AWS re:Invent

CMP 408-R

Using Elastic Fabric Adapter (EFA) to scale HPC workloads on AWS

Chethan Rao

Senior Product Manager, HPC

Amazon Web Services

AWS re:Invent

Agenda

HPC on AWS and EFA

Running EFA using AWS ParallelCluster

EFA Scaling Performance

Related breakouts

CMP402-R: Setting up your first HPC cluster

CMP409-R: Selecting the right instance for your HPC workloads

CMP418-R: Using AWS ParallelCluster to simplify cluster management

HPC on AWS and EFA

AWS re:Invent

AWS Services to get started with HPC on AWS

Amazon CloudWatch

Data management & data transfer	Compute & networking	Storage	Automation & orchestration	Visualization
AWS DataSync	Amazon EC2 instances (CPU, GPU, FPGA)	Amazon EBS	AWS Batch	NICE DCV
AWS Snowball	Amazon EC2 Spot	Amazon FSx for Lustre	AWS ParallelCluster	Amazon AppStream 2.0
AWS Snowmobile	AWS Auto Scaling	Amazon EFS	NICE EnginFrame	
AWS DirectConnect	Placement groups	Amazon S3		
	Enhanced networking			
	Elastic Fabric Adapter			

Amazon IAM (Identity and Access Management)

AWS Budgets

Elastic Fabric Adapter (EFA)

Scale tightly-coupled HPC applications on AWS

P3dn | G4dn[1] | C5n | i3en | M5n/ M5dn | R5n/ R5dn | C5an/ C5adn[1]

NVIDIA V100 and T4 GPUs

Custom Intel® Xeon® Scalable processor

AMD EPYC "Rome" processor

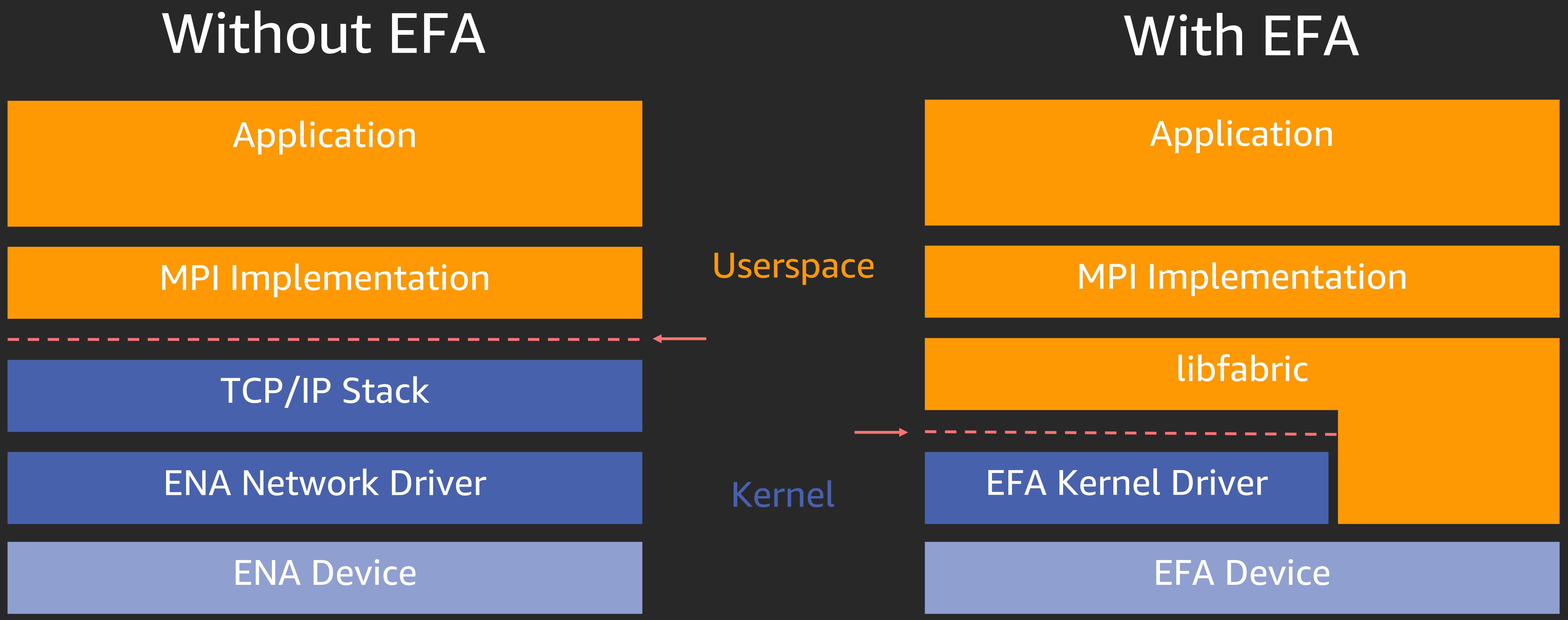
EFA

AWS' HPC/ML Network Interface

- Instance flexibility
- Infrastructure elasticity
- High data throughput
- Low latency message passing
- Faster application time-to-completion

[1] Coming soon!

HPC software stack in Amazon EC2



Scalable Reliable Datagram (SRD)

A reliable high-performance lower-latency network transport

Guaranteed delivery

- Does not consume any resources on EC2 instances

Network aware multipath routing

- Optimally utilizes all network paths (ECMP), no hot-spots

Orders of magnitude lower tail latency & jitter

- Fast recovery from network events

No ordering guarantees

- No head-of-line blocking

TCP vs Infiniband vs SRD

TCP	Infiniband	SRD
Stream	Messages	Messages
In-order	In-order	Out-of-order
Single path	Single (ish) path	ECMP spraying with load balancing
High limit on retransmit timeout (>50ms)	Static user-configured timeout (log scale)	Dynamically estimated timeout (µs resolution)
Loss-based congestion control	Semi-static rate limiting (limited set of supported rates)	Dynamic rate limiting
Inefficient software stack	Transport offload with scaling limitations	Scalable transport offload (same number of QPs regardless cluster size)

Running EFA using AWS ParallelCluster

AWS re:Invent

AWS ParallelCluster

ALINUX
CENTOS 6/7
UBUNTU 16/18

DCV
EFA
OPENMPI
INTELMPI
NCCL

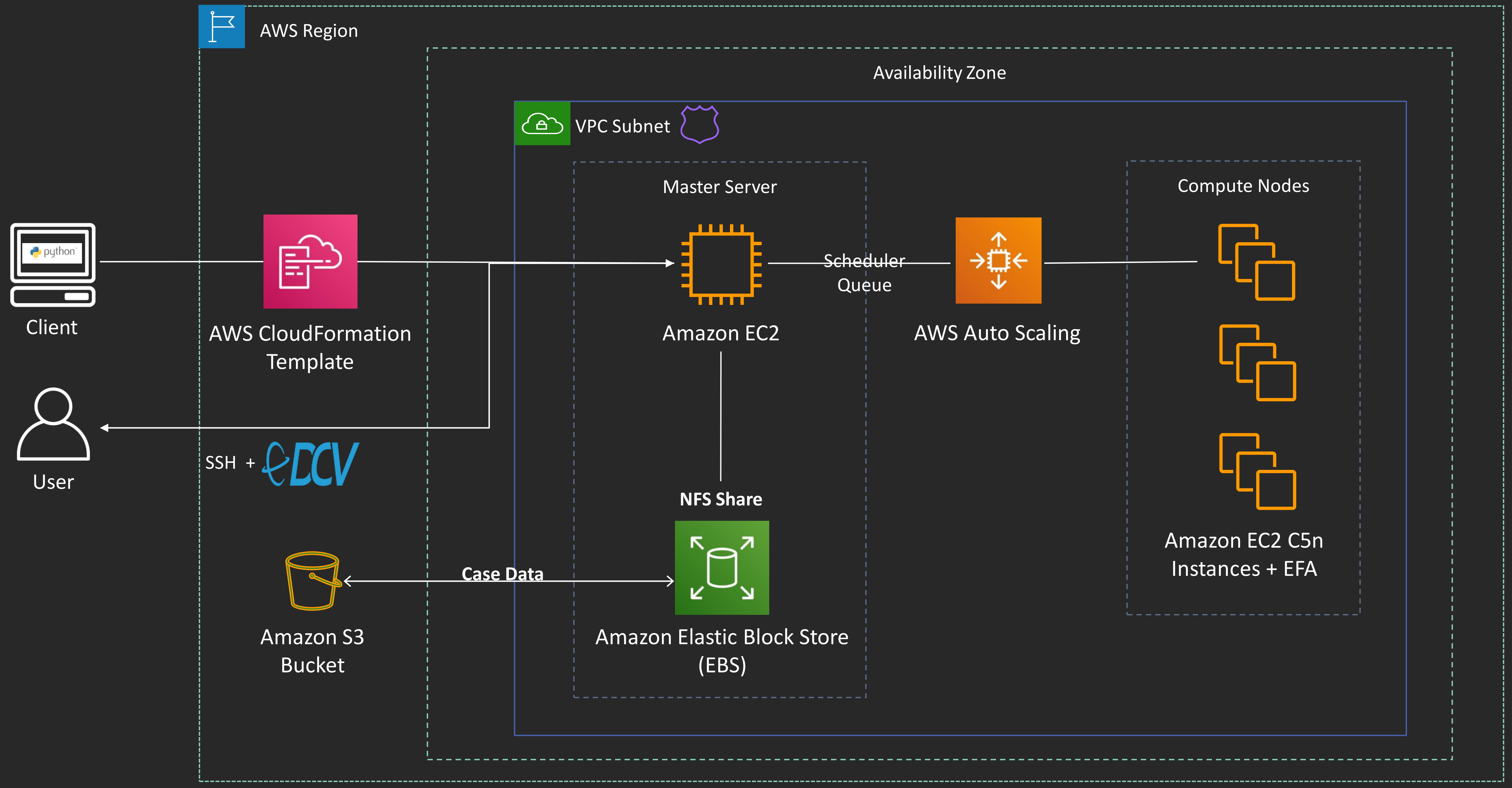
SLURM
SGE
TORQUE
AWS BATCH

FSX
EFS
S3
EBS
RAID

ON-DEMAND
SPOT

VPC & SUBNETS

AWS ParallelCluster Architecture



Getting Started

Launch AWS Console

Step 1: Go to
https://dashboard.eventengine.run/dashboard

Step 3: Use your hashcode to login to your dashboard

Step 4: Open AWS Console

Step 5: Launch Cloud9

Step 6: Open IDE

Launch AWS ParallelCluster w/o EFA

Step 7: Go to https://bit.ly/sc19-hpc-cloud, Section III AWS ParallelCluster

Step 8: Complete steps (a) through (e)

```
[cluster default]
key_name = lab-3-your-key
vpc_settings = public
ebs_settings = myebs
compute_instance_type = c5n.18xlarge
master_instance_type = c5n.2xlarge
cluster_type = ondemand
placement_group = DYNAMIC
placement = compute
initial_queue_size = 2
max_queue_size = 8 → 3
disable_hyperthreading = true
s3_read_write_resource = *
scheduler = slurm
```

Install OSU Benchmarks

Note: select MPI first before install

```
$ module load intelmpi

$ wget http://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-5.6.2.tar.gz
$ tar -xvf osu-micro-benchmarks-5.6.2.tar.gz
$ cd ./osu-micro-benchmarks-5.6.2/
$ ./configure --prefix=$PWD/install CC=mpicc CXX=mpicxx
$ make -j 4; make install
```

Submit your HPC job

```
$ cat > c5n_OSU.sbatch << EOF
#!/bin/bash
#SBATCH --job-name=osu-latency-job
#SBATCH --ntasks=2 --nodes=2
#SBATCH --output=osu_latency.out

srun --mpi=pmi2 ./osu-micro-benchmarks-5.6.2/install/libexec/osu-micro-benchmarks/mpi/pt2pt/osu_latency
EOF

$ sbatch c5n_OSU.sbatch
```

Launch an EFA Cluster

Terminate your current cluster

$ pcluster delete hpclab-yourname

Update my-cluster-config.conf to enable EFA

enable_efa=compute

Launch the EFA cluster

$ pcluster create hpclab-yourname -c my-cluster-config.conf

Login, install OSU benchmarks, re-run OSU latency script

$ pcluster ssh hpclab-yourname -i ~/.ssh/lab-3-key

$ module load intelmpi

```
[cluster default]
key_name = lab-3-your-key
vpc_settings = public
ebs_settings = myebs
compute_instance_type = c5n.18xlarge
master_instance_type = c5n.2xlarge
cluster_type = ondemand
placement_group = DYNAMIC
placement = compute
initial_queue_size = 2
max_queue_size = 8 → 3
disable_hyperthreading = true
s3_read_write_resource = *
scheduler = slurm
enable_efa = compute
```

OSU Latency Comparison

Without EFA

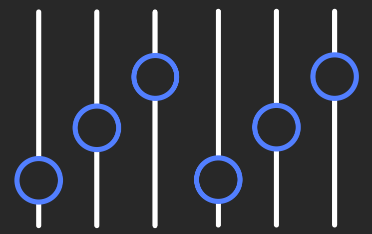
```
[ec2-user@ip-172-31-82-77 ~]$ cat osu_latency.out
# OSU MPI Latency Test v5.6.2
# Size          Latency (us)
0                      23.53
1                      23.24
2                      23.24
4                      23.23
8                      23.23
16                     23.23
32                     23.19
64                     23.26
128                    23.70
256                    23.81
512                    25.00
1024                   25.32
2048                   26.37
4096                   28.51
8192                   31.80
16384                  47.64
32768                  62.89
65536                  79.26
131072                118.89
262144                221.30
524288                420.69
1048576               841.02
2097152              2479.97
4194304              5833.30
```

With EFA

```
[ec2-user@ip-172-31-90-2 ~]$ cat osu_latency.out
# OSU MPI Latency Test v5.6.2
# Size          Latency (us)
0                      18.26
1                      18.35
2                      18.33
4                      21.81
8                      18.34
16                     18.33
32                     18.33
64                     18.39
128                    18.49
256                    18.64
512                    18.87
1024                   19.24
2048                   19.62
4096                   20.59
8192                   24.10
16384                  64.54
32768                  70.63
65536                  80.55
131072                 96.17
262144                144.74
524288                273.50
1048576               513.27
2097152              1015.75
4194304              1951.57
```

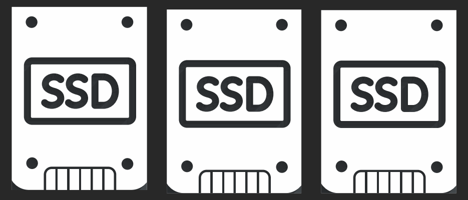
FSx for Lustre offers massively scalable file system performance

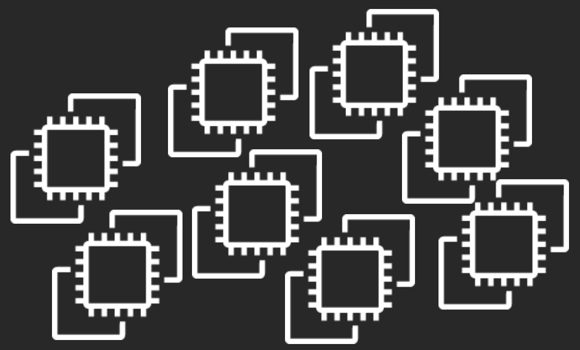
Parallel file system



100+ GiB/s throughput
Millions of IOPS
Consistent sub-millisecond latencies

SSD-based





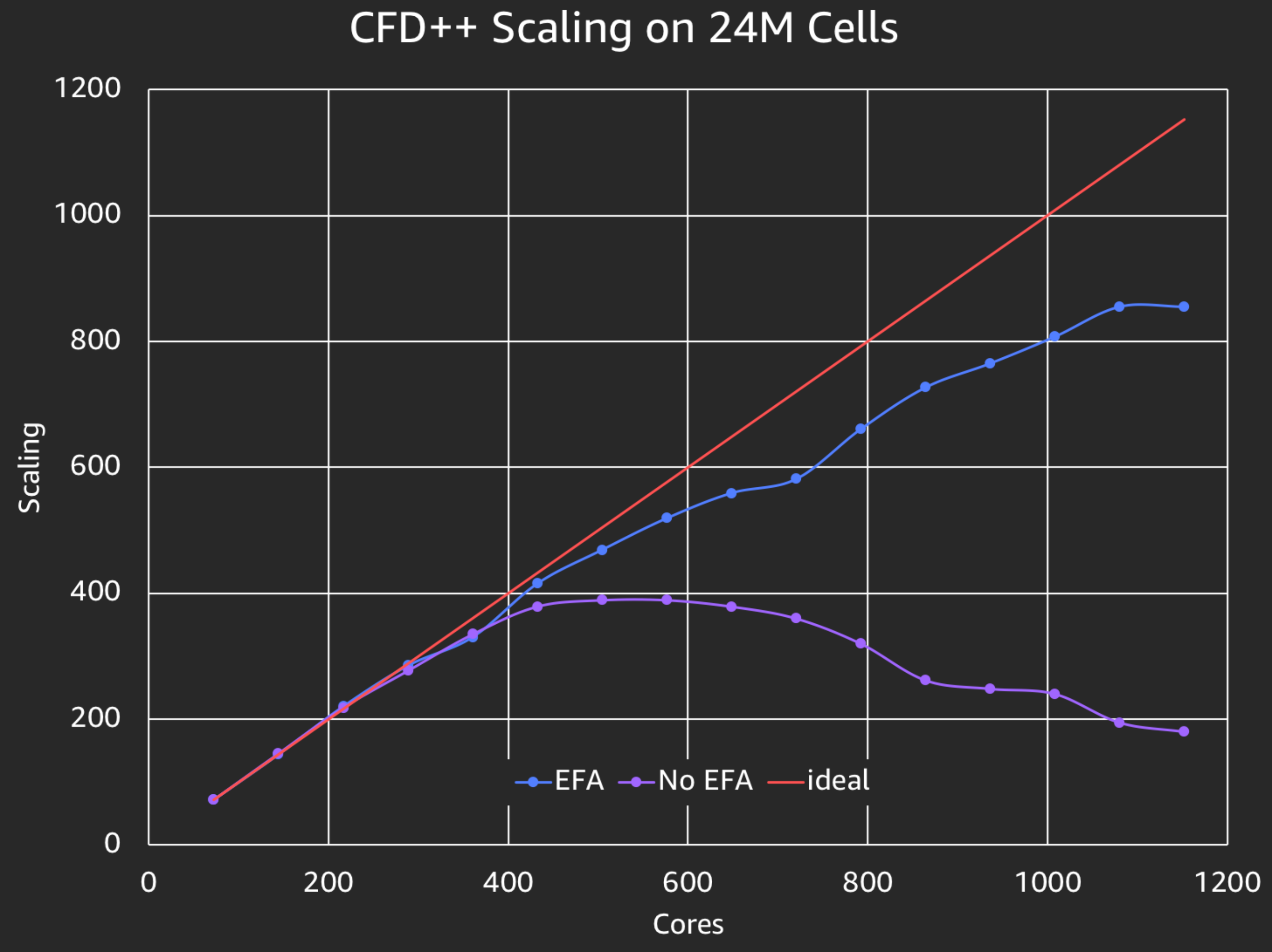
Supports hundreds of thousands of cores

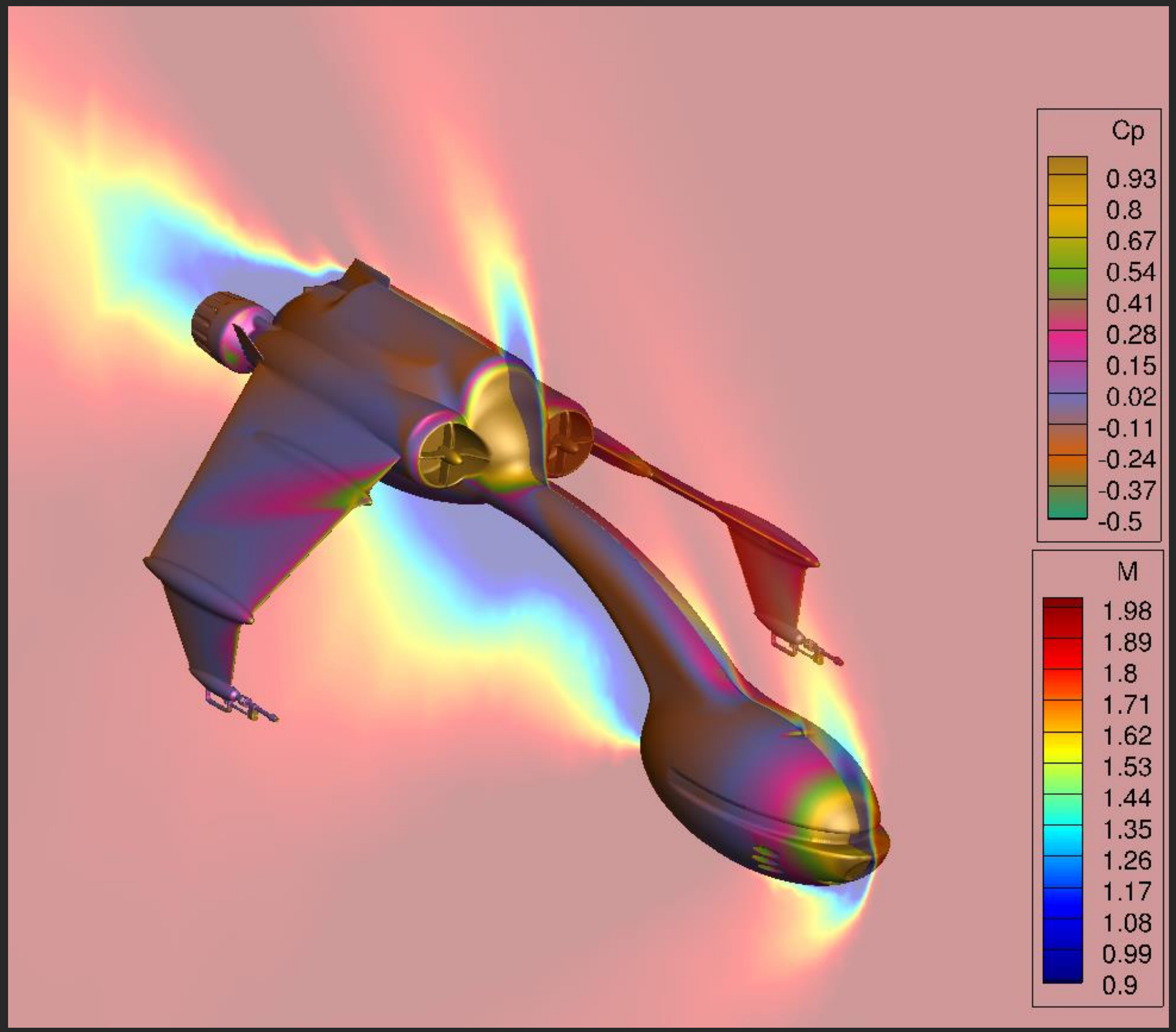
config

```
[aws]
aws_region_name = us-west-2

[cluster PlacementGroups]
key_name = PClusterKey
vpc_settings = public
placement_group = DYNAMIC
scheduler = slurm
cluster_type = spot
custom_ami = ami-0d750a1c552f67c2c
fsx_settings = fs_name

[fsx fs_name]
shared_dir = /fsx
storage_capacity = 3600
imported_file_chunk_size = 1024
export_path = s3://bucket/folder
import_path = s3://bucket
```

EFA Scaling Performance

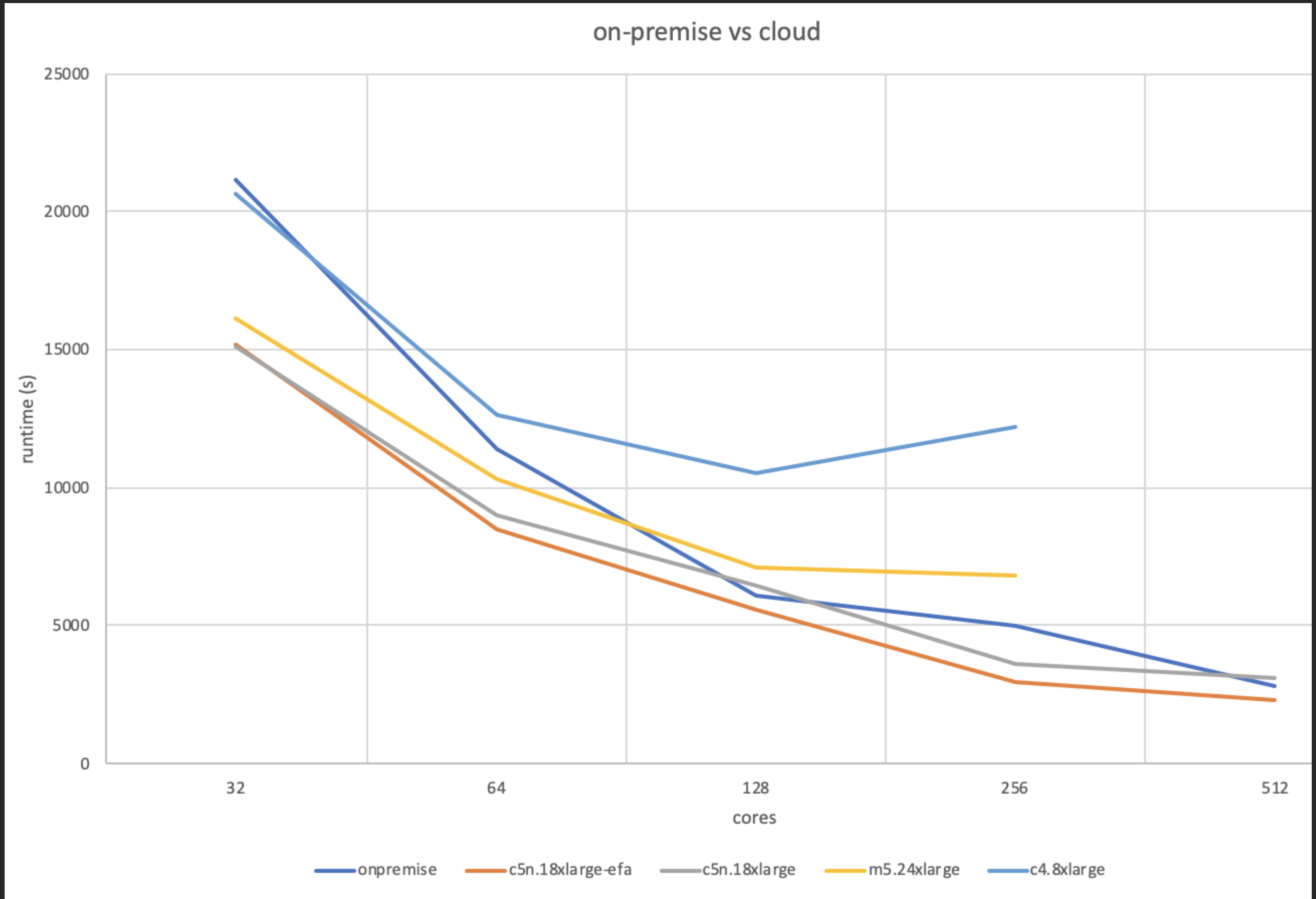
Metacomp CFD++

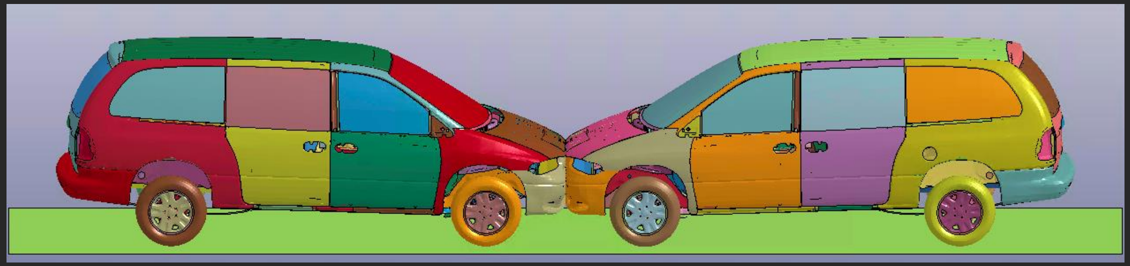




At ~1,200 cores (~33 nodes), EFA Vs non-EFA shows 4X scaling improvement with ~75% scaling efficiency

LSTC LS-DYNA



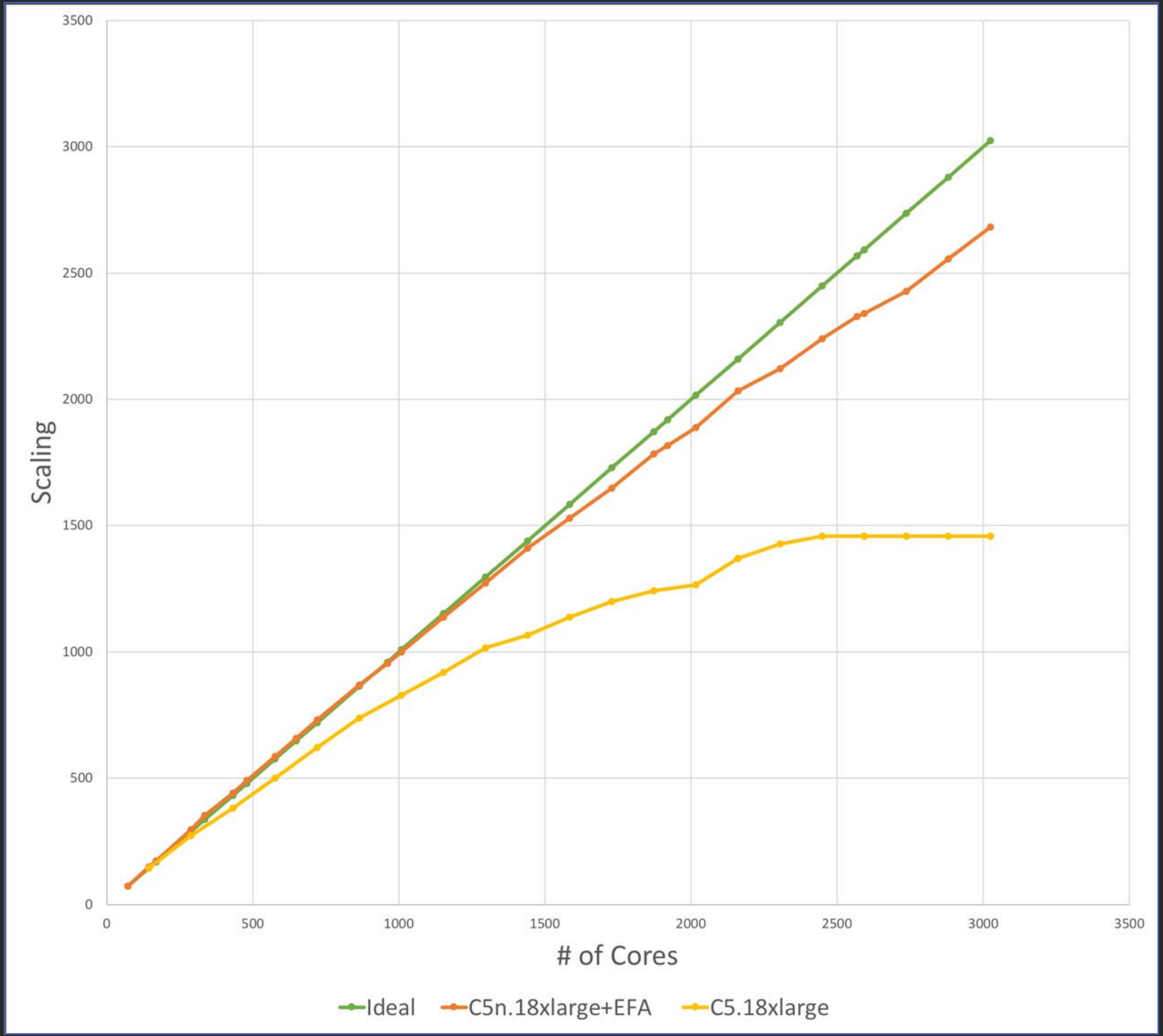


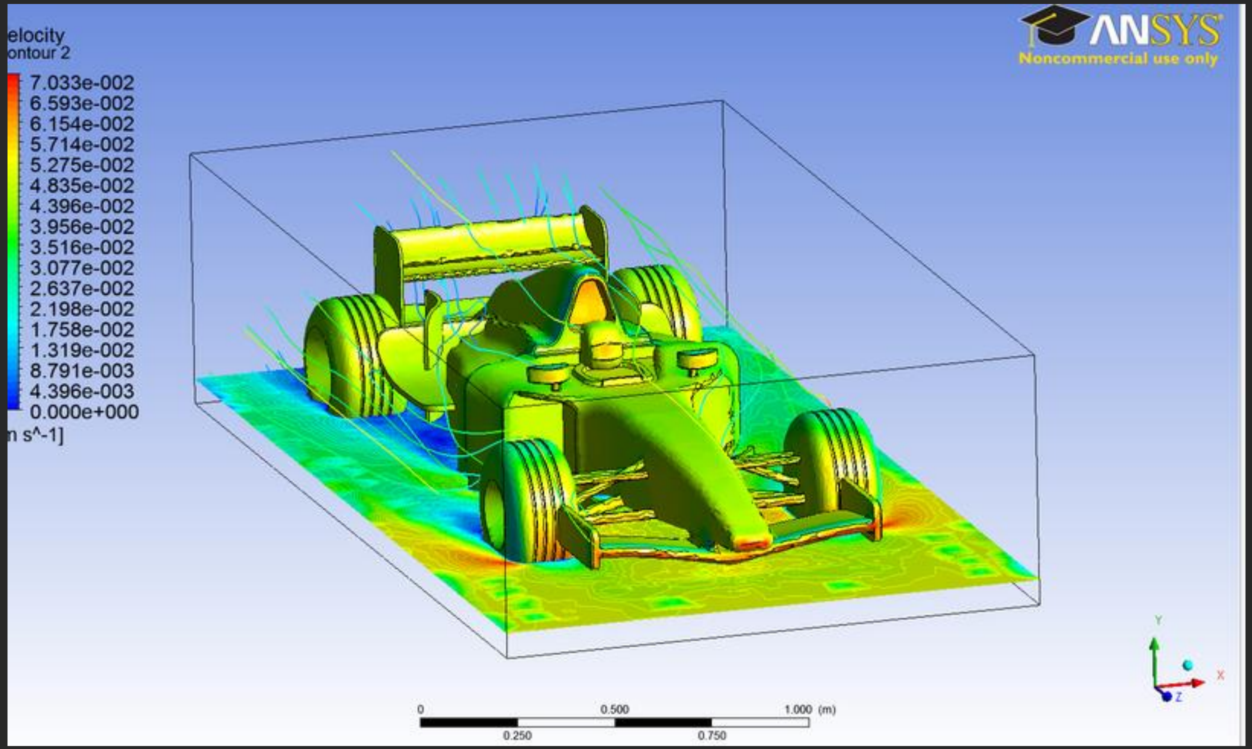
Car2Car time to completion with C5n + EFA Vs On-Premise, C5n, M5, and C4

At ~512 cores (~14 nodes), C5n+EFA shows ~25% faster time to completion over C5n w/o EFA

https://www.totalcae.com/learn/look-aws-efa-ls-dyna/

ANSYS Fluent



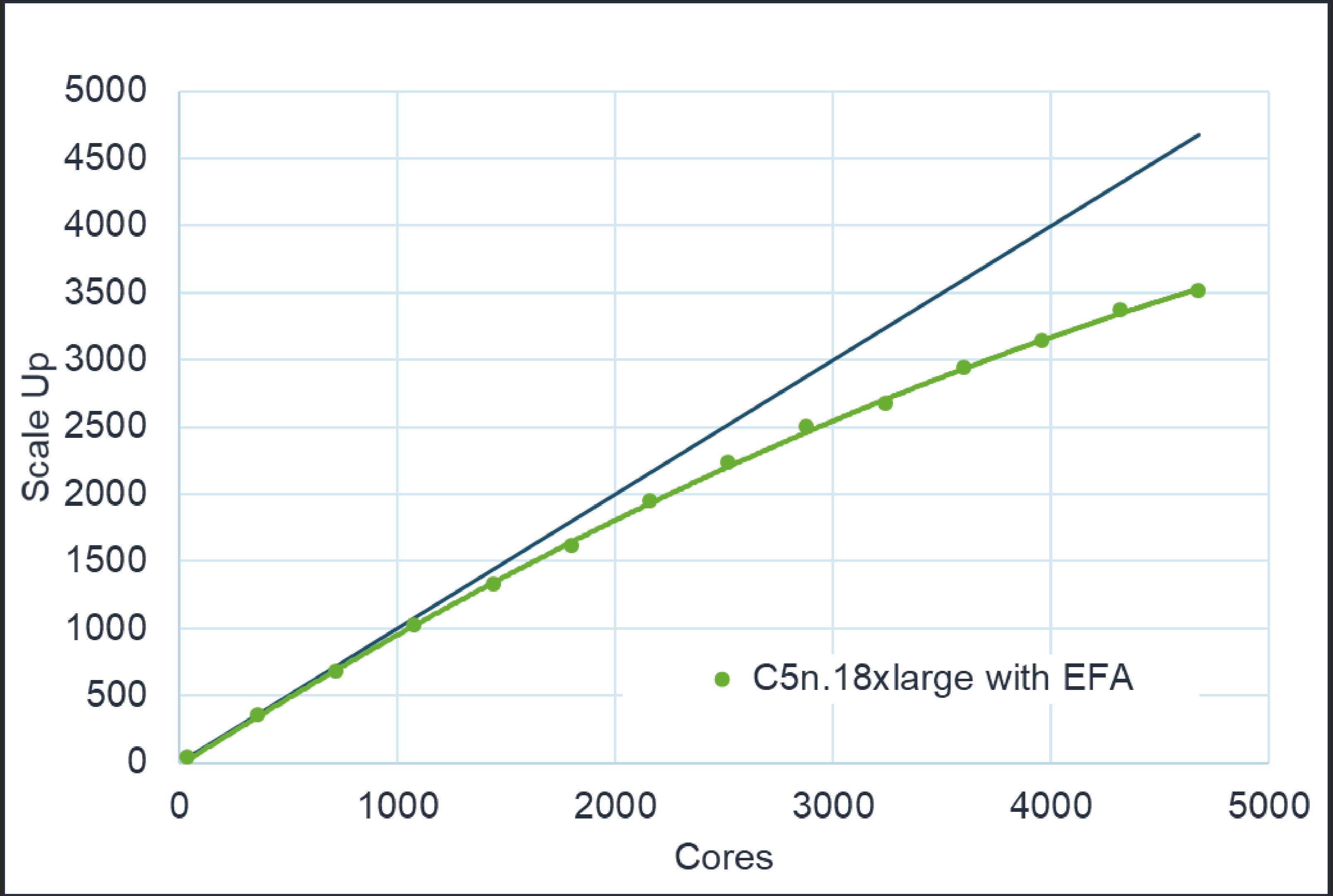


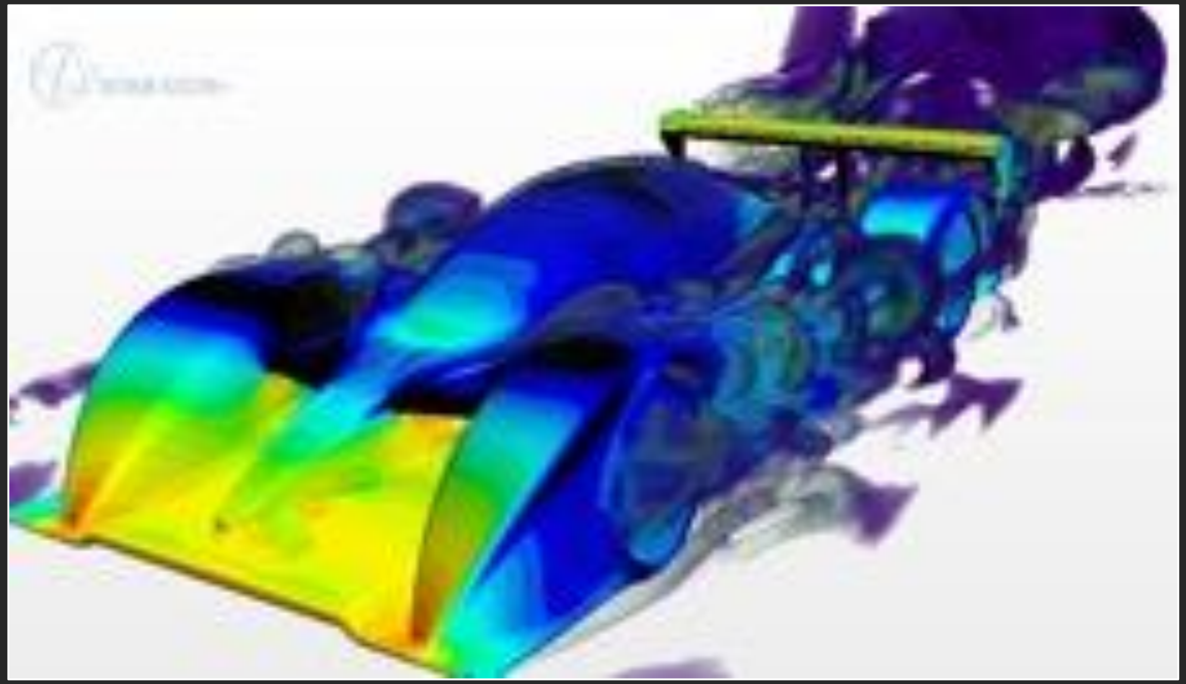
External flow over a Formula-1 Race Car (140M cell mesh)

At ~3,000 cores (~83 nodes), C5n+EFA shows ~89% scaling efficiency Vs ~48% using C5 w/o EFA

https://www.ansys.com/solutions/solutions-by-role/it-professionals/platform-support/benchmarks-overview/ansys-fluent-benchmarks/ansys-fluent-benchmarks-release-19/external-flow-over-a-formula-1-race-car

Siemens STAR-CCM+

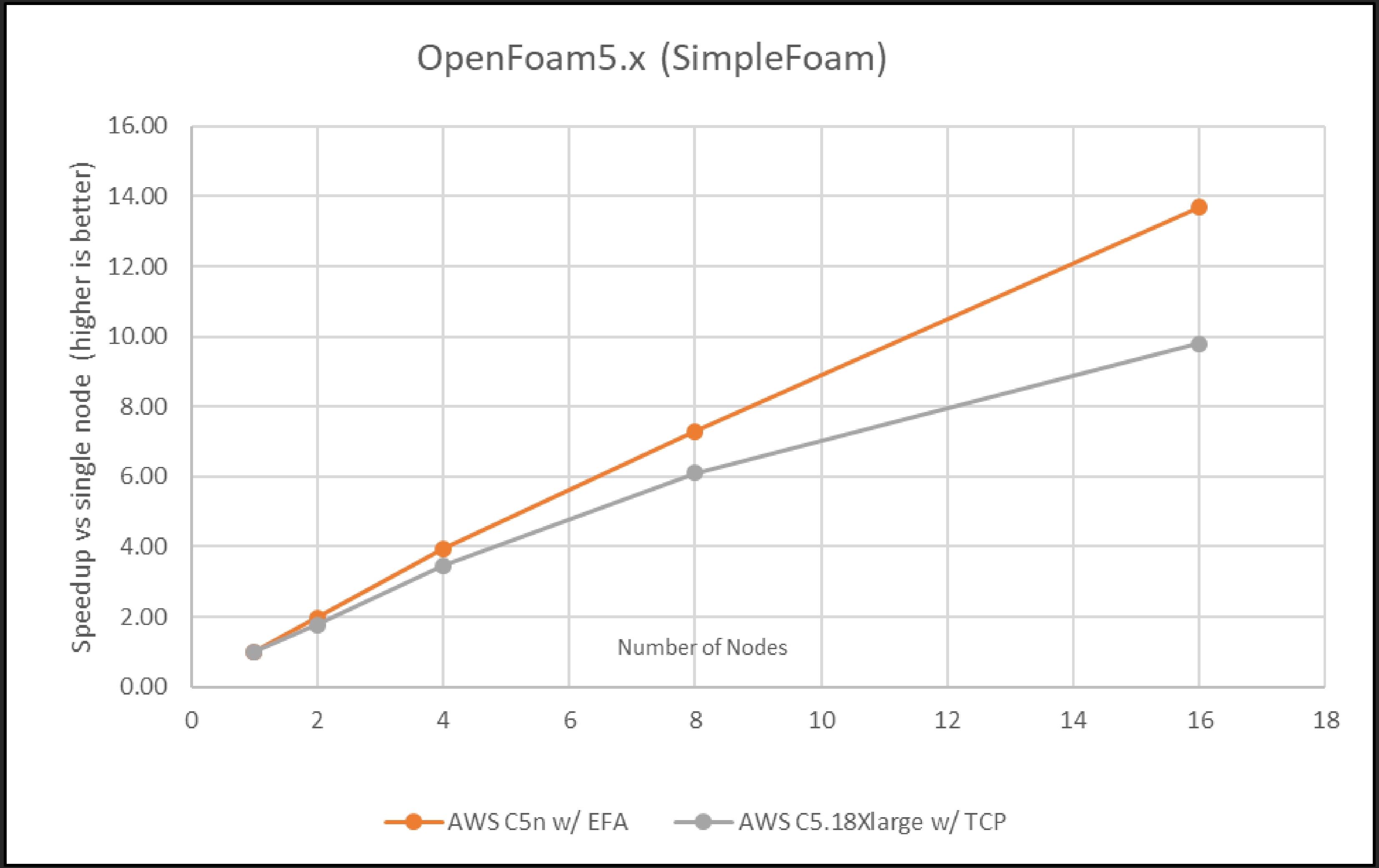


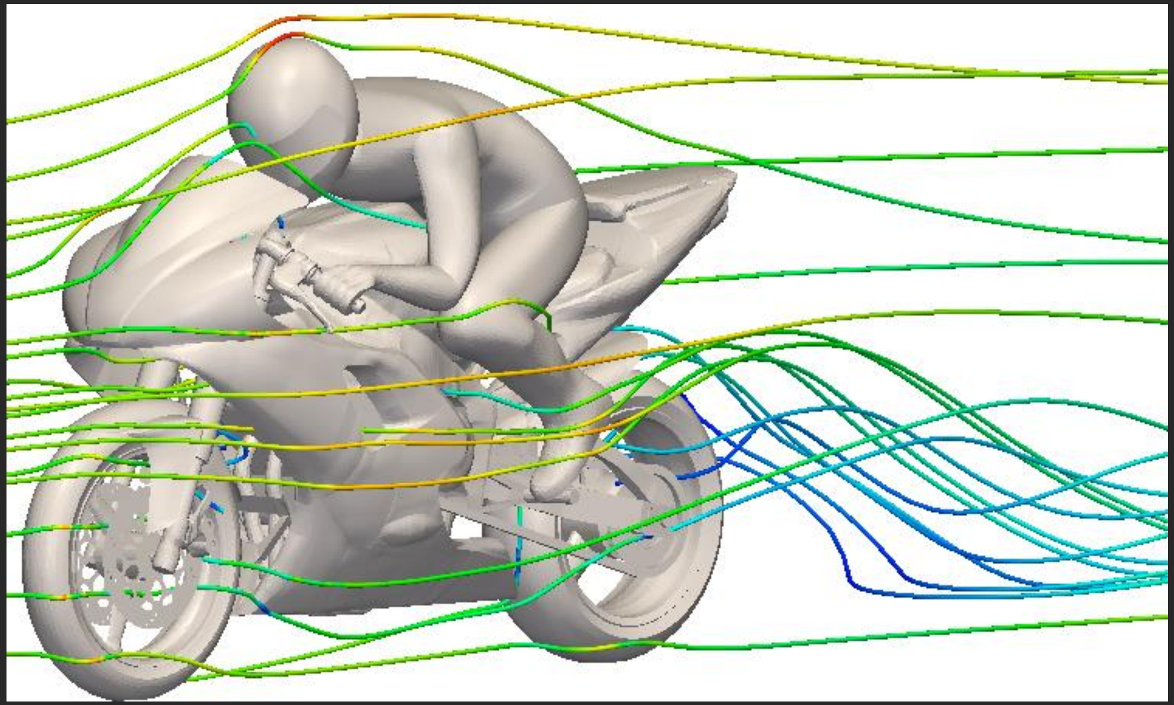


LeMans Racer (104M cell mesh) scaling with C5n+EFA with all cores fully utilized

At ~4,700 cores (~130 nodes), C5n+EFA shows ~76% scaling efficiency

OpenFOAM

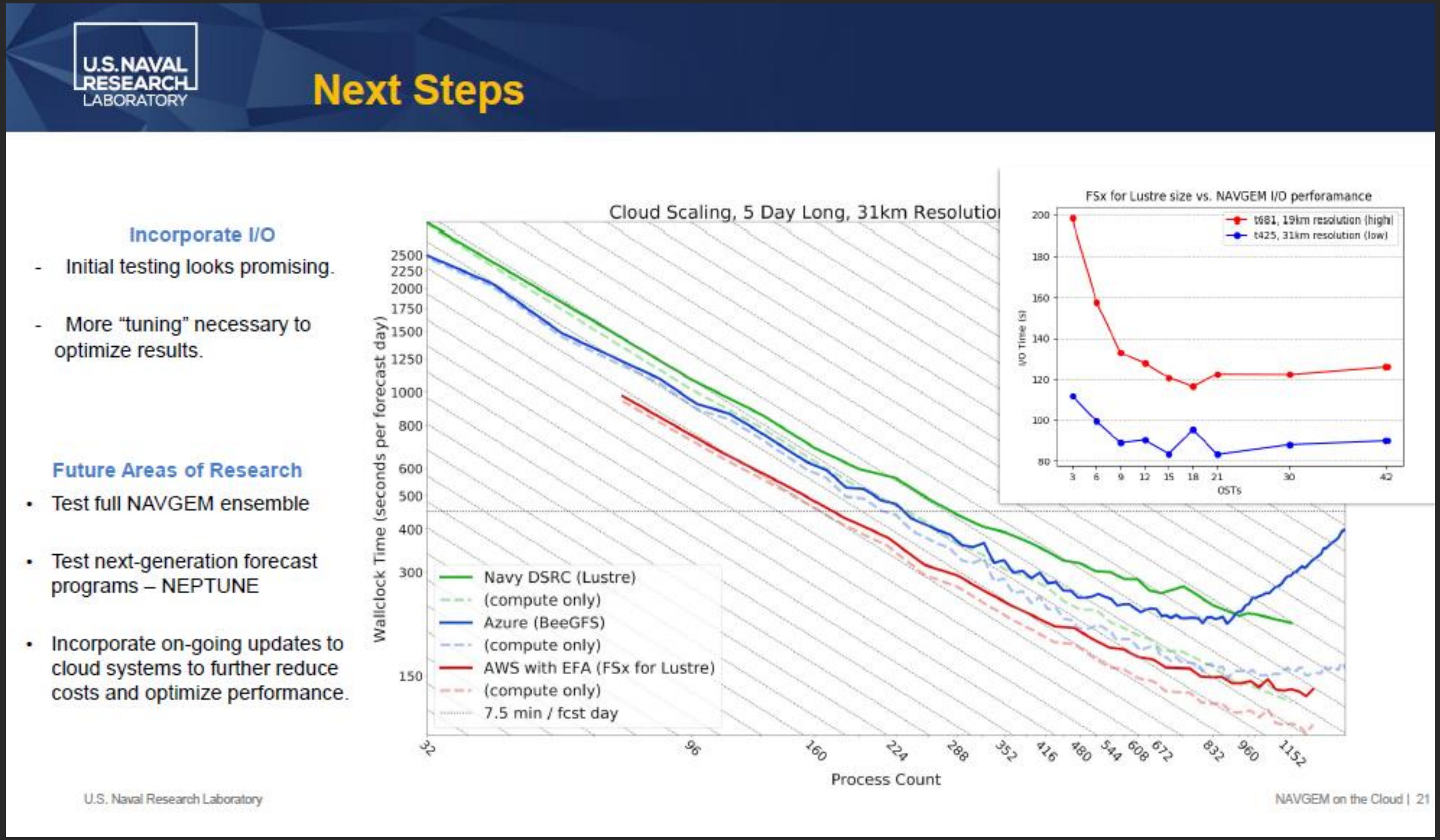




Motorbike (42M cell mesh) speedup with C5n+EFA Vs C5 without EFA

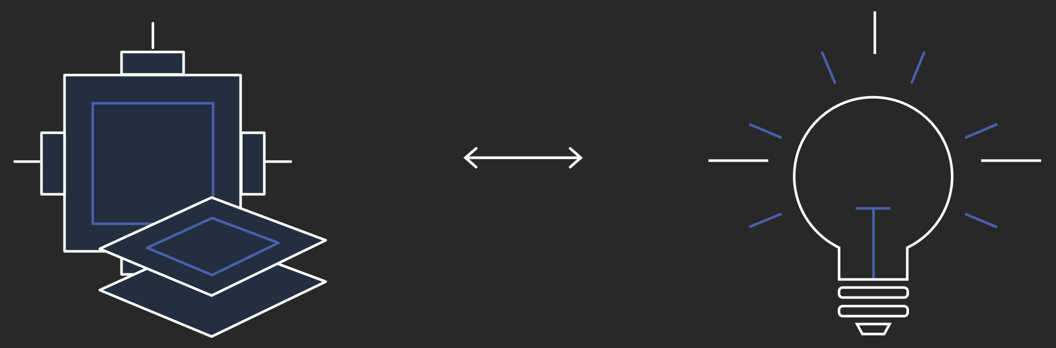
At 576 cores (16 nodes), C5n+EFA shows ~14X speedup Vs ~10X speedup with C5 without EFA

NRL NAVGEM



Navy Global Environmental Model (NAVGEM), 120hr forecast, 19km horizontal resolution

AWS w/ EFA and I/O (FSx for Lustre) is faster than compute-only times of on-prem solution



HPC on AWS

Flexible configuration and virtually unlimited scalability to grow and shrink your infrastructure as your HPC workloads dictate, not the other way around

Thank you!

Chethan Rao

raocheth@amazon.com

https://aws.amazon.com/hpc/efa/