The exaFOAM project: bringing OpenFOAM to exaSCALE

Exploitation of Exascale Systems for Open-Source Computational Fluid Dynamics by Mainstream Industry

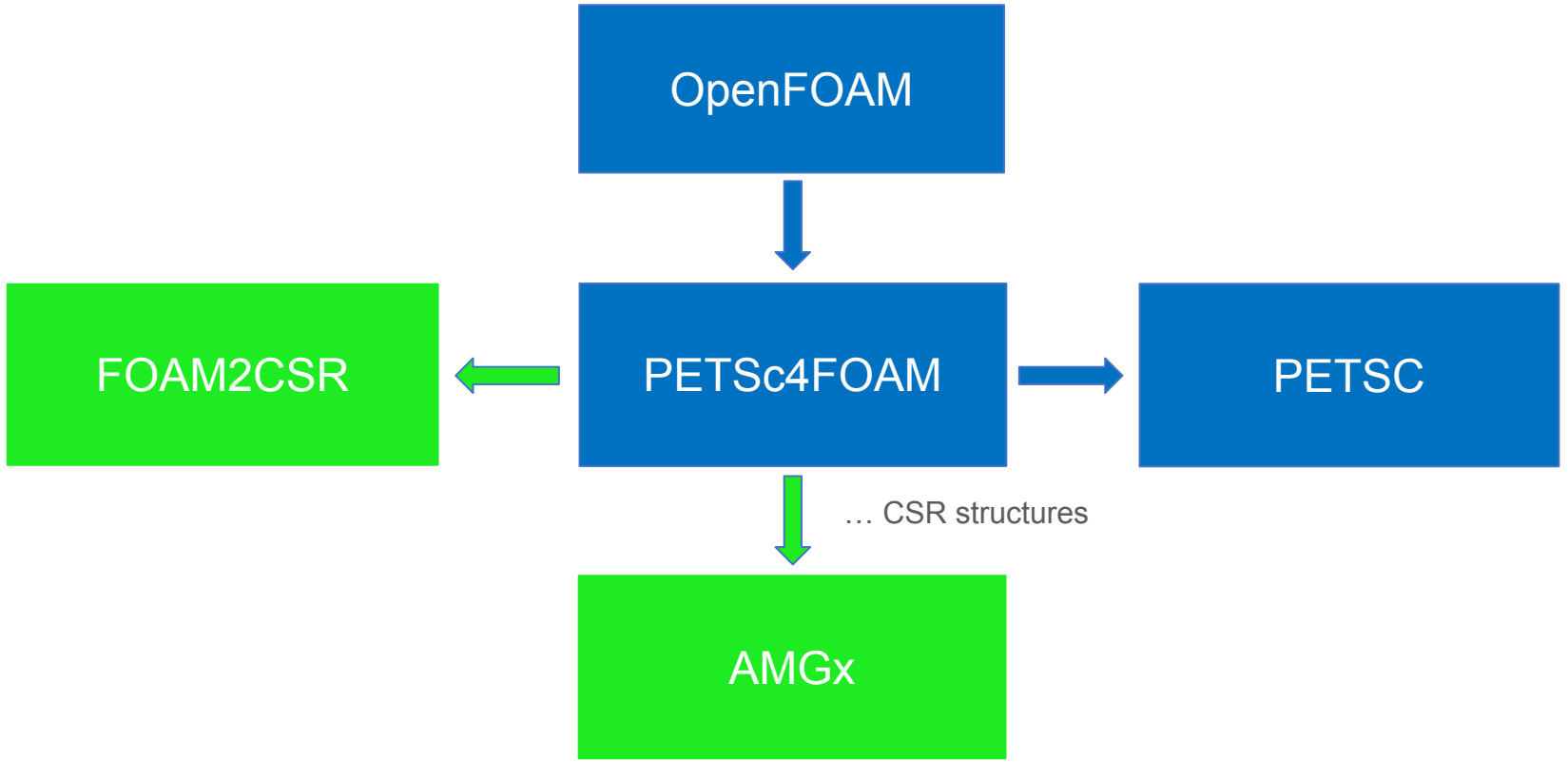
Funding Body: EuroHPC-03-2019
Call: H2020-JTI-EuroHPC-2019-1
Project Number: 9564167
Project Duration: 1st April 2021 to 31st March 2024

Hosted at:
Virtual meeting

2nd ExaFOAM workshop (Virtual)
22th March 2024

esi-group.com

Porting to Accelerators: hybrid approach



[1] M. Martineau, S. Posey, F. Spiga, AMGX GPU solver developments for OpenFOAM, 2020

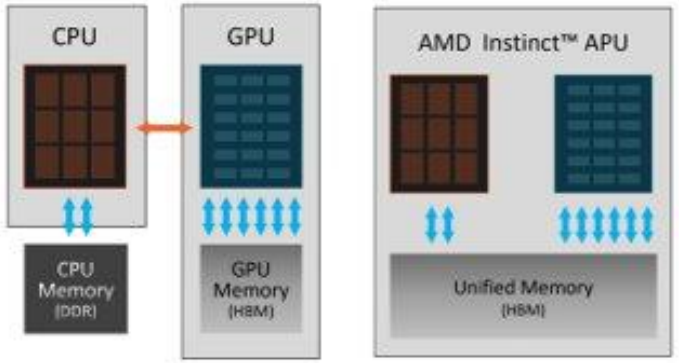
- **FOAM2CSR** has been implemented to convert the matrix from LDU-orderedCOO to CSR and to increase the amount of computational workload resident on the GPU [1]
- The **maximum speed-up** of this hybrid approach depends on how much time FOAM applications spend in the linear system solution. From experience this value is no more than **3X** on discrete GPUs (1 CPU-node vs 1 GPU-node)

exaFOAM

GPU porting approaches

We delivered a promising solution described in [1] based on **OpenMP target offloading** and **Unified Shared Memory** for APUs (e.g. AMD MI300A GPUs).

- Very few lines of code have been added (mainly #pragma directives)
- Usage of a Memory pool based on the Umpire library (OpenFOAM uses a lot of temporary arrays)
- No programming distinction between *host* and *device* memory spaces



[1] S. Tandon, L. Grinberg, D. Bercea, C. Bertolli, M. Olesen, S. Bnà and N. Malaya, Porting HPC applications to AMD Instinct TM MI300 APUs using unified memory and OpenMP, accepted, ISC High Performance 2024

exaFOAM

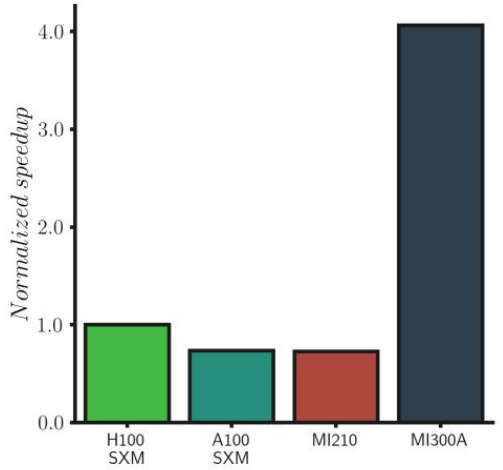
GPU porting approaches

Limitations to the openMP approach:

- Not all the compilers compile the code successfully (e.g. Nvidia compiler does not work)
- Not good performances on discrete GPUs (more than 65% of the time is spent in page migrations [1]). MI300A and GH are still not present in the current supercomputers. What about SMEs?
- This approach contains a lot of atomics operations. It relies on a good implementation of atomic on hardware (not the case with AMD MI200)

	CUDA		HIP		SYCL		Standard		OpenMP		OpenACC		Kokkos	
	C	F	C	F	C	F	C	F	C	F	C	F	C	F
NVIDIA	●	●	■	✖	▲	✖	●	●	◗	◗	●	●	▲	✖
AMD	■	✖	●	✖	■	✖	▲	✖	●	●	▲	▲	▲	✖
INTEL	■	✖	✖	✖	●	✖	◗	◗	●	●	✖	✖	▲	✖

● Full vendor support
◗ Partial vendor support
✖ Very limited or no support
▲ Comprehensive support, not from vendor
■ Indirect vendor support



[1] S. Tandon, L. Grinberg, D. Bercea, C. Bertolli, M. Olesen, S. Bnà and N. Malaya, Porting HPC applications to AMD Instinct TM MI300 APUs using unified memory and OpenMP, accepted, ISC High Performance 2024

exaFOAM

GPU porting approaches

ZeptoFoam, the successor of yoctoFOAM is a library built on top of OpenFOAM that refactors the core libraries (OpenFOAM, finiteVolume)

- Usage of a **Memory Pool** (currently only a naïve built-in implementation is available)
- Use of the **Visitor pattern** to separate, where possible, data structure from algorithms (e.g. same data structure (List of values) but several algorithms act on it (e.g. *std::memcpy* vs *cudaMemcpy* vs *hipMemCpy*)).
- Most of the kernels are confined into the **deviceFieldExecutor** class, the acronym name FOAM is still valid.
- **Explicit data managament** using the Memory Pool. This solution works fine both with the current discrete GPUs and APUs.
- All differential operators have been re-implemented in a new namespace (*devicefvm, devicefvc*)
- Rely on **third-party libraries** for linear algebra solvers and preconditioners (e.g. *AmgX* via *AmgX4Foam*). We implemented natively only *PCG* and *PBiCG* preconditioned by *Jacobi.*
- Regarding the *CUDA* implementation, we use *CUDA-aware MPI* wrapped by the *Pstream* library.

GPU porting approaches

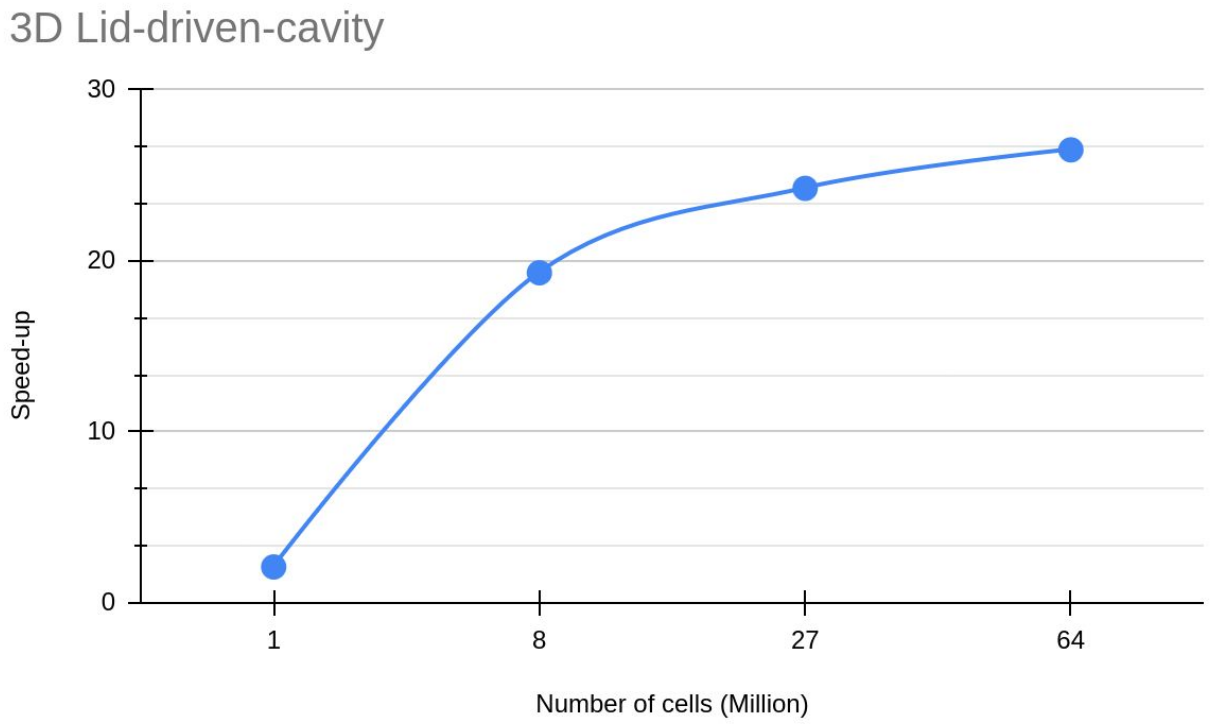
- Same **DSL**
- Implemented as a **library** built on-top of OpenFOAM, no modifications are required to the source code of OpenFOAM
- Same inputs as OpenFOAM, it relies on the same runTime Selection Table
- **Duplicate code**
- Better integration with OpenFOAM will reduce duplicate code but will probably requires modifications to the source code of OpenFOAM

```
deviceFvMatrix<vector> UEqn
(
      devicefvm::ddt(devU)
    + devicefvm::div(devphi, devU)
    - devicefvm::laplacian(devnu, devU)
);
solve(UEqn == -devicefvc::grad(devp));

// --- PISO loop
while (piso.correct())
{
    deviceVolumeField<scalar> rAU(UEqn.A().ref().reciprocal());
    deviceVolumeField<vector> HbyA(constrainHbyA(rAU*UEqn.H(), devU, devp));
    deviceSurfaceField<scalar> phiHbyA
    (
        "phiHbyA",
        devicefvc::flux(HbyA)
      // + fvc::interpolate(rAU)*fvc::ddtCorr(U, phi)
    );

    adjustPhi(phiHbyA, devU, devp);

    // Update the pressure BCs to ensure flux consistency
    constrainPressure(devp, devU, phiHbyA, rAU);

    // Non-orthogonal pressure corrector loop
    while (piso.correctNonOrthogonal())
    {
        // Pressure corrector
        deviceFvMatrix<scalar> pEqn
        (
            devicefvm::laplacian(rAU, devp) == devicefvc::div(phiHbyA)
        );

        pEqn.setReference(pRefCell, pRefValue);

        pEqn.solve(p.select(piso.finalInnerIter()));

        if (piso.finalNonOrthogonalIter())
        {
            devphi = phiHbyA - pEqn.flux();
        }
    }

    // #include "continuityErrs.H"

    devU = HbyA - rAU*devicefvc::grad(devp);
    devU.correctBoundaryConditions();
```

exaFOAM

zeptoFOAM results

3D-Lid-driven-Cavity: **saturation** tests on Leonardo (1 node with 4 GPUs A100 "Da Vinci")
PCG+Jacobi for p (fixed Iters: 250 (S), 650 (M), 1600 (L), 3000(XL)), BiCG+Jacobi for U



exaFOAM

zeptoFOAM results

3D-Lid-driven-Cavity: **saturation+strong scaling** tests on Leonardo (CPU Intel Ice Lake (32 core) vs GPU A100 "Da Vinci")
PCG+Jacobi for p (fixed Iters: 250 (S), 650 (M), 1600 (L), 3000 (XL)), BiCG+Jacobi for U

Case	32CPUs (1 CPU node)	1 GPU	2 GPUs	4 GPUs (1 GPU node)	1 CPU node/ 1 GPU node
S (1M)	3.43 [s]	1.92 [s]	1.62 [s]	1.63 [s]	2.1x
M (8M)	166.97 [s]	27.63 [s]	15.73 [s]	8.65 [s]	19.3x
L (27M)	1416.31 [s]	204.73 [s]	109.75 [s]	58.42 [s]	24.2x
XL (64M)	6250.03 [s]	N.A.	N.A.	236.10 [s]	26.5x

exaFOAM

zeptoFOAM results

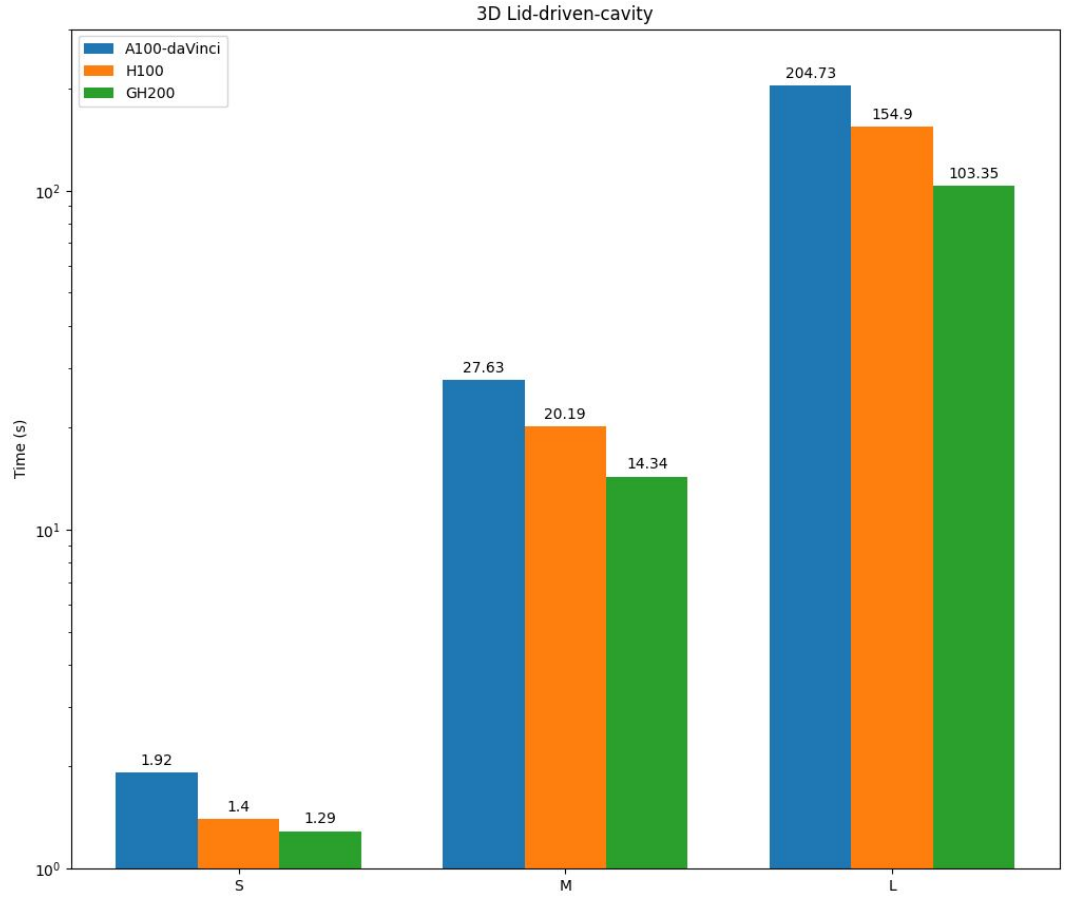
3D-Lid-driven-Cavity: **solver comparison** on Leonardo (1 GPU A100 "Da Vinci")
PCG+Jacobi vs PCG+AMG+BlockJacobi with AmgX for p (fixed iters: 250 (S), 650 (M)), BiCG+Jacobi for U

Case	PCG+Jacobi (1 GPU)	PCG+AMG (1 GPU)	Speed-up
S (1M)	4.62 [s]	3.28 [s]	1.4x
M (8M)	66.8 [s]	23.73 [s]	2.8x

exaFOAM

zeptoFOAM results

3D-Lid-driven-Cavity: **hardware comparison** between A100 "Da Vinci", H100 and GH200
PCG+Jacobi for p (fixed iters: 250 (S), 650 (M), 1600 (L)), BiCG+Jacobi for U



exaFOAM

Any Questions?

Simone Bnà

WP3 leader, CINECA

simone.bna@cineca.it

+(39) 051 6171938