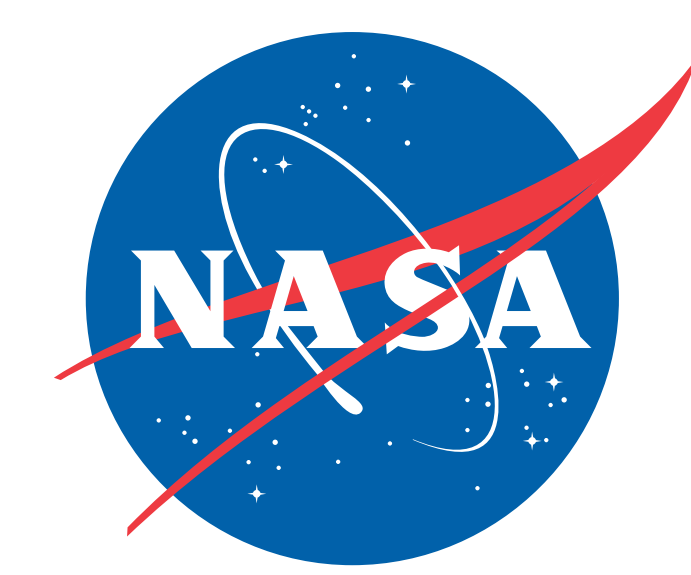




Improvements to a Batch Pentadiagonal Solver on NVIDIA GPUs



Dr. Charles W Jackson¹ and Dr. Mohammad Zubair²

¹Computational AeroSciences Branch,
NASA Langley Research Center, Hampton, VA
charles.w.jackson@nasa.gov

²Computer Science Department,
Old Dominion University, Norfolk, VA
zubair@cs.odu.edu

Introduction

This poster presents the recent work on OVERFLOW to port the batched-pentadiagonal solver to NVIDIA GPUs. This algorithm is used by OVERFLOW to solve the 3D Euler equations using a finite-difference method on curvilinear, structured meshes with an implicit, approximate-factorization linear scheme [1]. Each 1D pencil of the grid (in one grid direction) inverts a diagonalized form of the linearized Euler equations, resulting in 5 scalar pentadiagonal systems for each grid line. Because of the diagonalization used, the first three systems share the same left-hand-side (LHS) matrix, while the other two are simple modifications of this LHS.

- A matrix with N unknowns can be stored as a set of vectors: **A[N], B[N], C[N], D[N], E[N]**
- There is a batch size of β pentadiagonal systems (5 * number of pencils in a grid direction)
- Each vector in a batch can be stored in an interleaved, **A[β , N]**, or sequential, **A[N, β]**, manner
- Different algorithms benefit from different ordering
- The same storage is used for the LHS and RHS

$$\begin{bmatrix} c_1 & d_1 & e_1 & 0 & \dots & \dots & \dots & 0 \\ b_2 & c_2 & d_2 & e_2 & \ddots & & & \vdots \\ a_3 & b_3 & c_3 & d_3 & e_3 & \ddots & & \vdots \\ 0 & a_4 & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & e_{n-3} & 0 \\ \vdots & & \ddots & a_{n-2} & b_{n-2} & c_{n-2} & d_{n-2} & e_{n-2} \\ \vdots & & & \ddots & a_{n-1} & b_{n-1} & c_{n-1} & d_{n-1} \\ 0 & \dots & \dots & \dots & 0 & a_n & b_n & c_n \end{bmatrix}$$

Initial Approaches

- Version 1: Use CUSPARSE library's implementation of the batched pentadiagonal solver (using QR factorization)
- Version 2a: Implement our own direct solver, which can reduce memory usage by reusing the same LHS for the first three RHS
- Version 2b: Modify the LHS for the 4th and 5th equations within the pentadiagonal solver, reducing the memory usage further (modifications of the **B** and **D** diagonals)

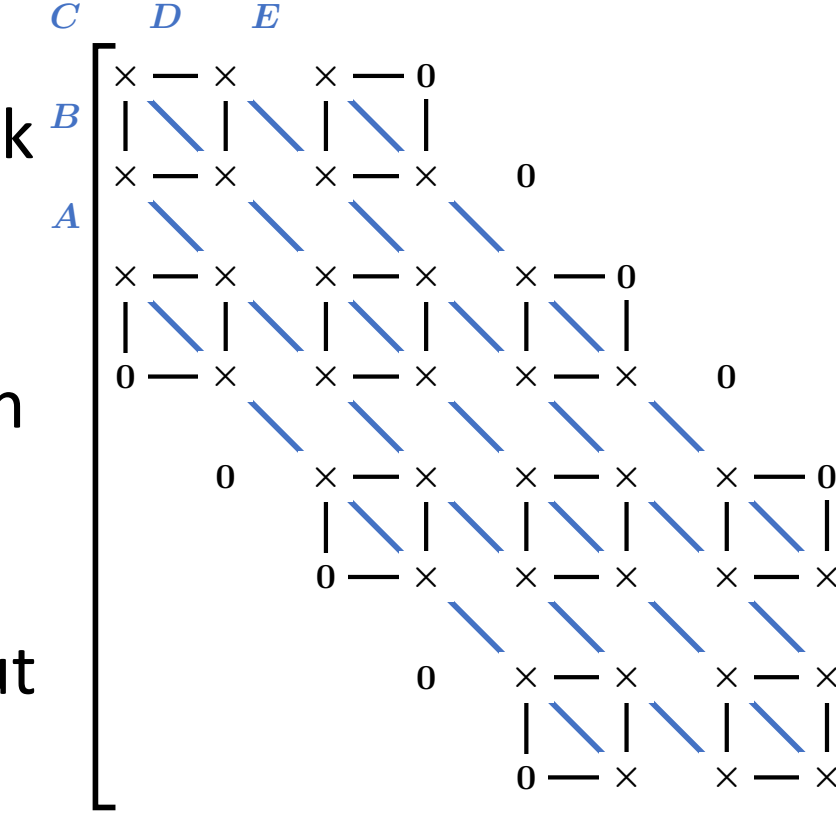
Interleaved Data Layout



- Because of the coarse-grained parallelism in versions 1 and 2, it is best to interleave the data (same as on CPU)
- In these versions, each thread solves a single system
- Threads **1** and **2** access the first entry of their A vectors at the same time, thus a coalesced load

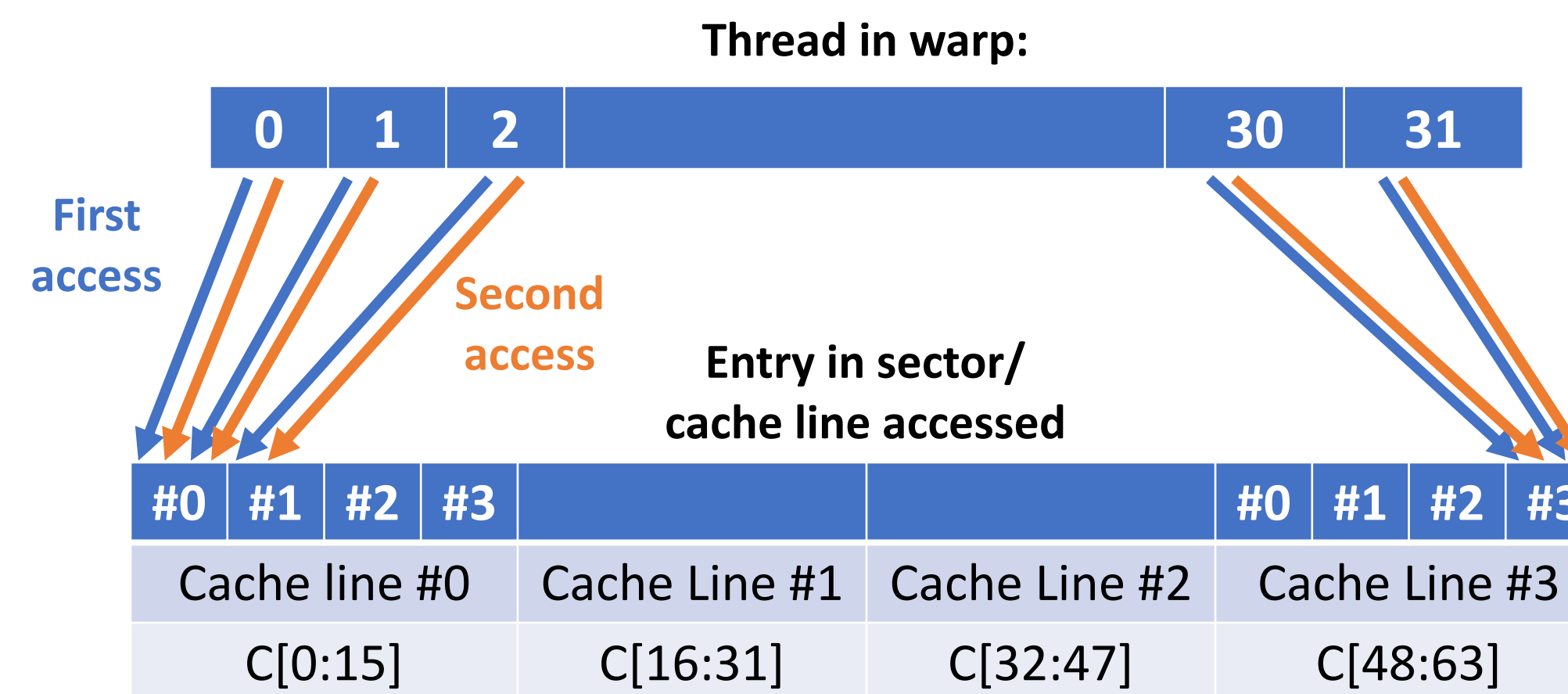
Parallel Cyclic Reduction Approach

- Version 3: Reformulate the pentadiagonal system as a 2x2 block tridiagonal system
- This exposes finer-grain parallelism through the parallel cyclic reduction (PCR) algorithm [2,3]
- There are scalar versions of these routines in the CUSPARSE library but no block-tridiagonal solvers
- Version 3 keeps the data of a pentadiagonal system in registers and uses shuffle instructions to share data among threads resulting in an effective utilization of the memory bandwidth



Memory Access for PCR

- Because of the fine grain parallelism, the systems do not need to be interleaved as in versions 1 and 2
- Construction of the matrix may be faster because it does not have to be interleaved



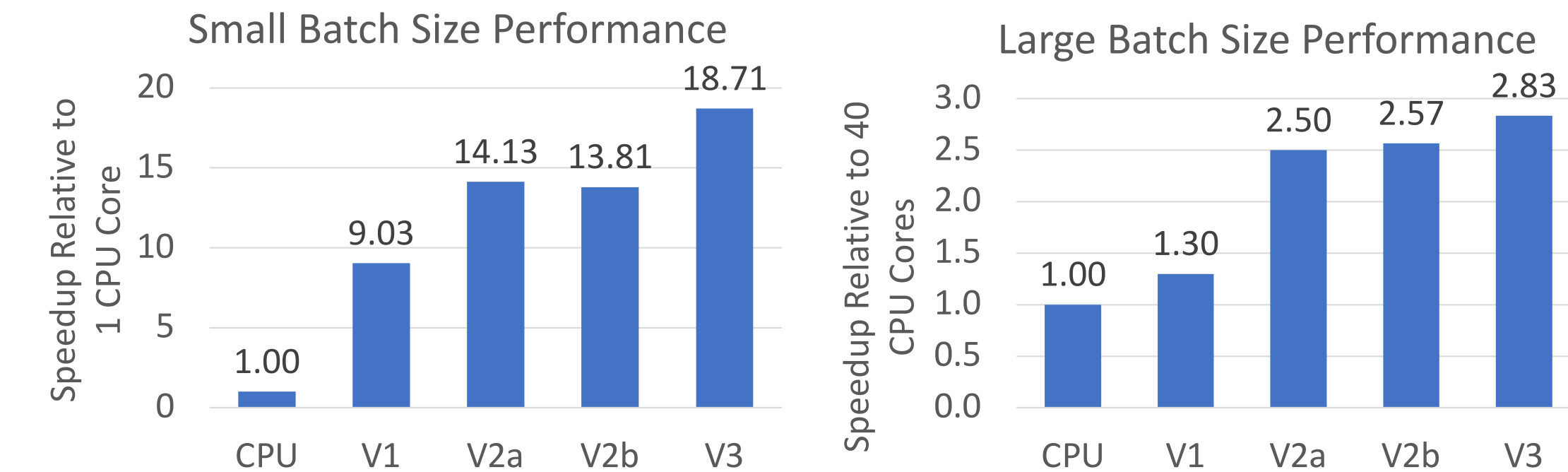
- The PCR algorithm causes uncoalesced memory accesses because the warp is loading every other entry at a time (blue arrows), loading 16 sectors instead of the ideal 8 (for double precision loads)
- However, the second load (orange arrows) is the other half of entries, which should still be in cache, thus no additional sectors need to be read, resulting in two loads using 16 sectors

Sources

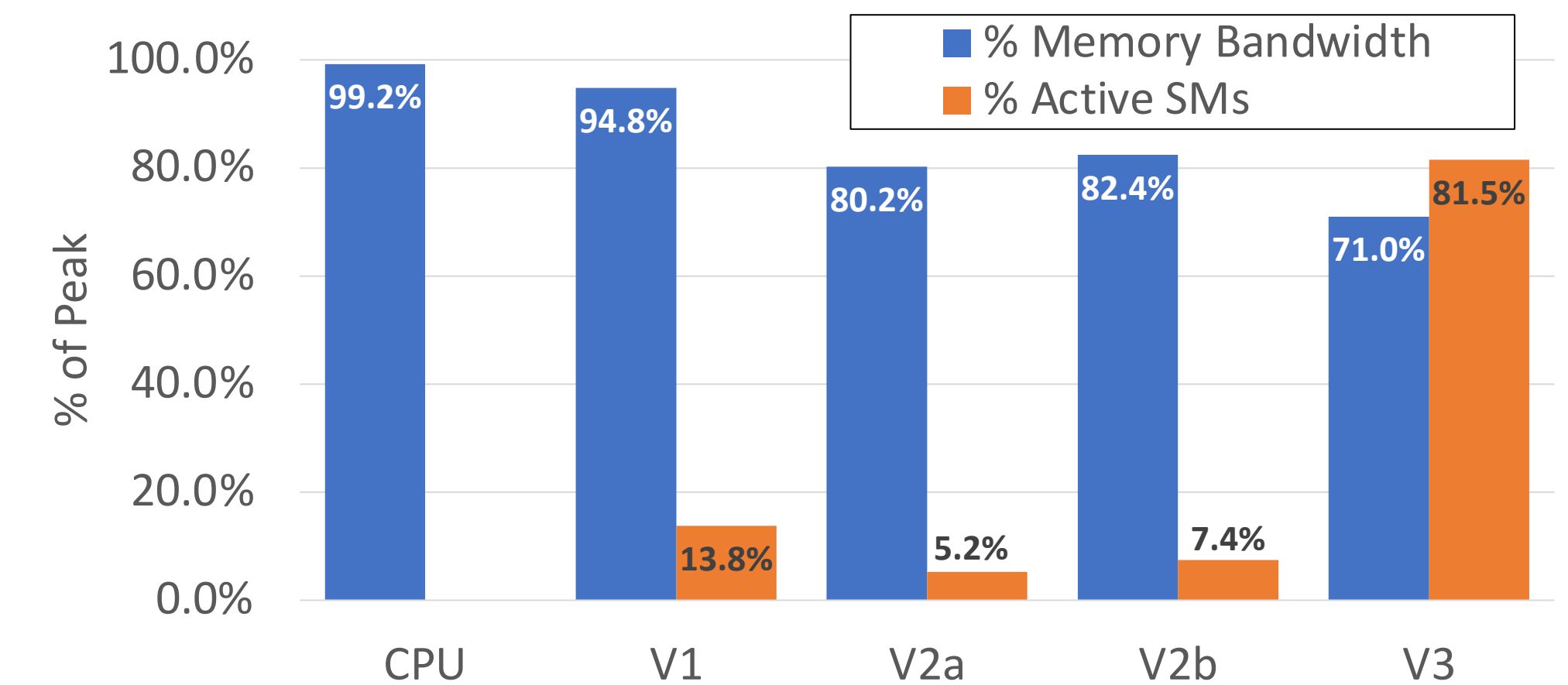
- [1] T. H. Pulliam and D. S. Chaussee. 1981. A Diagonal Form of an Implicit Approximate-Factorization Algorithm. *Journal of Computational Physics* 39, 2, 347-363 (1981). DOI: [10.1016/0021-9991\(81\)90156-X](https://doi.org/10.1016/0021-9991(81)90156-X)
- [2] E. László, M. Giles, and J. Appleyard. 2016. Manycore Algorithms for Batch Scalar and Block Tridiagonal Solvers. *ACM Transactions in Mathematical Software* 42, 4, Article 31 (July 2016). DOI: [10.1145/2830568](https://doi.org/10.1145/2830568)
- [3] W. Gander and G. H. Golub. 1997. Cyclic Reduction – History and Applications. *Scientific Computing*, Hong Kong. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.2066>

Performance

- Performance was measured on a single V100 GPU
- Time to solve a system with 64 unknowns with two batch sizes: 19,840 (compared to a single Skylake core) and 793,600 (compared to 40 Skylake cores on 2 sockets, 1 node)



Memory and Compute Usage for Large Batch Size



Conclusions

- Version 2 is approximately 92% faster than Version 1 due to reduced memory requirements
- Performing the modification of the LHS in the pentadiagonal solve has a minimal impact on the performance (<5%) compared to performing the modification in its own kernel
- Using PCR is much more performant than NVIDIA's current batch pentadiagonal solver (2.18x faster) but is currently only implemented for systems that have number of unknowns that are powers of 2.
- Combining reuse of LHS and PCR results in code that is 2.58x faster on a V100 than the CUSPARSE implementation and 3.87x faster than 40 Intel Xeon Skylake cores of 1 dual-socket node

Acknowledgements

The authors are grateful for the support provided by the NASA Office of Chief Engineer's Aerosciences Research and Analysis Program and continued support from the NASA Revolutionary Vertical Lift Technology (RVLT) Project of the Advanced AirVehicles Program (AAVP). The authors would also like to thank the staff at the NASA Advanced Supercomputing Division for their support and access to compute resources.