ORNL/TM-2023/3213

Developing and Scaling an OpenFOAM Model to Study Turbulent Flow in a HFIR Coolant Channel

Emilian Popov
Nicholas Mecham
Carter Edwardson

December 2023

OAK RIDGE National Laboratory

ORNL IS MANAGED BY UT-BATTELLE LLC FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2023/3213

Nuclear Energy and Fuel Cycle Division

DEVELOPING AND SCALING AN OPENFOAM MODEL TO STUDY TURBULENT FLOW IN A HFIR COOLANT CHANNEL

Emilian Popov: ORNL
Nicholas Mecham: NCSU
Carter Edwardson: UCSD

December 2023

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831
managed by
UT-BATTELLE LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR227

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDMENTS

The authors would like to acknowledge the support for this work provided by the Office of Material Management and Minimization of the US Department of Energy's National Nuclear Security Administration.

1. INTRODUCTION

Improving the understanding of how computational fluid dynamics (CFD) direct numerical simulations (DNS) of flows in the High Flux Isotope Reactor (HFIR) perform when run in parallel using the high-performance computing (HPC) platform Summit at the Oak Ridge Leadership Computing Facility (OLCF) is of particular importance to boost the computational tools used to support HFIR conversion to low enriched fuel (LEU). Evaluation of scaling performance was driven by the increasing importance of graphics processing unit (GPU) usage in HPC, which is becoming the standard for modern supercomputers such as Summit. The desired results are to obtain a strong positive correlation between the computational resources dedicated to a problem and the relative speed-up of the simulation in comparison to a benchmark. This capability will allow substantially improvement in HFIR flow analytical capabilities, specifically when predicting turbulence properties at high Reynolds numbers. The study leverages previous simulation results performed with code PHASTA (finite element) on HPC platforms Cori (NERSC) and Theta (ALCF) [1] with computing options provided in the computing platform OpenFOAM (finite volume) at OLCF. Transitioning from PHASTA to OpenFOAM will (1) eliminate dependence on third-party software for mesh generation and manipulation, (2) reduce resource needs by employing modern architectures, and (3) build expertise for future modeling of HFIR-specific problems like heat transfer in involute geometry, entrance effects, flow structure in channel corners, and so on—all important issues when defining the available thermal margins in the transition to LEU.

CPUs and GPUs differ significantly in their architecture and utilization, as discussed in the literature [2]. The most important differences are in the approach to computations and their memory. A single GPU contains a large quantity of cores, enabling it to perform with a much higher throughput than a CPU, but execution requires a different approach. GPU codes execute instructions using the Single-Instruction Multiple-Thread (SIMT) approach in which a single instruction is used for groups of threads called *warps*. A warp typically consists of 32 threads which must execute the same set of instructions, although on separate threads. Alternately, a CPU has far fewer cores that are much more flexible in their operation, excelling at quickly performing more complex serial computations. This is why GPUs have greater throughput when properly utilized. The second important difference is seen when comparing their memory spaces. Limited memory allocations and CPU–GPU communications cause a significant bottleneck in GPU-accelerated programs. Further study was required to properly take advantage of GPU resources. A comprehensive analysis of code performance and the model-specific features of turbulence constitutes the core of this work.

In this study, a DNS simulation of HFIR channel turbulence was performed with the finite volume CFD code OpenFOAM v2112 and CUDA v11.0 on Red Hat Enterprise Linux v8.2. The OpenFOAM installation had AMGx integrated to enable GPU acceleration and utilizes the PETSc4FOAM library. The computational resources and the problem size were scaled on CPU and CPU + GPU architectures to gain a better understanding of the performance of a DNS problem on modern computing hardware. The study aimed to analyze the scaling of the code exclusively on CPUs and then to examine the scaling of the codes with GPU acceleration enabled. Scaling studies included CPU and GPU acceleration on a mesh of varying resolution to analyze the impact of problem size relative to computational resources.

In the course of preparing the GPU configuration on Summit, mainly using the AMGX solvers, difficulties were encountered stemming from constant changes resulting from extensive ongoing development activities and the changing environment. This resulted in the inability to complete the GPU portion of the work. The code was compiled and tested, but production runs to assess acceleration were not performed because the used discretional compute time allocation expired as year-end approached. The Summit HPC platform is scheduled for decommissioning in 2024, making it unattractive for future use with Nvidia-based GPUs. Therefore, the work will be moved onto NERSC machines in FY24. An

application was prepared and submitted, and sufficient node-hours were awarded to continue the research in the next calendar year. This report summarizes work performed thus far, which mostly focused on CPU OpenFOAM computing.

2. OPENFOAM

The Open Field Operation and Manipulation (OpenFOAM) library is an open source CFD package written in C++. The codes are designed to be usable in a wide variety of applications for preprocessing, solving, and postprocessing numerical simulations, and they are also used in academia and industry. OpenFOAM's primary purpose is to create applications that can be categorized as either solvers or utilities. In continuum mechanics, solvers iteratively refine a solution until the value converges or a set number of iterations has been completed. Utilities are applications that perform functions unrelated to solving flow fields but are dedicated to creating a mesh or decomposing a mesh into subdomains for parallel processing.

By default, OpenFOAM is a serial program run entirely on CPUs, but it does have options for parallel computations which require the mesh to be divided into as many subdomains as there are processes to be executed in parallel. The decomposePar and redistributePar utilities were used to decompose the mesh into the desired number of subdomains and to redistribute the necessary information into the directories containing the processor information.

GPU acceleration of OpenFOAM was attempted by integrating NVIDIAs AMGx solvers in the program to offload the linear solver portion of the computations from the CPU onto the GPUs. Although AMGx enables parts of the computations to be offloaded from the CPU onto the GPUs, the CPUs are still required to construct the matrices based on the discretization schemes. CPUs are entirely responsible for pre- and post-processing in OpenFOAM. After the solving process begins, the AMGx wrapper copies the linear system onto the GPU for the AMGx solvers to compute [3]. When the linear system has been solved, the data are then offloaded from the GPU.

A major advantage of AMGx is its support for various solvers and its flexibility in being controllable using a configuration file that can be altered without the need to recompile the entire program [3], thus enabling the user to efficiently tailor solvers to the problem at hand. This step was found to cause a bottleneck because of the ongoing software development. The plug-in library PETSc—known for its high performance and scalability— was adopted, thus giving OpenFOAM access to state-of-the-art solvers and preconditioners [4]. In this study, GPU acceleration was planned by embedding AMGx in PETSc4FOAM, which is a PETSc plug-in for OpenFOAM. To maximize performance of the FOAM2CSR plug-in, an accelerated version of the LDU2CSR algorithm designed for OpenFOAM was used to convert between the default OpenFOAM matrix format (LDU) and the CSR matrix format used by AMGx [5].

3. SUMMIT ARCHITECTURE

The DNS simulation was run on the Summit supercomputer at Oak Ridge National Laboratory (ORNL), which is managed by OLCF. Summit has three types of nodes: log-in nodes for scripting, launch nodes for launching jobs, and the compute nodes where the jobs are run. The compute nodes contain two IBM Power9 processors and six NVIDIA Tesla V100 GPUs, as well as 512 GB of DDR4 RAM. Each Power9 processor contains 22 SMCs (SIMD Multi-Cores), with one SMC per socket being reserved for other system processes, thus providing 42 total cores per node, or up to 168 threads [6]. Each NVIDIA Tesla V100 GPUs has 16 GB of high bandwidth memory and 80 streaming multiprocessors (SMs) which contain 32 double-precision (DP) cores, 64 single precision (SP) cores, and 8 tensor cores each [6]. Using

one thread per core, each of the 80 SM GPU has one DP warp, two SP warps, and eight tensor cores to carry out independent parallel computations as compared to the 22 cores per CPU that can carry out one independent instruction each, although far more flexible.

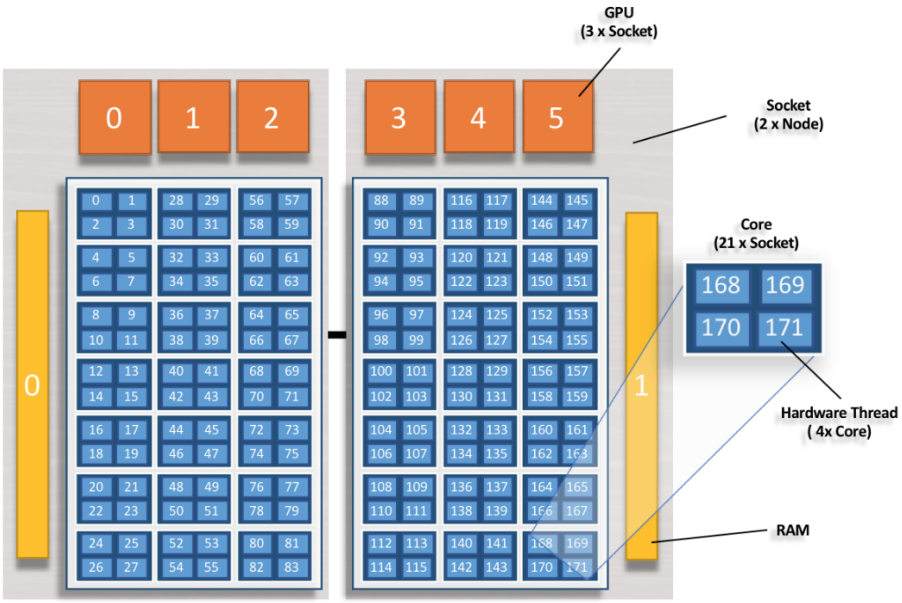

Figure 1. Summit node hardware configuration.

Summit nodes are divided into two sockets on which the components are housed, as shown in the light gray boxes in Figure 1. Each socket contains half the node's total resource: a CPU, three GPUs, and 256 GB of RAM. The division between the two sockets results in slower communication between the resources within the sockets. CPU–GPU communications are a common bottleneck for accelerated CFD simulations. To minimize potential communication delays within nodes, resource sets were utilized for the GPU-accelerated cases to divide and alter a node's appearance for a job in a manner that minimizes unnecessary communication between sockets. In this case, resource sets were used to divide nodes so that they will reduce message-passing interface (MPI) communications, and also to ensure (1) that they prioritize CPU–GPU interface by creating resource sets that do not span across sockets, and (2) that they choose resources that provide the most efficient paths for CPU–GPU communication.

4. OPENFOAM DNS STUDIES

Before moving to the scaling assessment, it was deemed important to confirm that OpenFOAM produces the same or very similar results to the baseline database results reported previously. Models and computing grids were developed, and runs were performed at OLCF Summit. This part of the work is reported in the section below.

4.1 COMPUTATIONAL CYCLIC (PERIODIC) DNS GRID

The DNS analysis aimed at replicating the results obtained previously with the PHASTA finite element software to support RANS data–informed modeling [1]. The DNS simulations were computed using the finite volume code OpenFOAM for solving the Navier–Stokes equations. For the DNS study, a computational grid was developed that was identical to that used in the PHASTA simulations. The grid was composed of hexahedral elements with curved edges to capture the involute shape of the HFIR

channel. For its generation, a simple domain decomposition was applied with the *blockMesh* utility provided as part of the OpenFOAM package. More information about the mesh generator can be found in the literature [7]. This grid generation method provided full control over the element distribution in any direction and allowed many elements to be generated as necessary to perform the DNS simulations. The geometry was specified using a combination of straight lines and high-order splines to match the dimensions of the HFIR channel. In total, 19 interpolation points were used to define each spline, and all arc lengths were verified to ±0.01 mm of nominal dimensions. Specifics on mesh refinement are provided below.

The DNS domain was chosen with the same length in the streamwise direction as the domain that was used to generate the HFIR flow database [1]. The domain encompasses the entire HFIR channel, with an axial (streamwise) length of 8δ, where δ is the half gap width. A cyclic (periodic) boundary condition was imposed on the inlet and outlet faces which translates the solution fields between these two faces. Initially, a coarse mesh was generated in which the first computational node for both gap-wise and side-wise walls was placed at $y^+=2$ within the viscous sublayer. This resulted in a total of 23.9 million cells configured as 153×68×2,296 in the streamwise, gapwise, and spanwise directions, respectively. The mesh was graded near the walls to capture the elevated shear. The boundary layers are shown in Figure 2. This initial grid was developed to obtain a converged solution before being refined to final resolution. Two consecutive refinements were performed, thus increasing the wall resolution to $y^+=0.5$. The final mesh dimensions are shown in Table 1, and the number of elements and compute partitions are given in Table 2, along with the database and RANS meshes. The reduction of computing resources by using simplified turbulence modeling is evident from the table. Developing a turbulence-averaging model of HFIR flow is one of the final objectives of this study. The database will be used to inform RANS model development.

The boundary conditions applied to this problem are listed in

Table 3. A pressure gradient force was applied to maintain a constant mean velocity of 16.3 m/s, thus corresponding to the mean velocity in the database [1].

Table 1. DNS cyclic mesh details compared to the database and RANS meshes

Problem	Reference	Re_τ	Lx/δ	Lz/δ	L^+x	L^+z	Δx^+	Δz^+	Δy_w^+	Δy_c^+	Ny
DNS	Present	960	2.6π	37.6π	7,682	113,556	12	12	0.5	12	272
RANS	Present	960	480	37.6π	437,745	113,556	2,155	2,155	20	2,155	21
Database	[1]	960	2.6π	37.6π	7,682	113,556	12	12	0.5	15	184

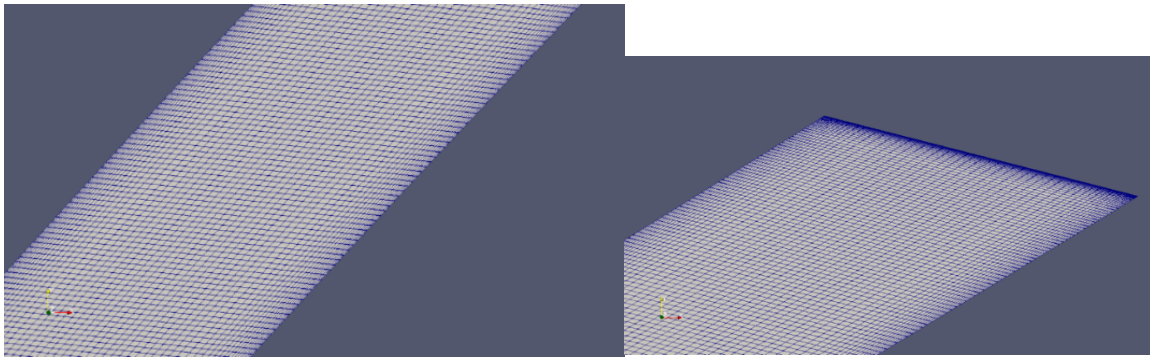

Figure 2. Large cyclic DNS grid: channel center (left), and channel corner (right).

Table 2. Grid refinement steps, mesh elements (millions), and computational partitions

Grid	DNS	Database mesh
Coarse	24 / 128	154 / 768
Intermediate	191 / 256	1,232 / 6,144
Fine	1,527 / 2048	9,809 / 49,152

Table 3. Boundary conditions for the DNS problem

Face	$p\left[\frac{m^2}{s^2}\right]$	$U\left[\frac{m}{s}\right]$
Inlet	cyclic	cyclic
Outlet	cyclic	cyclic
Walls	zeroGradient	noSlip

4.2 RESULTS AND COMPARISON TO THE DATABASE

The coarse mesh described above was initialized with a uniform streamwise velocity of 16.3 m/s. Nine blocks within the domain were temporarily defined as stationary obstacles via the topoSet utility to initialize turbulence, as shown in Figure 3. The blocks remained in the domain until turbulence visibly diffused throughout. Then the blocks were removed, and the simulation was allowed to progress until the low-velocity patches left by the blocks had visibly dissipated. At this point, the mesh was uniformly refined, and the solution field was mapped to continue the simulation on the intermediate mesh. The uniform refinement resulted in a 191 million cell mesh which was executed on 256 processors.

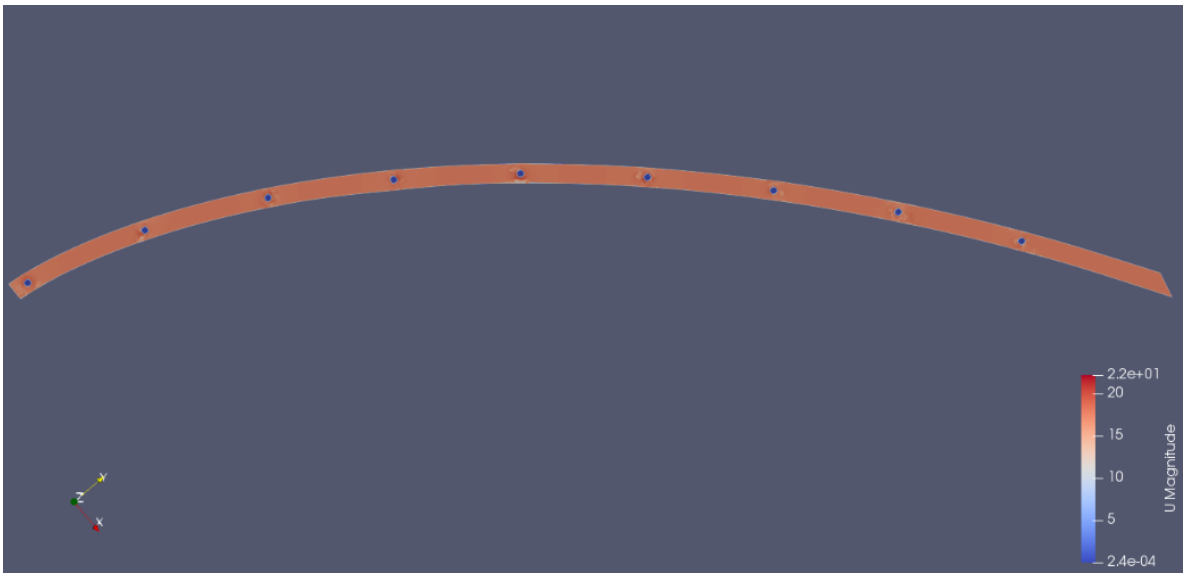

Figure 3. Initial DNS velocity field with turbulence-initializing blocks.

The intermediate mesh was allowed to develop turbulence and to achieve a steady, developed flow condition. The simulation was run for t^+=84, where the nondimensional time is defined as: $t^+ = t\, u_\tau/\delta$. This run time corresponds to approximately 205 residence times, which is sufficient to obtain steady turbulence. The simulation was run in a constant velocity mode to compute the pressure driving force.

Based on this variable, an approximate pressure loss was determined and is plotted in Figure 4 for the entire period of the intermediate mesh run. The pressure drop fluctuates at approximately 69 psi, which is close to the expected core pressure drop of approximately 80 psi, assuming approximately 20 psi channel inlet and exit losses. The total pressure loss measured over the HFIR core was approximately 100 psi. The lower-than-expected pressure drop in the intermediate mesh runs resulted from the mesh not being fine enough to properly predict the correct dissipation. As soon as the mesh is refined to its final resolution, it should reach a pressure drop close to the actual pressure drop. This encouraging result shows that the simulation is capable of predicting real operational values.

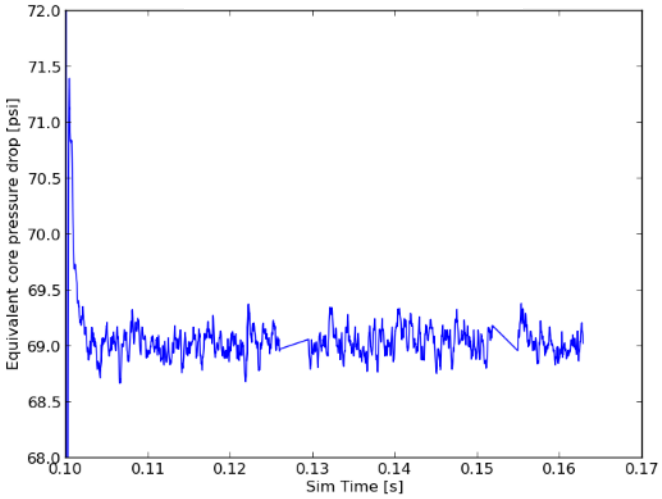

Figure 4. Pressure drop extrapolated to the actual HFIR core size.

The intermediate mesh allows the turbulent velocity eddies to be visualized primarily because of its relatively small size (191 M). The instantaneous velocity snapshots in the channel and in the sharp corner are shown in Figure 5. In the left image, the entire domain is visible. The merit of these plots is dubious because they do not provide a quantitative measure of velocity or turbulence. The velocity snapshots only confirm that the flow is indeed turbulent and that wall boundary layers develop.

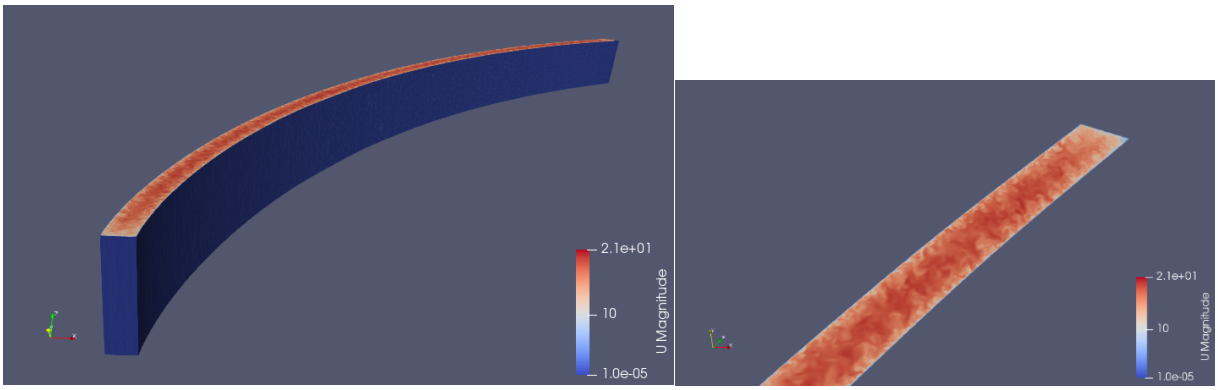

Figure 5. Snapshot of developed instantaneous velocities in the entire domain (left) and in the low curvature corner (right).

To quantitively assess the turbulence, the instantaneous velocity data were statistically processed, and some major properties are discussed and compared to the database in the paragraphs below.

It is not practically feasible to use all data for each element to accumulate statistics for time averaging. Over 8,450 virtual probes were inserted into the domain at approximately the same locations as in the database (Figure 6) to accumulate data. Note that the locations are approximately the same as a result of the generation of the domain via blockMesh, where the walls do not share the exact same coordinates as in the database version. This should not affect the accuracy of comparison.

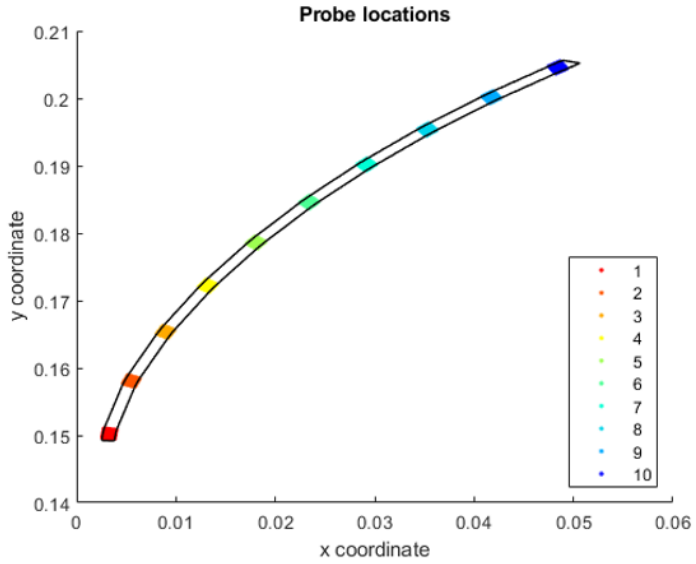

Figure 6. Virtual probe locations in the database and current the DNS study.

The simulation was then allowed to progress for t^+=18 (27,000 time steps) to accumulate approximately 232 million datapoints for statistics, the results of which are plotted in Figure 7.

Analysis of results compared with the database results indicates a lower dissipation computed by OpenFOAM, which leads to elevated turbulence intensity. Although the turbulent kinetic energy (TKE) Figure 7(b) is generally lower in the OpenFOAM simulation, combined with the much lower dissipation Figure 7(c), it produces a velocity profile Figure 7(a) that does not match accurately with the database profile. A measure for this mismatch is the specific dissipation rate Figure 7(d), which is much lower in the OpenFOAM run. This behavior was expected because the mesh has not been refined to final resolution. The discrepancy should disappear when the final mesh density is achieved.

Figure 7(e) depicts a property of major interest in RANS turbulence modeling: the turbulent viscosity. Turbulent (eddy) viscosity is the ultimate quantity returned to the flow solver from the RANS model for all models based on eddy viscosity. Here, the turbulent viscosity was computed using the DNS results, and it represents a benchmark for the RANS modeling. It is provided as an illustration for the capability of the DNS level of simulations to be used when setting objectives for low order models. It is important to note that both DNS simulations produce a qualitatively identical result, which is a measure of method coherence.

The reported work was performed at a division computing platform with limited resources and did not allow resolution to be increased further to achieve the planned final mesh size. The simulation was moved for computing to the Summit OLCF machine, and the computations were continued there. The mesh was successfully refined to its final resolution of 1.53B elements on 2,048 partitions, but that was the limit, as discussed below. The project used an allocation of compute time that was not sufficient to perform all runs with the fine mesh. The work is ongoing and will continue into the next year, when a larger compute

time is expected to be awarded to the project. A proposal was submitted to the Energy Research Computing Allocation Proposal (ERCAP) program, and an allocation of 5,320 node-hours was received.

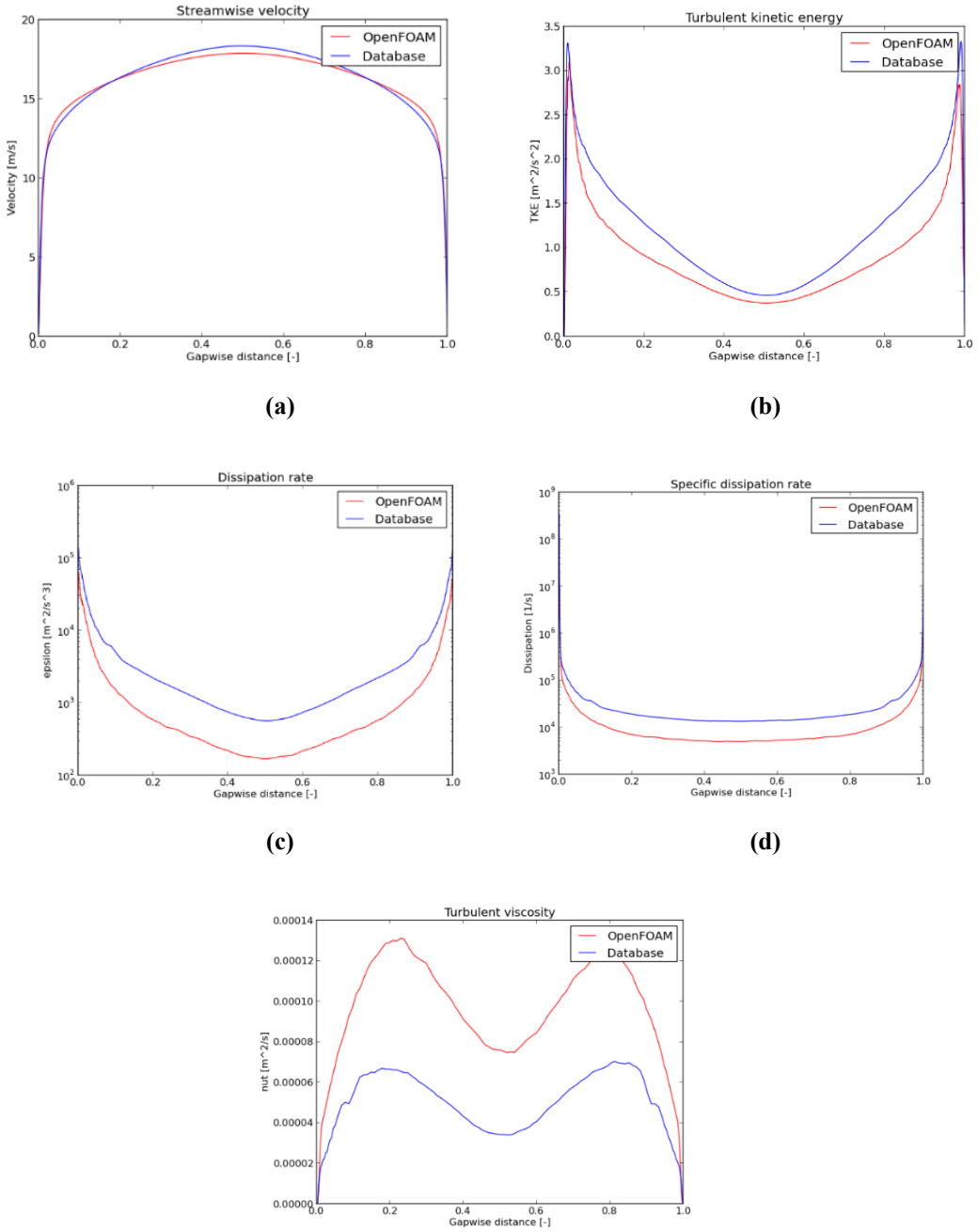

Figure 7. Intermediate mesh results of OpenFOAM DNS compared against database results.

5. OLCF SUMMIT KEY PERFORMANCE METRICS

This scaling study focused on the strong scaling of CPU simulation on an intermediate mesh and a fine mesh. The intermediate mesh contained 191 million cells, and the fine mesh contained approximately 1.53 billion cells. The intermediate mesh CPU simulation was benchmarked at 8 nodes and was scaled up to 64 nodes. The fine mesh study included only multi-threading evaluation because it was impossible to redistribute the mesh over more than 2,048 ranks because of a face/cell numbering issue related to exceeding the 32-bit (2^{31}) integer limit. To overcome this problem, OpenFOAM must be recompiled with 64-bit integers, and the base mesh (coarse) must be regenerated. In effect, this sets the problem back to its initial step. In view of the current state of the project, this work was deferred until FY24, when more compute time is available. The CPU simulation used the Geometric Agglomerated Algebraic Multigrid (GAMG) solver with the GaussSeidel smoother for pressure with no preconditioner. Velocity was solved using the smoothSolver and the same smoother with no preconditioner.

Metrics were used to provide a comprehensive understanding of the simulation scaling behavior and to determine the effectiveness of scaling a specific algorithm or architecture. The metrics measure the performance and efficiency of a system as the computational resources or the problem size increase. Beyond quantifying performance as the problem scales on a system, these metrics can also be a valuable tool in understanding what impacts system performance and hinders its scalability.

The first and most important performance metric that was measured in this study was the speedup, or the relative factor by which the computation time per time step reduces in relation to the benchmark case for a given scaling method. The primary method of scaling that was analyzed was the strong scaling, in which the achieved speedup was compared with the ideal speedup. (Theoretical speedup was based on number of processing units.) This was accomplished by analyzing the scaling speedup for a constant problem size while increasing the computational resources utilized on a given problem. The ideal speedup would be the same factor by which the computational resources were increased, or the linear scaleup. The second method for studying strong scaling was to analyze the relative change in scaling with the higher resolution meshes. Studying the scaling in this manner yielded a direct metric that illustrated how effectively the problem size and the computational resources scale on a given problem.

To measure how efficiently the computational resources are being utilized, the core hours per time step were analyzed. In the case of insufficiency of available compute power, this important factor shows the optimal configuration that allows a problem to be solved and how much the performance degrades when the allocated resource is minimized. This analysis was accomplished using multi-threading as per processor configuration on Summit.

6. OPENFOAM DNS SCALING

The simulation to be scaled is a DNS of channel turbulence in the HFIR research reactor at ORNL. The DNS used a mesh containing approximately 191 million cells. OpenFOAM was run using PISO (Pressure Implicit with Splitting of Operators), a transient solver for incompressible turbulent flow. The CPU simulations were run using the GAMG solver for pressure and the smoothSolver for velocity, and both were used with the GaussSeidel smoother and no preconditioner. The simulation was scaled by repeatedly doubling the number of nodes/threads used to run the simulation over a different number of time steps.

6.1 CPU SCALING

The benchmark simulation for the CPU scaling was run on eight nodes or 256 CPU cores without use of the GPU nodes. The mesh was decomposed into as many subdomains as the number of MPI processes.

The scaling study was performed by doubling and redoubling the nodes from 8 up to 64 nodes, whereas MPI memory allocation during redistributePar became less reliable. The results of the scaling study are shown below.

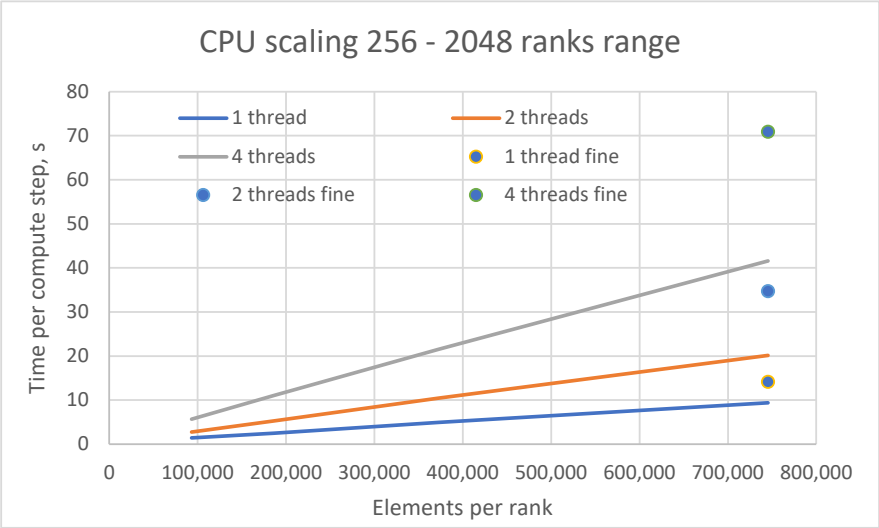

Figure 8. Strong scaling showing the linear speedup achieved with OpenFOAM on Summit.

As stated above, the CPU configuration allows for multithreading. This option was exprored as part of this assessment. The results of the speedup tests are shown in Figure 8 ranging up to the maximum number of threads per core available. The horizontal axis depicts the number of elements per core. The result demonstrates an almost perfect linear scaling within the range of ranks used. OpenFOAM performs quite well on all threads, but using hyperthreading clearly reduces problem execution linearly as well. For the largest number of elements per rank, one time step is computed for 10, 20, and slightly above 40 seconds for the three cases, which is a linear deterioration of performance. Hyperthreading seems to be more promising with a low number of elements, but more testing is needed to confirm this. Achieving a higher number of partitions was difficult because the Summit allocation had a hard time limit, and in view of the problem size, it was not sufficient to distribute over larger number of ranks.

The weak scaling—or the scaling of code when the number of elements is increased per the same number of partitions—could only be demonstrated on the 2,048 rank run because of the problems explained above. The results are indicated by the colored dots in Figure 8. The number of elements was increased eight times. The time to compute a step increased 9.67 times for the single thread and 12.5 times for both 2 and 4 threads per core. This result is quite positive in view of the large problem tested (1.53B). The single thread run time decreased almost proportionally to the increase in the number of elements, indicating that the code is capable of executing very large HPC problems.

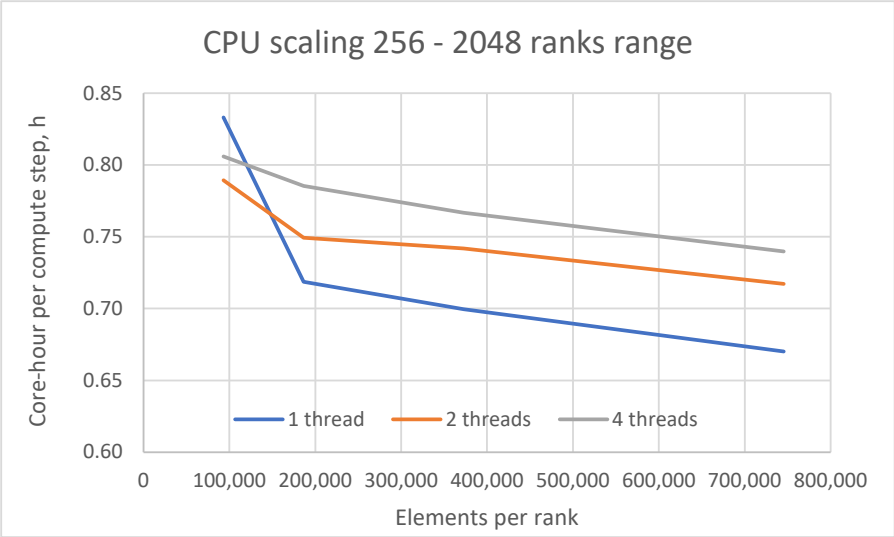

Figure 9. Resource utilization testing shows the aggregate core time for computing a time step.

The resource utilization was very coherent, regardless of threading (Figure 9). Although resource utilization varies in a narrow range, this analysis systematically illustrates that more elements per core provide better resouce usage. This result is more valid for single threading (blue), exhibiting a sharp degradation of efficiency at less than 200K elements per rank. It is notiable that any threading undergoes the same degradation, but some to a lesser extent. However, this finding should not affect performance because variation of the utilization metric is minimal.

The weak scaling of resource utilization was also computed but is not depicted in Figure 9 because it is not comparable to the data shown there. As the number of elements increased 8 times, the core-hours to compute a time-step increased to 9.41 for the single thread, 9.87 for two threads, and 10.08 for four threads per core. This corresponds to an increase ranging from 11.3–12.5 ×, which deviates from linear at approximately 40 to 55%. Apparently there is a toll to pay for running large problems which should not be impossible to bear on an HPC platform such as Summit. The result demonstrated that at elevated loads of core-elements, utilization efficiency drops, which is typical for CFD solutions requiring global commutications to solve. An optimum of resource utilization and speedup should be sought.

A summary of all statistics for the runs performed during CPU scaling is provided in Table 4. This information was used to derive the trends shown in the figures above.

Table 4. Summary of CPU scaling study runs.

Intermediate mesh CPU only			745,511	cells per partition		256 partitions				
run #	initial time	final time	steps	clock time, s	total time, s	time per step, s	based on total time	nodes	cores/node	threads
1	0.1452	0.145582	764	7179		9.40		8	32	1
2	0.14545	0.1458	700	6597	6637	9.42	9.48	8	32	1
3	0.1458	0.145978	356	7182	7229	20.17	20.31	4	32	2
4	0.145975	0.146061	172	7158	7225	41.62	42.01	2	32	4
Intermediate mesh CPU only			372,755	cells per partition		512 partitions				
5	0.145975	0.146325	700	3443	3488	4.92	4.98	16	32	1
6	0.146325	0.147025	1400	6917	6957	4.94	4.97	16	32	1
7	0.14685	0.147194	688	7177	7228	10.43	10.51	8	32	2
8	0.147025	0.147191	332	7168	7229	21.59	21.77	8	32	4
9	0.147025	0.147191	332	7159	7220	21.56	21.75	4	32	4
Intermediate mesh CPU only			186,378	cells per partition		1024 partitions				
10	0.145975	0.147225	2500	6316	6358	2.53	2.54	32	32	1
11	0.1472	0.147825	1250	6586	6629	5.27	5.30	16	32	2
12	0.1477	0.148012	624	6892	6944	11.04	11.13	8	32	4
Intermediate mesh CPU only			93,189	cells per partition		2048 partitions				
13	0.145975	0.146725	1500	2197	2248	1.46	1.50	64	32	1
14	0.1467	0.14895	4500	6665	6718	1.48	1.49	64	32	1
15	0.14895	0.14995	2000	5550	5657	2.78	2.83	32	32	2
16	0.14995	0.150582	1264	7164	7224	5.67	5.72	16	32	4

6.2 GUIDANCE ON FUTURE GPU SCALING

Previous studies have demonstrated that increasing the number of MPI processes beyond 4 per GPU becomes increasingly inefficient and in some cases is detrimental to performance [3]. Summit has the option to use resource sets within a node, which allows the user to alter a node's appearance for a particular job. Using resource sets, the nodes can be divided into multiple sub-nodes, which can be a useful approach for various reasons. In this case, the purpose is to prioritize CPU–GPU communications and to reduce MPI communications by creating resource sets that do not span sockets within the node.

The two options for scalable resource sets that would not span multiple sockets are one or three resource sets per socket. The first option is a nonbinding approach that would utilize 12 CPU cores, 12 MPI ranks, and 3 GPUs per socket in a single resource set. The nonbinding option allows any of the GPUs to work on any of the MPI ranks. This creates more work for the scheduler, but it could reduce underutilization of a GPU. This comes at the cost of potentially increasing memory transfers between CPU and GPU, which is a common bottleneck of GPU-accelerated CFD simulations.

The second option is a binding approach in which there are three resource sets in the socket, and each has four MPI ranks bound to four CPU cores and one GPU. This option reduces the workload on the job scheduler by making each core responsible for a single task and making the GPU responsible for each of those four tasks. This option may reduce the memory transfers between CPU and GPU, but it may also cause periods in which the tasks cannot fully utilize the GPUs. This problem is being minimized by selecting four MPI ranks per GPU. It is recommended that a benchmark simulation be run using each option to determine which one has the best performance for use in the GPU scaling study.

7. CONCLUSIONS

This work completes phase 3 of DNS task as defined in the LEU project work scope. The performed calculations demonstrate successful transition to an open-source platform OpenFOAM for computing HFIR channel turbulence at its most resolved scales using DNS methods. The results indicate that OpenFOAM is equally capable of computing instantaneous flow properties as the previously used PHASTA software. The comparison of major flow quantities demonstrate that this goal was achieved. The advantages of OpenFOAM are numerous with the most important ones being, a free platform not employing third party products, more advanced in terms of using hexahedral meshes (less memory intense), comprises all needed tools for massive data management, grid generation and manipulation, and more portable to large scale HPC platforms like Summit at OLCF.

The results obtained in this phase, together with the database [1] constitute the backbone of any further simplification of turbulence, like RANS, needed for engineering fast-track analyses of HFIR flows. The derived quantities (e.g., the turbulent viscosity, kinetic energy, and dissipation rate) laid the objectives for the next project phase namely, data-informed RANS modeling. The computed within the DNS study general turbulence criteria are the standard for any approximation and set the metrics required to achieve a high-fidelity and low computational cost models for improving the prediction of HFIR flow and heat-transfer conditions.

8. REFERENCES

[1] Popov E. et al., "Numerical Database of HFIR Flow Turbulence", Oak Ridge: ORNL TM-2022/2814, 2022.

[2] Bielawski R. et al., Highly-scalable GPU-accelerated compressible reacting flow solver for modeling high-speed flows, The Elsevier Ltd, 2023.

[3] T. Rathnayake, S. Jayasena and M. Narayana, OPENFOAM ON GPUS USING AMGX, Society for Modeling & Simulation International, 2017.

[4] Bna S. Spisso I. Olesen M. Rossi G., PETSc4FOAM: a library to plug-in PETSc into the OpenFOAM framework, Zenodo, 2020.

[5] M. Martineau, S. Posey and F. Spiga, AMGX GPU SOLVER DEVELOPMENTS FOR OPENFOAM, nvidia, 2020.

[6] OLCF, Summit User Guide.

[7] OpenCFD Ltd., "OpenFOAM: User Guide," 2019. [Online]. [Accessed 8 February 2023].