Introduction to PyFR: A Scalable Open-Source CFD Flow Solver

Sambit Mishra

March 26, 2024

ACES
ACCELERATING COMPUTING FOR EMERGING SCIENCES

High Performance Research Computing
DIVISION OF RESEARCH

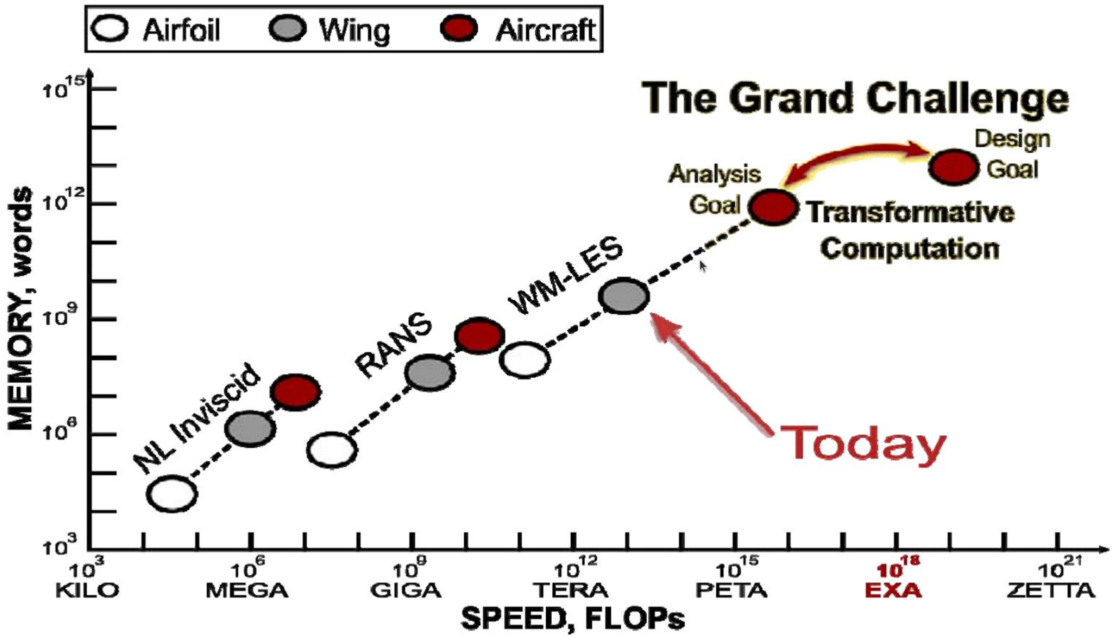
PyFR

Motivation

Computational fluid dynamics (CFD) simulations are performed across diverse disciplines …

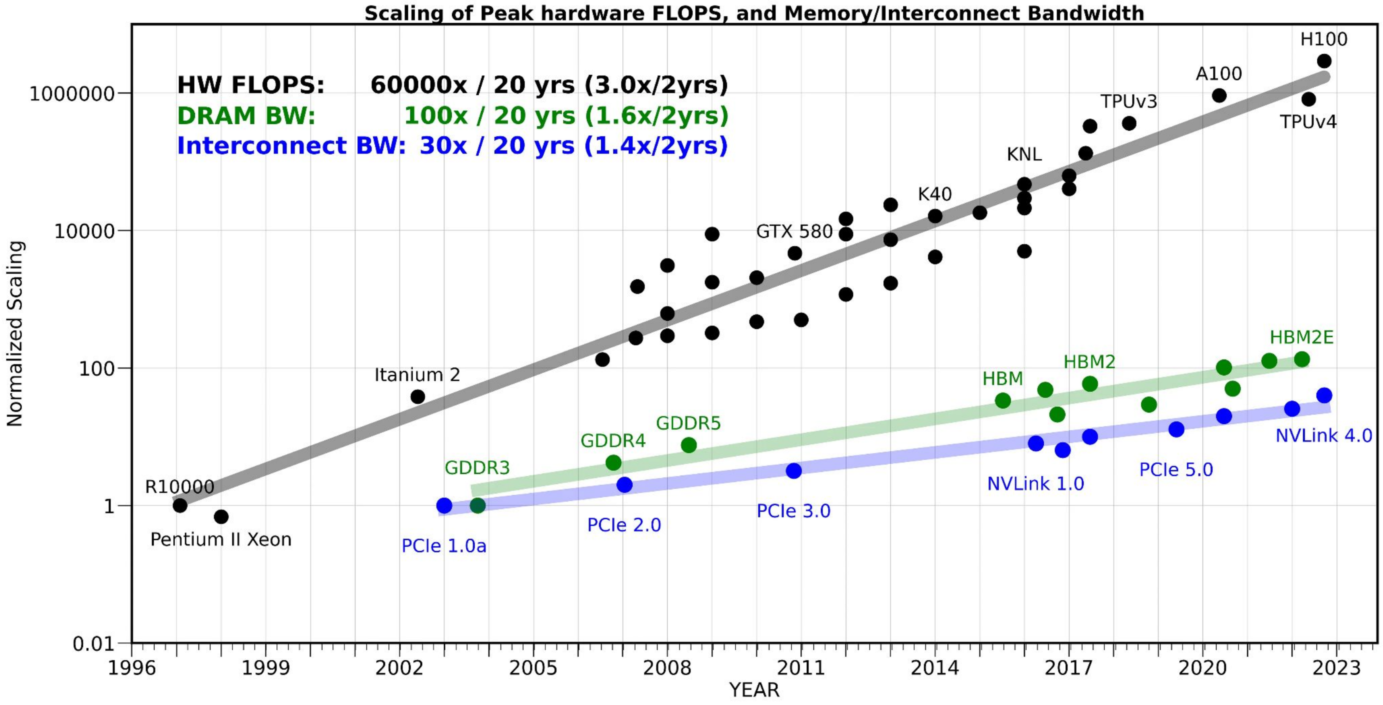
… yet their application is restricted to a small but significant area.

Motivation



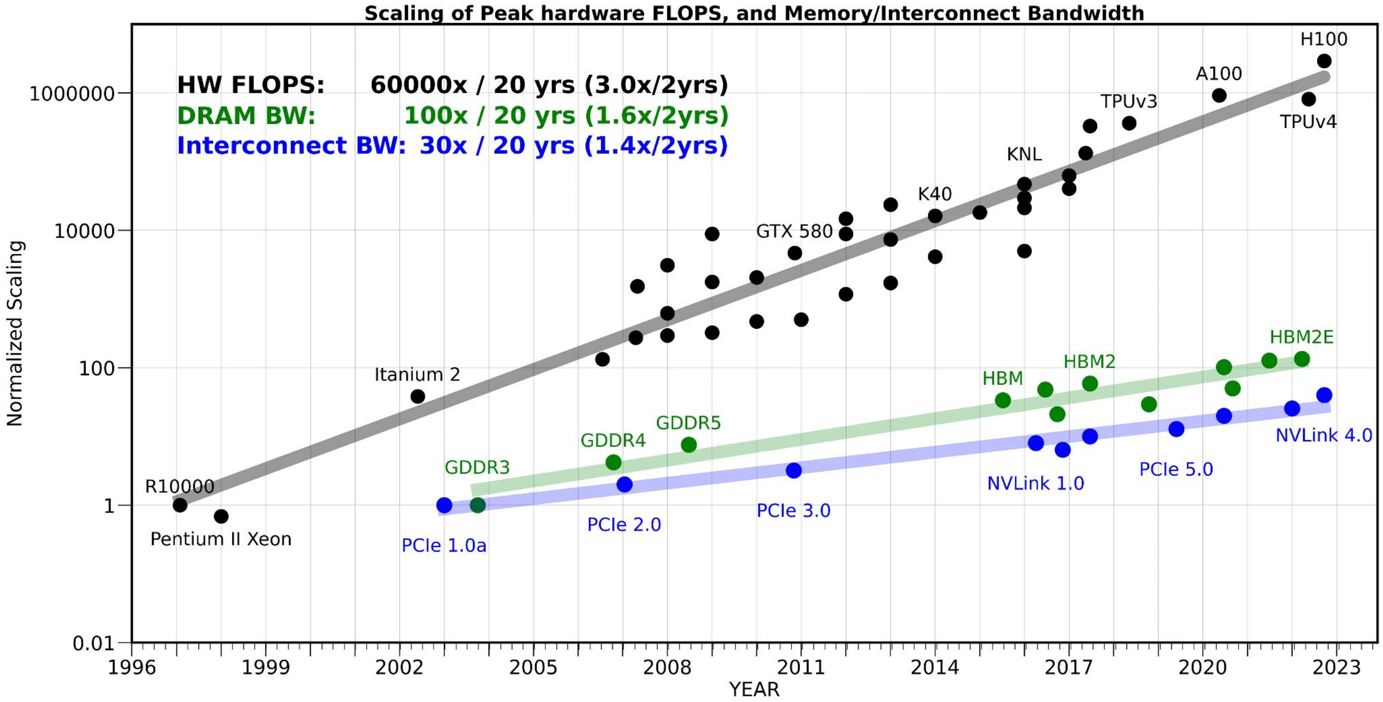
https://nyuscholars.nyu.edu/en/publications/the-opportunities-and-challenges-of-exascale-computing

Motivation



https://medium.com/riselab/ai-and-memory-wall-2cb4265cb0b8

Motivation



FLOP/s improvements have been faster than memory bandwidth improvements

https://medium.com/riselab/ai-and-memory-wall-2cb4265cb0b8

Introduction to PyFR

How does PyFR tackle these challenges?

Introduction to PyFR

How does PyFR tackle these challenges?

- Use of a high-level language (Python) to set up the system …
  … and a low-level language (domain-specific) for performance portability

Introduction to PyFR

How does PyFR tackle these challenges?

- Use of a high-level language (Python) to set up the system …
  … and a low-level language (domain-specific) for performance portability

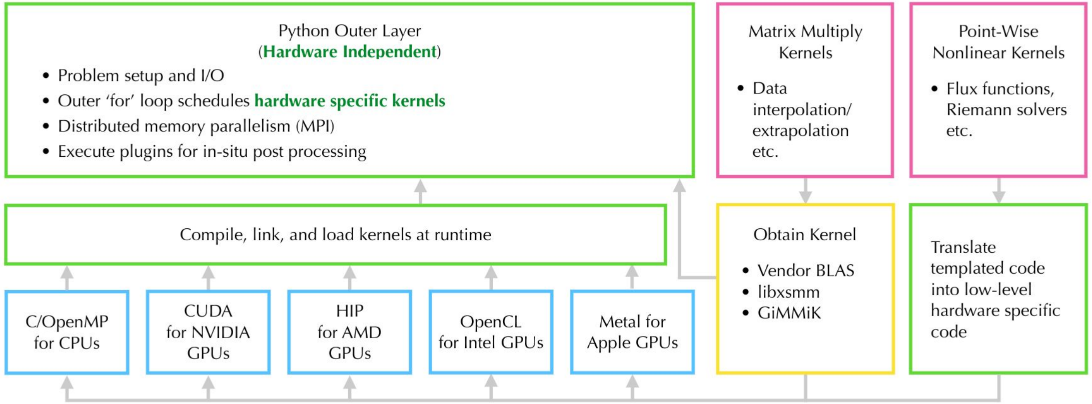
- extreme abstraction!

Introduction to PyFR

How does PyFR tackle these challenges?

- Use of a high-level language (Python) to set up the system …
  … and a low-level language (domain-specific) for performance portability

- extreme abstraction!

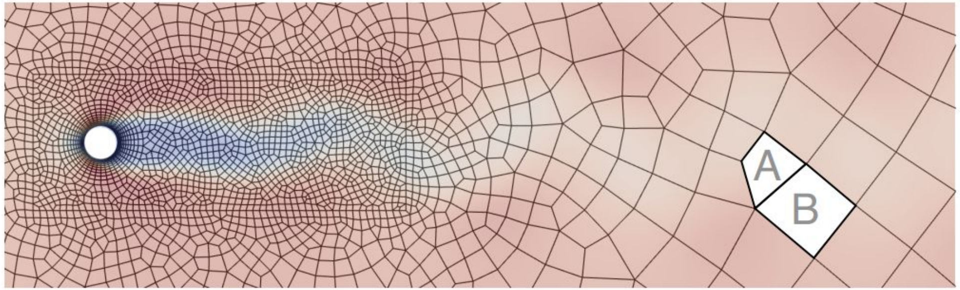


Introduction to PyFR

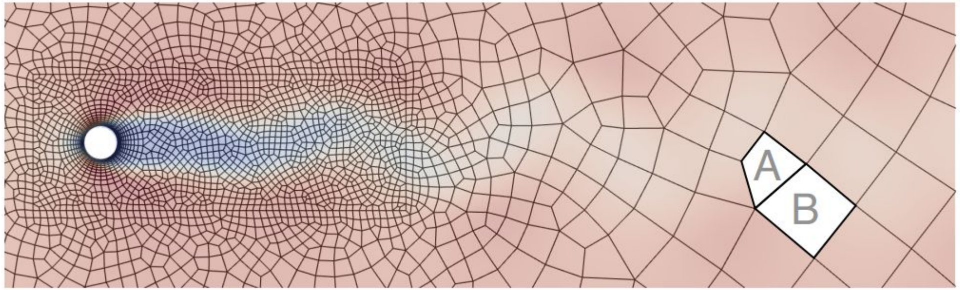
How does PyFR tackle these challenges?

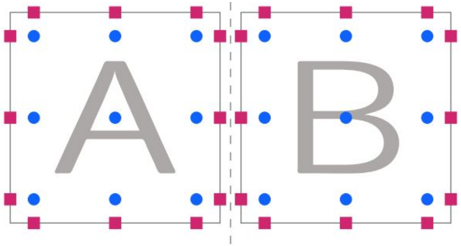
- Use of a high-level language (Python) to set up the system …
  … and a low-level language (domain-specific) for performance portability
- extreme abstraction!
- point-to-point non-blocking communication
- computation-communication overlap

Introduction to PyFR



Introduction to PyFR





Introduction to PyFR

Notable achievements

- scale to a large number of GPUs
    - strong-scaled up to 18,000 NVIDIA K20X Titan GPUs [1]
    - currently performing a 48,000 GPUs run on Frontier

- heterogeneous computing across CPUs and GPUs
    - ran a simulation with OpenMP backend on CPU, OpenCL backend on AMD W9100 GPU and CUDA backend on NVIDIA K40c GPU [2]

- Finalist for the 2016 ACM Gordon Bell Prize for Supercomputing

[1] https://doi.org/10.1109/SC.2016.1
[2] https://doi.org/10.1016/j.compfluid.2015.07.016

Introduction to PyFR

Incompressible flow past submarine [1]

Flow past airfoil for Martian helicopters [2]

Triple-point shock interaction,
animation provided by Tarik Dzanic

[1] https://doi.org/10.52843/cassyni.y2hggy
[2] https://doi.org/10.52843/cassyni.6r5ry1

PyFR v2.0.0

Governing Equations	Compressible and incompressible; Euler and Navier Stokes
Spatial Discretization	Arbitrary order Flux Reconstruction on mixed unstructured grids (triangles, quadrilaterals, tetrahedra, pyramids, prisms, hexahedra)
Temporal Discretization	Explicit adaptive Runge-Kutta schemes and implicit SDIRK.
Stabilisation	Anti-aliasing, modal filtering, entropy filtering
Shock capturing	Artificial viscosity and entropy filtering
Platforms	ARM and x86 CPU clusters AMD, Apple, Intel and NVIDIA GPU clusters
Plugins	NaN checker, file writer, progress bar, fluid force, turbulence generation, Ffowcs–Williams Hawkings, in-situ visualisation, time averaging, point sampler, expression integrator

Tutorial outline

This course shall focus on running PyFR simulations on the ACES testbed.

Tutorial outline

This course shall focus on running PyFR simulations on the ACES testbed.

To understand the fundamentals of the flux reconstruction approach, the design philosophy of PyFR, and its achievements, participants are requested to refer to resources provided towards the end of this short course.

Start with https://www.pyfr.org/

PyFR v2.0.0

Governing Equations	**Compressible** and incompressible; Euler and **Navier Stokes**
Spatial Discretization	Arbitrary order Flux Reconstruction on mixed unstructured grids (triangles, quadrilaterals, tetrahedra, pyramids, prisms, **hexahedra**)
Temporal Discretization	**Explicit adaptive Runge-Kutta schemes** and implicit SDIRK.
Stabilisation	Anti-aliasing, modal filtering, entropy filtering
Shock capturing	Artificial viscosity and entropy filtering
Platforms	ARM and **x86 CPU clusters** AMD, Apple, **Intel and NVIDIA GPU clusters**
Plugins	NaN checker, **file writer**, **progress bar**, fluid force, turbulence generation, Ffowcs–Williams Hawkings, **in-situ visualisation**, time averaging, point sampler, expression integrator

Tutorial outline

This short-course is divided into four parts.

Participants are requested to complete the questions asked over the polls.

Tutorial outline

This short-course is divided into four parts.

Participants are requested to complete the questions asked over the polls.

Course will only progress after all participants are on the same page.

Tutorial outline

This short-course is divided into four parts.

Participants are requested to complete the questions asked over the polls.

Course will only progress after all participants are on the same page.

The final postprocessing of the test simulation shall be performed offline in Paraview: https://www.paraview.org/download/.

Tutorial outline

Recommended process for learning how to use PyFR:

1. Familiarise yourself with PyFR with examples:
   https://pyfr.readthedocs.io/en/latest/examples.html

2. If stuck, go through the user's guide:
   https://pyfr.readthedocs.io/en/latest/user_guide.html

3. If still stuck, ask on the Discourse forum:
   https://pyfr.discourse.group/

Tutorial outline

Recommended process for learning how to use PyFR *for this course*:

1. Familiarise yourself with PyFR with examples: https://pyfr.readthedocs.io/en/latest/examples.html
2. If stuck, ASK ME ~~go through the user's guide: https://pyfr.readthedocs.io/en/latest/user_guide.html~~
3. If still stuck, ASK ME ~~ask on the Discourse forum: https://pyfr.discourse.group/~~

Tutorial outline

~~**PART 1:** A brief introduction and motivation to use PyFR~~
BREAK: [[Q&A session]]

PART 2: Setting up of PyFR on the ACES testbed (hands-on)
BREAK: [[Q&A session]]

PART 3: Preprocessing and running simulations (hands-on)
BREAK: [[Q&A session]]

PART 4: Viewing results and post-processing
BREAK: [[Q&A session]]

END OF PART 1

10 minutes break

10:00

Any questions?

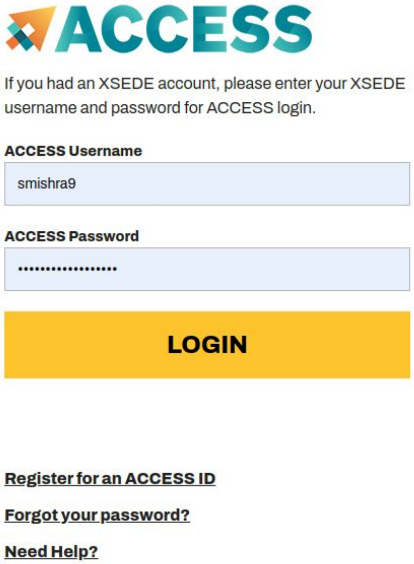
- PyFR Homepage: https://www.pyfr.org/
- PyFR usage documentation: https://pyfr.readthedocs.io/en/latest/index.html
- PyFR community: https://pyfr.discourse.group/
- All scripts and case files are available at `/scratch/training/pyfr`

PART 2

Setting up of PyFR on the ACES testbed

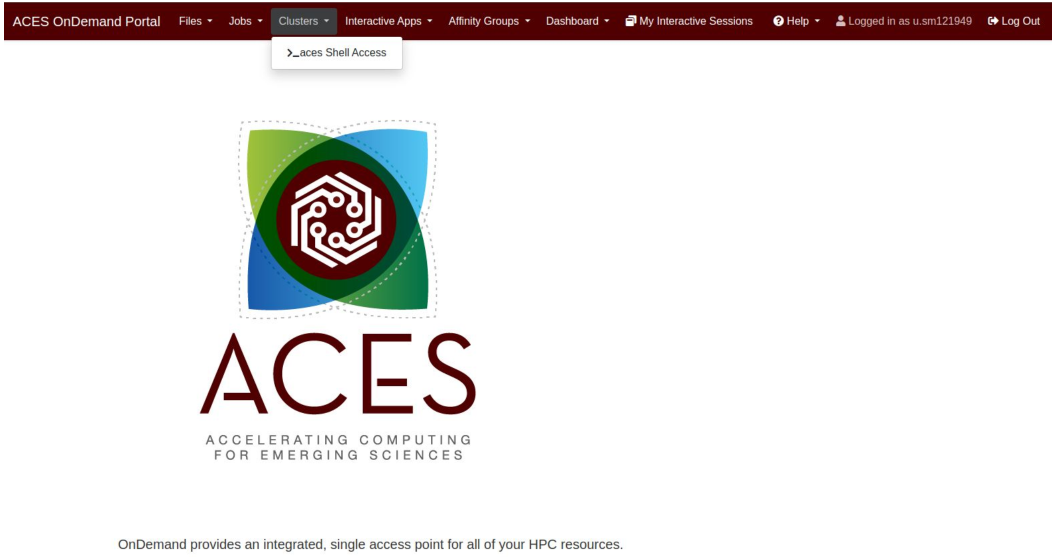
PyFR setup

Log on to access portal for ACES



PyFR setup

Open terminal on ACES: Clusters → aces Shell Access



PyFR setup

Run `source /scratch/training/pyfr/.local/setup_script`

```
Host: login.aces                                                    Themes: Default
[2024-03-25 18:12:07][u.sm121949@alogin3 ~]$ source /scratch/training/pyfr/.local/setup_script
====================================================================================================
Welcome to the Introduction to PyFR Short Course!
====================================================================================================


This script is designed to streamline your setup and get you started with PyFR, a powerful CFD solver.


====================================================================================================
For a comprehensive understanding of PyFR and its capabilities, refer to the official documentation:
https://pyfr.readthedocs.io/en/latest/index.html
====================================================================================================


Use this script to support your learning and exploration of PyFR. It's here to help!


====================================================================================================


Available Commands:
        setup_all: This will install all required software and dependencies for the course.
        test_all: Performs a quick simulation test using the Taylor-Green Vortex case to ensure everything is set up correctly.

For detailed instructions and more information on each step of the setup, view the script directly:
        cat /setup_script
====================================================================================================
To visualize your simulation results, you'll need ParaView. Install it locally from the link below:
https://www.paraview.org/download/
====================================================================================================
Enjoy your journey into computational fluid dynamics with PyFR. Let's dive in!
====================================================================================================
Run pyfr_help to see all available functions
Run pyfr_cleanup_all to clean up all paths and directories created by the functions
[2024-03-25 18:12:24][u.sm121949@alogin3 ~]$
```

PyFR setup

Run `setup_all`

```
[2024-03-26 07:40:13][u.sm121949@alogin3 ~]$ setup_all
====================================================================================================
Cleaning up all environment paths to ensure no path clashes on ACES test bed.
We clean up the following paths: PATH, CPATH, CPPATH, LDPATH, LIBRARY_PATH, LD_LIBRARY_PATH, PKG_CONFIG_PATH
The following directories shall be used for installations not available as modules:
        DOWNLOAD_LOC=/scratch/training/pyfr/.local/downloads
        LOCAL_LOC=/scratch/training/pyfr/.local/install
====================================================================================================
====================================================================================================
The following modules have been loaded:

Currently Loaded Modules:
  1) CUDA/12.3.0      7) XZ/5.4.2            13) UCX/1.14.1        19) FlexiBLAS/3.3.1     25) libreadline/8.2  31) cryptography/41.0.1
  2) GCCcore/12.3.0   8) libxml2/2.11.4      14) libfabric/1.18.0  20) FFTW/3.3.10         26) Tcl/8.6.13       32) virtualenv/20.23.1
  3) zlib/1.2.13      9) libpciaccess/0.17   15) PMIx/4.2.4        21) FFTW.MPI/3.3.10     27) SQLite/3.42.0    33) Python-bundle-PyPI/2023.06
  4) binutils/2.40   10) hwloc/2.9.1         16) UCC/1.2.0         22) ScaLAPACK/2.2.0-fb  28) libffi/3.4.4     34) Cython/3.0.7
  5) GCC/12.3.0      11) OpenSSL/1.1         17) OpenMPI/4.1.5     23) bzip2/1.0.8         29) Python/3.11.3    35) numpy/1.26.4
  6) numactl/2.0.16  12) libevent/2.1.12     18) OpenBLAS/0.3.23   24) ncurses/6.4         30) cffi/1.15.1      36) PyFR/2.0.0



====================================================================================================
====================================================================================================
The Ascent in-situ visualization library has been set up.
We have added Ascent to the path as follows:
        export PYFR_ASCENT_MPI_LIBRARY_PATH=/scratch/training/pyfr/.local/downloads/ascent/0.9.2/scripts/build_ascent/install/ascent-develop/lib/libascent_mpi.so
====================================================================================================
====================================================================================================
The METIS partitioning software has been set up.
We have added METIS to the path as PYFR_METIS_LIBRARY_PATH as follows:
        export PYFR_METIS_LIBRARY_PATH=/scratch/training/pyfr/.local/install/metis/lib/libmetis.so
====================================================================================================
====================================================================================================
The libxsmm library has been set up.
We have added libxsmm to the path as PYFR_XSMM_LIBRARY_PATH as follows:
        export PYFR_XSMM_LIBRARY_PATH=/scratch/training/pyfr/.local/downloads/libxsmm/lib/libxsmm.so
====================================================================================================
====================================================================================================
Cloning into 'PyFR-Test-Cases'...
remote: Enumerating objects: 72, done.
remote: Counting objects: 100% (43/43), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 72 (delta 20), reused 28 (delta 15), pack-reused 29
Receiving objects: 100% (72/72), 2.65 MiB | 13.69 MiB/s, done.
Resolving deltas: 100% (25/25), done.
Test cases for PyFR have been set up.
Commands used:
        rm -rf /scratch/user/u.sm121949/short-course_pyfr/01_introduction/
        mkdir -p /scratch/user/u.sm121949/short-course_pyfr/01_introduction/
        cd /scratch/user/u.sm121949/short-course_pyfr/01_introduction/
        export case_name="taylor-green"
        ml WebProxy
        git clone https://github.com/PyFR/PyFR-Test-Cases.git
        cd /scratch/user/u.sm121949/short-course_pyfr/01_introduction/PyFR-Test-Cases/3d-taylor-green
====================================================================================================
[2024-03-26 07:40:18][u.sm121949@alogin3 3d-taylor-green]$
```

https://pyfr.readthedocs.io/en/latest/index.html

PyFR setup

If confused, look into `pyfr_help`

```
[2024-03-26 07:47:12][u.sm121949@alogin3 3d-taylor-green]$ pyfr_help
==========================================================================================
Available Commands:
        pyfr_srun_help: Provides commands for setting up an interactive bash session on a compute node.
        setup_all: This will install all required software and dependencies for the course.
                setup_base: Sets up the base directories for downloads and installations.
                setup_modules: Loads the required modules for the course.
                setup_ascent: Installs the Ascent in-situ visualization library.
                setup_metis: Installs the METIS partitioning software.
                setup_libxsmm: Installs the libxsmm library.
                setup_tests_for_pyfr: Sets up the PyFR test cases.
        test_all: Performs a quick simulation test using the Taylor-Green Vortex case to ensure everything is set up correctly.
                pyfr_decompress_and_import: Decompresses and imports the Taylor-Green Vortex mesh file.
                quick_test: Runs a quick simulation test using the Taylor-Green Vortex case.
                        quick_test_one_nvidia_gpu: Runs a quick simulation test using the Taylor-Green Vortex case on a single NVIDIA GPU.
                        quick_test_one_max_gpu: Runs a quick simulation test using the Taylor-Green Vortex case on a single GPU.
                        quick_test_one_cpu: Runs a quick simulation test using the Taylor-Green Vortex case on a single CPU core.
        make_a_tgv_video: Creates a video of the Taylor-Green Vortex simulation from Ascent PNG files.
[2024-03-26 07:47:14][u.sm121949@alogin3 3d-taylor-green]$
```

https://pyfr.readthedocs.io/en/latest/index.html

PyFR setup

If the scripts do not run as expected, close the terminal window and rerun the two commands

```
source /scratch/training/pyfr/.local/setup_script
setup_all
```

https://pyfr.readthedocs.io/en/latest/index.html

Tutorial outline

~~**PART 1:** A brief introduction and motivation to use PyFR~~
~~**BREAK:** [[Q&A session]]~~

~~**PART 2:** Setting up of PyFR on the ACES testbed (hands-on)~~
BREAK: [[Q&A session]]

PART 3: Preprocessing and running simulations (hands-on)
BREAK: [[Q&A session]]

PART 4: Viewing results and post-processing
BREAK: [[Q&A session]]

10 minutes break

10:00

Any questions?

- PyFR Homepage: https://www.pyfr.org/
- PyFR usage documentation: https://pyfr.readthedocs.io/en/latest/index.html
- PyFR community: https://pyfr.discourse.group/
- All scripts and case files are available at `/scratch/training/pyfr`

PART 3

Preprocessing and running simulations

Running a simulation on ACES

1. Complete PyFR and test cases setup
    a. Ensure that the directories and files have been set up
2. Log into a compute node
3. Preprocess
4. Partition the mesh
5. Run simulation
6. Postprocess

Running a simulation on ACES

1. ~~Complete PyFR and test cases setup~~
    a. ~~Ensure that the directories and files have been set up~~

2. **Log into a compute node**

3. Preprocess

4. Partition the mesh

5. Run simulation

6. Postprocess

Running a simulation on ACES

Log into a compute node

- CPU node
- NVIDIA H100 GPU node
- Intel MAX 1100 GPU node

Running a simulation on ACES

Log into a compute node

- CPU node
  - Check availability
- NVIDIA H100 GPU node
  - Check availability
- Intel MAX 1100 GPU node
  - Check availability

00:30

Running a simulation on ACES

Log into a compute node

- CPU node
    - Check availability: `pestat -pcpu`
- NVIDIA H100 GPU node
    - Check availability: `pestat -pgpu -G`
- Intel MAX 1100 GPU node
    - Check availability: `pestat -ppvc -G`

Running a simulation on ACES

Log into a compute node

- CPU node
  - ~~Check availability: `pestat -pcpu`~~
  - Log into a node
- NVIDIA H100 GPU node
  - ~~Check availability: `pestat -pgpu -G`~~
  - Log into a node
- Intel MAX 1100 GPU node
  - ~~Check availability: `pestat -ppvc -G`~~
  - Log into a node

00:30

Running a simulation on ACES

Log into a compute node

- CPU node
  - ~~Check availability: `pestat -pcpu`~~
  - Log into a node: `srun -N 1 --reservation=training -pcpu --mem=50G --time=02:00:00 --tasks-per-node=24 --pty bash`
- NVIDIA H100 GPU node
  - ~~Check availability: `pestat -pgpu -G`~~
  - Log into a node: `srun -N 1 --reservation=training -pgpu --mem=50G --gres=gpu:h100:2 --time=02:00:00 --tasks-per-node=1  --pty bash`
- Intel MAX 1100 GPU node
  - ~~Check availability: `pestat -ppvc -G`~~
  - Log into a node: `srun -N 1 -ppvc --mem=50G --gres=gpu:pvc:2 --time=02:00:00 --tasks-per-node=2  --pty bash`

00:30

Running a simulation on ACES

Log into a compute node

- CPU node
  - ~~Check availability: `pestat -pcpu`~~
  - Log into a node: `srun -N 1 --reservation=training -pcpu --mem=50G --time=02:00:00 --tasks-per-node=24 --pty bash`
- NVIDIA H100 GPU node
  - ~~Check availability: `pestat -pgpu -G`~~
  - Log into a node: `srun -N 1 --reservation=training -pgpu --mem=50G --gres=gpu:h100:2 --time=02:00:00 --tasks-per-node=2  --pty bash`
- Intel MAX 1100 GPU node
  - ~~Check availability: `pestat -ppvc -G`~~
  - Log into a node: `srun -N 1 -ppvc --mem=50G --gres=gpu:pvc:2 --time=02:00:00 --tasks-per-node=2  --pty bash`

Make sure you are within a compute node: `[2024-03-26 08:31:50][u.sm121949@ac073 3d-taylor-green]$`

00:30

Running a simulation on ACES

1. ~~Complete PyFR and test cases setup~~
   - ~~Ensure that the directories and files have been set up~~

2. ~~Log into a compute node~~

3. **Preprocess**

4. Partition the mesh

5. Run simulation

6. Postprocess

Running a simulation on ACES

Preprocess

- Uncompress mesh file

00:30

Running a simulation on ACES

Preprocess

- Uncompress mesh file: `unxz ${MESH_MSH_XZ} ${MESH_MSH}`

00:30

Running a simulation on ACES

Preprocess

- ~~Uncompress mesh file: `unxz ${MESH_MSH_XZ} ${MESH_MSH}`~~
- Import to PyFR format

00:30

https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Preprocess

- ~~Uncompress mesh file: `unxz ${MESH_MSH_XZ} ${MESH_MSH}`~~
- Import to PyFR format: `pyfr import ${MESH_MSH} ${MESH_PYFRM}`

00:30

https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Preprocess

- ~~Uncompress mesh file: `unxz ${MESH_MSH_XZ} ${MESH_MSH}`~~
- ~~Import to PyFR format: `pyfr import ${MESH_MSH} ${MESH_PYFRM}`~~
- Set up the configuration file.

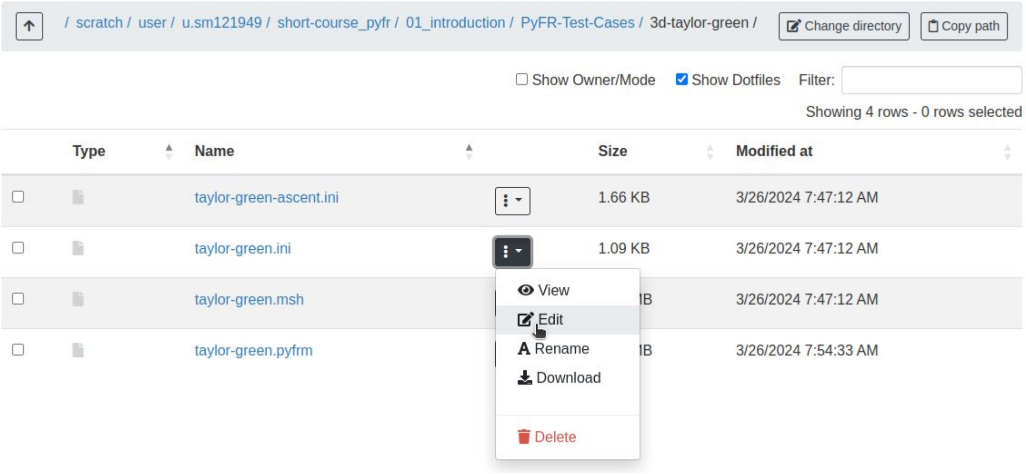
00:30

https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Preprocess

- ~~Uncompress mesh file: `unxz ${MESH_MSH_XZ} ${MESH_MSH}`~~
- ~~Import to PyFR format: `pyfr import ${MESH_MSH} ${MESH_PYFRM}`~~
- Set up the configuration file.



00:30

https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Preprocess

- ~~Uncompress mesh file: `unxz ${MESH_MSH_XZ} ${MESH_MSH}`~~
- ~~Import to PyFR format: `pyfr import ${MESH_MSH} ${MESH_PYFRM}`~~
- Set up the configuration file.

```
[backend]
precision = single

[constants]
gamma = 1.4
mu = 6.25e-4
Pr = 0.71
Ps = 111.607

[solver]
system = navier-stokes
order = 3

[solver-time-integrator]
scheme = rk34
controller = pi
tstart = 0
tend = 10 ; 20
dt = 5e-4
dt-max = 2e-3
atol = 0.00001
rtol = 0.00001
errest-norm = uniform
safety-fact = 0.9
min-fact = 0.3
max-fact = 2.5

[solver-interfaces]
riemann-solver = rusanov
ldg-beta = 0.5
ldg-tau = 0.1

[solver-interfaces-quad]
flux-pts = gauss-legendre

[solver-elements-hex]
soln-pts = gauss-legendre

```

```

[solver-interfaces-quad]
flux-pts = gauss-legendre

[solver-elements-hex]
soln-pts = gauss-legendre

[soln-plugin-integrate]
nsteps = 500
file = integral.csv
header = true

int-vol = 1

; vor1 = (grad_w_y - grad_v_z)
; vor2 = (grad_u_z - grad_w_x)
; vor3 = (grad_v_x - grad_u_y)

; int-E = rho*(u*u + v*v + w*w)
; int-enst = rho*(%(vor1)s*%(vor1)s + %(vor2)s*%(vor2)s + %(vor3)s*%(vor3)s)

[soln-plugin-writer]
dt-out = 1
basedir = .
basename = taylor-green-{t:.2f}

[soln-plugin-dtstats]
flushsteps = 1000
file = dtstats.csv
header = true

[soln-ics]
rho = 1 + (1.0/(16.0*Ps))*(cos(2*x) + cos(2*y))*(cos(2*z) + 2)
u = sin(x)*cos(y)*cos(z)
v = -cos(x)*sin(y)*cos(z)
w = 0
p = Ps + (1.0/16.0)*(cos(2*x) + cos(2*y))*(cos(2*z) + 2)

```

00:30

https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Preprocess

- ~~Uncompress mesh file: `unxz ${MESH_MSH_XZ} ${MESH_MSH}`~~
- ~~Import to PyFR format: `pyfr import ${MESH_MSH} ${MESH_PYFRM}`~~
- Set up the configuration file.

```
[backend]
precision = single

[constants]
gamma = 1.4
mu = 6.25e-4
Pr = 0.71
Ps = 111.607

[solver]
system = navier-stokes
order = 3

[solver-time-integrator]
scheme = rk34
controller = pi
tstart = 0
tend = 10 ; 20
dt = 5e-4
dt-max = 2e-3
atol = 0.00001
rtol = 0.00001
errest-norm = uniform
safety-fact = 0.9
min-fact = 0.3
max-fact = 2.5

[solver-interfaces]
riemann-solver = rusanov
ldg-beta = 0.5
ldg-tau = 0.1

[solver-interfaces-quad]
flux-pts = gauss-legendre

[solver-elements-hex]
soln-pts = gauss-legendre

```

```

[solver-interfaces-quad]
flux-pts = gauss-legendre

[solver-elements-hex]
soln-pts = gauss-legendre

[soln-plugin-integrate]
nsteps = 500
file = integral.csv
header = true

int-vol = 1

; vor1 = (grad_w_y - grad_v_z)
; vor2 = (grad_u_z - grad_w_x)
; vor3 = (grad_v_x - grad_u_y)

; int-E = rho*(u*u + v*v + w*w)
; int-enst = rho*(%(vor1)s*%(vor1)s + %(vor2)s*%(vor2)s + %(vor3)s*%(vor3)s)

[soln-plugin-writer]
dt-out = 1
basedir = .
basename = taylor-green-{t:.2f}

[soln-plugin-dtstats]
flushsteps = 1000
file = dtstats.csv
header = true

[soln-ics]
rho = 1 + (1.0/(16.0*Ps))*(cos(2*x) + cos(2*y))*(cos(2*z) + 2)
u = sin(x)*cos(y)*cos(z)
v = -cos(x)*sin(y)*cos(z)
w = 0
p = Ps + (1.0/16.0)*(cos(2*x) + cos(2*y))*(cos(2*z) + 2)

```

Q: What does this do?

00:30

https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

1. ~~Complete PyFR and test cases setup~~
   - ~~Ensure that the directories and files have been set up~~
2. ~~Log into a compute node~~
3. ~~Preprocess~~
4. **Partition**
5. Run
6. Postprocess

00:30

Running a simulation on ACES

Partition (if simulation is too slow, or memory requirements are too big)

- On a CPU node
  - Check available physical cores
- On a GPU node
  - Check available devices

00:30

Running a simulation on ACES

Partition (if simulation is too slow, or memory requirements are too big)

- ~~On a CPU node~~
    - ~~Check available physical cores: `lscpu`~~
- ~~On a GPU node~~
    - ~~Check available devices: `clinfo -l`~~
- Partition mesh

00:30

https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Partition (if simulation is too slow, or memory requirements are too big)

- ~~On a CPU node~~
  - ~~Check available physical cores: `lscpu`~~
- ~~On a GPU node~~
  - ~~Check available devices: `clinfo -l`~~
- Partition mesh: `pyfr partition 8 ${MESH_PYFRM}`

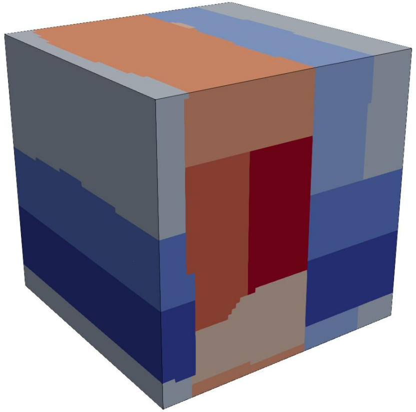
00:30

https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Partition (if simulation is too slow, or memory requirements are too big)

- ~~On a CPU node~~
  - ~~Check available physical cores: `lscpu`~~
- ~~On a GPU node~~
  - ~~Check available devices: `clinfo -l`~~
- Partition mesh: `pyfr partition 8 ${MESH_PYFRM}`



00:30

https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

1. ~~Complete PyFR and test cases setup~~
    a. ~~Ensure that the directories and files have been set up~~

2. ~~Log into a compute node~~

3. ~~Preprocess~~

4. ~~Partition~~

5. **Run**

6. Postprocess

00:30

Running a simulation on ACES

Run

- Run simulation

00:30

https://www.open-mpi.org/doc/v4.1/man1/mpirun.1.php
https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Run

- Run simulation: `mpirun -n ${NRANKS} pyfr --progress run --backend ${BACKEND} ${MESH_PYFRM} ${CONFIG_INI}`

00:30

https://www.open-mpi.org/doc/v4.1/man1/mpirun.1.php
https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Run

- ~~Run simulation: `mpirun -n ${NRANKS} pyfr --progress run --backend ${BACKEND} ${MESH_PYFRM} ${CONFIG_INI}`~~
- Check simulation

00:30

https://www.open-mpi.org/doc/v4.1/man1/mpirun.1.php
https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Run

- ~~Run simulation: `mpirun -n ${NRANKS} pyfr --progress run --backend ${BACKEND} ${MESH_PYFRM} ${CONFIG_INI}`~~
- Check simulation

```
[2024-03-26 08:40:34][u.sm121949@ac073 3d-taylor-green]$ mpirun -mca btl ^ofi -n 24 pyfr --progress run -b openmp ${case_name}.pyfrm ${case_name}-ascent.ini
   0.3% [>                                                                    ] 0.06/20.00 ela: 00:00:08 rem: 00:49:52
```

```
[2024-03-26 08:43:02][u.sm121949@ac052 3d-taylor-green]$ mpirun -mca btl ^ofi -n 2 pyfr --progress run -b cuda ${case_name}.pyfrm ${case_name}-ascent.ini
   2.7% [===>                                                                 ] 0.55/20.00 ela: 00:00:04 rem: 00:02:42
```

```
[2024-03-26 08:47:23][u.sm121949@ac051 3d-taylor-green]$ mpirun -mca btl ^ofi -n 2 pyfr --progress run -b opencl ${case_name}.pyfrm ${case_name}-ascent.ini
   2.6% [===>                                                                 ] 0.52/20.00 ela: 00:00:07 rem: 00:04:23
```

00:30

https://www.open-mpi.org/doc/v4.1/man1/mpirun.1.php
https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Run

- ~~Run simulation: `mpirun -n ${NRANKS} pyfr --progress run --backend ${BACKEND} ${MESH_PYFRM} ${CONFIG_INI}`~~
- Check simulation
  - on the CPU node
  - on NVIDIA H100 GPU node
  - on Intel MAX GPU node

```
[2024-03-26 08:40:34][u.sm121949@ac073 3d-taylor-green]$ mpirun -mca btl ^ofi -n 24 pyfr --progress run -b openmp ${case_name}.pyfrm ${case_name}-ascent.ini
  0.3% [>                                                                    ] 0.06/20.00 ela: 00:00:08 rem: 00:49:52
```

```
[2024-03-26 08:43:02][u.sm121949@ac052 3d-taylor-green]$ mpirun -mca btl ^ofi -n 2 pyfr --progress run -b cuda ${case_name}.pyfrm ${case_name}-ascent.ini
  2.7% [===>                                                                 ] 0.55/20.00 ela: 00:00:04 rem: 00:02:42
```

```
[2024-03-26 08:47:23][u.sm121949@ac051 3d-taylor-green]$ mpirun -mca btl ^ofi -n 2 pyfr --progress run -b opencl ${case_name}.pyfrm ${case_name}-ascent.ini
  2.6% [===>                                                                 ] 0.52/20.00 ela: 00:00:07 rem: 00:04:23
```

00:30

https://www.open-mpi.org/doc/v4.1/man1/mpirun.1.php
https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Run

- ~~Run simulation: `mpirun -n ${NRANKS} pyfr --progress run --backend ${BACKEND} ${MESH_PYFRM} ${CONFIG_INI}`~~
- Check simulation
  - on the CPU node: `top`
  - on NVIDIA H100 GPU node: `nvidia-smi`
  - on Intel MAX GPU node: `sysmon`

```
[2024-03-26 08:40:34][u.sm121949@ac073 3d-taylor-green]$ mpirun -mca btl ^ofi -n 24 pyfr --progress run -b openmp ${case_name}.pyfrm ${case_name}-ascent.ini
   0.3% [>                                                                                              ] 0.06/20.00 ela: 00:00:08 rem: 00:49:52
```

```
[2024-03-26 08:43:02][u.sm121949@ac052 3d-taylor-green]$ mpirun -mca btl ^ofi -n 2 pyfr --progress run -b cuda ${case_name}.pyfrm ${case_name}-ascent.ini
   2.7% [===>                                                                                           ] 0.55/20.00 ela: 00:00:04 rem: 00:02:42
```

```
[2024-03-26 08:47:23][u.sm121949@ac051 3d-taylor-green]$ mpirun -mca btl ^ofi -n 2 pyfr --progress run -b opencl ${case_name}.pyfrm ${case_name}-ascent.ini
   2.6% [===>                                                                                           ] 0.52/20.00 ela: 00:00:07 rem: 00:04:23
```

00:30

https://www.open-mpi.org/doc/v4.1/man1/mpirun.1.php
https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

Run

- ~~Run simulation:~~ ~~`mpirun -n ${NRANKS} pyfr --progress run --backend ${BACKEND} ${MESH_PYFRM} ${CONFIG_INI}`~~
- ~~Check simulation~~
  - ~~on the CPU node:~~ ~~`top`~~
  - ~~on NVIDIA H100 GPU node:~~ ~~`nvidia-smi`~~
  - ~~on Intel MAX GPU node:~~ ~~`sysmon`~~
- Ensure that all output files are being appropriately created

00:30

https://www.open-mpi.org/doc/v4.1/man1/mpirun.1.php
https://pyfr.readthedocs.io/en/latest/index.html

Running a simulation on ACES

1. ~~Complete PyFR and test cases setup~~
    - ~~Ensure that the directories and files have been set up~~

2. ~~Log into a compute node~~

3. ~~Preprocess~~

4. ~~Partition~~

5. ~~Run~~

6. **Postprocess**

00:30

10 minutes break

10:00

Any questions?

Please install Paraview for the final part: https://www.paraview.org/download/

- PyFR Homepage: https://www.pyfr.org/
- PyFR usage documentation: https://pyfr.readthedocs.io/en/latest/index.html
- PyFR community: https://pyfr.discourse.group/
- All scripts and case files are available at `/scratch/training/pyfr`

PART 4
Viewing results and post-processing

Tutorial outline

~~**PART 1:** A brief introduction and motivation to use PyFR~~

~~**PART 2:** Setting up of PyFR on the ACES testbed (hands-on)~~
~~**BREAK:** [[Q&A session]]~~

~~**PART 3:** Preprocessing and running simulations (hands-on)~~
~~**BREAK:** [[Q&A session]]~~

PART 4: Viewing results and post-processing
BREAK: [[Q&A session]]

Running a simulation on ACES

1. ~~Complete PyFR and test cases setup~~
   - ~~Ensure that the directories and files have been set up~~
2. ~~Log into a compute node~~
3. ~~Preprocess~~
4. ~~Partition (if needed)~~
5. ~~Run~~
6. **Postprocess**

00:30

Running a simulation on ACES

Postprocess

- (Optional) Run `make_a_tgv_video.`

https://pyfr.readthedocs.io/en/latest/index.html
https://www.paraview.org/download/

00:30

Running a simulation on ACES

Postprocess

- (Optional) Run `make_a_tgv_video.` It runs the following:

```
ml FFmpeg/6.0
ffmpeg -framerate 10 -pattern_type glob -i 'taylor-green-*.png' -c:v libx264 -r 30 -pix_fmt yuv420p output.mp4
```

https://pyfr.readthedocs.io/en/latest/index.html
https://www.paraview.org/download/

00:30

Running a simulation on ACES

Postprocess

- ~~(Optional) Run `make_a_tgv_video`. It runs the following:~~
  - ~~`ml FFmpeg/6.0`~~
  - ~~`ffmpeg -framerate 10 -pattern_type glob -i 'taylor-green-*.png' -c:v libx264 -r 30 -pix_fmt yuv420p output.mp4`~~

- Export solution file: `pyfr export ${MESH_PYFRM} ${MESH_PYFRS} ${SOLN_VTU}`

https://pyfr.readthedocs.io/en/latest/index.html
https://www.paraview.org/download/

00:30

Running a simulation on ACES

Postprocess

- ~~(Optional) Run `make_a_tgv_video`. It runs the following:~~
  - ~~`ml FFmpeg/6.0`~~
  - ~~`ffmpeg -framerate 10 -pattern_type glob -i 'taylor-green-*.png' -c:v libx264 -r 30 -pix_fmt yuv420p output.mp4`~~
- ~~Export solution file: `pyfr export ${MESH_PYFRM} ${MESH_PYFRS} ${SOLN_VTU}`~~
- Download the vtu file.

https://pyfr.readthedocs.io/en/latest/index.html
https://www.paraview.org/download/

00:30

Running a simulation on ACES

Postprocess

- ~~(Optional) Run `make_a_tgv_video`. It runs the following:~~
  - ~~`ml FFmpeg/6.0`~~
  - ~~`ffmpeg -framerate 10 -pattern_type glob -i 'taylor-green-*.png' -c:v libx264 -r 30 -pix_fmt yuv420p output.mp4`~~
- ~~Export solution file: `pyfr export ${MESH_PYFRM} ${MESH_PYFRS} ${SOLN_VTU}`~~
- ~~Download the vtu file.~~
- Open Paraview.

https://pyfr.readthedocs.io/en/latest/index.html
https://www.paraview.org/download/

00:30

Wish to repeat the simulation?

1. `source /scratch/training/pyfr/.local/setup_script`
2. `setup_all`
3. `test_all`

User support

1. Research-level implementation of PyFR:
   - Literature search for PyFR papers
   - All presentations of researchers using PyFR: https://cassyni.com/s/pyfr

2. PyFR Homepage: https://www.pyfr.org/

3. PyFR usage documentation: https://pyfr.readthedocs.io/en/latest/index.html

4. PyFR community: https://pyfr.discourse.group/

Participants can reach out to me for help (creating unstructured mesh files, set up SBATCH scripts etc.)

Acknowledgements

This work was supported by

- the National Science Foundation (NSF), award numbers:
  - 2112356 - ACES - Accelerating Computing for Emerging Sciences
  - 1925764 - SWEETER - SouthWest Expertise in Expanding, Training, Education and Research
  - 2019129 - FASTER - Fostering Accelerated Scientific Transformations, Education, and Research
- Staff and students at Texas A&M High-Performance Research Computing.

High Performance Research Computing

DIVISION OF RESEARCH

https://hprc.tamu.edu

HPRC Helpdesk:

help@hprc.tamu.edu
Phone: 979-845-0219

Help us help you. Please include details in your request for support, such as, Cluster (ACES, Faster, Grace, ViDaL), NetID (UserID), Job information (Job id(s), Location of your jobfile, input/output files, Application, Module(s) loaded, Error messages, etc), and Steps you have taken, so we can reproduce the problem.