AMGX GPU SOLVER DEVELOPMENTS FOR OPENFOAM

Matt Martineau, Stan Posey, Filippo Spiga 14-Oct-2020

SUMMARY

- Extended **PETSc4FOAM** library (from members of the HPC TC) to accelerate pressure solves with AmgX

- Early results of the AmgX solver library used to accelerate the OpenFOAM pressure solve on **GPUs** achieved ~**4x** to ~**8x speedups**

- A new library, **FOAM2CSR**, for low-overhead conversion between OpenFOAM LDU matrices and GPU-resident CSR matrices

- Multi-GPU/multi-node capability, with ongoing performance optimisation

PETSc4FOAM: A Library to plug-in PETSc into the OpenFOAM Framework

AMGX FOR OPENFOAM

Open-source sparse iterative solver library

- Fully GPU accelerated library and highly configurable
- **Algebraic multigrid (AMG)** preconditioning
  - In this study: **PCG + AMG**
- All results refer to the *v2.1.x pre-release branch*
  - Significant (>2x) setup performance increases
  - Improved support for new versions of CUDA (10, 11)

```
"solver": {
    "preconditioner": {
        "solver": "AMG",
        "cycle": "V",
        "smoother": {
            "solver": "BLOCK_JACOBI"
        },
        "max_iters": 1,
        "max_levels": 25,
        "interpolator": "D2",
        "presweeps": 1,
        "postsweeps": 1
    },
    "solver": "PCG",
    "max_iters": 100,
    "convergence": "ABSOLUTE",
    "tolerance": 1e-04,
    "norm": "L1"
}
```

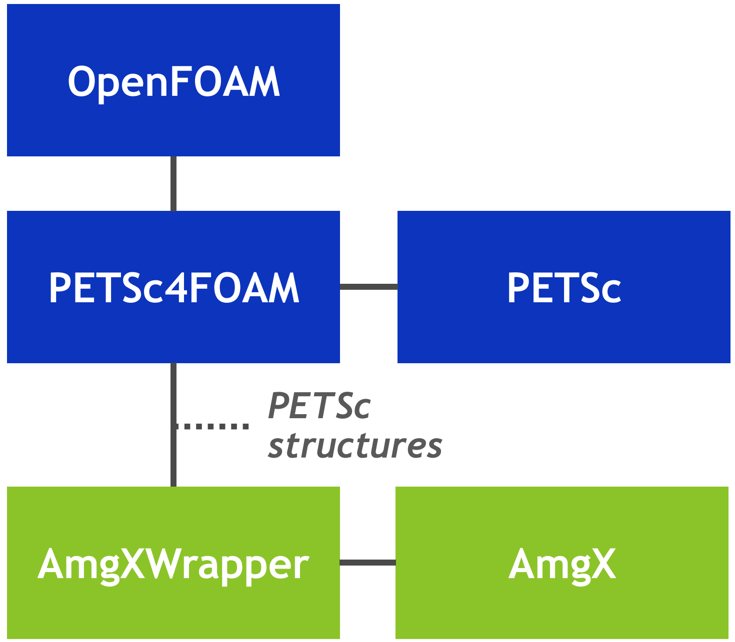
AmgX configuration for AMG + PCG

https://github.com/NVIDIA/AMGX

BASIC INITIAL SOLUTION

First pass at GPU acceleration

- Extending **PETSc4FOAM** infrastructure to call into **AmgXWrapper** to drive **AmgX**
    - AmgXWrapper accepts **PETSc** data structures
- Initial performance slower than CPU
    - *Much performance critical code not resident on the GPU*



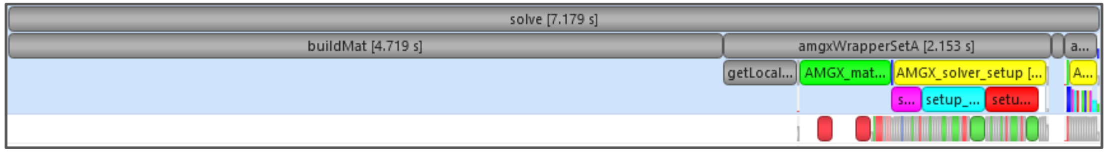
High level call structure for initial acceleration approach using AmgX

https://github.com/barbagroup/AmgXWrapper

INITIAL SOLUTION PROFILING

Searching for optimisation potential

- 1 MM cell case on DGX-1 using V100 and single BDW core



- Accelerate **buildMat** and reduce overhead of **amgxWrapperSetA**

- **AMGX_solver_setup** required on first step; in subsequent steps this can be replaced with **AMGX_solver_resetup**

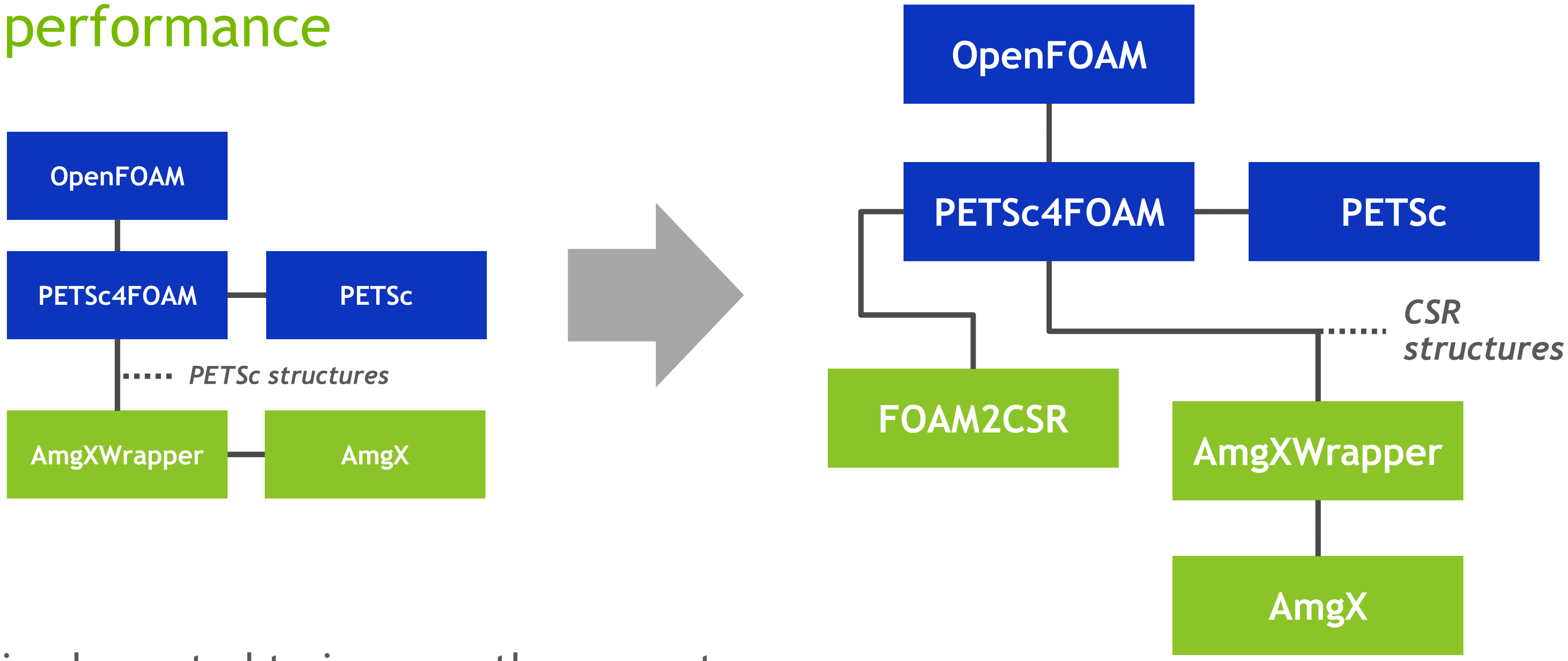
- *“Wrap PETSc vector”* can be avoided

Task	Time
Build matrix	4.7s
Get local matrix	0.4s
Upload matrix	0.6s
Setup	1.1s
Wrap PETSc vector	0.1s
Pressure solve	0.2s

Out of 7.2s pressure solve time only 1.4s is effective GPU work

ADAPTED SOLUTION

To improve performance



- FOAM2CSR implemented to increase the amount of computational workload resident on the GPU
- AmgXWrapper is extended and optimised to support CSR and improve host utilization

FOAM2CSR APPROACH

OpenFOAM LDU to GPU-resident CSR

FOAM2CSR Algorithm:

(1) Copy/reorganise LDU matrix data ready for conversion

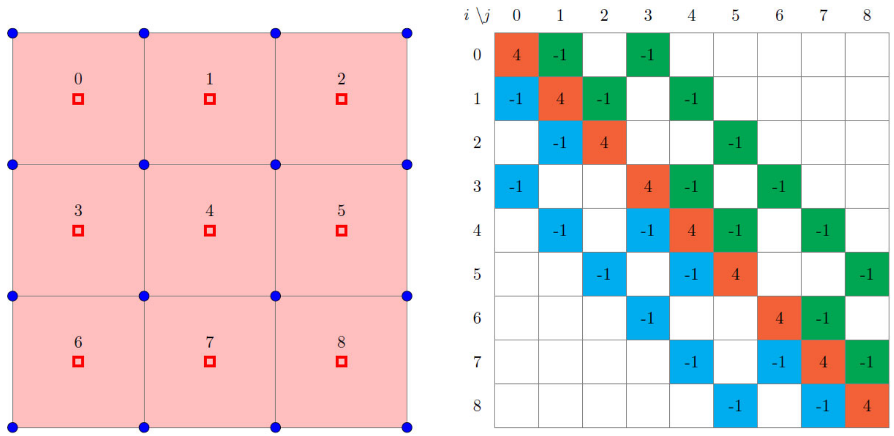
```
diagAddr    = [ 0, 1, …, Nrows ]
perm        = [ 0, 1, …, Nnz ]
colIndices = [ diagAddr upperAddr lowerAddr ]
rowIndices = [ diagAddr lowerAddr upperAddr ]
values      = [ diag     upper     lower     ]
```

(2) Sort **perm** and **rowIndices**, by **rowIndices** (radix sort)

(3) Collapse **rowIndices** to **rowOffsets** (exclusive scan)

(4) Sort **colIndices** and **values** by **perm**

- After the first step - low overhead conversion



upper = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]

diag = [4 4 4 4 4 4 4 4 4]

lower = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]

upperAddr = [1 3 2 4 5 4 6 5 7 8 7 8]

lowerAddr = [0 0 1 1 2 3 3 4 4 5 6 7]

LDU matrix visualisation
(taken from S. Bnà, I. Spisso, M. Olesen, G. Rossi.
PETSc4FOAM: A Library to plug-in PETSc into the OpenFOAM Framework)

AMGX / AMGXWRAPPER CHANGES

Improving integration and performance

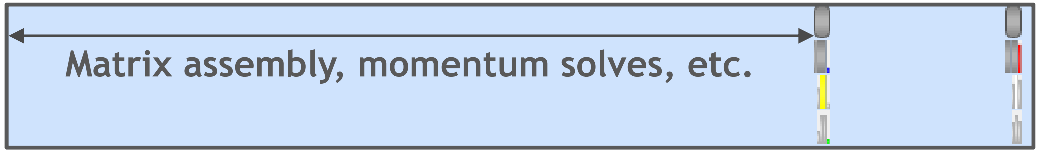
- Added **OpenFOAM residual calculation** to AmgX
- Fixed the default partitioning scheme in AmgX
- We extended AmgXWrapper to:
  - Handle **raw CSR inputs**, either *host or device pointers*
  - Support updating **matrix coefficients only**, and **re-setup**, a fast setup for timesteps where sparsity patterns persist
  - Perform **matrix consolidation** using CUDA IPC calls

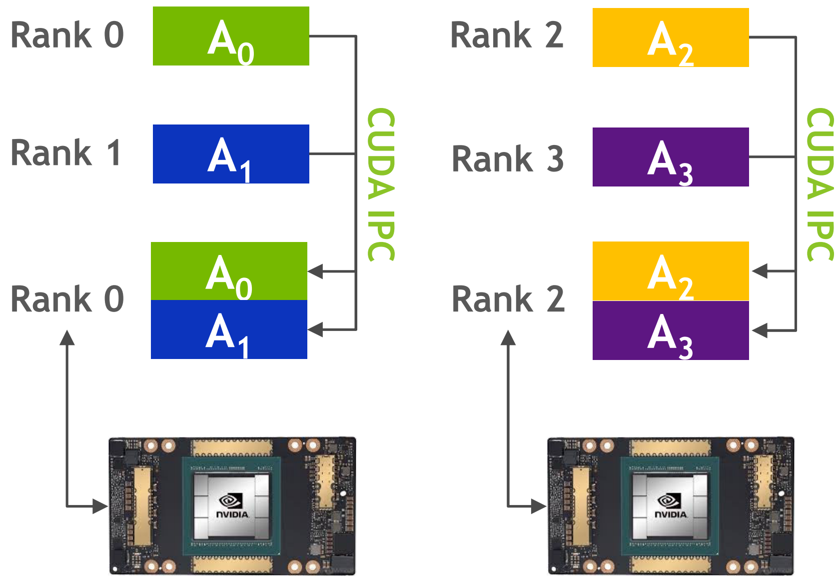
```
/* Upload CSR matrix to AmgX */
ErrorCode setA(
    const int nGlobalRows,
    const int nLocalRows,
    const int nLocalNz,
    const int* rowOffsets,
    const int* colIndicesGlobal,
    const double* values,
    const int* partData);

/* Update CSR matrix values in AmgX */
ErrorCode updateA(
    const int nLocalRows,
    const int nLocalNz,
    const double* values);

/* Performs the linear solve in AmgX */
ErrorCode solve(
    dobule* solution,
    const double* rhs,
    const int nRows);
```

https://github.com/barbagroup/AmgXWrapper

AMGXWRAPPER CONSOLIDATION

Merging matrix elements for performance



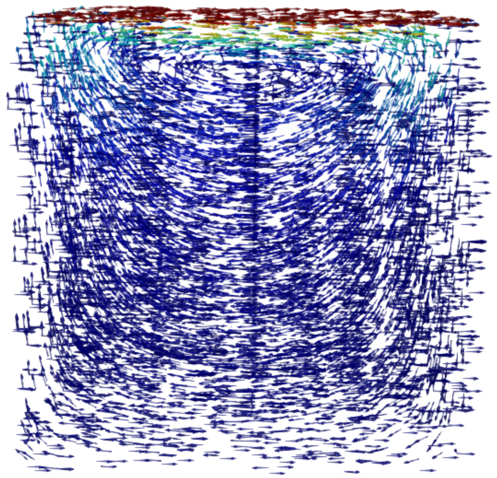


- Performance limited by the *single core restriction* due to the CPU-resident momentum solves etc.
- We developed a consolidation feature in AmgXWrapper that is low overhead
- CPU cores can be saturated for improved simulation runtime - around **8x wallclock speedup** single GPU

EXPERIMENTAL SETUP

Problem and system

- Using the HPC committee 3D lid-driven cavity model described in the PRACE whitepaper https://develop.openfoam.com/committees/hpc
- The medium (M) test problem fits adequately on a single GPU (200x200x200 or total 8 MM cells)



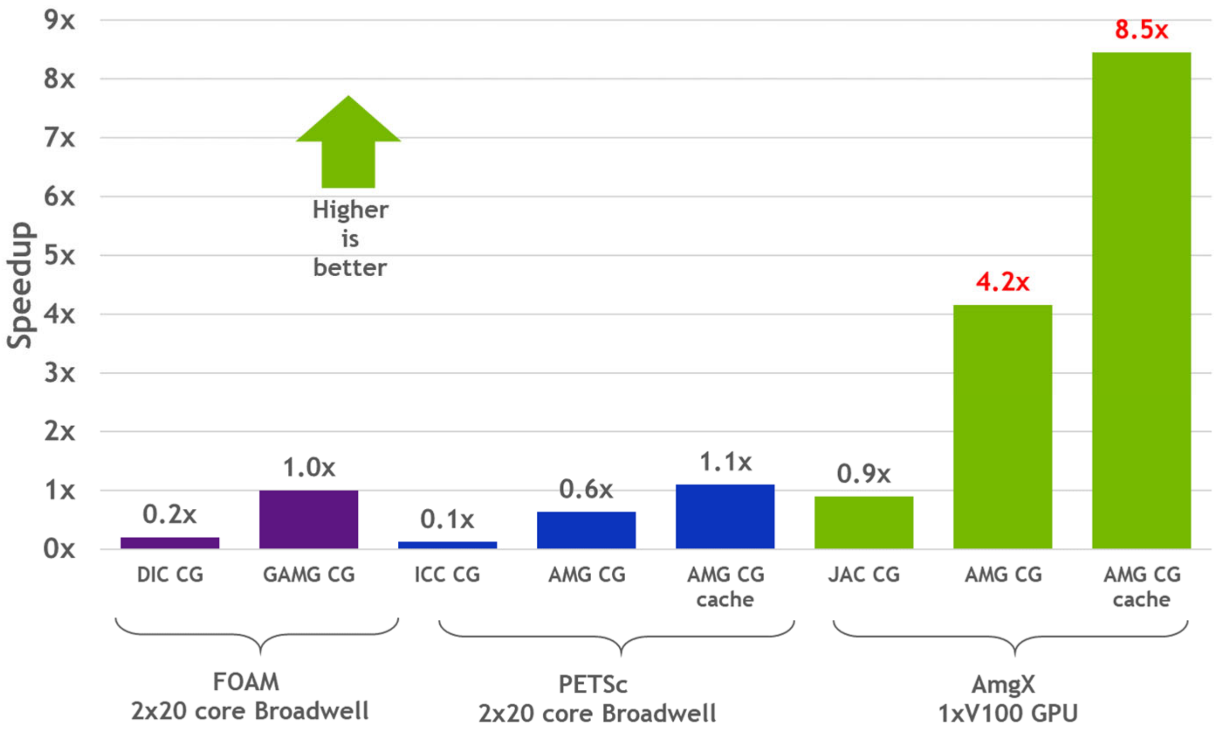
Lid Driven cavity
(M, 200x200x200, 20 steps) solution,
accelerated with AmgX

DGX-1 Feature	System Specifications
GPUs	8 x V100
CPUs	2 x 20 core E5-2698 v4
GPU memory	128 GB total system
GPU bandwidth	900 GB/s per GPU

Figure 1: Details of the test system NVIDIA DGX-1

RESULTS - PRESSURE SOLVE

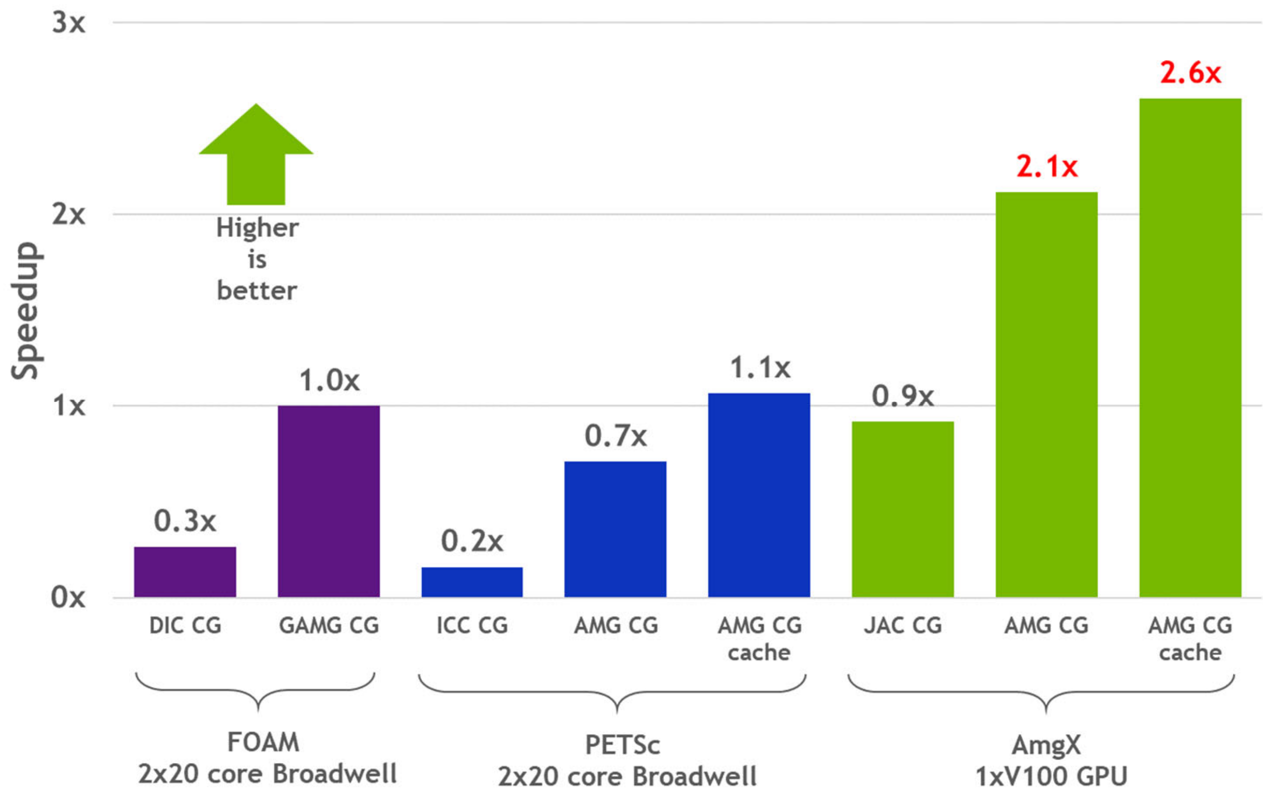
M problem (8 MM cells, 100 iters) on DGX-1



- Measuring all steps required to fulfil the pressure solve, i.e. LDU2CSR, comms, memory copies, solve etc.
- Significant GPU speedups over FOAM-GAMG of ~4x to ~8x
- New A100 GPU 1.6x faster than V100 for A100 speedups over FOAM-GAMG of ~6x to ~13x
- Still room for improvement

RESULTS - WALLCLOCK

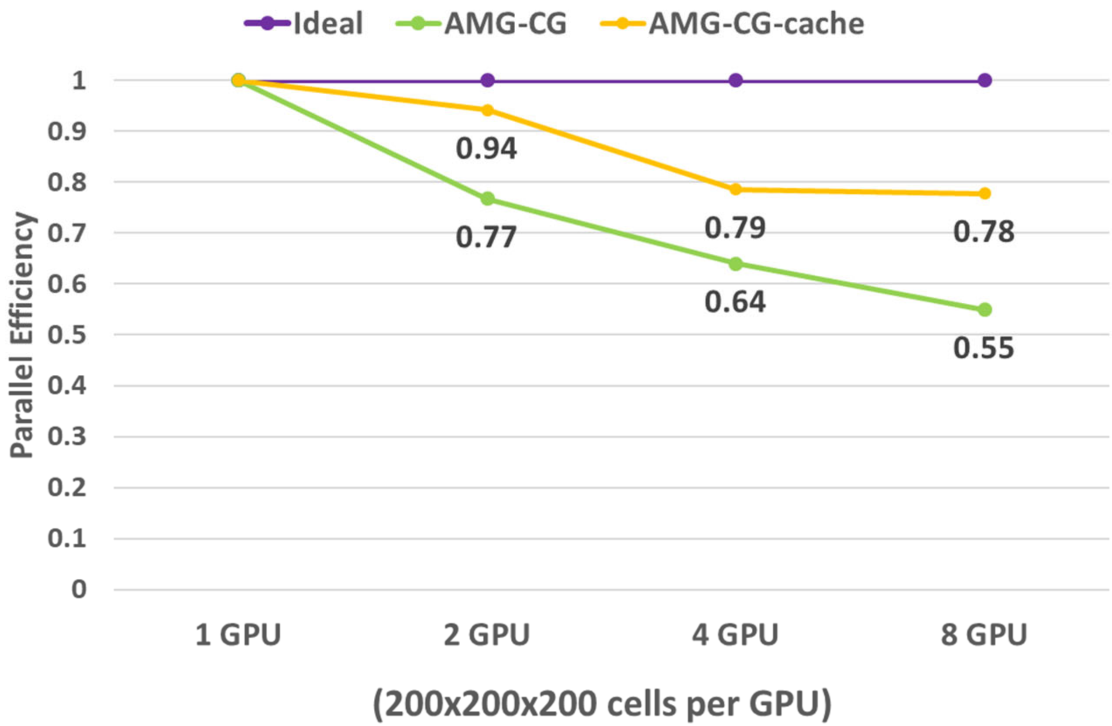
M problem (8 MM cells, 100 iters) on DGX-1



- Full solution wallclock is reasonable considering only pressure solve (now ~35% of total) is GPU accelerated
- Overhead of acceleration (i.e. copying data to and from the GPU) is small
- Could be greatly improved by accelerating matrix assembly and momentum solves

RESULTS – WEAK SCALING

Preliminary Multi-GPU Results on DGX-1



- The solver can be run on multiple GPUs across multiple nodes
- Consolidation for CPU cores also works in multi-GPU configuration
- Data movement is minimal, but could be removed if prior steps accelerated
- Setup scaling limits non-cached case
  - *The limitation is well understood, and we are currently optimising*

CONCLUSIONS

- Early results showcase the OpenFOAM pressure solve accelerated on NVIDIA **V100** GPUs using AmgX, achieving **~4x** to **~8x speedup**
- A new library, **FOAM2CSR**, was developed for low-overhead conversion between OpenFOAM LDU matrices and GPU-resident CSR matrices
- Changes to AmgX, and AmgXWrapper, enable integration with OpenFOAM and improved performance
- The multi-GPU/multi-node implementation is fully functional and performance optimisation is ongoing