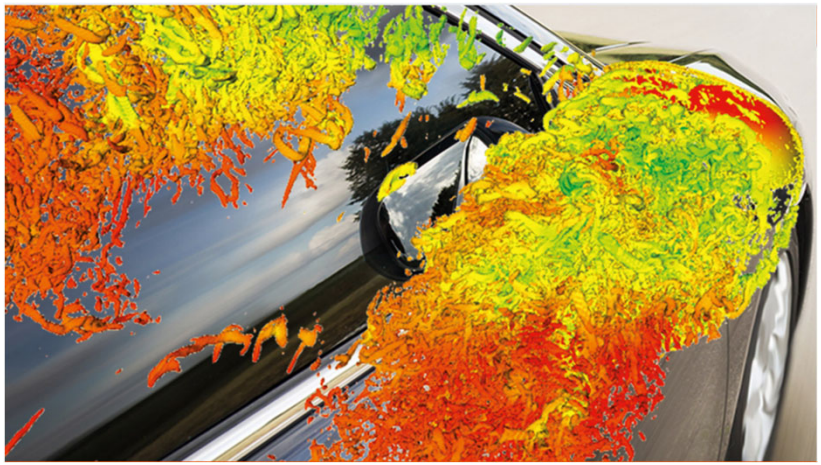


8th OpenFOAM Conference 2020

13 October 2020 - 15 October 2020
Worldwide, Online
09:00 am - 06:00 pm CEST (Central European Summer Time)

esi get it right®

OpenFOAM CONFERENCE

GPU accelerated OpenFOAM simulations with PETSc4FOAM

Oct. 14th, 2020, Industry Session

S. Zampini*, S. Bna'**, M.Valentini**, I. Spisso**

** Extreme Computing Research Center,
King Abdullah University for Science and Technology, Saudi Arabia

** SuperComputing Applications and Innovation (SCAI) Department, CINECA, Italy

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

CINECA

www.esi-group.com

Outline of the presentation

- Motivation
- PETSc https://www.mcs.anl.gov/petsc/
- PETSc4FOAM https://develop.openfoam.com/modules/external-solver/
- Examples of usage
- Test cases with accelerated workstations (V100)
- Closing remarks and future work

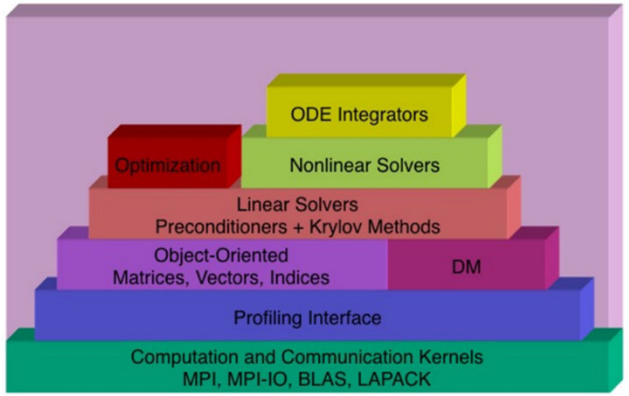
This presentation is not intended to provide definitive answers, but only to present the new tools.

Motivation

- OpenFOAM did well with native linear solvers within a homogenous hardware ecosystem and with medium scale jobs
- Now application toolboxes need to provide solutions with performance portability across different architectures
  - NVIDIA GPUs with CUDA
  - AMD GPUs with HIP
  - Intel discrete GPU with DPC++
  - ARM
  - NEC SX-Aurora TSUBASA
- And these solutions must scale from laptops to large machines
  - Fugaku: Fujitsu A64FX SoC
  - ANL A21: Cray + Intel Xeon + XE (or NVIDIA?)
  - ORNL Frontier: Cray + AMD Epyc + AMD Radeon
  - ORNL El Capitan: Cray + AMD Epyc + AMD Radeon

Portable Extensible Toolkit for Scientific computing (PETSc)

- Nonlinear and time dependent solvers
- Adopted by many applications as non-linear algebra backend
- Supports for CUDA (cuBLAS + CUSPARSE), MKL, OpenCL
- Actively developed: ~20 developers around the world
- Committed to provide support for AMD GPUs and INTEL XE within ECP
- Very powerful command line options customizations
- Interfaces with many third party libraries



PETSc4FOAM

Interfaces the OpenFOAM linear algebra solver layer with PETSc

https://develop.openfoam.com/modules/external-solver/

Credits to Mark Olesen (ESI) and Simone Bna' (CINECA)

- Easy-to-use, dictionary based customization
- Do not reinvent the wheel and maintain a single code base to have access to latest developments of the PETSc library
- Can benefit from third party libraries solvers
  - Direct solvers: UMFPACK, MUMPS, SUPERLU_DIST, STRUMPACK
  - Multigrid: HYPRE (LLNL), ML (Sandia)
  - Any other solver that PETSc may be interfaced with in the future

PETSc4FOAM example of usage

- Add to the controlDict: `libs (petscFoam);`
- Customize the solution step in the `fvSolution` dictionary:

```
solvers
{
    p
    {
        solver petsc;
        options
        {
            ksp_type cg;
            mat_type aijcusparse;
            pc_type gamg;
        }
        tolerance 1.e-04;
        relTol 0;
        maxIter 3000;
    }
...
```

```
...
    another_equation_name
    {
        solver petsc;
        options
        {
            ksp_type gmres;
            mat_type aijmkl;
            pc_type bjacobi;
        }
        tolerance ...
        relTol ...
        maxIter ...
    }
...
```

Results: experimental setting

- Small development box
  - Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz (Skylake)
  - 2 V100 GPUs
- 1 node of Marconi100 https://www.hpc.cineca.it/hardware/marconi100
  - IBM POWER9 AC922 @ 3.1 GHz
  - 4 V100 GPUs, Nvlink 2.0, 16GB
  - Ranked 9th in the 06/2020 Top500
- icoFoam, 20 time steps, 2 pressure correctors, 8 million cells
  - Fits in 1 GPU
- Cuda 10.2, PETSc version 3.14
- 3 different matrix types for PETSc: **`AIJ`**, **`AIJCUSPARSE`** and **`AIJKOKKOS`**.

Results: experimental setting (cont'd)

- Compare native OpenFOAM PCG + GAMG solver against a smoothed aggregation AMG (PCGAMG) from PETSc for pressure solve
- Easy test case for linear solves: uniform grid, laminar flow
- Timings via PETSc `-log_view` (`system/petscOptions`)
- Using the following customization for PETSc PCGAMG

```
pc_type gamg;
pc_gamg_type agg;
pc_gamg_agg_nsmooths 1;
pc_gamg_coarse_eq_limit 100;
pc_gamg_reuse_interpolation true;
pc_gamg_square_graph 10;
pc_gamg_threshold 0.0;
pc_gamg_threshold_scale 0.5;
pc_gamg_smoothprolongator_ksp_type cg;
pc_gamg_use_sa_esteig true;
```

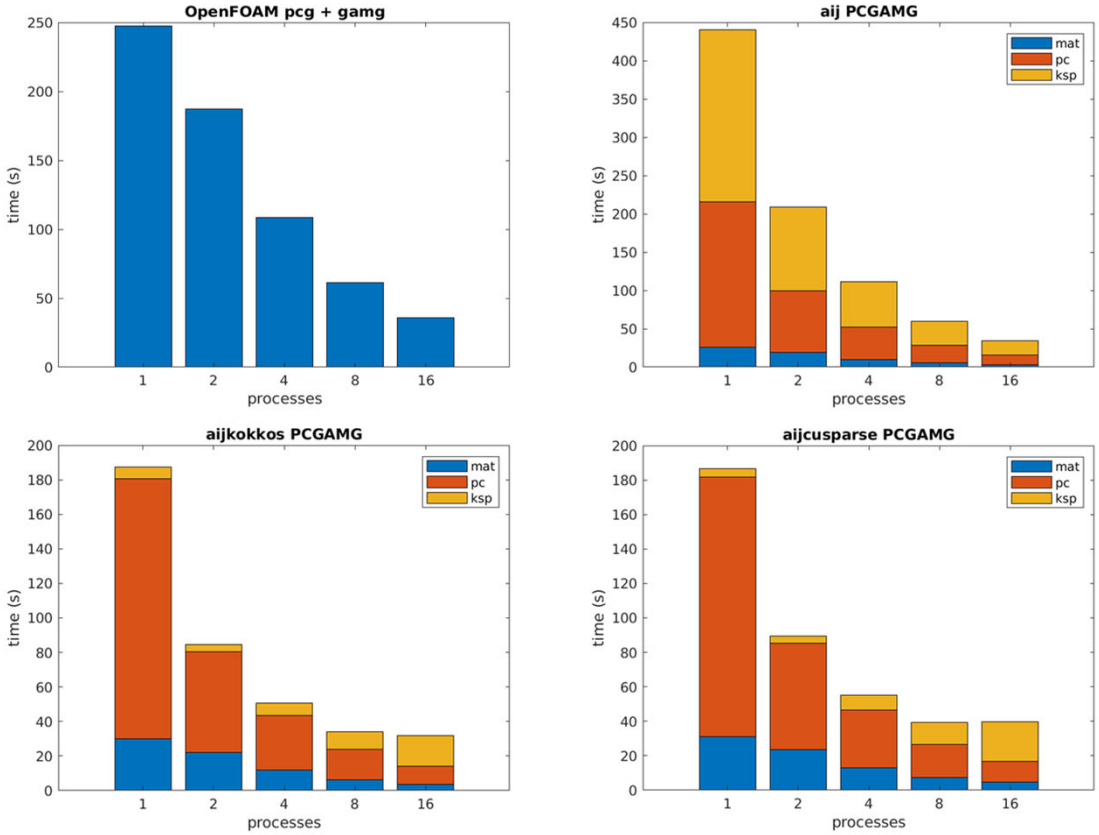
```
mg_levels_ksp_max_it 1;
mg_levels_esteig_ksp_type cg;
mg_coarse_ksp_type cg;
mg_coarse_ksp_max_it 2;
mg_levels_ksp_type chebyshev;
mg_levels_pc_type jacobi;
```

Example output from -log_view

```
Summary of Stages:   ----- Time ------  ----- Flop ------  --- Messages ---  -- Message Lengths --  -- Reductions --
                        Avg     %Total     Avg     %Total    Count   %Total     Avg         %Total    Count   %Total
 0:      Main Stage: 1.9932e+02  51.6%  4.1408e+09   1.5%  0.000e+00   0.0%  0.000e+00        0.0%  0.000e+00   0.0%
 1:      foam_p_mat: 3.0991e+01   8.0%  0.0000e+00   0.0%  0.000e+00   0.0%  0.000e+00        0.0%  0.000e+00   0.0%
 2:       foam_p_pc: 1.5064e+02  39.0%  2.5918e+10   9.5%  0.000e+00   0.0%  0.000e+00        0.0%  0.000e+00   0.0%
 3:      foam_p_ksp: 5.1290e+00   1.3%  2.4360e+11  89.0%  0.000e+00   0.0%  0.000e+00        0.0%  0.000e+00   0.0%
```

```
Event                Count      Time (sec)     Flop                              --- Global ---  --- Stage ----  Total   GPU    - CpuToGpu -   - GpuToCpu - GPU
                   Max Ratio  Max     Ratio   Max  Ratio  Mess   AvgLen  Reduct  %T %F %M %L %R  %T %F %M %L %R Mflop/s Mflop/s Count   Size   Count   Size  %F
---------------------------------------------------------------------------------------------------------------------------------------------------------------

--- Event Stage 0: Main Stage

MatMult              40 1.0 3.8947e-01 1.0 4.14e+09 1.0 0.0e+00 0.0e+00 0.0e+00  0  2  0  0  0   0100  0  0  0 10632   82664     20 1.28e+03    0 0.00e+00 100
VecSet               83 1.0 9.2592e-03 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0     0       0      0 0.00e+00    0 0.00e+00  0
VecCUDACopyTo        20 1.0 3.3929e-01 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0     0       0     20 1.28e+03    0 0.00e+00  0
VecCUDACopyFrom      60 1.0 5.2349e-01 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0     0       0      0 0.00e+00   60 3.84e+03  0

--- Event Stage 1: foam_p_mat

MatAssemblyBegin     20 1.0 1.5497e-05 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0     0       0      0 0.00e+00    0 0.00e+00  0
MatAssemblyEnd       20 1.0 1.5209e+00 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   5  0  0  0  0     0       0     20 9.18e+03    0 0.00e+00  0
MatCUSPARSCopyTo     20 1.0 1.3966e+00 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   5  0  0  0  0     0       0     20 9.18e+03    0 0.00e+00  0
```

Skylake: breakdown for pressure solve timings



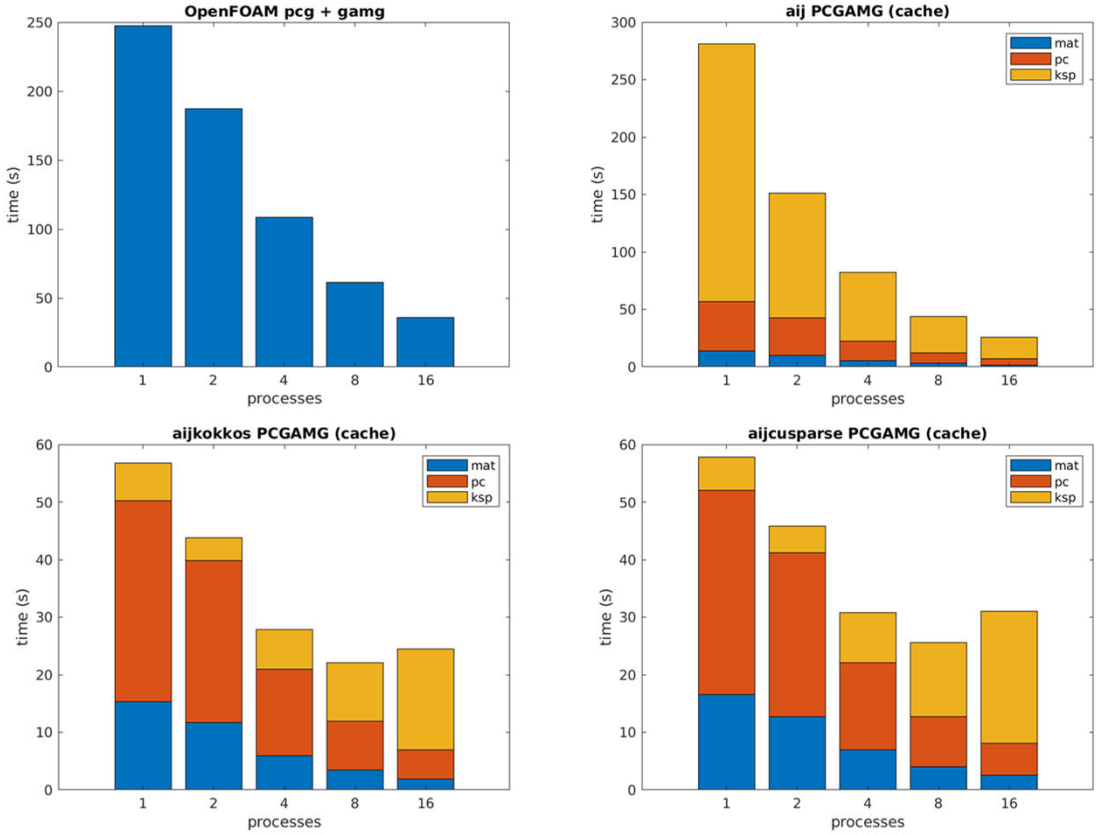
Improve performances, caching

- Pressure operator are the same for 2 consecutive runs
- Preconditioner construction can be lagged between successive time steps

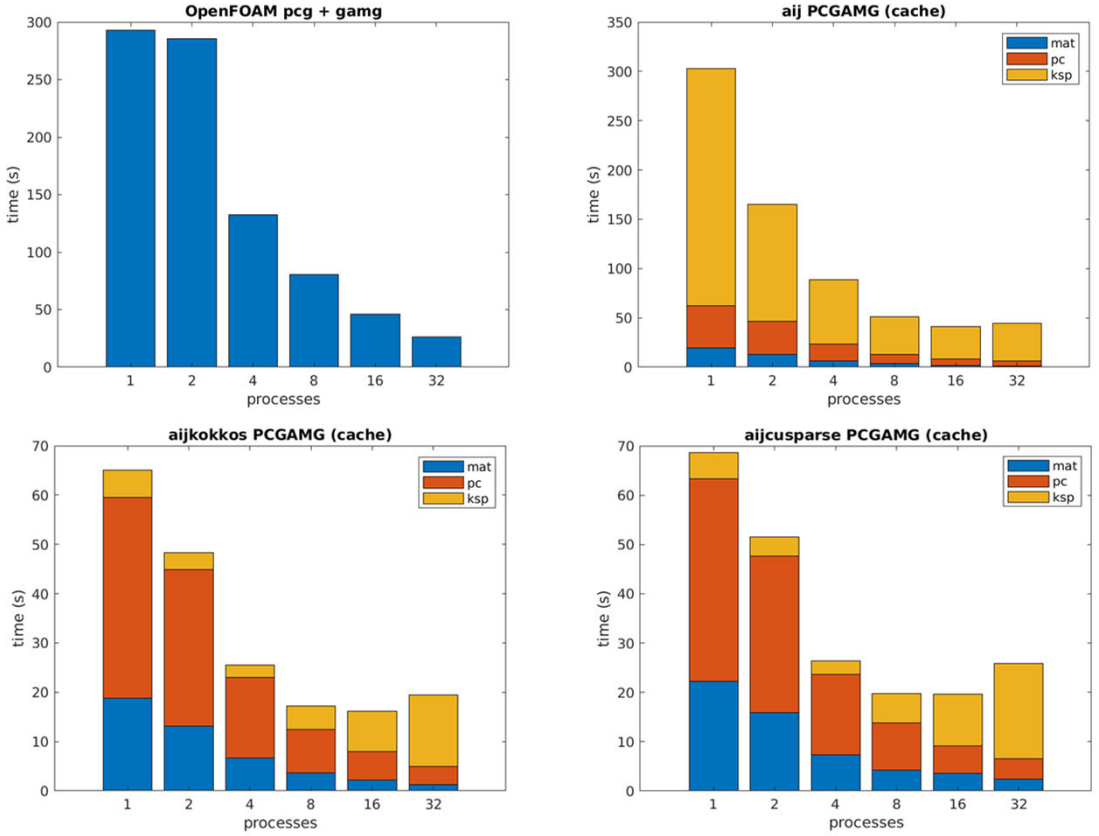
```
caching
{
  matrix
  {
    update periodic;
    periodicCoeffs
    {
      frequency  2;
    }
  }
...
```

```
...
  preconditioner
  {
    update periodic;
    periodicCoeffs
    {
      frequency  40; //setup once
    }
  }
}
```

Skylake: breakdown for pressure solve timings



Marconi100: breakdown for pressure solve timings



Conclusions and future work

- Presented PETSc4FOAM capabilities
  - Customizable
  - Performance portable (native, third party)
  - OpenFOAM code base should not care about solver and architecture characteristics
- Comprehensive analysis can only be done on users' tests cases!
  - Maybe CG + Jacobi is the fastest solution for you?
- Preliminary results indicate good speedup when not oversubscribing the GPU (Summit with Volta Multiprocessing System?)
- Future work
  - Speed up matrix assembly
  - Geometric multigrid (ML) and AMGX https://github.com/NVIDIA/AMGX
  - Based on users' feedback ☺