JSASS-2025-2011

Performance and scaling of FaSTAR-GPU for CRM high-lift stall configurations on JSS3

Lusher, D.J.[(1)], Sansica, A.[(1)], Matsuzaki, T.[(2)]
[(1)]*Japan Aerospace Exploration Agency, 7-44-1 Jindaiji Higashi-machi, Chofu-shi, Tokyo 182-8522, Japan*
[(2)]*FMIC R&D INC., 105, 2-7-13 Kyounanchou, Musashino-shi, Tokyo 180-0023, Japan*

Keywords: CFD, GPU, FaSTAR-GPU, scaling, CRM-HL, RANS

Abstract

A large set of Reynolds Averaged Navier-Stokes (RANS)-based production simulations were performed on the full build-up configuration of the NASA High-Lift Common Research Model (CRM-HL) aircraft. The OpenACC accelerated solver, FaSTAR-GPU, is used for the work. Sensitivity to mesh refinement is shown for two mesh families from Heldenmesh and Pointwise, using the unstructured mixed element grids provided by the 5th High-Lift Prediction Workshop (HLPW5). Results are shown on grids between 12 million and 1500 million cells. Good agreement is found between the two mesh families at the finest refinement levels, with lift predictions within 1% of each other. Perfect agreement is found for cross-validations between FaSTAR-CPU and FaSTAR-GPU results. Quicker convergence to steady solutions is observed for the DP-LUR (FaSTAR-GPU) time-stepping algorithm when compared to the traditional LU-SGS (FaSTAR-CPU) method and settings. Computational performance and scaling are investigated on up to 120 Nvidia V100 GPUs on the JSS3 supercomputer. The level of obtained strong scaling is demonstrated for between 1 and 30 GPU nodes. The optimal number of MPI domains per GPU is also investigated, with a one-to-one mapping between CPU-core and GPU found to provide the highest computational efficiency.

1. Introduction

To reduce the time-to-market of future aircraft, it is critical to predict the flight envelope accurately before building prototypes for flight tests. It is well known that most low-fidelity CFD methods that rely on approximate RANS-based solutions, turbulence models, or wall models, struggle to match experimental predictions in off-design regions of the flight envelope [1], such as during stall. The High-Lift Prediction Workshop (HLPW) series [https://hiliftpw.larc.nasa.gov/] aims to assess the numerical prediction capability of current CFD technology considering the high-lift version of the NASA Common Research Model.

The present work uses the geometries and grids provided by the latest 5th edition of this workshop (HLPW5) to investigate mesh sensitivity and computational scaling for RANS-based steady simulations in the OpenACC accelerated solver, FaSTAR-GPU [2]. Cross-validation is also provided against the traditional CPU-based FaSTAR-CPU [3; 4] code. Performance in the context of iterative convergence, multi-GPU scaling, and optimal MPI domain composition strategy, are also investigated.

2. Computational Method

All simulations are performed in JAXA's unstructured compressible CFD solver, FaSTAR [3; 4]. A new GPU accelerated (MPI+OpenACC) version of the code (FaSTAR-GPU [2]) is the main focus of this study. A cell-vertex finite volume method is applied with the Harten-Lax-van Leer-Einfeldt-Wada (HLLEW) scheme[5] used as default for convective terms in the Reynolds-Averaged Navier-Stokes equations. A Green–Gauss/Weighted-Least-Squares (GLSQ) method[6] is applied to perform gradient computations. Stabilization is achieved via a Hishida (van Leer-type) slope limiter and U-MUSCL scheme [7]. The baseline turbulence model used for all comparisons is the no-ft2 term variant of the classic Spalart-Allmaras 1992 model [8]. For time-integration to steady solutions, the CPU-based solver uses a Lower/Upper Symmetric Gauss-Seidel (LU-SGS) [9] time-stepping scheme. The GPU-based solver instead uses a Data Parallel Lower Upper Relaxation (DP-LUR) method [10].In both cases, a Courant–Friedrichs–Lewy (CFL)-fixed local time step is used. All calculations were performed with a CFL number of 1. The different convergence characteristics of the two time-stepping methods is to be reported. For the DP-LUR method on GPU, $k = 13$ Jacobi sub-steps are required for stability on the CRM-HL case. This is con-

siderably higher than the $k = 6$ found to be required for simpler geometries such as the two-dimensional OAT15A airfoil or clean CRM configuration.

2.1. Flow Configurations and Mesh

The flow configuration is taken to be the conditions imposed by the 10th edition of the Aerodynamic Prediction Challenge (APC-10). The geometry is taken to be the full build-up aircraft configuration corresponding to Task 3 at the HLPW5. This version of the High-Lift Common Research Model (CRM-HL) is far more geometrically complex than the clean configurations used for validation purposes within the workshop. This model contains flaps, slats, nacelle and pylon, all of which add additional computational cost and increased difficulty in obtaining converged numerically stable solutions. Uniform freestream conditions are imposed such that $M = 0.2$, $Re = 1.09 \times 10^6$, and $T_{\text{static}} = 283.3K$. An angle of attack sweep is performed for certain mesh levels and grid topologies over the range $\alpha = [6.85°, 8.95°, 11.04°, 13.12°, 14.15°, 15.17°, 16.19°, 17.20°, 18.20°]$. Initial conditions are generated by performing a warm-start procedure to help improve the convergence and reliability of the steady solutions. The benefits of warm-started simulations over cold start was discussed on the previous HLPW4 configuration in [11].

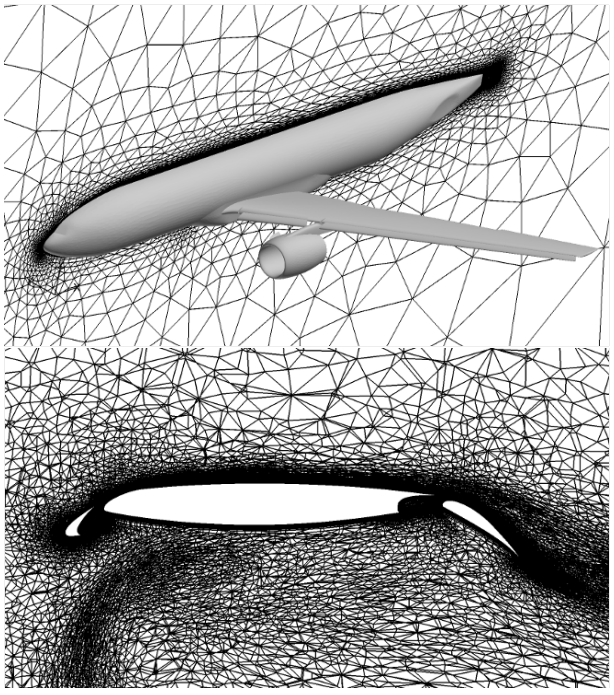

Figure 1: Heldenmesh mixed element unstructured grids on the LVL-f medium refinement level, provided by HLPW5. Number of nodes: 58,556,136.

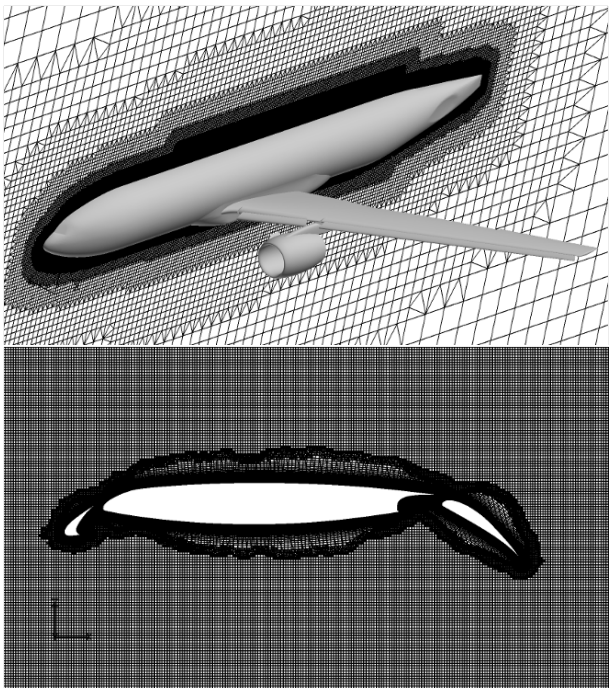

Figure 2: Pointwise mixed element unstructured grids on the LVL-C medium refinement level, provided by HLPW5. Number of nodes: 242,352,881.

Mesh sensitivity studies typically focus on convergence of aerodynamic forces for simulations performed on successive levels of refinement within the same family of grids. However, RANS simulations can also be sensitive to the different approaches taken by competing meshing strategies. In this work we compare families of grids provided by Helden Aerospace™(Heldenmesh), and Cadence™(Pointwise). The Heldenmesh and Pointwise meshes are the ones provided by the HLPW5: https://hiliftpw.larc.nasa.gov/Workshop5/grids.html. Four and five levels of mesh refinement are simulated for the Pointwise and Heldenmesh families, respectively, for the baseline configuration of $Re = 1.09 \times 10^6$. The Pointwise meshes range in cell counts between 52 and 1230 million, whereas the Heldenmesh ones have between 12 to 1500 million. The Heldenmesh grids use prisms and pyramids within the viscous layers, with tetrahedrals in the volume. The Pointwise meshes are hexahedra dominant. The baseline meshes selected to perform the first comparisons are the medium Pointwise LVL-C, and Heldenmesh LVL-f grids. Example three-dimensional and side cut views of the two mesh approaches are shown in Figure 1 (Heldenmesh) and Figure 2 (Pointwise).

Each case and mesh level is initially started from uniform flow at an angle of attack of $\alpha = 0°$. The solution is advanced for 10,000 iterations before being restarted at $\alpha = 2°$. This incremental increase of the AoA is per-

formed in repeated steps until initial restart conditions are generated for all of the target AoA above. An additional 200,000-250,000 iterations are then performed depending on the case. Presented values of aerodynamic forces are averaged over the final 30,000 iterations.

3. Results

3.1. Mesh refinement study

Figure 3 and Figure 4 show two views of surface skin-friction in the case of Heldenmesh LVL-f and Pointwise LVL-c grids at a moderate angle of attack of $\alpha = 8.95°$, when the flow is mainly attached on the main part of the wing. Both mesh approaches deliver solutions that are qualitatively very similar. The regions of local flow separation are shown in dark blue, encircled by the dashed white contours indicating zero skin-friction iso-lines. At this angle of attack, the flow separation is minimal and is limited to the flaps and in the outboard region near the wing tip. Additional flow separation is observed on the nacelle around the main pylon area. The skin-friction patterns generated by the slats at the front of the wing agree well between the two mesh options. No significant differences are observed between the two meshing approaches on the medium level grids.

In figure 5, a full angle of attack sweep is presented for both of the mesh families on the Pointwise LVL-C and Heldenmesh LVL-f grids. Both mesh families have reasonable agreement for the lift curve, with slightly lower values observed for the Heldenmesh grids. Note that, additional refinement of the Heldenmesh grids in the next section reduces this difference in C_L between the two mesh families (figure 6). The drag curve matches very well. The moment coefficient has a similar offset to the lift curve which is reasonable constant throughout the entire angle of attack range tested.

Figure 6 shows quantitative comparisons of lift, drag, and pitching moment coefficients for the two mesh types over multiple levels of refinement (Pointwise LVL-A, B, C, D, and Heldenmesh LVL-c, m, f, g, r) at $\alpha = 8.95°$. The final values of the aerodynamic coefficients are given at each mesh level as a function of the inverse of the total cell count. The Pointwise meshes provided to the HLPW5 workshop provide more consistent predictions between the mesh refinement levels than is found for the Heldenmesh ones. The provided Heldenmesh grids show large variations between the refinement levels. However, the coarsest provided Heldenmesh grid is far coarser than the Pointwise one, and once the finest Heldenmesh grids are used (LVL-g, LVL-r), the values start to level off and match the predictions of the Pointwise grids. Similar trends are also observed for the drag and pitching moment coefficients. The three finest Pointwise grids form a straight line with a lift coefficient predicted to be just under 1.85, compared to 1.83 for the finest level Heldenmesh grid. This corresponds to a relative discrepancy in C_L of 1%. At the LVL-r (Heldenmesh) and LVL-D (Pointwise), excellent agreement is found for both the drag and moment coefficients which overlap exactly.

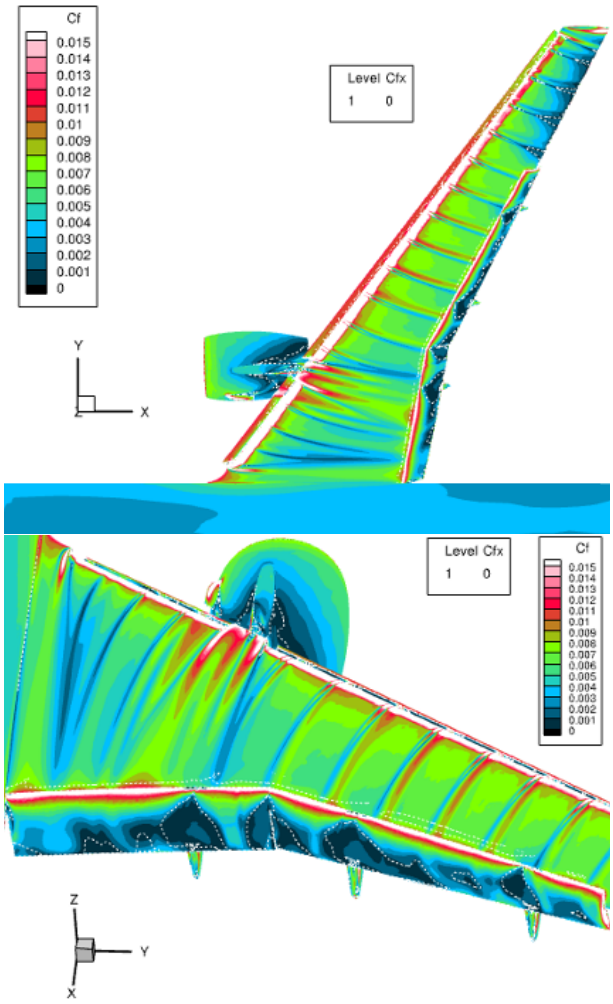

Figure 3: Skin-friction contours for the Heldenmesh LVL-f grid at an angle of attack of $\alpha = 8.95°$. The white line encloses regions of local flow separation.

3.2. LU-SGS vs DP-LUR convergence comparison

In this section, comparison is made between the convergence characteristics to steady solutions using both the LU-SGS (FaSTAR-CPU) and DP-LUR (FaSTAR-GPU) time-stepping algorithms. The purpose of this test is that, time to solution is the most useful measure of computational performance and efficiency when considering steady-RANS solutions. Comparisons are made using 20 TOKI-SORA CPU nodes (960 CPU cores), and 1 TOKI-RURI Nvidia V100 GPU node (4 GPUs). Figure 7 shows

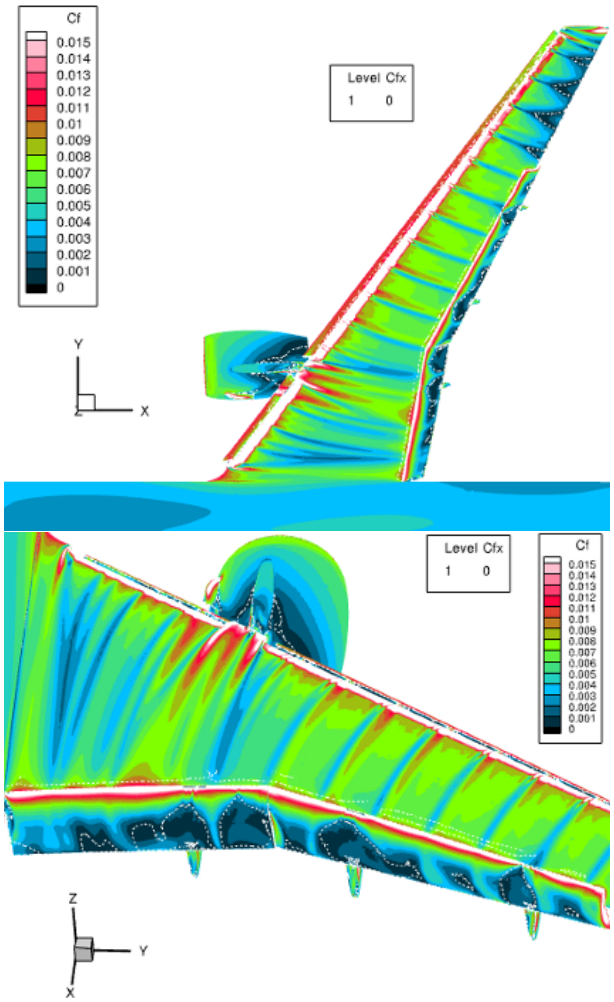

Figure 4: Skin-friction contours for the Pointwise LVL-C grid at an angle of attack of $\alpha = 8.95°$. The white line encloses regions of local flow separation.

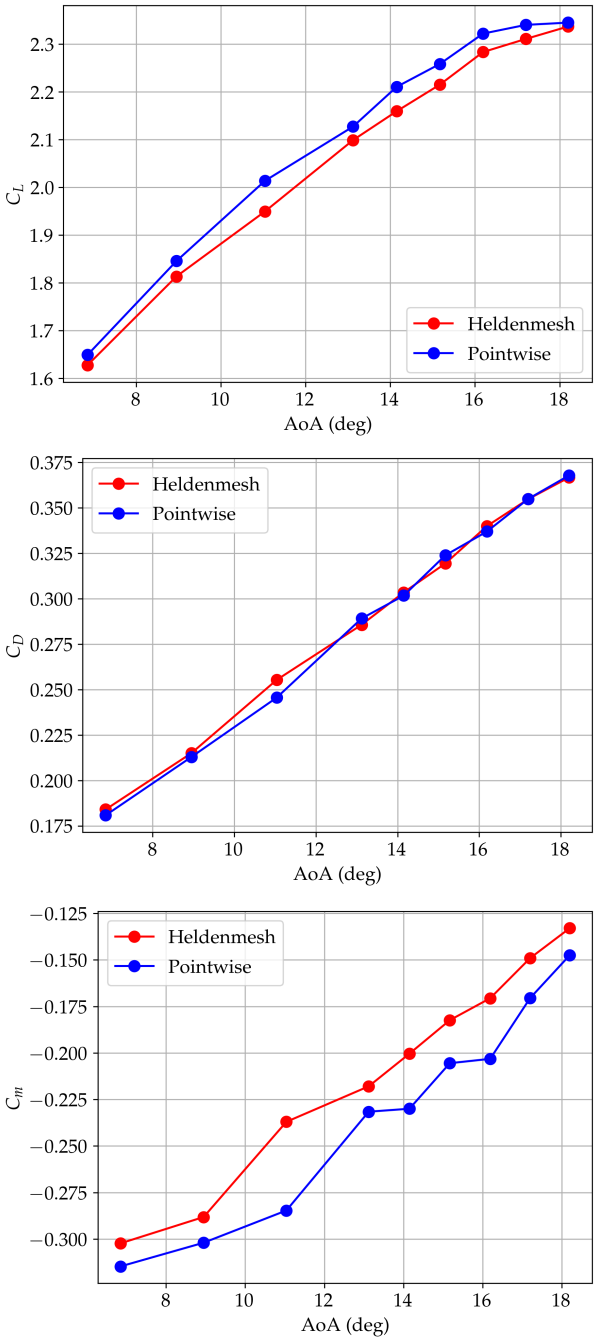

Figure 5: Lift, drag, and pitching moment coefficient polars as a function of angle of attack, for both the Heldenmesh LVL-f and Pointwise LVL-C grids.

the convergence of the lift coefficient between runs in FaSTAR-CPU (black) and FaSTAR-GPU (red, dotted). The same CFL number of 1 is used in both solvers. Perfect agreement is observed between the two solvers, confirming the correctness of the parallel implementation (MPI vs OpenACC+MPI) between the two solvers. Despite the different time-integration methods, the final values of C_L between the CPU and GPU codes match to within 0.0006% of each other. Similar agreement was observed for the drag and moment coefficients.

Figure 8 shows the iterative convergence for the residual of the density array on both CPU and GPU architectures. These plots highlight the different convergence characteristics between the LU-SGS and DP-LUR time-stepping algorithms in the two codes. When the x-axis is scaled with raw iteration number, the quicker convergence of the DP-LUR method is clear to see. The horizontal dashed line at 10^{-9} is selected as the measure of convergence in this case. The DP-LUR method reaches this convergence criteria by a factor of 1.45× fewer iterations than with LU-SGS. To comment on the performance impact this can have, the x-axis is rescaled in the

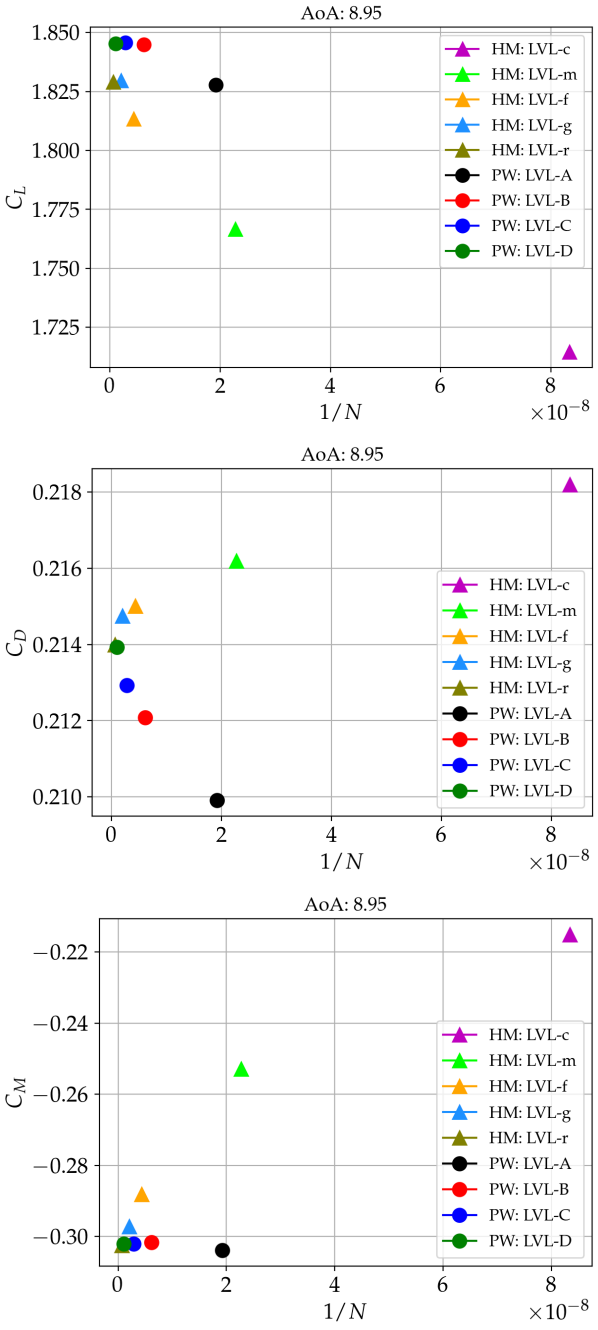

Figure 6: Mesh refinement study for both the Heldenmesh and Pointwise grids, at an angle of attack of $\alpha = 8.95°$. Showing coefficients of lift, drag, and pitching moment on nine different meshes.

lower plot to be based on the physical runtime in minutes. The choice of 20 CPU and 1 GPU nodes is relatively arbitrary, but represents typical ratios that users routinely use for this problem size. The time difference in reaching convergence between the two time-stepping methods is shown by the vertical dashed black lines. For this ratio of

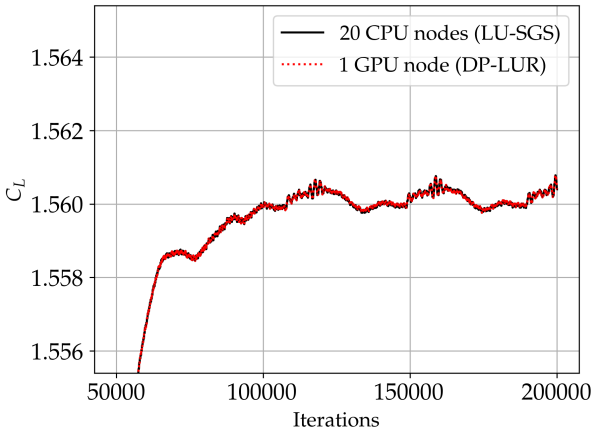

Figure 7: Iterative convergence of lift coefficient at $\alpha = 6$ on the LVL-f Heldenmesh grid. Showing excellent agreement between the CPU- and GPU-based solvers. Results between the two codes match to within 0.0006%.

computational nodes, the DP-LUR algorithm reaches the convergence criteria around 27% quicker.

3.3. Multi-GPU scaling

The MPI-based multi-GPU scaling is investigated in this section in the context of strong scaling for a fixed mesh size and increasing numbers of nodes/GPUs. Each node consists of 4 Nvidia V100 GPUs. The Heldenmesh LVL-f medium grid is used to perform the multi-GPU scaling tests. Figure 9 shows the runtime and relative speed-up factor to perform a fixed number of 500 iterations in both FaSTAR-CPU and FaSTAR-GPU. The CPU-based simulation is performed on 10 TOKI-SORA nodes (480 CPU cores). The GPU-based simulations are performed on between 1 and 30 nodes (4 and 120 GPUs, respectively). The red bars correspond to the speed-up relative to the runtime on the 10 CPU nodes, with the decrease in runtime shown on the second axis in blue. A linear dashed black line is also shown to indicate ideal strong scaling. The results show that even with only 1 GPU node, the single GPU node is 1.7x faster than the same calculation on 10 CPU nodes (giving an equivalent ratio of around 200 CPU cores per GPU). As additional GPU nodes are added, additional speed-up is obtained up to 17.4x on 120 GPU nodes. It is important to note that, the CRM-HL case selected is a very challenging one in terms of numerical stability with the DP-LUR algorithm that requires $k = 13$ iterations of the linear DP-LUR Jacobi iteration. For simpler test cases such as the clean CRM configuration, only $k = 6$ iterations are required. In that case, the speed-up in figure 9 would be doubled relative to the CPU result with LU-SGS. Optimisation of the time-stepping algorithm is ongoing work to improve the efficiency of the GPU code.

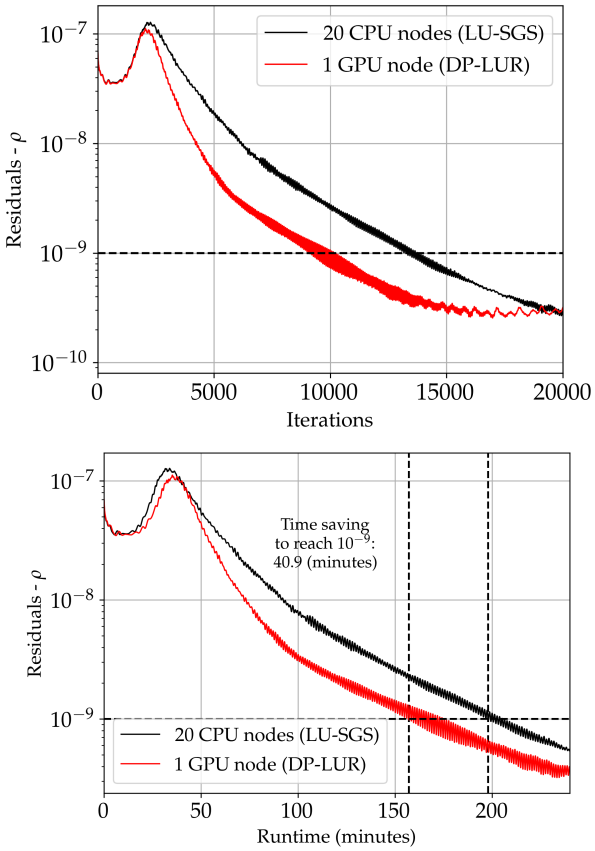

Figure 8: Iterative convergence of residuals, scaled by raw iterations and physical runtime. The comparison is performed on 1 GPU TOKI-RURI node (4 GPUs), and 20 TOKI-SORA CPU nodes (960 cores).

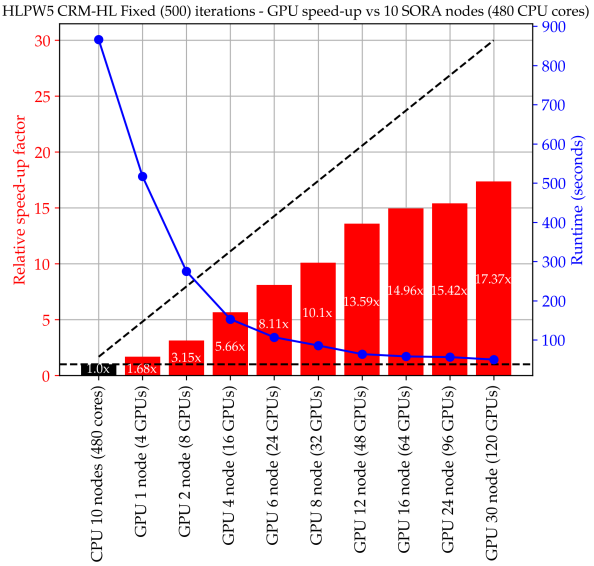

Figure 9: Multi-GPU scaling for a fixed 500 iterations at $\alpha = 6$ on the LVL-f Heldenmesh grid.

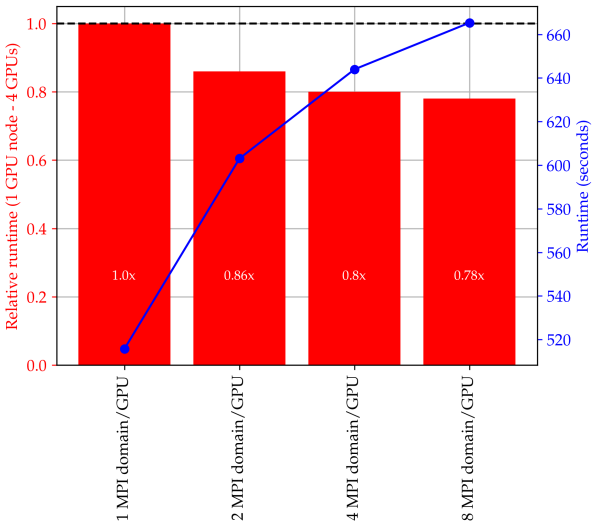

Figure 10: Relationship between the mapping of GPUs and number of MPI domains for a fixed 500 iterations at $\alpha = 6$ on the LVL-f Heldenmesh grid.

3.4. MPI domains per GPU

The previous section examined the strong scaling obtained by FaSTAR-GPU over MPI when adding additional nodes/GPUs at a fixed problem (mesh) size. In that case, a one-to-one mapping was used between MPI ranks (domains, in FaSTAR), and GPUs. That is, one individual CPU core is assigned to run a single process on each of the GPUs, such that only four of the thirty six available CPU cores on the TOKI-RURI node are used. In this section, the effect of mapping additional MPI ranks (domains) to each GPU is examined. The purpose of this is to understand what the optimal number of MPI ranks is per GPU.

The current mesh pre-processing in FaSTAR is aimed towards CPU-based execution where the number of domains is large (i.e. > 1000). GPU execution in comparison has far fewer number of MPI domains, because each GPU is much more powerful than an individual CPU core. For the medium level meshes for the cases presented, MPI decomposition is typically only required on 4-8 MPI domains when using GPU execution. The downside of this is that the pre-processing routines which are performed on CPU are very slow in comparison, as they can only be parallelized over 4-8 CPU nodes with much larger cell counts within each domain. Figure 10 shows the runtime for 500 iterations on the LVL-f Heldenmesh grid on a single GPU node (4 GPUs). In this case the hardware is fixed, but the different solution points represent the solution being decomposed into additional do-

mains. Each of the four GPUs are then assigned either 1, 2, 4, or 8 MPI domains each, within the single node. It is shown that the optimal configuration is a one-to-one mapping between CPU core (MPI domain) and GPU. Running additional MPI domains on each GPU leads to a computational slowdown by decomposing the problem into smaller chunks and adding unnecessary additional communication. Even with only 4 MPI domains per GPU, a slowdown of 20% is observed compared to the one-to-one optimal mapping. In conclusion, further work is required to also accelerate the parallel pre-processing routines in FaSTAR to be better suited to the low number of MPI domains required for execution within the GPU accelerated version of the solver. A GPU accelerated version of the FaSTAR pre-processing may be beneficial in the future to improve the mesh pre-processing times required.

4. Conclusions

A large set of RANS-based simulations has been presented in the OpenACC accelerated FaSTAR-GPU code. Simulations were performed on up to 120 Nvidia V100 GPUs on the JSS3 supercomputer. The case considered was the full build-up version of the NASA CRM-HL full aircraft model. Flow conditions were taken from those used at the 10th Aerodynamic Prediction Challenge (APC-10). Sensitivity to mesh refinement was reported for two different types of mesh topology, over 4-5 levels of refinement in each case. The grids were those provided by the High Lift Prediction Workshop 5, generated in both the Pointwise, and Heldenmesh meshing software. Large-scale simulations were performed on meshes comprised of up to 1.5 billion cells.

Excellent agreement was observed between the two meshing techniques when simulating the full angle of attack sweep at fixed mesh refinement levels. Qualitative comparisons of surface skin-friction contours showed only minimal differences between the two mesh families, which agreed well. Cross-validation between CPU- and GPU-based versions of the FaSTAR solver was demonstrated. Comparison of iterative convergence for two different time-stepping algorithms showed the DP-LUR method on GPU reached convergence quicker than LU-SGS on CPU. For a fixed mesh size, good multi-GPU strong scaling was demonstrated on between 4 and 120 GPUs. For a fixed number of GPUs, the optimal number of MPI domains per GPU was investigated. The highest computational efficiency was obtained when applying a one-to-one mapping between CPU cores and GPUs (one MPI domain per GPU).

Acknowledgements

Computational time was provided by the JAXA JSS3 supercomputer.

References

[1] C. L. Rumsey, J. P. Slotnick, and C. Woeber, *HLPW-4/GMGW-3: Overview and Workshop Summary*. 2022.

[2] P. Zehner and A. Hashimoto, “Acceleration of the data-parallel lower-upper relaxation time-integration method on gpu for an unstructured cfd solver,” *Computers & Fluids*, vol. 256, p. 105842, 2023.

[3] A. Hashimoto, K. Murakami, T. Aoyama, K. Ishiko, M. Hishida, M. Sakashita, and P. Lahur, “Toward the Fastest Unstructured CFD Code ’FaSTAR’,” *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, no. AIAA 2012-1075, 2012.

[4] T. Ishida, A. Hashimoto, Y. Ohmichi, T. Aoyama, and K. Takekawa, “Transonic buffet simulation over NASA-CRM by unsteady-FaSTAR code,” *AIAA Paper 2017-0494*, 2017.

[5] S. Obayashi and G. P. Guruswamy, “Convergence acceleration of an aeroelastic Navier-Stokes solver,” *AIAA Journal*, vol. 33, pp. 1134–1141, 1995.

[6] E. Shima, K. Kitamura, and T. Haga, “Green–gauss/weighted-least-squares hybrid gradient reconstruction for arbitrary polyhedra unstructured grids,” *AIAA Journal*, vol. 51, pp. 2740–2747, 2013.

[7] C. O. Burg, “Higher order variable extrapolation for unstructured finite volume rans flow solvers,” *17th AIAA Computational Fluid Dynamics Conference*, 2005.

[8] P. R. Spalart and S. R. Allmaras, “A one-equation turbulence model for aerodynamic flows,” in *30th Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings*, vol. AIAA Paper, 1992.

[9] D. Sharov and K. Nakahashi, “Reordering of hybrid unstructured grids for lower-upper symmetric Gauss-Seidel computations,” *AIAA Journal*, vol. 36, pp. 484–486, 1998.

[10] M. J. Wright, G. V. Candler, and D. Bose, “Data-parallel line relaxation method for the navier-stokes equations,” *AIAA journal*, vol. 36, no. 9, pp. 1603–1609, 1998.

[11] M. Zauner, A. Sansica, T. Matsuzaki, D. James Lusher, and A. Hashimoto, “Free-air simulation sensitivities on nasa’s high-lift common research model,” *AIAA Journal*, pp. 1–25, 2024.