MFC: Performant Multiphase Flow Simulation at Leadership-Class scale using OpenACC

Spencer Bryngelson

Georgia Institute of Technology

June 6, 2023

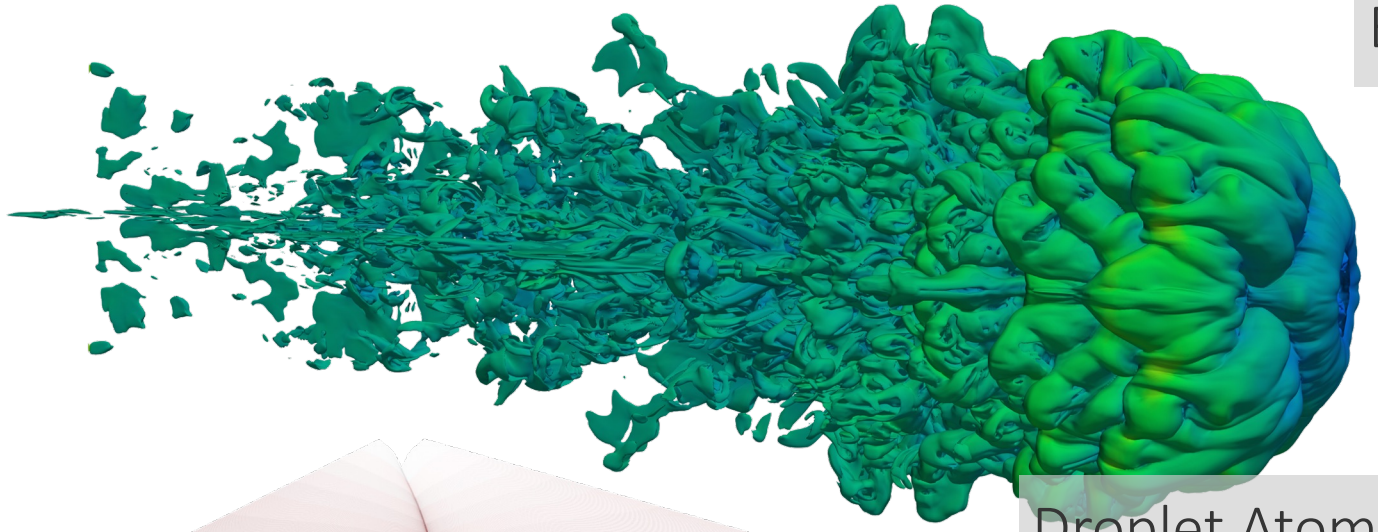
OpenACC Webinar

Computational Physics Group

https://comp-physics.group

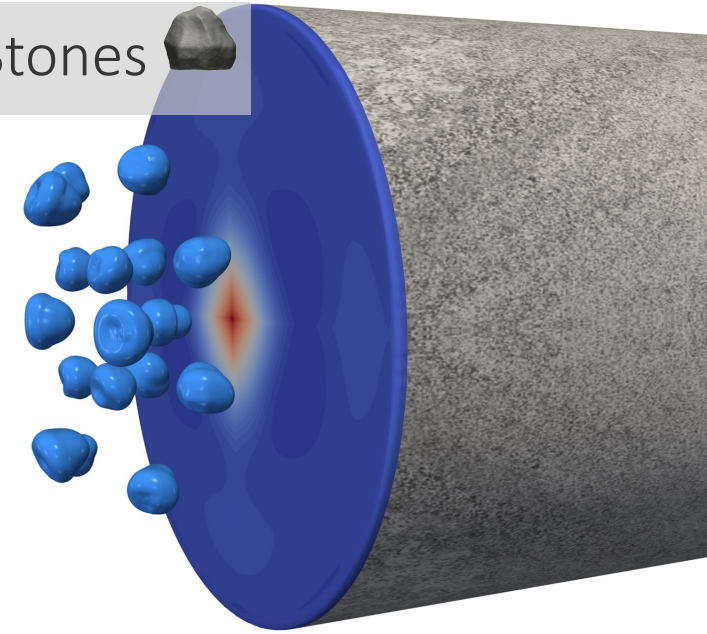
Atomizing water droplet. Vorticity shown.

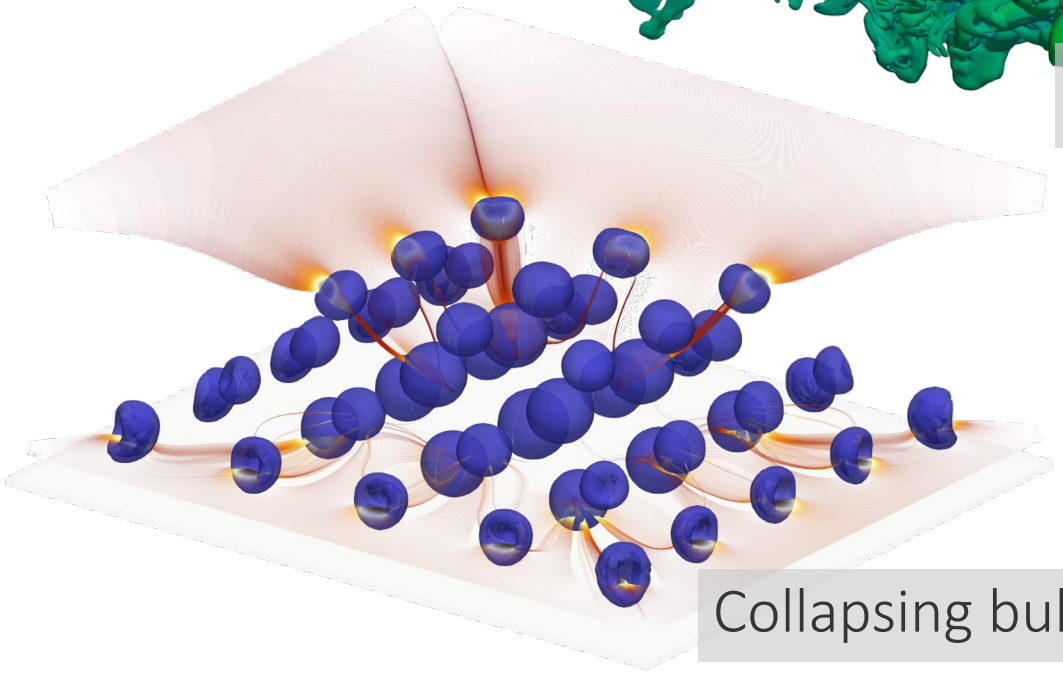
Computational fluid dynamics



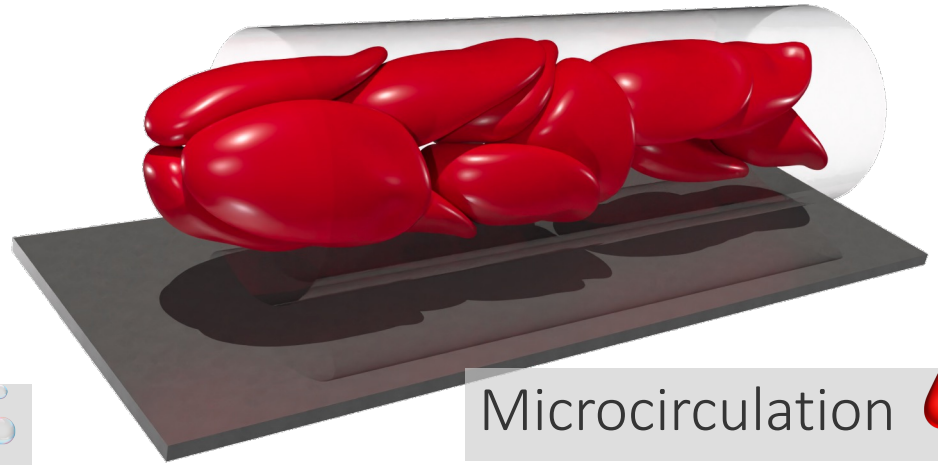
Droplet Atomization 😮‍💨

Breakup Kidney Stones





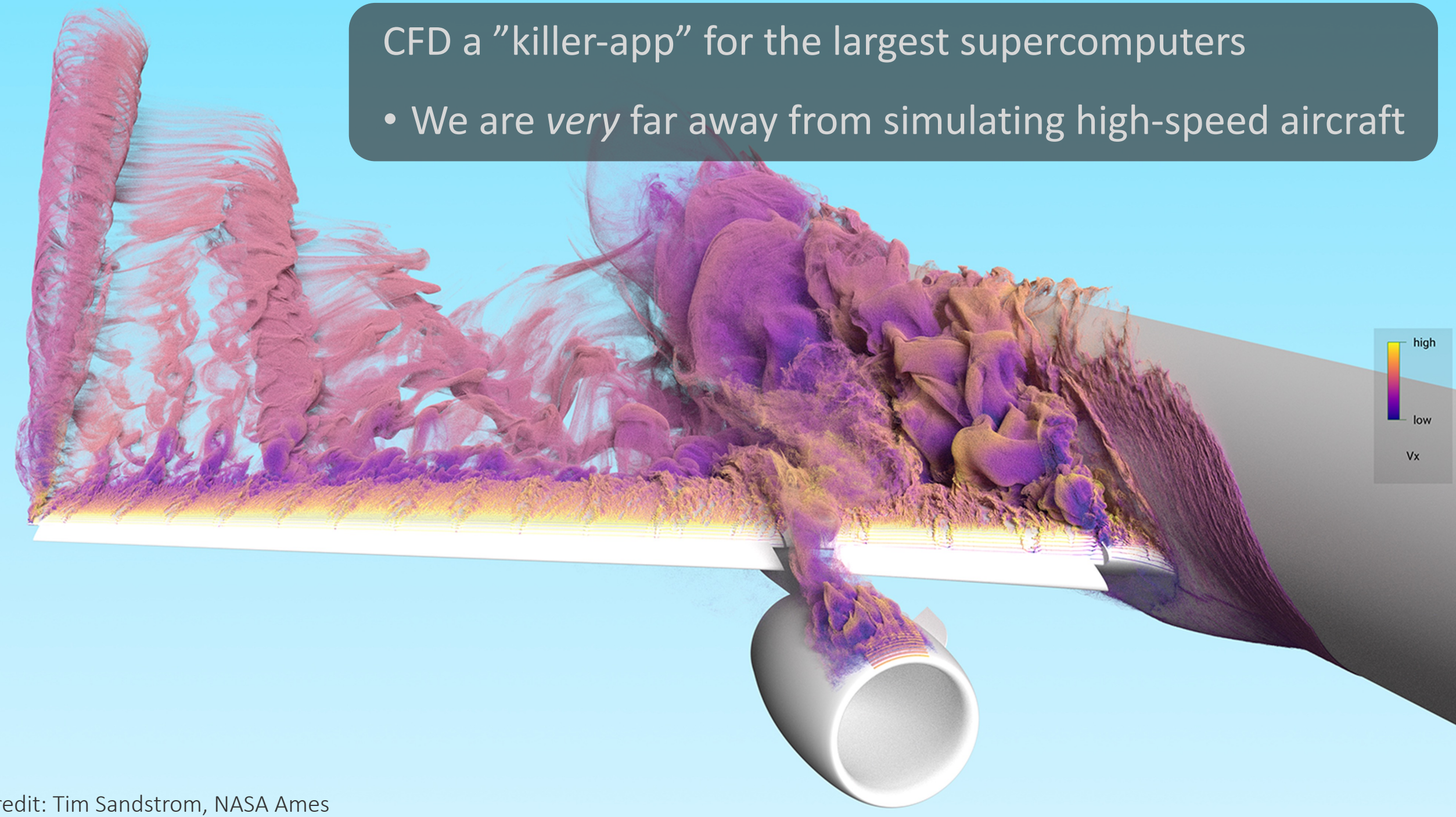
Collapsing bubble clouds 🫧



Microcirculation 🩸

CFD a ”killer-app” for the largest supercomputers

- We are *very* far away from simulating high-speed aircraft



Credit: Tim Sandstrom, NASA Ames

GPUs are enabling computation

- FLOPS on new supercomputers come from GPUs, e.g.,
  - US: Summit, Frontier, El Capitan
  - Europe: LUMI, Leonardo
- Even at double precision
  - 1 Intel Xeon Gold: 1 TFLOPS
  - 1 NVIDIA A100: 10 TFLOPS
  - 1 AMD MI250X: 50 TFLOPS

Goal

Efficient flow simulation on such computers
without armory of developers

Model and numerics

- Compressible, multi-phase
- **Diffuse-interfaces**
  - Baer–Nunziato-like, 4,5,6,7-eqn
- **Finite volume:** WENO (shock-capturing)
- **Riemann solve:** HLLC
- **Time stepping:** Explicit RK
- **Extra!**: Sub-grid models, phase change, surface tension, sharpening, …

$$\frac{\partial \alpha_i \rho_i}{\partial t} + \nabla \cdot (\alpha_i \rho_i \boldsymbol{u}) = 0,$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u} + p\boldsymbol{I}) = 0,$$

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot [(\rho E + p)\boldsymbol{u}] = 0,$$

$$\frac{\partial \alpha_i}{\partial t} + \boldsymbol{u} \cdot \nabla \alpha_i = -k \nabla \cdot \boldsymbol{u}$$

B., Schmidmayer, et. al *CPC* (2021)

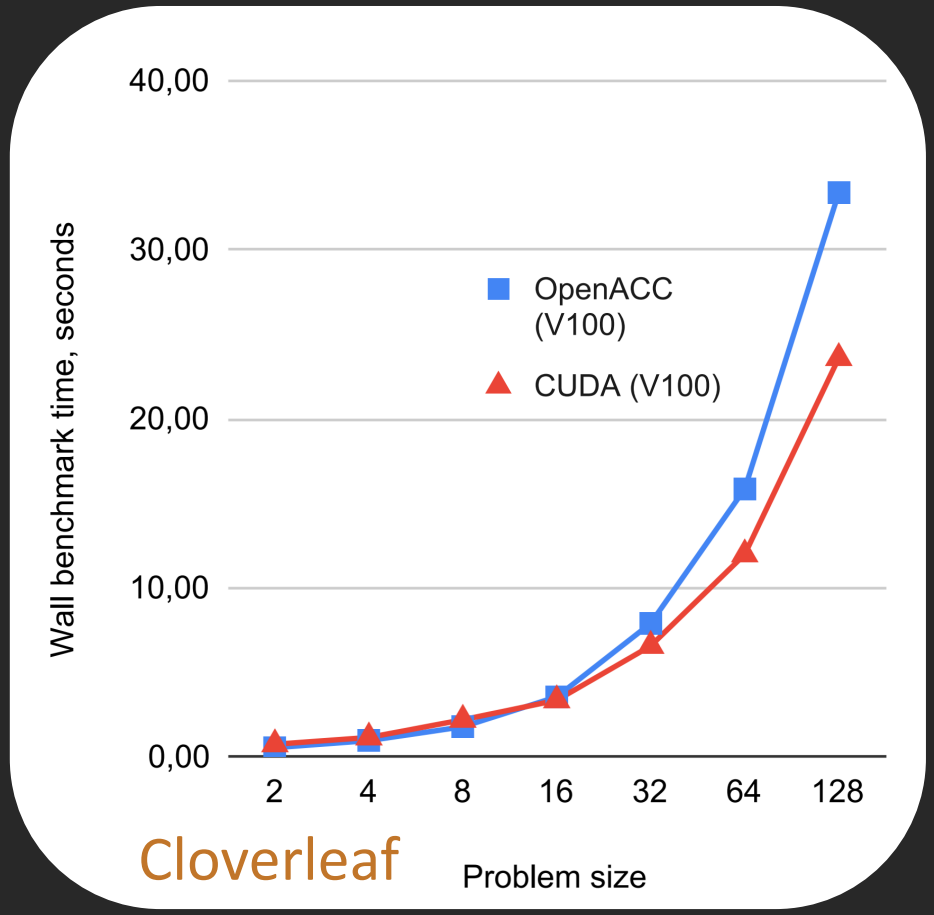
Our strategy: Take control of your code

- **Abstractions**: Avoid. Often unnecessary. *Slow*.
- **Directives**: we use OpenACC
  - Performance-competitive with CUDA/HIP, control if needed, *portable*
- **Meta-programming**: we use Fypp (Python-based)
  - Provide compiler with run-time constants, expose optimizations

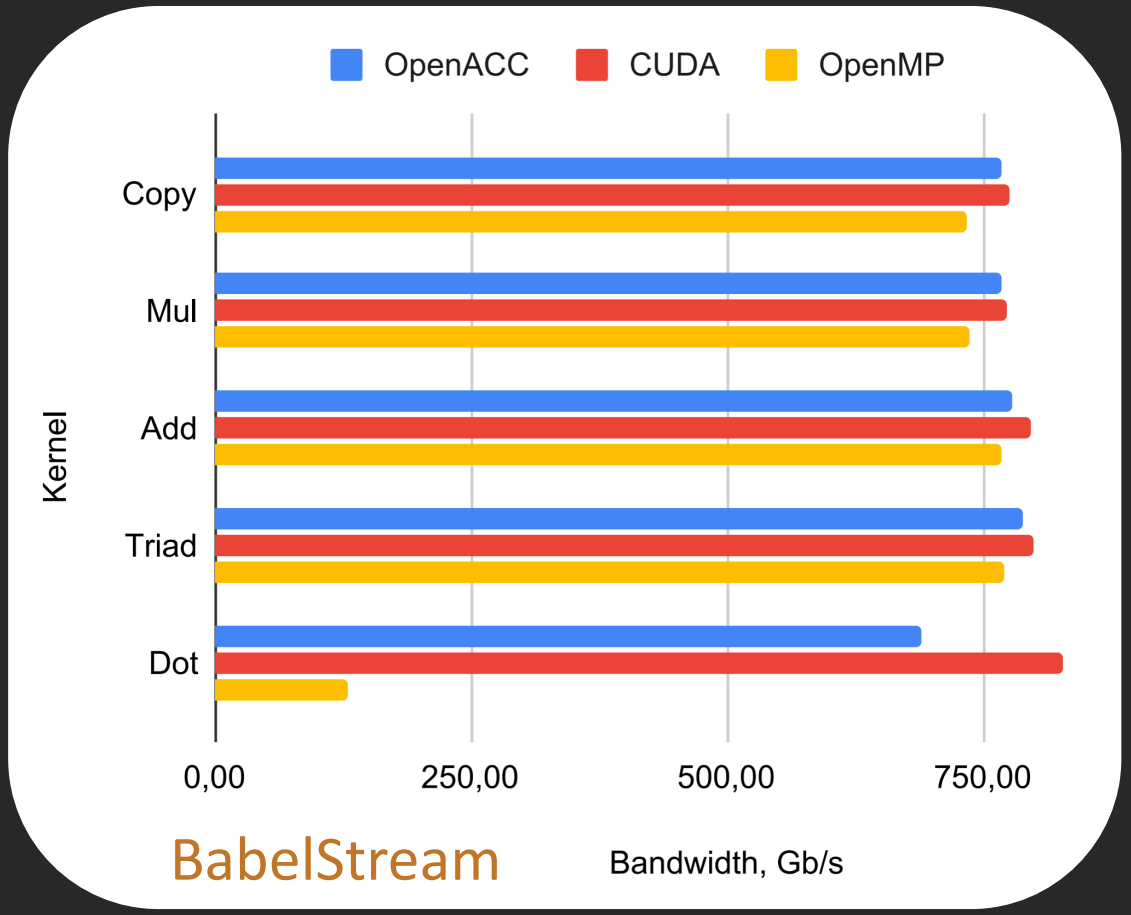
My view: Compilers are good, use them.

Tuned backends and kernel duplication → debt.

Aside: Is OpenACC that great?



Cloverleaf



BabelStream

OpenACC competitive [!] with CUDA (**nvhpc** compilers)

Khalilov & Tomoveev *J. Phys. Conf.* (2021)

Snippet: Mostly benign Fortran

Python!

GPU collapsed loops

Sufficient GPU work

Hot loop!

```
#:for WENO_DIR, XYZ in [(1, 'x'), (2, 'y')]
if (weno_dir == ${WENO_DIR}$) then
  !$acc parallel loop collapse(2) default(present)
  do k = dir2%end, dir2%end
    do j = dir1%beg, dir1%end
      !$acc loop seq
      do i = 1, num_eq
        dvd(0) = v_rs_ws_${XYZ}$(j + 1, k, i) &
                 - v_rs_ws_${XYZ}$(j, k, i)
        dvd(-1) = v_rs_ws_${XYZ}$(j, k, i) &
                  - v_rs_ws_${XYZ}$(j - 1, k, i)
      end do
    end do
  end do
end if
```

https://github.com/MFlowCode

Snippet: Manual routine inlining

Define macro →

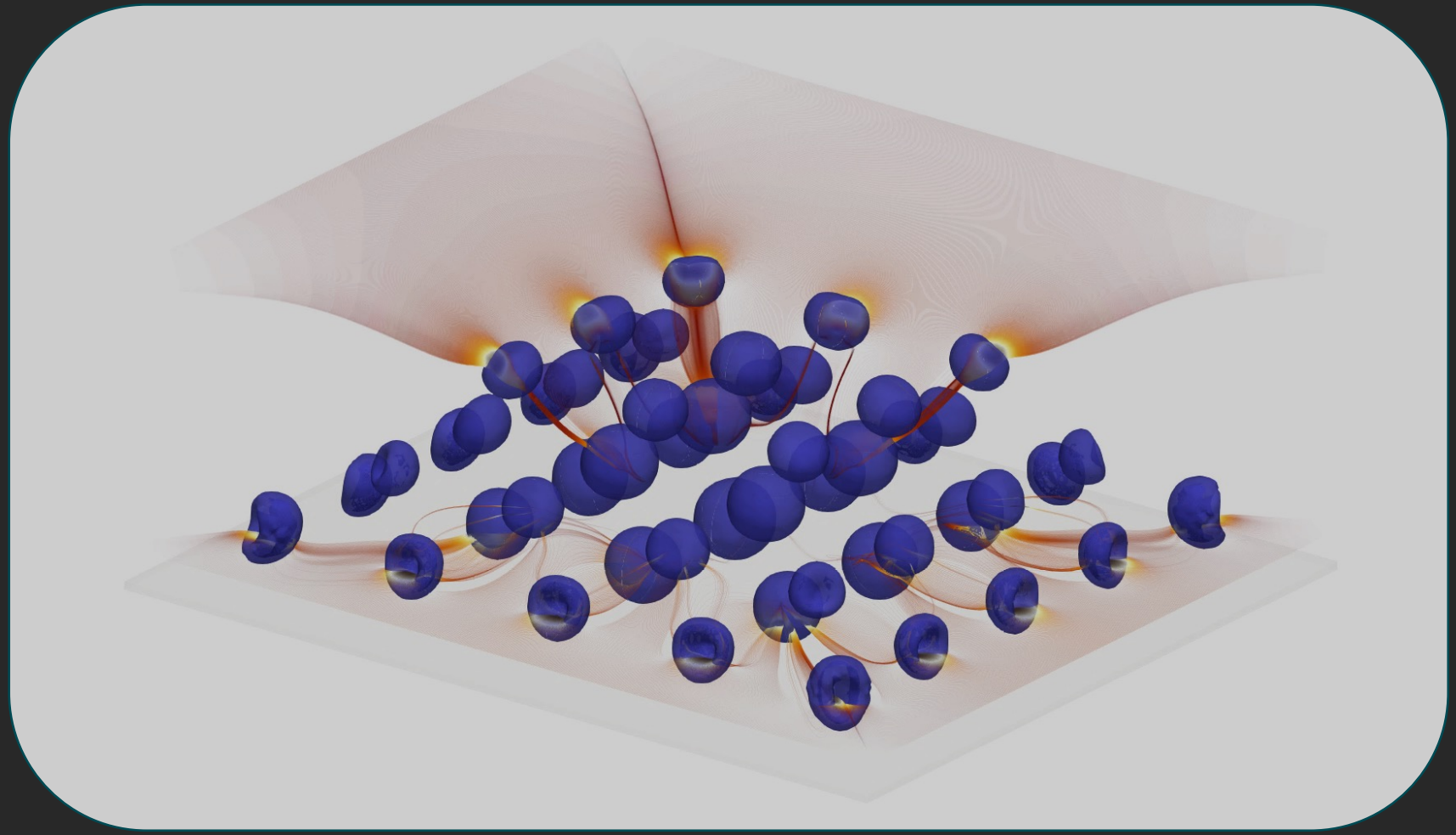
```
#:def s_compute_speed_of_sound()
    subroutine s_compute_speed_of_sound(pres, rho, adv, c)

        real(kind(0d0)), intent(IN) :: pres, rho
        real(kind(0d0)), dimension(num_fluids), intent(IN) :: adv
        real(kind(0d0)), intent(OUT) :: c
        integer :: q

        c = 0d0
        !$acc loop seq
        do q = 1, num_fluids
            c = c + adv(q)*(1d0/gammas(q) + 1d0)* &
                (pres + pi_infs(q)/(gammas(q) + 1d0))
        end do
        c = c/rho
    end subroutine s_compute_speed_of_sound
#:enddef
```

Inline as needed →

```
@:s_compute_speed_of_sound()
```

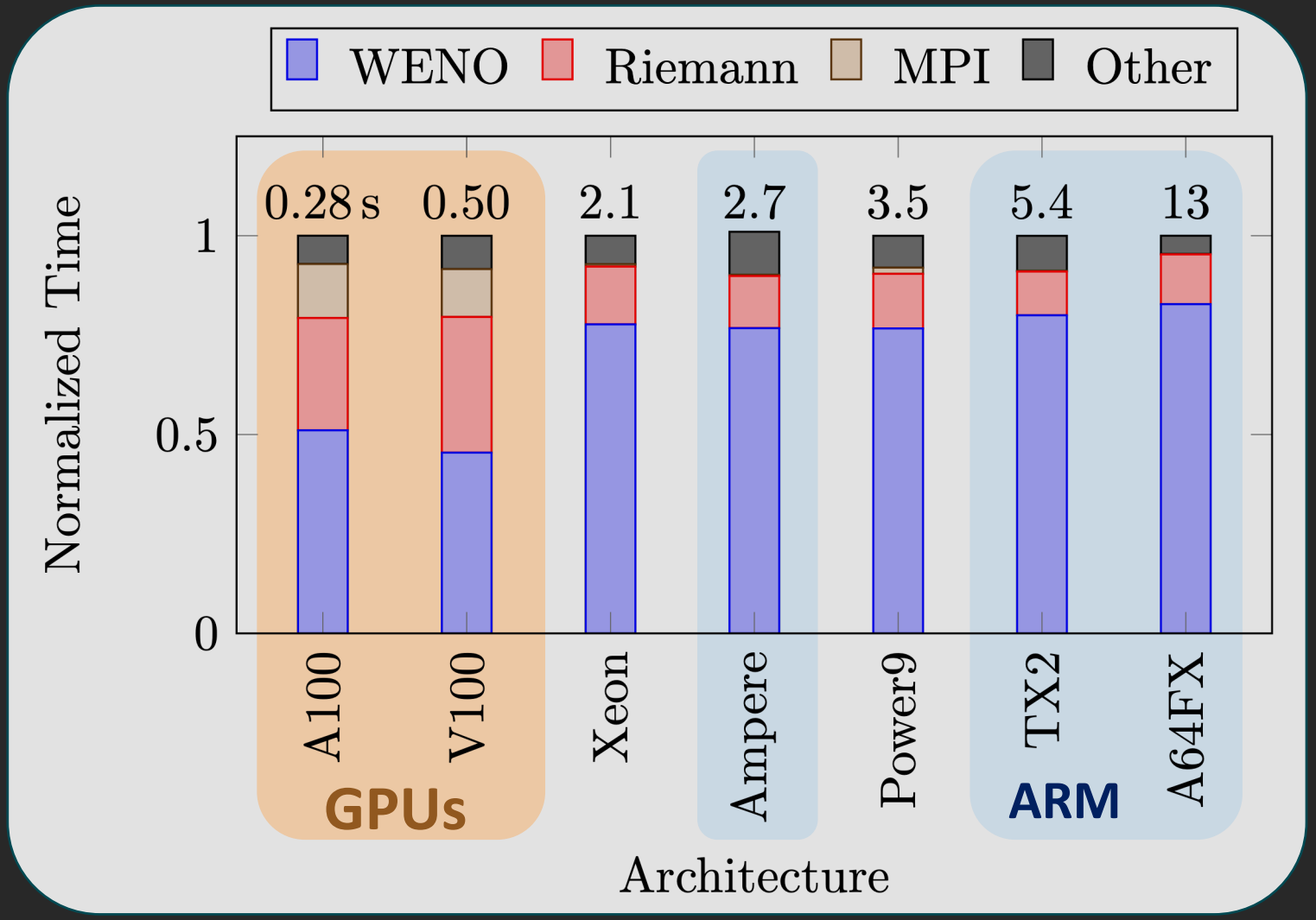
https://github.com/MFlowCode

A test case



Example simulation: Shock–bubble–wall collapse dynamics

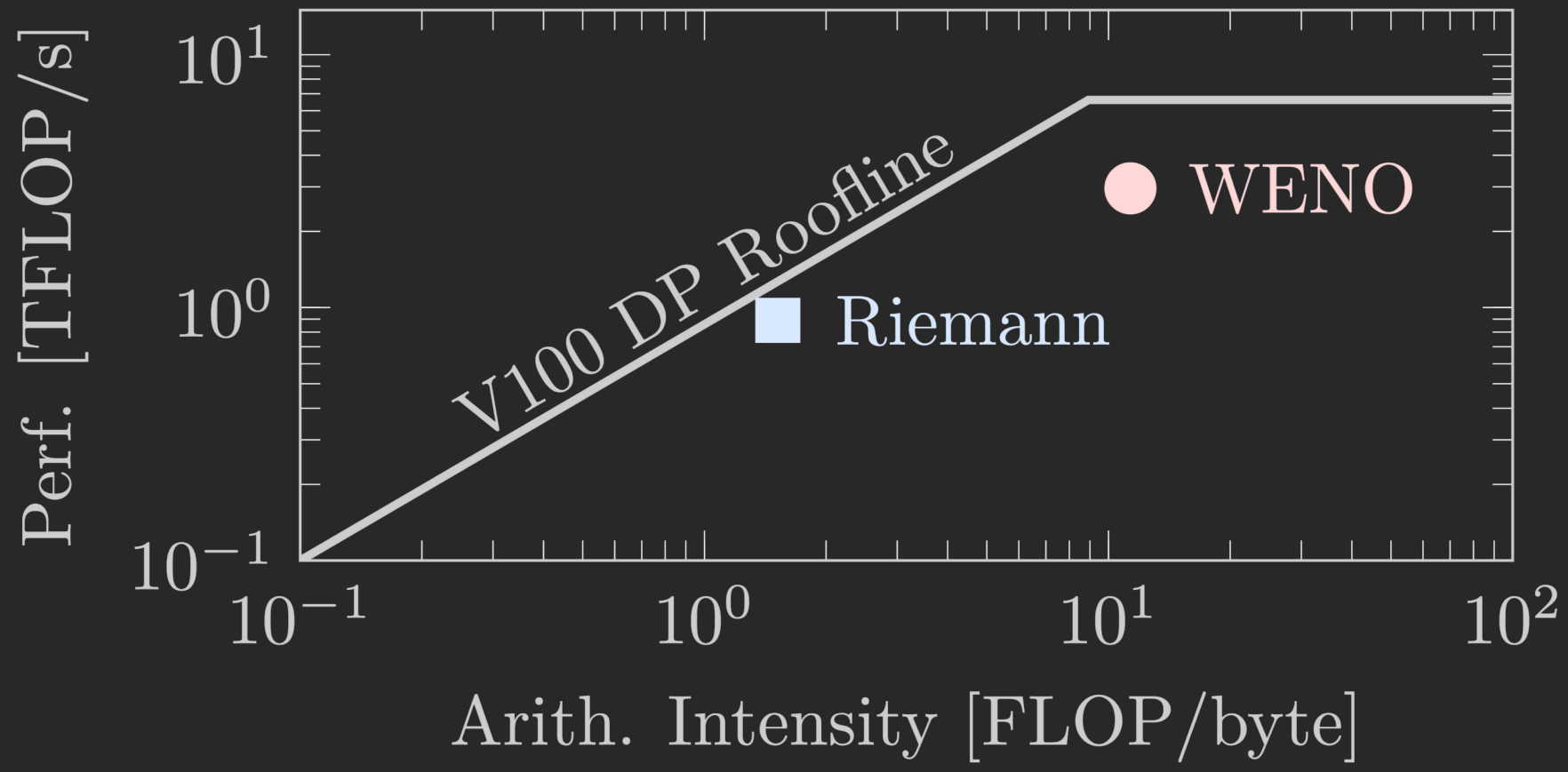
Results: Architecture-friendly



- 1 A100 ≈ 8 Xeon CPUs
  ≈ 10x factor earlier!
  ≈ 50% peak FLOPS
- ARM CPUs competitive
  Ready for upcoming arch. like NV GraceHopper
- MPI time modest
  Due to RDMA, CUDA-aware MPI
  (OpenACC can indeed do this)

Elwasif, B., et al. *HPC Asia* (2023)

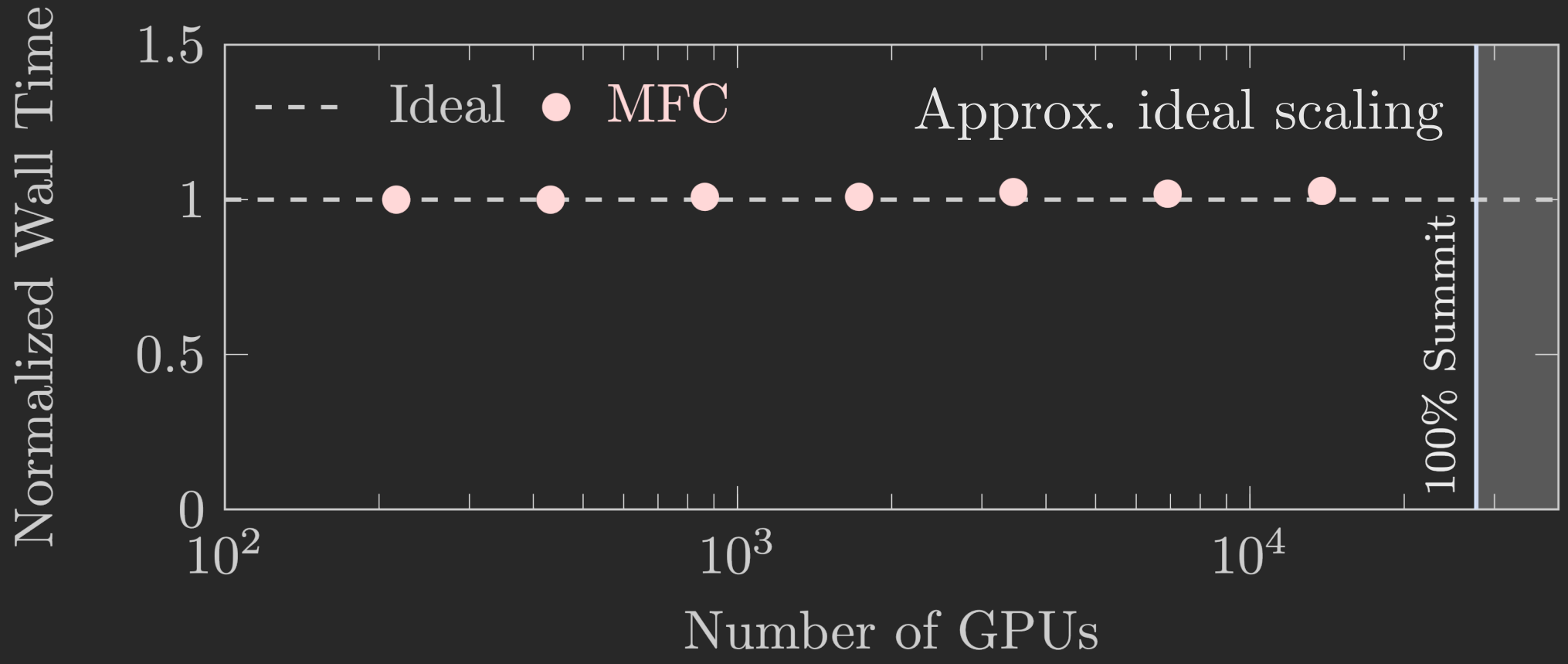
Results: Efficient use of GPU resources



Expensive kernels near FMA roofline

Radhakrishnan et al. arXiv (2023)

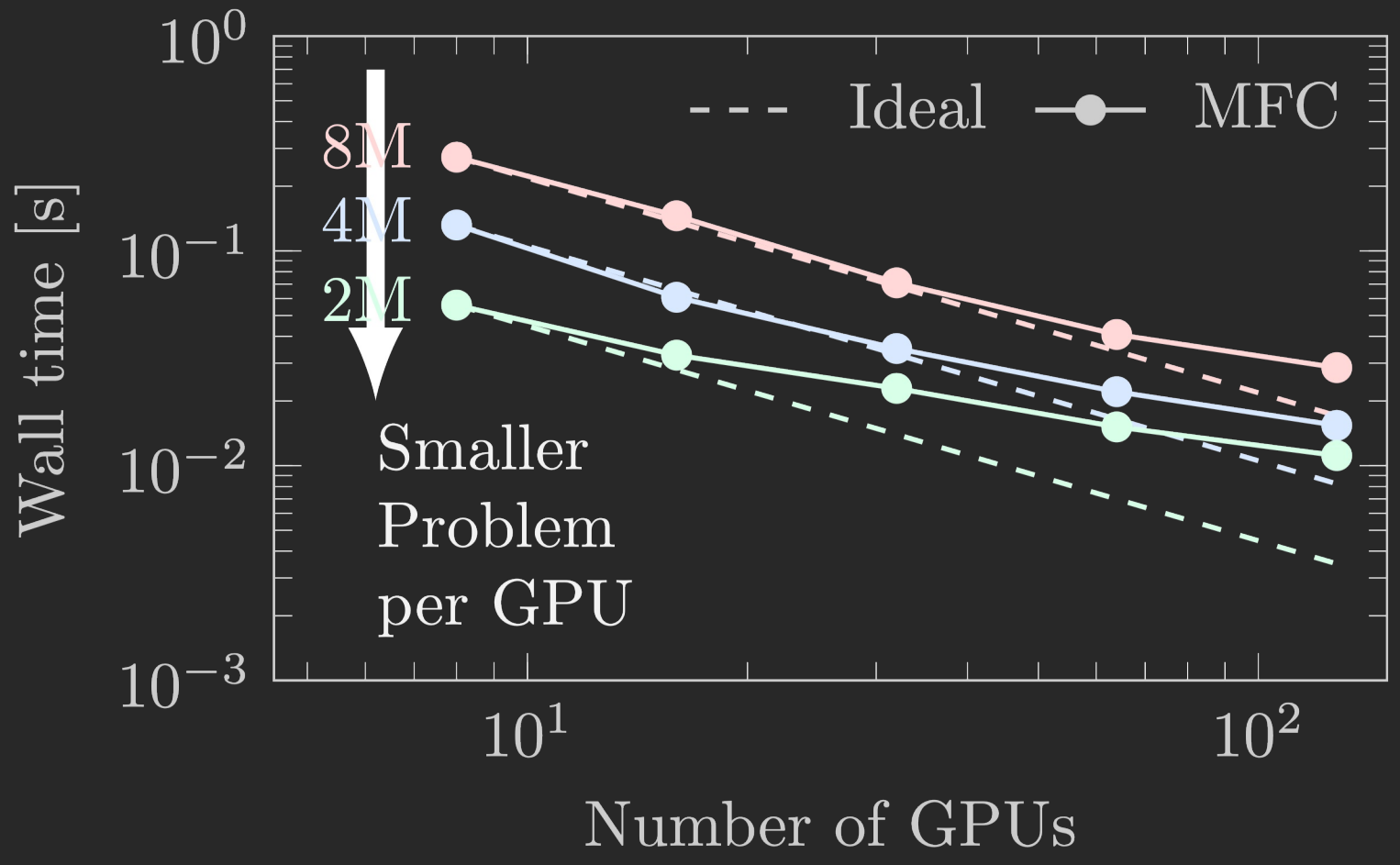
Results: Weak scaling on **Summit**



Punchline: 100 PFLOPS, 100B grid points, 0.1 s/time-step

Radhakrishnan et al. arXiv (2023)

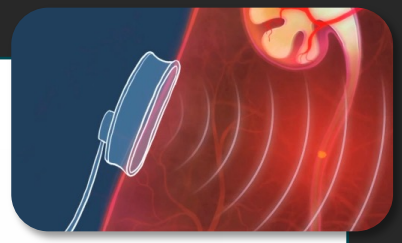
Results: Strong scaling

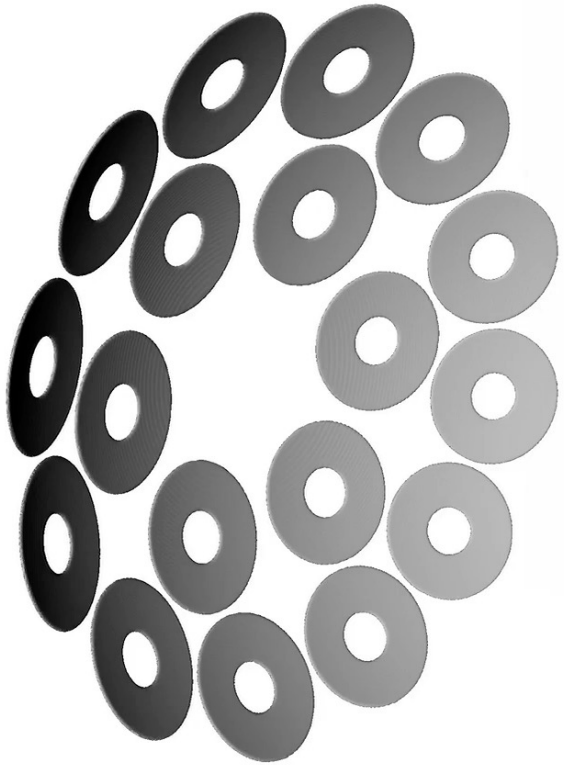


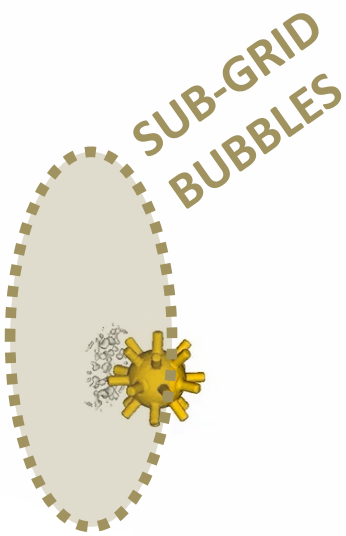
Radhakrishnan et al. arXiv (2023)

Example simulations

Kidney stone treatment

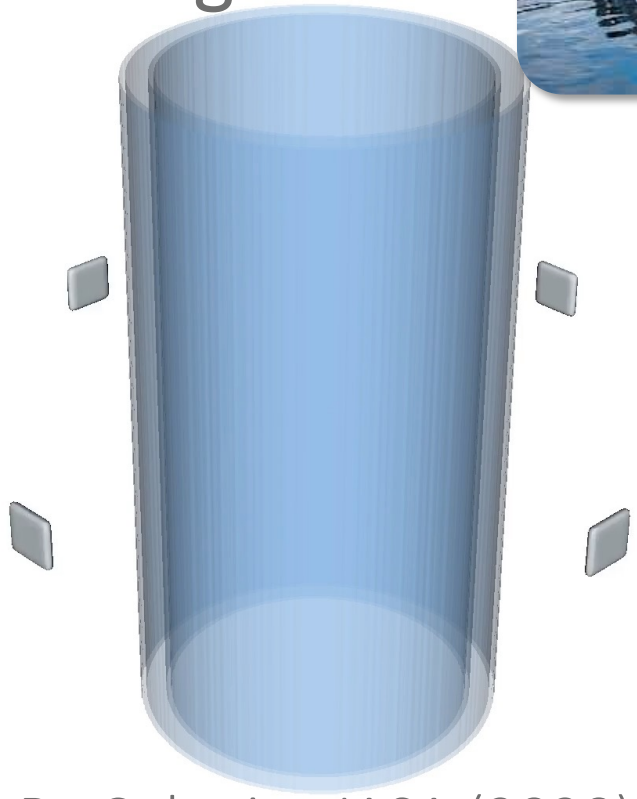






Credit: Kazuki Maeda

Feeding whales



B., Colonius JASA (2020)

Maintenance strategy

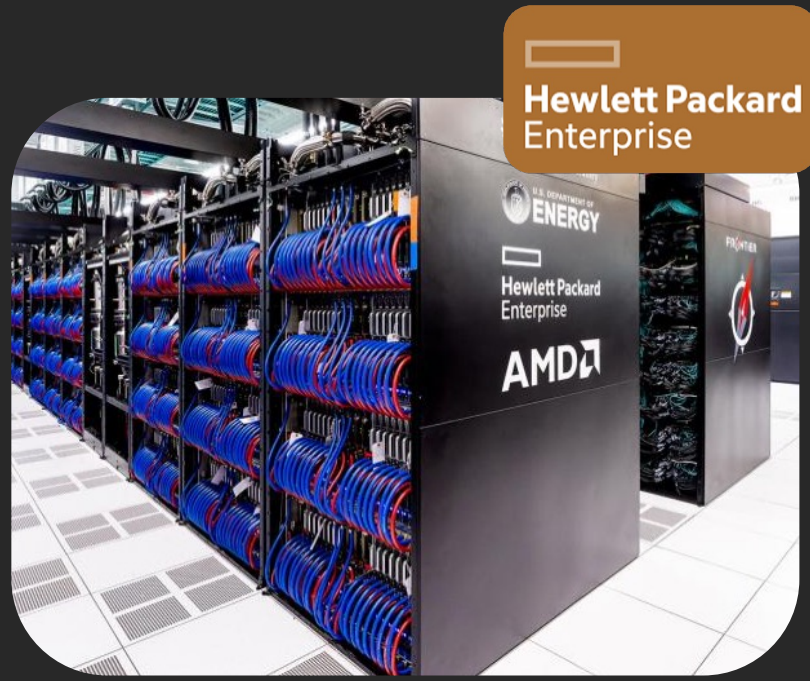
We maintain

- **Correctness:** Maintain easy-to-write *tests*, touch all code features
- **Performance:** Avoid unexpected slow-downs via *benchmarking*
- **Documentation**: All doc. writing happens in one place

By

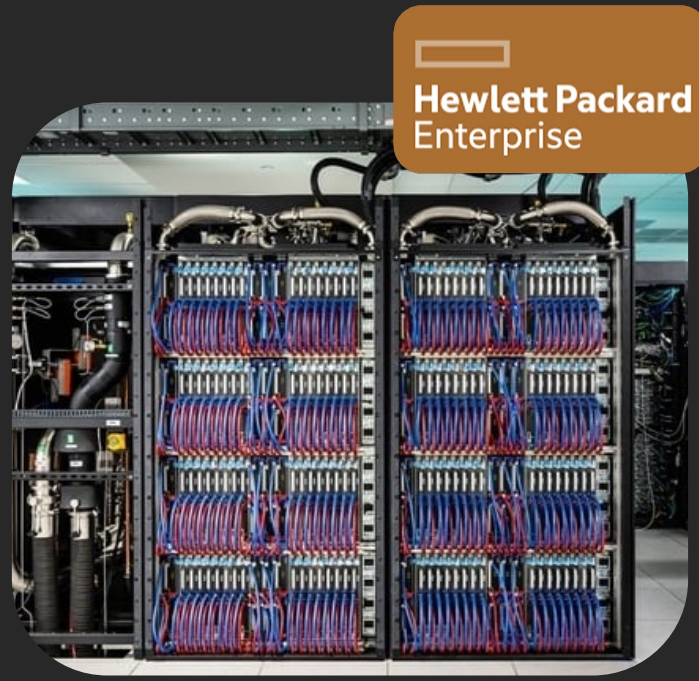
- **CI**: Do *every* commit, automatically, with diff. compilers/hardware
- **CD**: Deploy documentation and code to Docker, Website, …

So… what about CORAL-2?



ORNL Frontier (AMD MI250X)

LLNL El Capitan (AMD MI300)



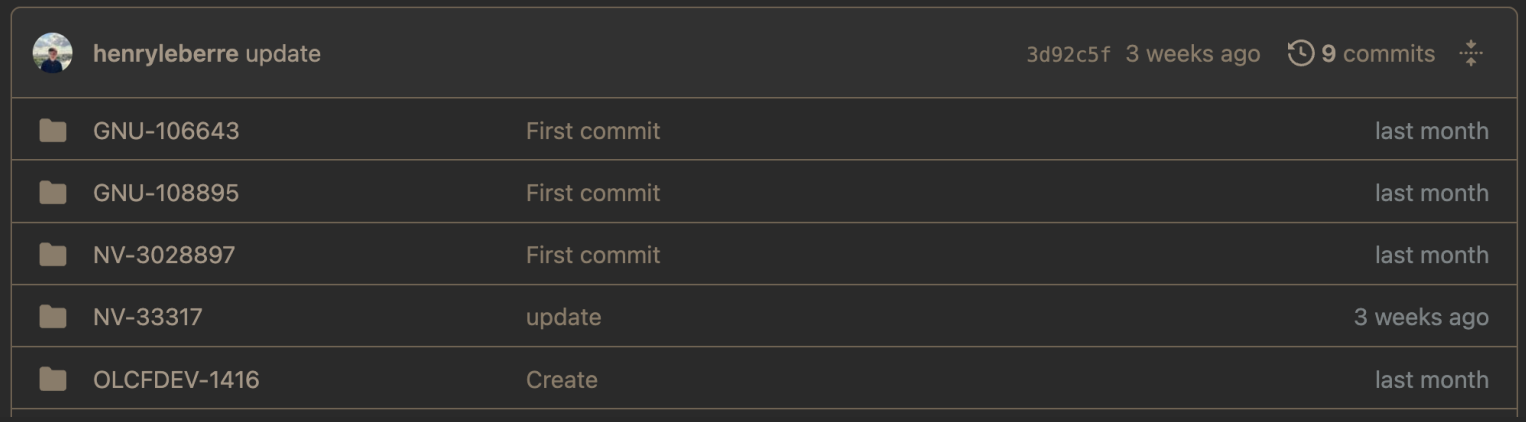
ANL Sunspot (Intel PV)
[Aurora early-readiness]

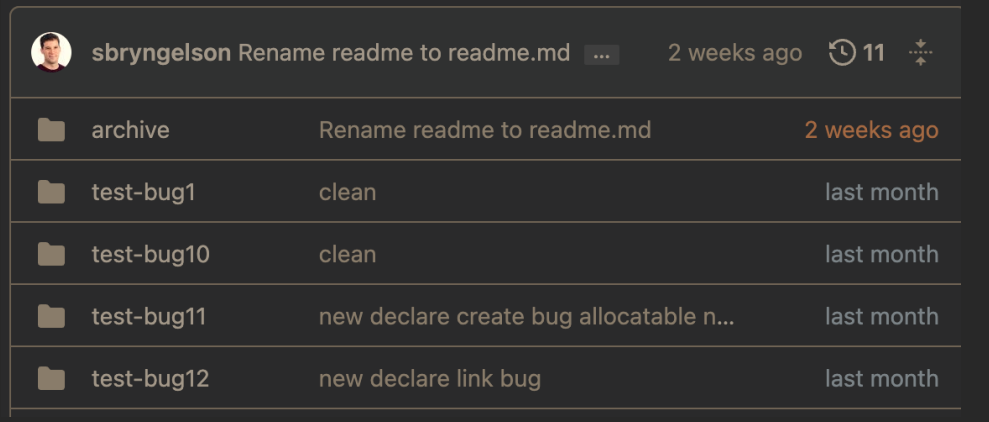
New hardware means… we adapt… to HPE!

So… what about CORAL-2?

Right now: On OLCF **Crusher** (**Frontier** early-readiness)

- OpenACC on AMD MI200-series
  - GNU (v13): spotty coverage, fixes a bit slow, open (GNU Bugzilla)
  - Cray CCE (v15): better coverage, faster fixes, proprietary (OLCF Office Hours)
- Finding, filing, following CCE/GNU compiler bugs

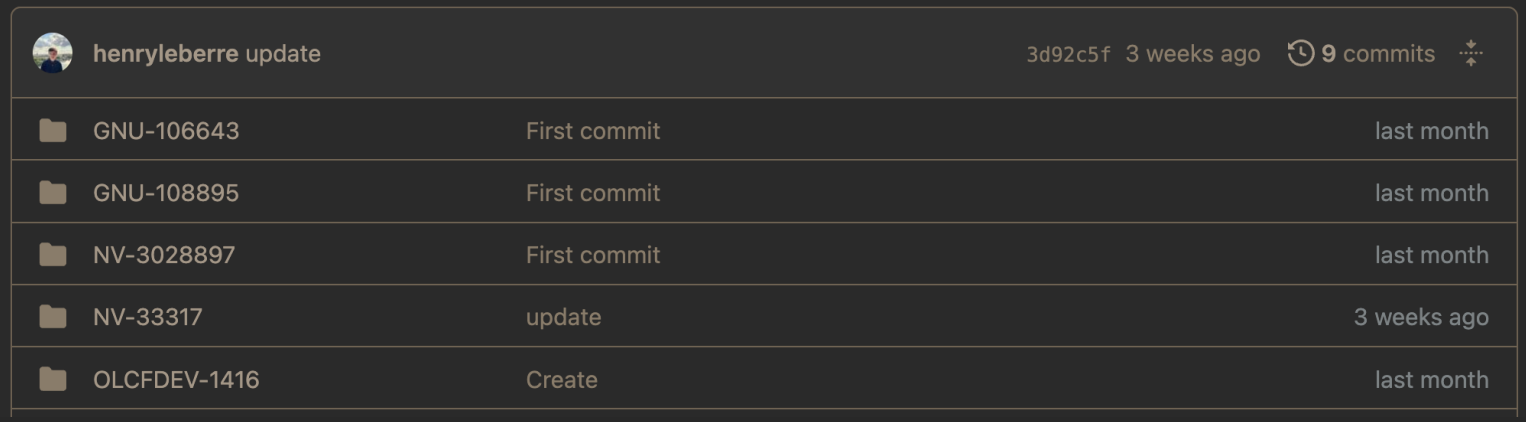


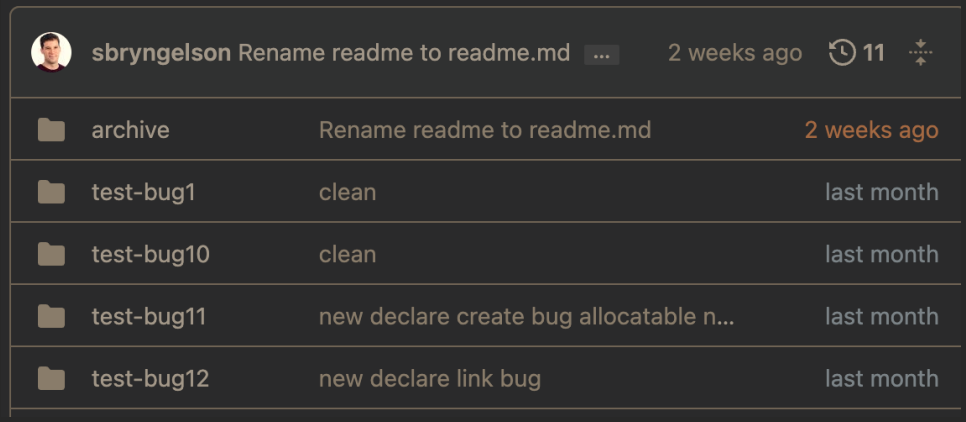


So… what about CORAL-2?

Note: New compiler support for OpenACC and OpenMP offload

- Many compilers using shared kernel generator
- So, OpenACC and OpenMP often *share* bugs
- Can’t (always) fix problems by switching to OpenMP





Workarounds exist!

Known Issues

Open Issues

Running

OLCFDEV-1684: libfabric CXI provider fails to init in single-node runs on Frontier

It was reported to us that single-node jobsteps cannot initialize the libfabric CXI provider.

```
FATAL ERROR (proc 1): in gasnetc_ofi_init() at[..]/gasnet_ofi.c:1164: fi_domain failed: -38(Function not implemented)
```

This issue has been fixed in Slurm 23.02 which we are currently testing and expect to deploy on the system as soon as fully verified.

OLCFDEV-1510: MPI_Comm_spawn function and server not working

The MPI_Comm_spawn function within mpich and the spawn server for multi-node spawning are not working on Crusher.

Error reported:

```
MPICH ERROR [Rank 0] [job id 255254.0] [Thu Jan 26 17:41:16 2023] [crusher189] - Abort(403262725) (rank 0 in comm 0): Fatal error in PMPI_Comm_spawn:
PMPI_Comm_spawn(140): MPI_Comm_spawn(cmd="child", argv=(nil), maxprocs=4, MPI_INFO_NULL, root=0, MPI_COMM_NULL, intercomm=0x7fffffff690c, errors=(nil)
PMPI_Comm_spawn(80).: Null communicator
```

OLCFDEV-496: srun -m plane=N distribution results in errors or incorrect distribution

Use of the *-m plane=N* distribution option with *srun* could result in errors or incorrect distribution of tasks.

Certain combinations of values could result in an error like:

```
$ srun  -n 108  --ntasks-per-node=54 -d 1  -m plane=54  -c 1 -A NNNNN -N2 -t2 /bin/hostname
srun: job 53451 queued and waiting for resources
srun: job 53451 has been allocated resources
srun: error: Unable to create step for job 53451: More processors requested than permitted
```

There may also be other scenarios unrelated to the *plane* issue that could produce a similar error. We are continuing to investigate this issue. If you encounter this error message or a similar error with a scenario that is not documented here, please report it by contacting OLCF.

Other combinations may run without error but result in an incorrect distribution, for example:

```
$ srun  -n 96 --ntasks-per-node=48 -m plane=48 -c 1 -A NNNNN -N2 -t2 /bin/hostname | uniq -c
srun: job 53453 queued and waiting for resources
srun: job 53453 has been allocated resources
     32 crusher003
     64 crusher001
```

There is no known workaround at this time, other than considering other ways to phrase your jobstep request.

OLCFDEV-937: GPU-Aware MPI hang when *rocm* not loaded

If using GPU-Aware MPI on Crusher, users will experience hangs in their application at runtime if the *rocm* modulefile is not loaded.

If you encounter a hang, please double check the modules you have loaded inside your job allocation (i.e., *module -t list*) to ensure the *rocm* version used for building is loaded at runtime.

Programming

OLCFDEV-689: Silent hangs in applications using HIP cooperative groups or the rocGDB debugger

Users in ROCm 4.5 may observe silent hangs in applications using HIP cooperative groups or the rocGDB debugger.

The MI250X device supports more CUs than can be used in a cooperative dispatch. Setting the environment variable *HSA_COOP_CU_COUNT* to 1 will cause ROCr to return the correct CU count for cooperative groups through the *HSA_AMD_AGENT_INFO_COOPERATIVE_COMPUTE_UNIT_COUNT* attribute of *hsa_agent_get_info()*. Future ROCm releases will make *HSA_COOP_CU_COUNT=1* the default.

From ROCm 5.2 onward on the MI250X GPU, *hipDeviceProp_t.multiProcessorCount* will return 96 for the number of compute units (CUs), not 110. *hipDeviceGetAttribute(hipDeviceAttributeMultiprocessorCount)* will return 96, not 110 CUs. See also: https://rocmdocs.amd.com/en/latest/Current_Release_Notes/Current-Release-Notes.html#new-environment-variable

If you experience deadlocks or significant performance degradations, please submit a ticket by emailing help@olcf.ornl.gov.

OLCFDEV-518: Querying AMD device name returns an empty string

When querying *deviceProp.name* or using *rocminfo* while using ROCm 4.5.0 or later, users will see an empty string. The vendor has identified a fix for this issue and we expect it to be available in a future ROCm release.

Additional information can be found at: [https://rocmdocs.amd.com/en/latest/Current_Release_Notes/Current-Release-Notes.html#clinfo-and-rocminfo-do-not-display-marketing-name]

Open Issues w/Workaround

Running

OLCFDEV-1655: Occasional seg-fault during MPI_Init

Occasionally, some applications may encounter a segmentation fault during MPI_Init. This is sometimes accompanied by messages about *pmi_allgather*, MPICH, or CXI failures. Some examples of this failure's signature in stdout/stderr:

```
srun: error: frontier00572: task 1918: Segmentation fault (core dumped)
srun: launch/slurm: _step_signal: Terminating StepId=1291836.0
slurmstepd: error: *** STEP 1291836.0 ON frontier00001 CANCELLED AT 2023-03-31T04:21:06 ***
```

or

```
srun: error: frontier00175: task 314: Segmentation fault
srun: launch/slurm: _step_signal: Terminating StepId=1291835.0
slurmstepd: error: *** STEP 1291835.0 ON frontier00001 CANCELLED AT 2023-03-31T04:18:45 ***
Fri Mar 31 04:18:45 2023: 'PE_9168]:inet_recv:inet_recv: unexpected EOF Success
Fri Mar 31 04:18:45 2023: [PE_9168]:_pmi_network_allgather:_pmi_inet_recv from controller failed
Fri Mar 31 04:18:45 2023: [PE_9168]:_pmi_allgather:_pmi_network_allgather failed...
slurmstepd: error: Failed to destroy CXI Service ID 5 (cxi2): -16
slurmstepd: error: Failed to destroy CXI Service ID 5 (cxi2): -16
slurmstepd: error: switch_g_job_postfini: Device or resource busy
```

or

```
Fri Mar 31 05:03:19 2023: [PE_552]:inet_recv:inet_recv: recv error (fd=4) Connection reset by peer
Fri Mar 31 05:03:19 2023: [PE_552]:_pmi_network_allgather:_pmi_inet_recv from controller failed
Fri Mar 31 05:03:19 2023: [PE_552]:_pmi_allgather:_pmi_network_allgather failed
Fri Mar 31 05:03:19 2023: [PE_576]:inet_recv:inet_recv: recv error (fd=4) Connection reset by peer
Fri Mar 31 05:03:19 2023: [PE_576]:_pmi_network_allgather:_pmi_inet_recv from controller failed
Fri Mar 31 05:03:19 2023: [PE_576]:_pmi_allgather:_pmi_network_allgather failed
MPICH ERROR [Rank 0] [job id 1291841.0] [Fri Mar 31 05:03:19 2023] [frontier00073] - Abort(1616271) (rank 0 in comm 0): Fatal error in PMPI_Init: Othe
MPIR_Init_thread(170).......:
MPID_Init(501)..............:
MPIDI_OFI_mpi_init_hook(805):
MPIDU_bc_table_create(204) .: PMI Allgather failed: aborting job:
```

docs.olcf.ornl.gov/systems/crusher_quick_start_guide.html#known-issues

Making workarounds workable

Allocation →

Confusing compiler workaround →

```
@:ACC_SETUP_VFs(dq_prim_qp)
```

Fypp-powered workarounds

```
#:def ACC_SETUP_VFs(*args)
    block
        integer :: macros_setup_vfs_i

        @:LOG({'@:ACC_SETUP_VFs(${', '.join(args)}$)'})

        #:for arg in args
            !$acc enter data copyin(${arg}$)
            !$acc enter data copyin(${arg}$%vf)
            if (allocated(${arg}$%vf)) then
                do macros_setup_vfs_i = lbound(${arg}$%vf, 1), ubound(${arg}$%vf, 1)
                    if (associated(${arg}$%vf(macros_setup_vfs_i)%sf)) then
                        !$acc enter data copyin(${arg}$%vf(macros_setup_vfs_i))
                        !$acc enter data create(${arg}$%vf(macros_setup_vfs_i)%sf)
                    end if
                end do
            end if
        #:endfor
    end block
#:enddef
```

Take away

At least for compressible multi-phase CFD, current leadership-class performance without too much suffering

NVHPC awesome, better support needed for CCE/GNU

Download me: mflowcode.github.io

Thank you!

This work was supported by many students, faculty, scientists

Download me: mflowcode.github.io