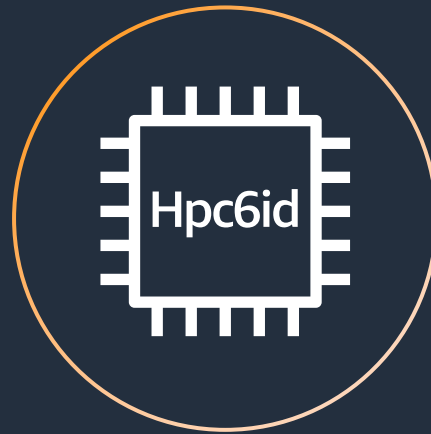
HPC instance choice for diverse workloads

X86 - AMD



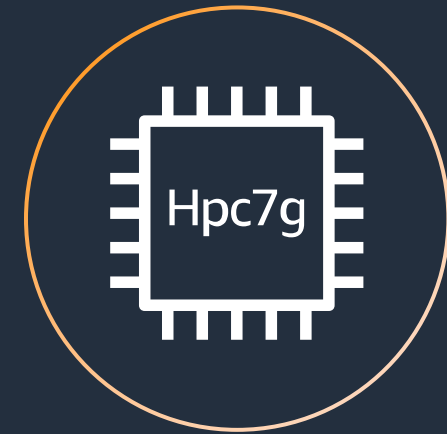
- AMD Milan
- 96 physical cores / instance
- Up to 3.6 GHz
- 384 GB RAM
- 100 Gbps EFA

X86 - Intel



- Intel Ice Lake
- 64 physical cores / instance
- Up to 3.5 GHz
- 1 TB RAM
- 15.2 TB NVMe
- 200 Gbps EFA

ARM - AWS (coming soon)

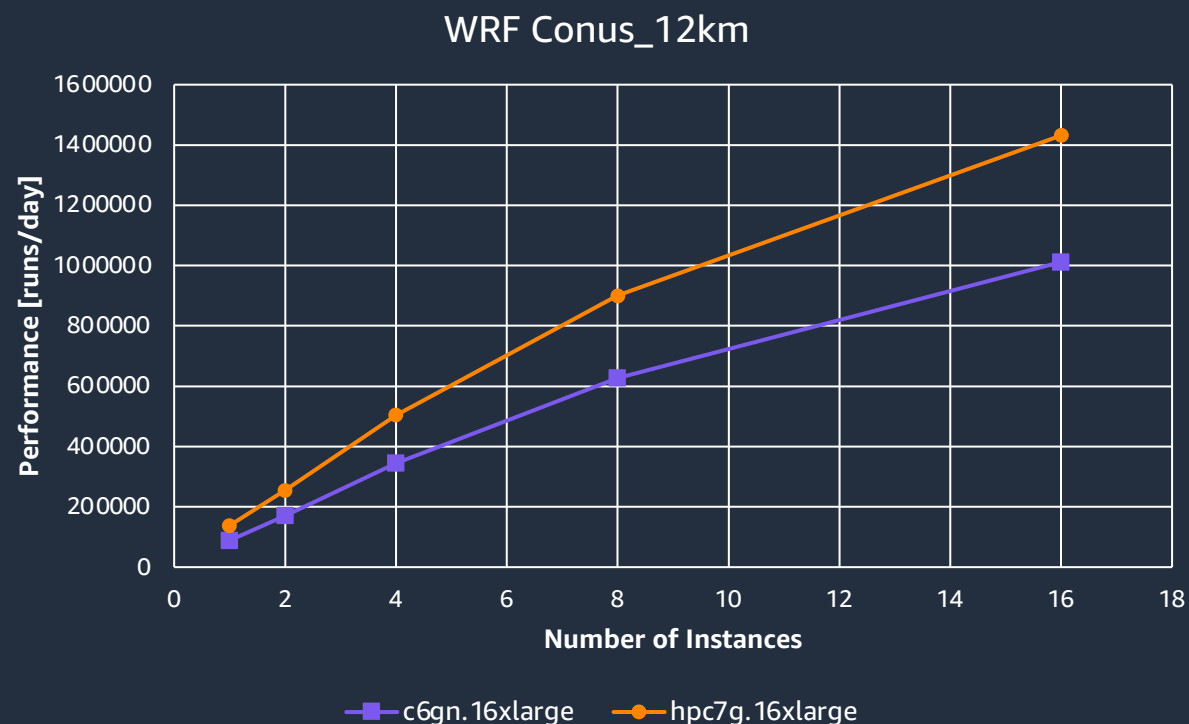


- AWS Graviton3E
- 64 physical cores / instance
- Up to 2.6 GHz
- 128 GB RAM
- 200 Gbps EFA



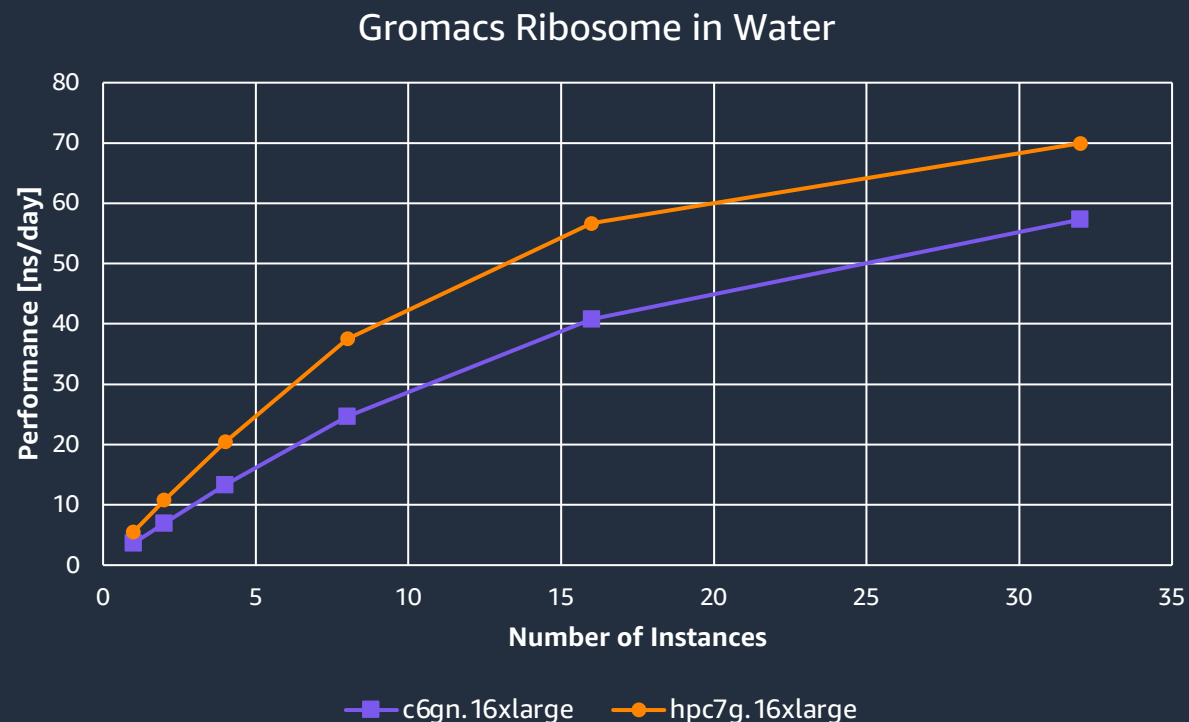
Example Hpc7g Performance: WRF

- Continental United States
12x12km (127,500 grid points)
- **56%** speed-up on single instance
- **41%** speed-up at 16 instances



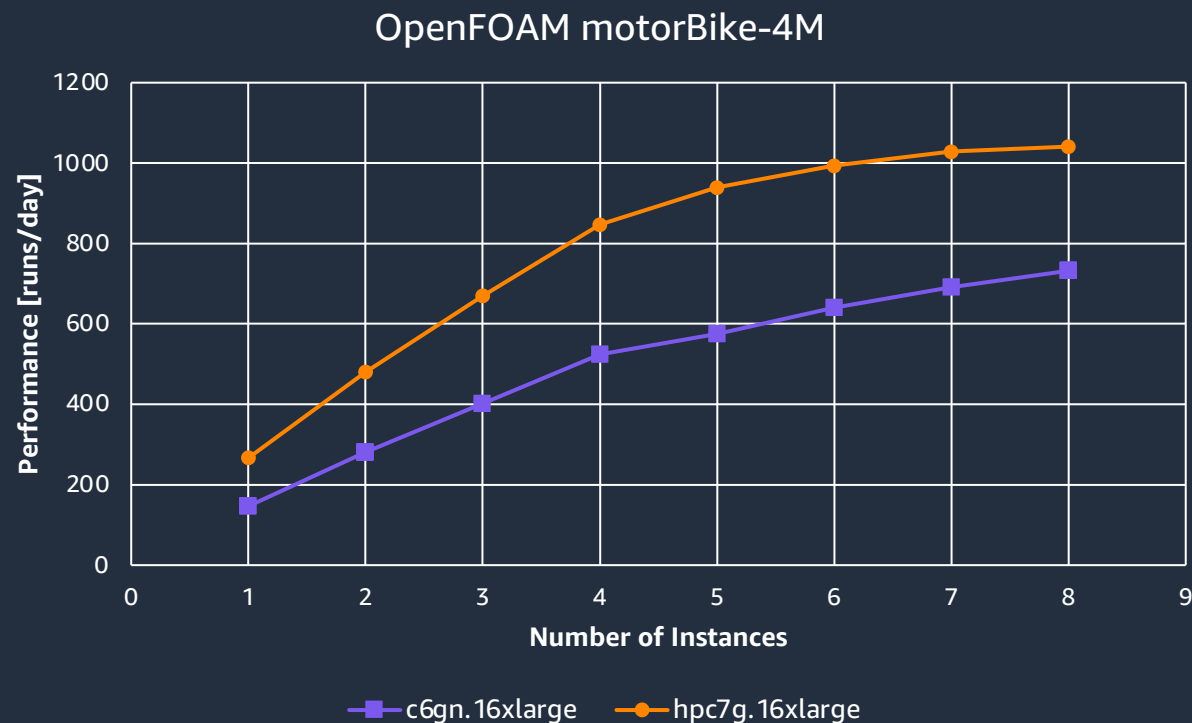
Example Hpc7g Performance: Gromacs

- 2M atoms Ribosome in Water
- 53% speed-up on single instance
- 22% speed-up at 32 instances



Example Hpc7g Performance: OpenFOAM

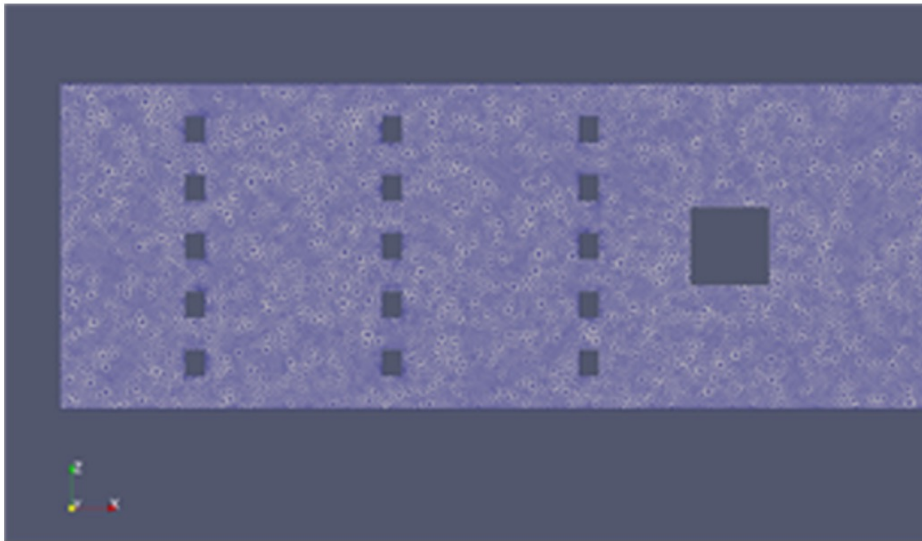
- Small motorBike test case (4M cells) to show scaling bottlenecks
- 82% speed up on single instance
- 42% speed-up on 8 instances



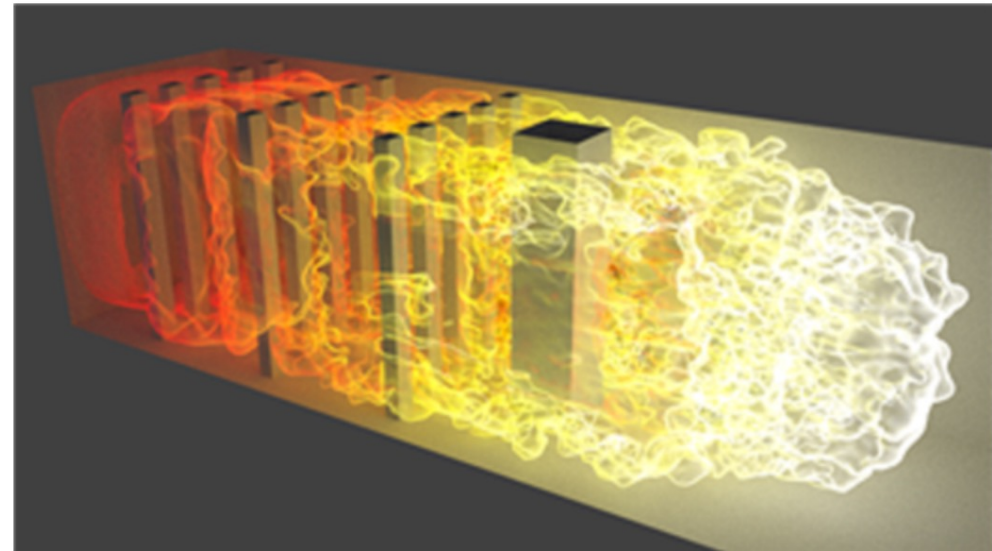


CERFACS: Hpc7g use case

- It is a state-of-the-art simulation of an explosion. Modelling and algorithms match industrial applications used by CERFACS partners (AIRBUS, SAFRAN, Total).
- From the previous collaboration benchmark we updated the AVBP version to 7.9. The grid has 60M Tetra.

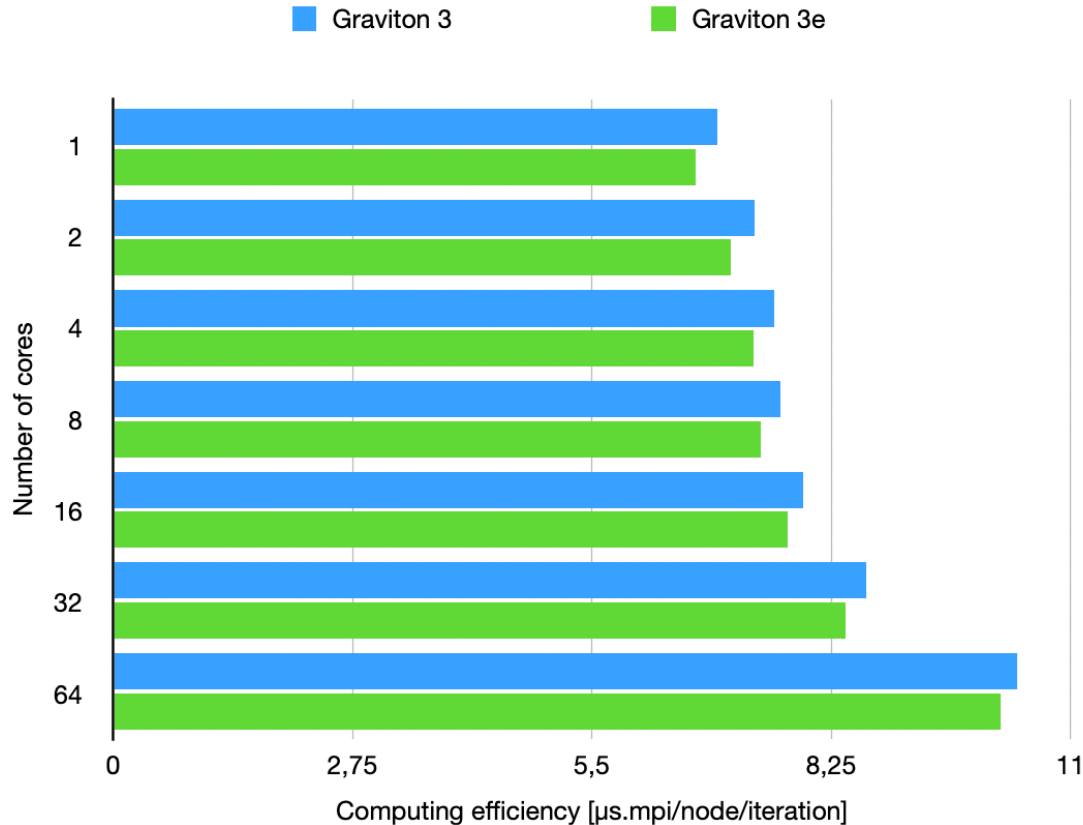


Top view of a mid-cut of the grid.



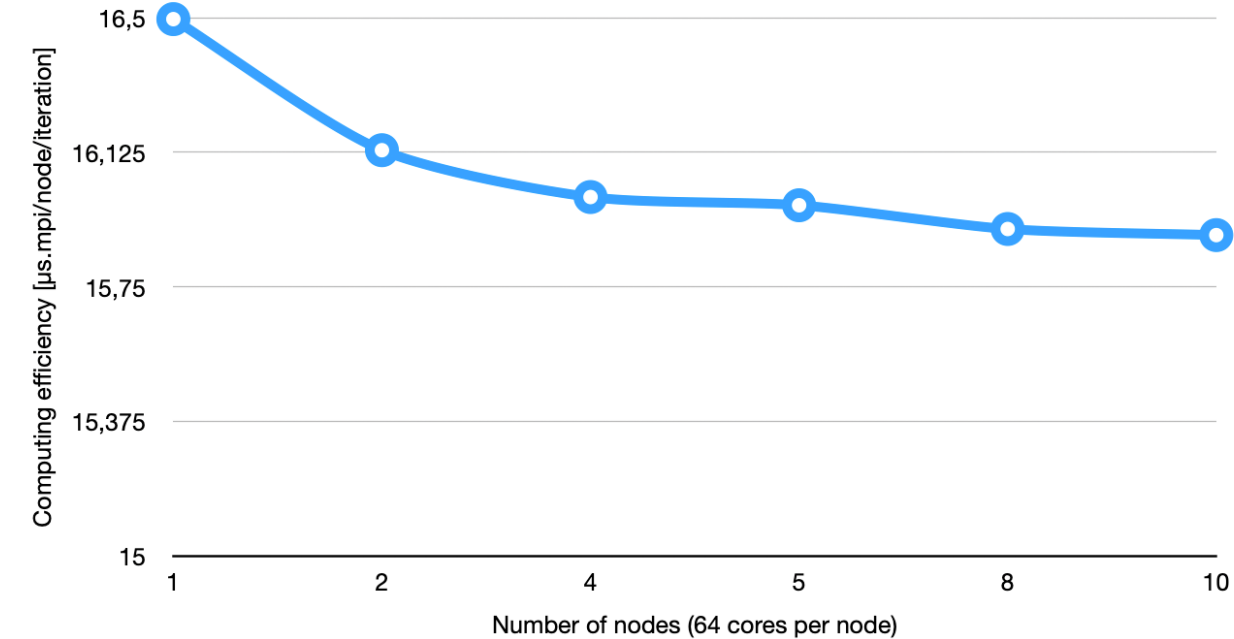
Isosurface of temperature rendering (Gullaud et al)

Single node performance



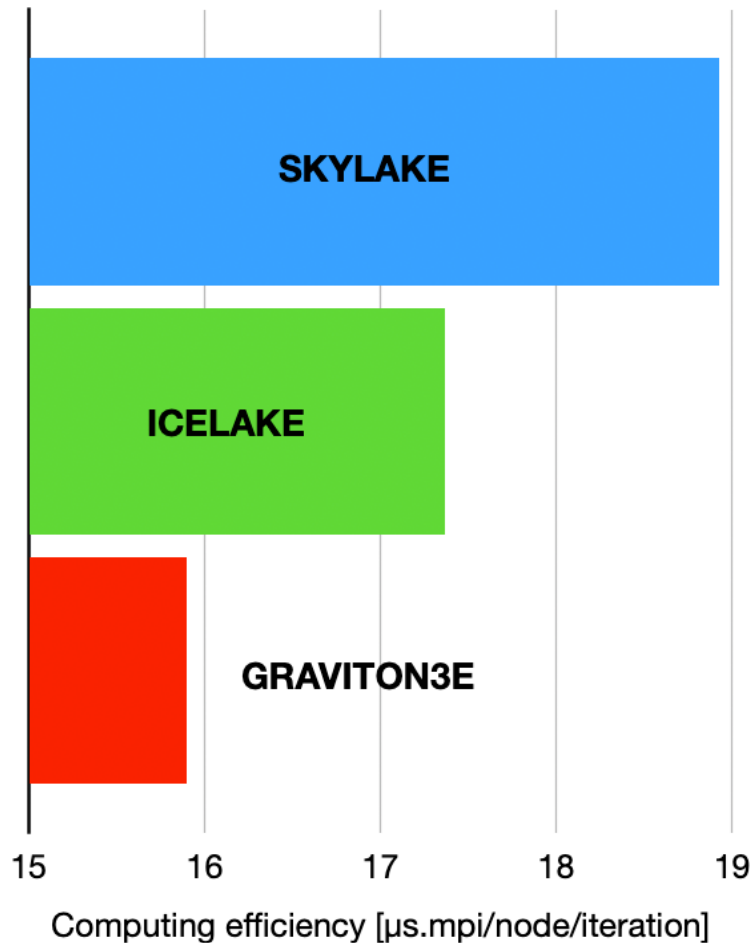
- This graph shows the comparative performance of graviton3 and 3e.
- Performance is measure using 10 iteration without I/O
- Overall graviton3e is 4% more efficient than graviton3
- Strong scaling from 1 core to full node (64 cores) is excellent. For further tests full nodes will be used.
- Graviton3e is 186% faster than graviton2

Multi node Performance



- This graph shows the strong scaling of graviton3e from 1 to 10 nodes. Performance measured in production mode with 1000 iterations.
- Perfect scaling would be a straight line.
- Here we can see that performance is near perfect, with a 3.7% loss from 1 to 10 nodes.

Multi node Performance



- This graph shows the efficiency of graviton3 in a production mode measured using 6000 iterations.
- 15 skylake nodes (540 cores) versus 10 Icelake nodes (760 cores) versus 10 graviton3 (640 cores) are compared here in terms of computing efficiency per mpi per node per iteration.
- Graviton3 is 16% and 8.4% faster than comparable Skylake and Icelake nodes.

Simcenter STAR-CCM+



Vorticity: Magnitude (1/s)

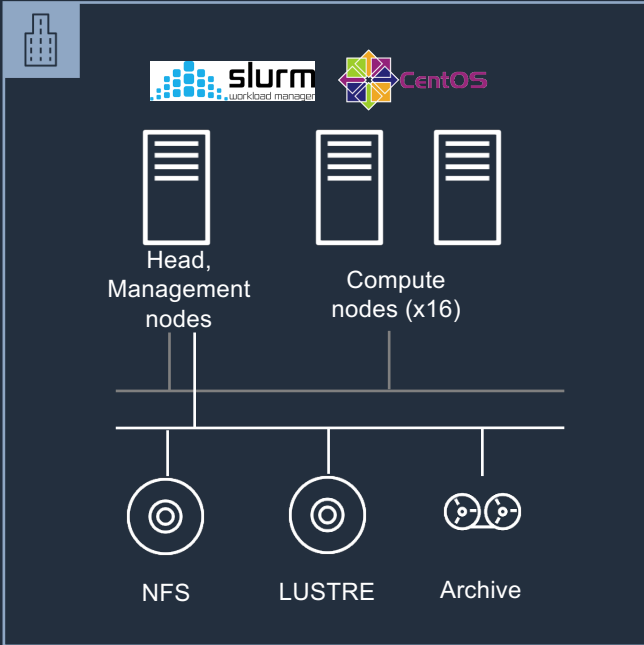


Wall Shear Stress: Magnitude (Pa)

Tooling



ON-PREMISES



PARALLELCLUSTER RECIPE (YAML)

```

Image:
  Os: centos7

HeadNode:
  InstanceType: c5.4xlarge
  Networking:
    SubnetId: subnet-b46032ec
  Ssh: KeyName: My_PC3_KeyPair
  AllowedIps: 0.0.0.0/0

Scheduling:
  Scheduler: slurm
SlurmSettings:
  ScaledownIdleTime: 10
  Dns:
    DisableManagedDns: true
SlurmQueues:
  - Name: q1_ondemand
    ComputeSettings:
      LocalStorage:
        RootVolume:
          Size: 100
      CapacityType: ONDEMAND
      ComputeResources:
        - Name: compute-resource-1
          InstanceType: c5.n18xlarge
          Efa:
            Enabled: true
            MinCount: 0
            MaxCount: 64
      Networking:
        SubnetIds:
          - subnet-a12321bc
        PlacementGroup:
          Enabled: true

SharedStorage:
  - MountDir: /shared
    Name: myebs
    StorageType: Ebs
    EbsSettings:
      VolumeType: gp3
      Size: 100
  - MountDir: /lustre
    Name: myfsx
    StorageType: FsxLustre
    FsxLustreSettings:
      StorageCapacity: 1200
      DeploymentType: SCRATCH_2
      ImportPath: s3://myhpcbucket
  
```

OS

Head node

Scheduler Settings

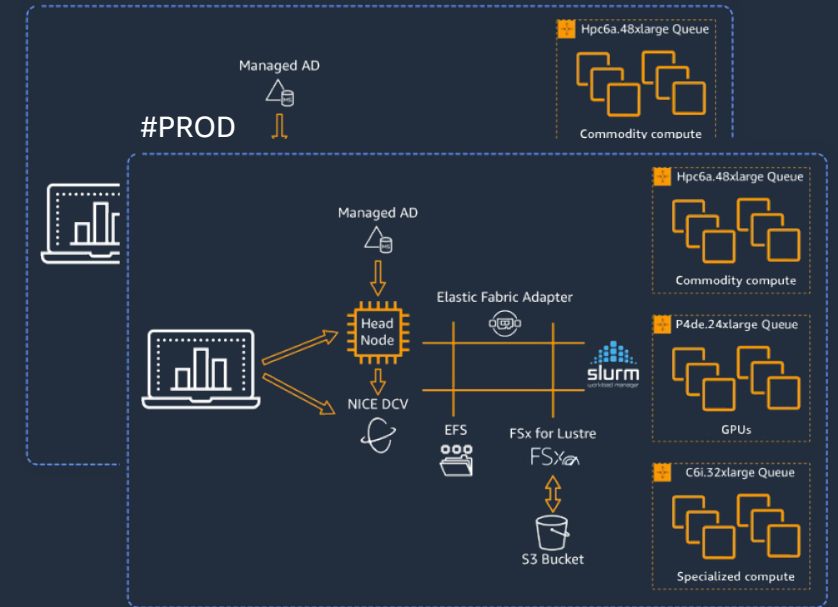
Compute nodes

Network settings

NFS Storage

Lustre storage

#DEV



Recipe becomes the manageable asset.

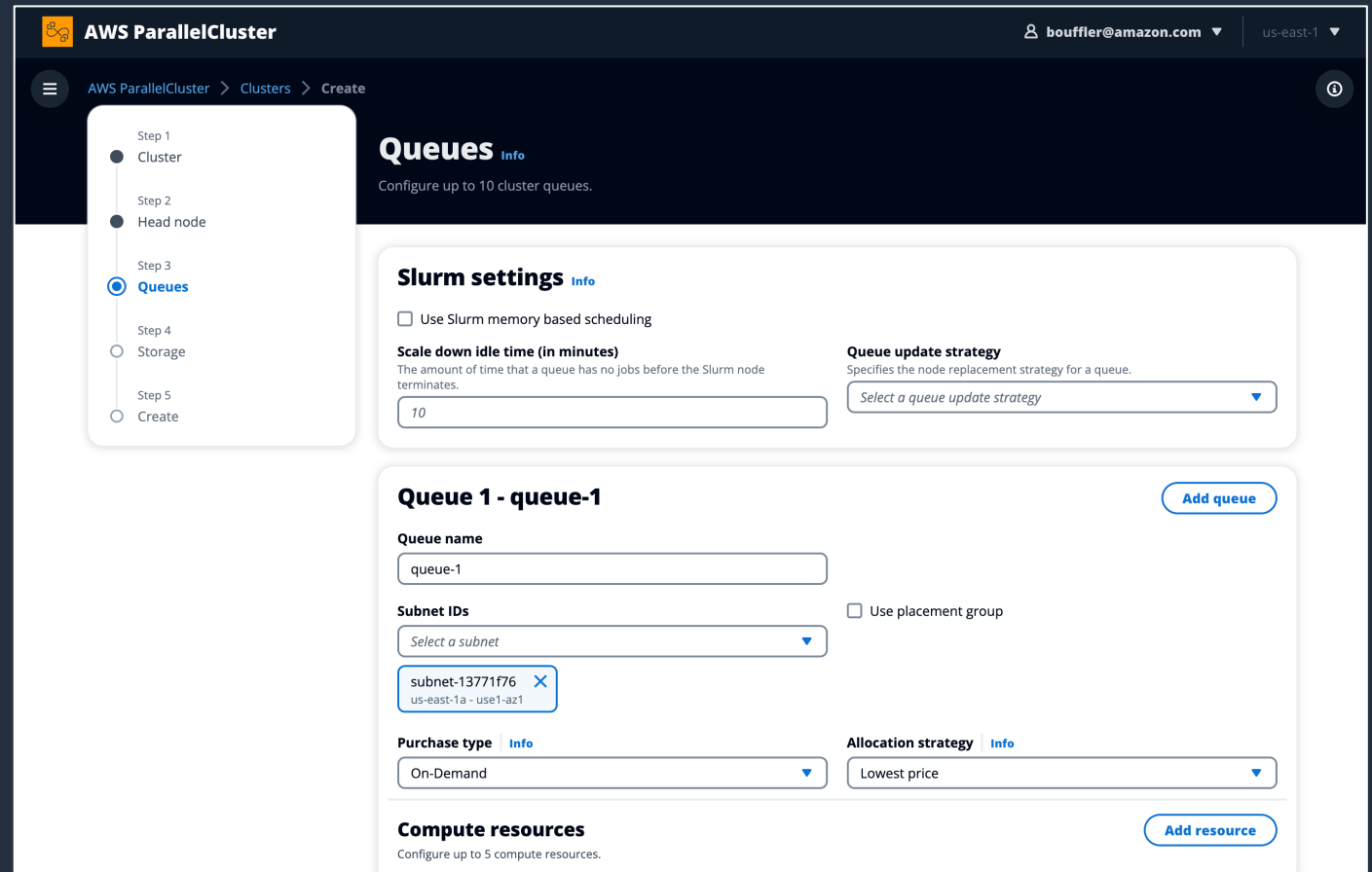
AWS ParallelCluster UI

Guided interface for designing and deploying ParallelCluster

Design and deploy multiple clusters through a guided process

Manage and debug all your clusters (multi-region) through a single dashboard interface.

Connect to clusters securely through remote desktop or command line interface



Spack Configs for AWS ParallelCluster

- Validated best-practices from AWS HPC Performance Engineering
- Fixes and general optimizations for any application
- Tuned configurations for common HPC workloads
- Supports Intel, AMD, Graviton
- Easy to use

Application	Speed-up
OpenFOAM	N/A
WRF	11.9
MPAS	N/A
GROMACS	1.16
Quantum Espresso	1.53
Devito	N/A
LAMMPS	16.6

Speedup relative to original Spack configs. N/A indicates no original Spack build was available.



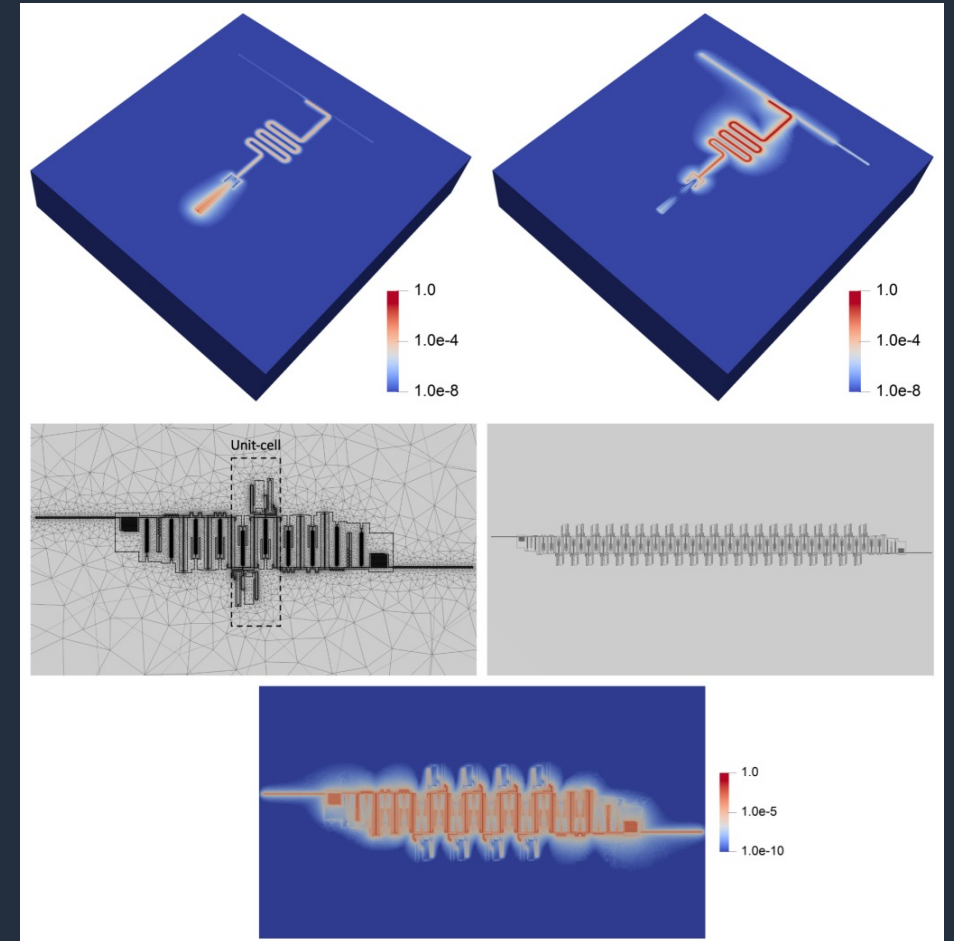
Using Spack Configs for AWS ParallelCluster

The screenshot shows the AWS ParallelCluster console interface. On the left, a vertical navigation pane lists six steps: Step 1 Source, Step 2 Cluster properties, Step 3 Head node (selected), Step 4 Shared storage, Step 5 Queues, and Step 6 Create. The main content area is titled 'Head node' with the subtitle 'Configure the head node.' Under the 'Advanced options' section, there are three script configuration options: 'Run script on node start' (script: `/home/ec2-user/start.sh`), 'Run script on node configured' (script: `https://raw.githubusercontent.com/spack/spack-configs/main/AWS/parall`), and 'Run script on node updated' (script: `/home/ec2-user/start.sh`). The 'Run script on node configured' section is highlighted with a blue border. Below these options is a section for 'IAM Policies'.

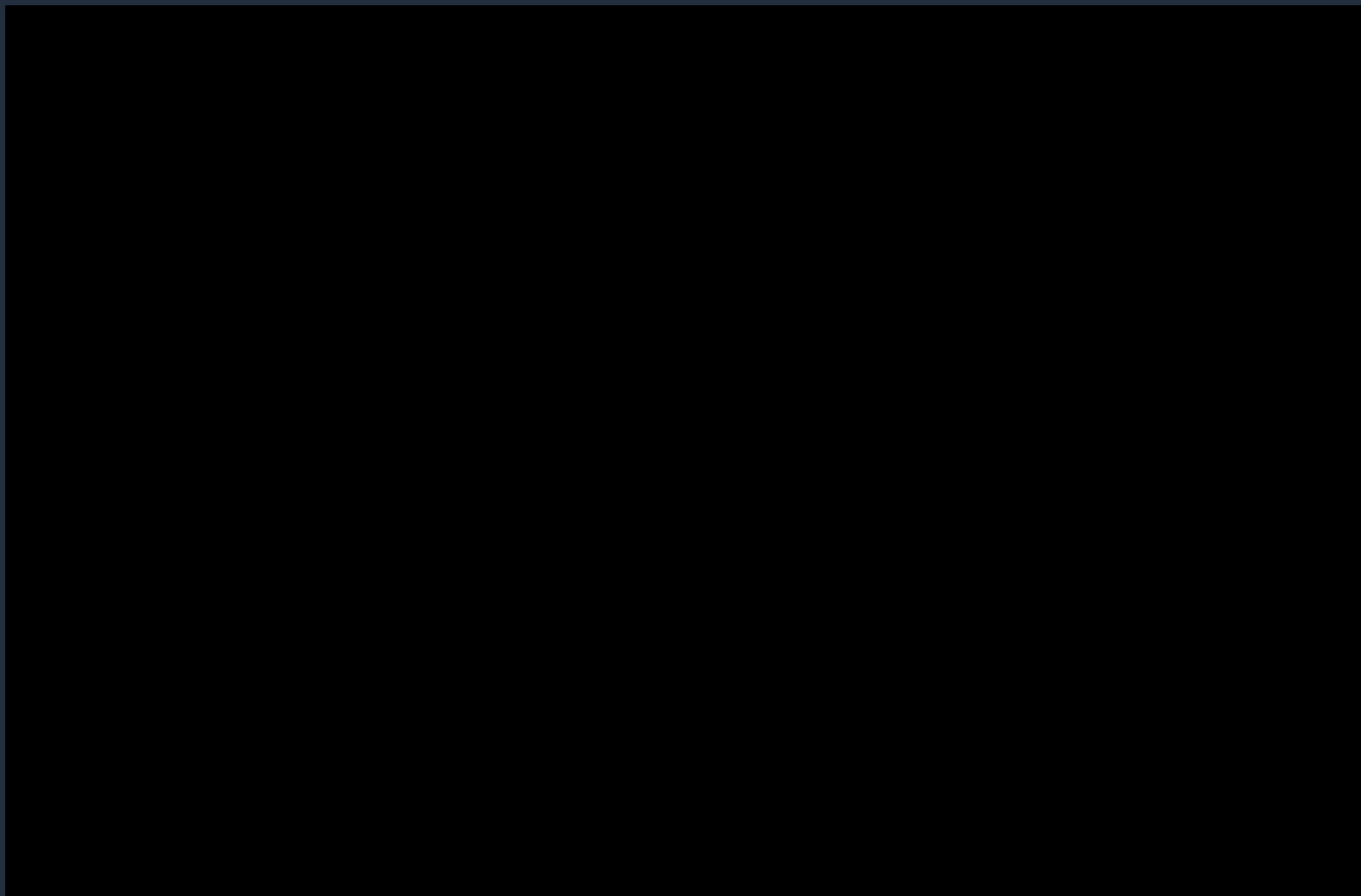
- Configure cluster to run script on Head Node during boot-up
- Script runs asynchronously
- Parameterizable – specify what packages to install

Palace: 3D Finite Element Solver for Computational Electromagnetics

- Enable large-scale 3D simulations of complex electromagnetics models & enable the design of quantum computing hardware
- Optimized for EFA and EC2 HPC instances
- Leverages [MFEM](#) from LLNL
- Deep dependency stack. Great for Spack.



From zero to PALACE in 4 minutes...



- Start with naive c7gn head node
- Re-use postinstall.sh script to install Spack
- Install GCC 12
- Optimize dependencies for neoverse V1
- Everything is in **build cache** except for **armpl-gcc**
- Python, Open MPI, & Petsc installed
- **84 packages in 4 minutes. Plus architecture optimizations!**

Outcomes and next steps

Installation *sans* build cache is ~1 hour on a full G7 instance

Added ARMPL support for Palace (should boost performance)

Implemented using Spack CICD pipelines

PALACE, OpenFOAM, GROMACS fully available in build cache


- OPENFOAM install: 16 seconds
- GROMACS install: 1 second

Lingering issues with Icelake, Skylake, AMD fixed soon



MPIs and NCCL


- EFA is presented as a **libfabric provider**
- **Known MPIs:**
 - Intel MPI supports EFA
 - MVAPICH supports EFA
 - AWS provides support for Open MPI on EFA
- **NVIDIA's Collective Communications Library (NCCL)** supports EFA.



Intel® MPI Library
Deliver flexible, efficient, and scalable cluster messaging.



Open MPI



MVAPICH



NVIDIA/nccl
Optimized primitives for collective multi-GPU communication



NVIDIA.

What else?

Dear Boof,

We look after an open-source thing called **foo** which is used by the **bar** community to resolve **fubars** in labs all around the world.

We have this slightly crazy idea that if we were able to build and maintain an Arm64 variant, we could resolve 2x more fubars than most labs can currently afford.

Sincerely, AHUG people

Dear Boof,

We look after a tool that nearly every scientific developer could use to **eliminate the daily crap** they need to do in order to get a science code developed quickly on Arm64.

Can we talk?

Sincerely

Dear Boof,

I have a pain in **all the diodes** down my left-hand side.

Sincerely, Marvin



Thank you!

Boof

Twitter: @boofla

Email: bouffler@amazon.com

The screenshot shows the DAY1HPC website, which is AWS Engineering's HPC community site. The URL is https://day1hpc.com. The navigation menu includes Tech Notes, Learn, Products, and About. The main banner features the title "Elastic Fabric Adapter" with the subtitle "High-speed networking for MPI and NCCL codes. Runs at cloud scale." Below the banner are two article cards. The first card is titled "Introducing GPU health checks in AWS ParallelCluster 3.6" and includes the text: "AWS ParallelCluster 3.6.0 can now detect GPU failures in HPC and AI/ML tasks. Health checks run at the start of Slurm jobs and if they fail, the job is requeued on another instance. This can increase reliability and prevent wasted spend. ...". The second card is titled "Creating and tweaking Slurm queues for your clusters" and includes the text: "Compute queues are where all the action is on an HPC cluster. In the cloud, you get the chance to customize the queues to match the code you're going to run on them - which is actually pretty fancy when you think about it. Matt Vaughn ...". A "Read More" button is visible below the second article card. On the right side of the page, there is a "Categories" list with items: AWS Batch, AWS ParallelCluster, Elastic Fabric Adapter, NICE DCV, AI/ML, CAE/CFD, Financial Services, Climate/Environment/Weather, and Life Sciences. Below the categories is a "Latest Articles" section with a link to "Introducing GPU health checks in AWS ParallelCluster 3.6".