ECP Milestone Report

Support CEED-enabled ECP applications in their preparation for Aurora/Frontier

WBS 2.2.6.06, Milestone CEED-MS35

Tzanio Kolev
Paul Fischer
Ahmad Abdelfattah
Valeria Barra
Natalie Beams
Jed Brown
Jean-Sylvain Camier
Noel Chalmers
Veselin Dobrev
Stefan Kerkemeier
Yu-Hsiang Lan
Elia Merzari
Misun Min
Malachi Phillips
Thilina Ratnayaka
Kris Rowe
Jeremy Thompson
Ananias Tomboulides
Stanimire Tomov
Vladimir Tomov
Tim Warburton

September 30, 2020

ECP Milestone Report
Support CEED-enabled ECP applications in their preparation for Aurora/Frontier
WBS 2.2.6.06, Milestone CEED-MS35

Office of Advanced Scientific Computing Research
Office of Science
US Department of Energy

Office of Advanced Simulation and Computing
National Nuclear Security Administration
US Department of Energy

September 30, 2020

ECP Milestone Report
Support CEED-enabled ECP applications in their preparation for Aurora/Frontier
WBS 2.2.6.06, Milestone CEED-MS35

Approvals

Submitted by:

Tzanio Kolev, LLNL
CEED PI

Date

Approval:

Andrew R. Siegel, Argonne National Laboratory
Director, Applications Development
Exascale Computing Project

Date

Revision Log

Version	Creation Date	Description	Approval Date
1.0	October 5, 2020	Original	

EXECUTIVE SUMMARY

The goal of this milestone was to help CEED-enabled ECP applications (particularly ExaSMR, MARBL, ExaAM, ExaWind and E3SM) in their preparations for the Aurora and Frontier architectures. This work included collaboration with ECP vendors and porting and optimization of CEED's benchmarks and miniapps to early access hardware.

As part of this milestone, we also made the best bake-off problems and bake-off kernel implementation from Nek, MFEM, libParanumal and the external community available in the latest libCEED release, libCEED-0.7. During the milestone period we also organized, in virtual form, the fourth CEED Annual meeting (CEED4AM) which included representatives from ECP applications, vendors and software technology projects.

The specific tasks addressed in this milestone were:

- Work with vendors to port and run CEED benchmarks on early access systems for Aurora and Frontier.
- Make the best BP/BK implementations from Nek, MFEM, libParanumal and external community available in libCEED.
- Organize the next CEED Annual meeting (CEED4AM).
- Optimize CEED applications and miniapps for Aurora and Frontier architectures.

The artifacts delivered include the next libCEED release, libCEED-0.7, and a number of developments integrated within applications to improve their GPU performance and capabilities. See the CEED website, `http://ceed.exascaleproject.org` and the CEED GitHub organization, `http://github.com/ceed` for more details.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The goal of this milestone was to help CEED-enabled ECP applications (particularly ExaSMR, MARBL, ExaAM, ExaWind and E3SM) in their preparations for the Aurora and Frontier architectures. This work included collaboration with ECP vendors and porting and optimization of CEED's benchmarks and miniapps to early access hardware.

As part of this milestone, we also made the best bake-off problems and bake-off kernel implementation from Nek, MFEM, libParanumal and the external community available in the latest libCEED release, libCEED-0.7. During the milestone period we also organized, in virtual form, the fourth CEED Annual meeting (CEED4AM), which included representatives from ECP applications, vendors and software technology projects.

The artifacts delivered include the next libCEED release, libCEED-0.7, and a number of developments integrated within applications to improve their GPU performance and capabilities. See the CEED website, `http://ceed.exascaleproject.org` and the CEED GitHub organization, `http://github.com/ceed` for more details.

2. PERFORMANCE IMPROVEMENTS FOR SUMMIT

While this milestone was focused on performance improvements for Aurora and Frontier, in this section we also report on our ongoing efforts and recent results from our work on Summit and NVIDIA GPUs.

2.1 Nek5000/RS development on Summit

Nek5000/RS is a thermal-fluids code based on the spectral element method (SEM) that is used for a wide range of scientific applications, including reactor thermal-hydraulics, thermal convection, ocean modeling, combustion, vascular flows, and fundamental studies of turbulence. NekCEM supports both an SEM and an SE discontinuous-Galerkin (SEDG) formulation for applications in electromagnetics, drift-diffusion, and quantum-mechanical systems. These codes have scaled to millions of MPI ranks using the Nek-based *gslib* communication library to handle all near-neighbor and other stencil type communications (e.g., for algebraic multigrid). Tensor contractions constitute the principal computational kernel, which leads to high CPU performance with a minor amount of tuning. For portability reasons, NekRS—the GPU variant of Nek5000—was built using OCCA, the concurrent compute abstraction coming from Tim Warburton's group. OCCA supports Cuda, HIP, OpenCL, and OpenMP. For GPU-based platforms, node-level parallelism requires kernels written at a higher level than simple tensor contractions. The principal NekRS kernels are based on fast kernels developed in libParanumal, which is also coming from the Warburton group at Virginia Tech.

NekRS is performing remarkably well on Summit. A year ago, record simulation sizes for Nek5000 on Mira used about 15 million spectral elements and typical simulation times for strong-scale production level runs were about 1 second per timestep. Today, problems of E=175 million elements (with $N = 7$, for a total of $n = EN^3 = 60$B grid points) require far less than 1 second per timestep, despite the fact that the strong-scale limit for each V100 is about 1-2 million points, whereas for Mira it was about 4000 points per core.

Developments over the past few months have resulted in significant speedup in NekRS. Advances include

- Optimization of the characteristics-based timestepper, including replacing 5-stage RK time-advancement with a 4-stage variant, improved kernels, and overlapped communication. (Sustains 70% of SMEM bandwidth on the V100s.)
- Development of an extended variant of *gslib* that selects from several communication strategies, including pack/unpack on the host or device, and GPU-direct or host-based communication. Runtime adaptation picks the fastest mode depending on the data type (e.g., FP32 or FP64) and optional kernel overlap (e.g., $A\underline{u}$).
- Development of overlapping additive Schwarz (ASM) smoothing (in FP32), including standard and restricted variants, for p-multigrid. Local ASM solves are performed using the fast diagonalization method which is implemented with efficient tensor contractions.

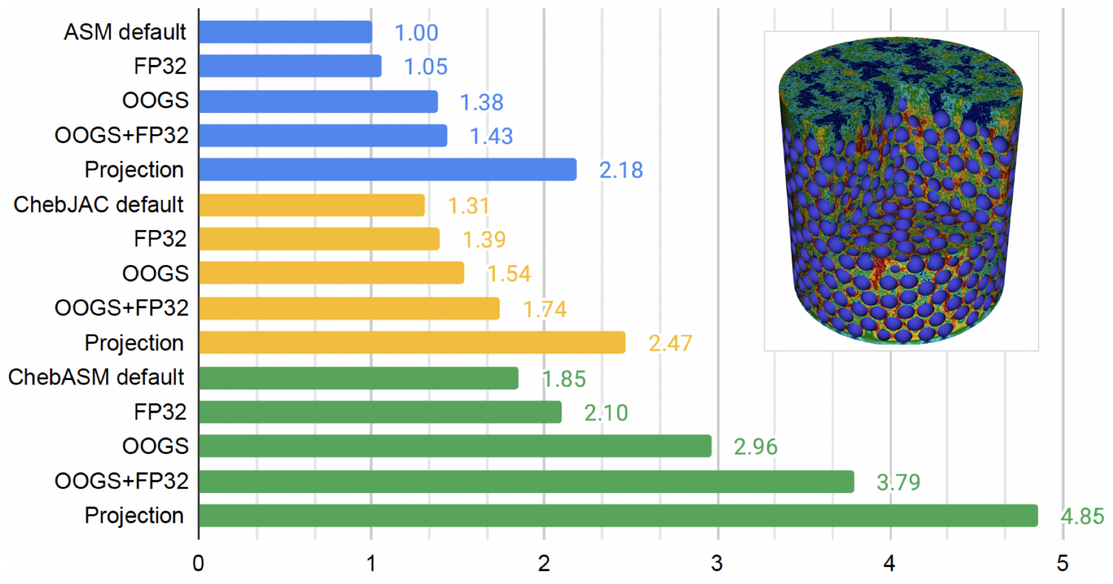


Figure 1: NekRS pressure-solve cost breakdown for turbulent flow through 1568 pebbles simulated using 66 V100 GPUs on Summit. The all-hex mesh comprises $E = 524,386$ elements of order $N = 7$ ($n = 179,864,398$).

- Development of Chebyshev-accelerated ASM smoothing. This approach combines the ASM with Chebyshev-accelerated Jacobi smoothing originally implemented in libParanumal.
- Implementation of projection-based initial guesses [10, 1].

As shown in Fig. 1 When coupled with FP32, optimized *gslib*, and projection, the Chebyshev-Schwarz preconditioner is $> 4.8\times$ faster than the baseline ASM method in isolation.

2.2 Improvements in MAGMA basis actions for NVIDIA GPUs

As report in previous CEED reports, for tensor bases, the best libCEED CUDA backend performance is achieved by operator fusion with runtime compilation in the `cuda-gen` backend. That is, the `cuda-gen` backend creates one kernel to perform the entire high-level operator application, rather than applying each sub-operator with separate kernels. In some cases, fusion may not be possible, such as when there is not enough GPU memory available for the fused kernel, or a need for a QFunction provided through an external library or source, which cannot be converted to code for runtime compilation. Thus it is necessary to have performant "stand-alone" kernels for the computationally-intensive basis actions. To that end, the CEED MAGMA team has made substantial improvements to the basis computations of the MAGMA backend.

The main idea of the new tensor-basis kernels is to formulate the tensor contractions as a series of fused device-level batch-BLAS matrix multiplications (GEMMs) operating on fast GPU memory. The device-level basis actions operate only on the shared memory or registers, and no device routine allocates shared memory or register arrays, except for temporary scalar variables. All device routines have the same thread configurations.

Prior to the latest additions to the MAGMA backend, the best non-fused CUDA backend was `cuda-shared`, which utilizes the GPU's shared memory to increase performance. In Figure 2, we show the rate of degrees of freedom (DOFs) processed per second inside MFEM's CG solver for the diffusion benchmark (BP3), for `cuda-shared` (dash/square) and the new MAGMA backend (circle/solid). The plot on the right side of the figure shows the ratio of this DOFs processing metric for the MAGMA backend compared to `cuda-shared`; the benefit of the new approach across all problems sizes begins for basis functions of order $p >= 3$, consistently reaching between 10%-20% improvement over `cuda-shared` for seventh and eighth order basis functions.

The entire basis calculation is formulated as one large GEMM for all local elements, which is then split into the optimal batch count and size for the order of the basis functions, number of quadrature points, and total number of elements. These optimal parameters are chosen by a wrapper developed from specialized

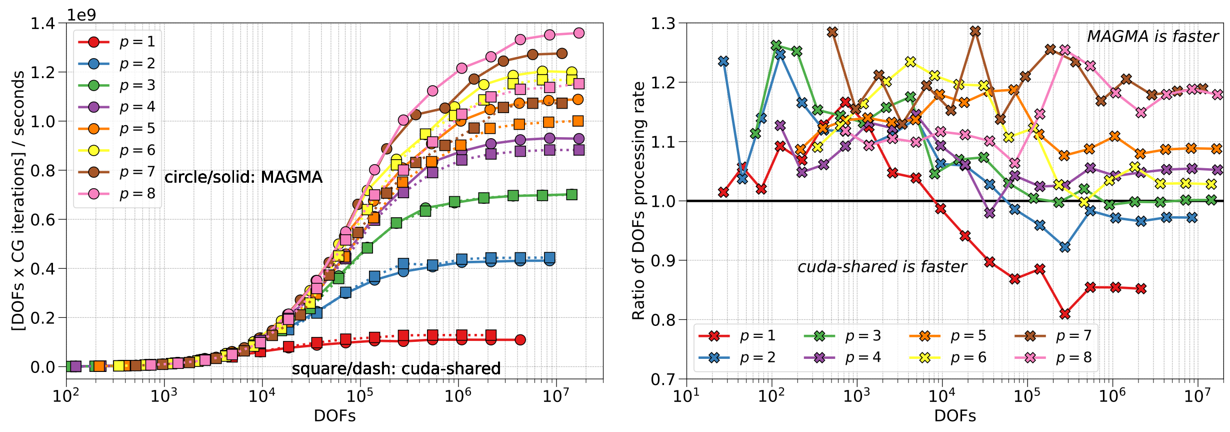


Figure 2: Left: MAGMA and `cuda-shared` backend performance for tensor-basis diffusion (BP3) MFEM benchmark in terms of DOFs processed per second on a single V100 GPU with CUDA 10.2, for basis functions of order $p = 1$ to 8. Right: The ratio of DOFs processed by MAGMA divided by `cuda-shared`.

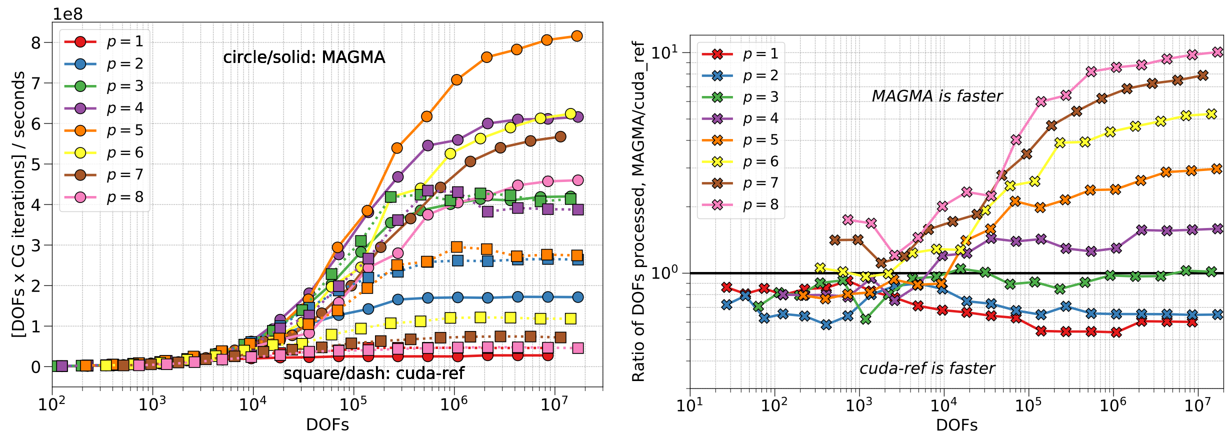
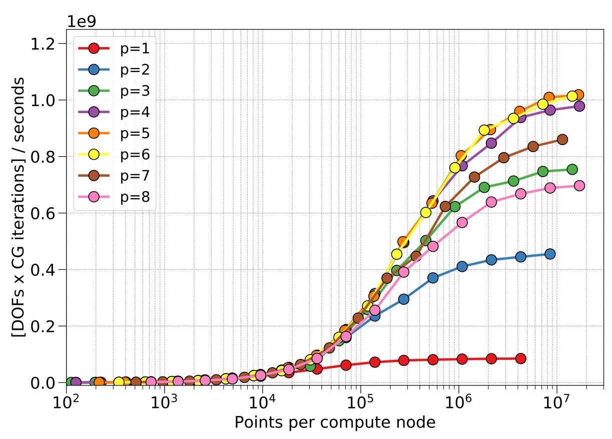


Figure 3: Left: MAGMA and cuda-ref backend performance for non-tensor-basis diffusion (BP3) MFEM benchmark in terms of DOFs processed per second on a single V100 GPU with CUDA 10.2, for basis functions of order $p = 1$ to 8. Right: The ratio of DOFs processed by MAGMA divided by cuda-ref.

tuning sweeps for the NVIDIA V100 GPU across typical sizes seen in libCEED use. The wrapper will also automatically decide whether to call the MAGMA library or cuBLAS for the batch GEMM routine. In Figure 3 we show the same diffusion benchmark results as in Figure 2 for tetrahedron elements. Starting from the meshes of the tensor benchmarks, each hexahedron element is divided into six tetrahedron elements. The results are now compared to the `cuda-ref` reference backend, as `cuda-shared` does not support non-tensor bases. The new batch-GEMM implementation is twice as fast as `cuda-shared` basis function order greater than five and greater than approximately 10^5 degrees of freedom in the mesh; up to ten times speedup is seen for eighth-order basis functions.

3. PORTING AND PERFORMANCE IMPROVEMENTS FOR FRONTIER

In this section we report recent porting and performance result from our work on systems targeting Frontier and AMD GPUs. We already reported extensively on our HIP porting efforts in Section 6.1 of the CEED-MS34



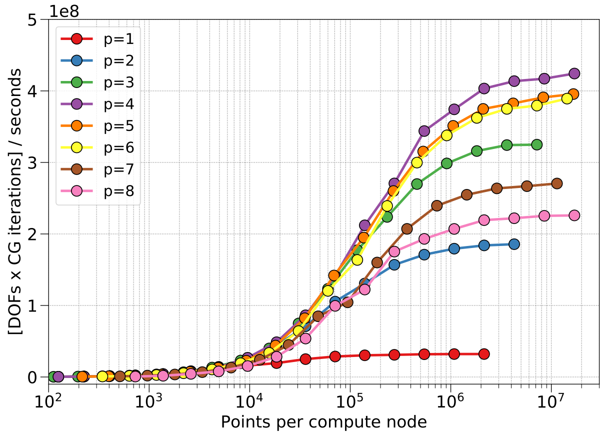


Figure 4: Left: `hip-ref` MFEM benchmark performance for the mass problem (BP1) on a single AMD MI50 GPU with ROCm 3.7, for basis functions of order $p = 1$ to 8. Right: The diffusion benchmark (BP3).

milestone report.

3.1 libCEED HIP backend

libCEED has recently added its first HIP backend, `hip-ref`. It is based on the `cuda-ref` backend. Like all the pure CUDA backends, it relies on runtime compilation of the kernels, through `hiprtc` instead of `nvrtc`. While an experimental HIP backend had been developed with ROCm 3.3, the new compiler and runtime of ROCm 3.5 improved the use and performance of `hiprtc` enough to allow the `hip-ref` backend, re-factored for ROCm 3.5, to officially be included as part of libCEED. In Figure 4, we see the performance of the `hip-ref` backend on an AMD MI50 GPU, for the mass and diffusion MFEM benchmarks problems.

hipMAGMA backend

An experimental HIP mode of the MAGMA backend is currently under development. Figure 5 shows the same benchmarks as in Figure 4 for the hipMAGMA backend. In the tensor-basis case, hipMAGMA backend shares the same device kernels as the standard CUDA MAGMA backend. In Figure 4, we see the performance of the `hip-ref` backend on an AMD MI50 GPU, for the mass and diffusion MFEM benchmarks problems.

For non-tensor basis problems, we consider the tetrahedron version of BP3, as shown for the `cuda-shared` and MAGMA backends in Section 3. The left side of Figure 6 shows both hipMAGMA and `hip-ref` for the diffusion benchmark, while the right side shows the ratio of the hipMAGMA backend's processing rate for DOFs to that of `hip-ref`. Though the overall performance of the hipMAGMA backend is lower than that of its CUDA counterpart from Figure 3, this is not surprising, as we have not yet performed the proper tuning sweeps for the batch GEMM wrapper for AMD hardware. Both the hipMAGMA and CUDA MAGMA backends achieve the best non-tensor performance for fifth order basis functions, though the gains of the GEMM approach over `hip-ref` continues to increase for higher orders of basis functions. Interestingly, the ratio between the ported hipMAGMA backend and the ported `hip-ref` backend is very similar to corresponding CUDA plot.

4. PORTING AND PERFORMANCE IMPROVEMENTS FOR AURORA

In this section we report recent porting and performance result from our work on systems targeting Aurora and Intel GPUs.

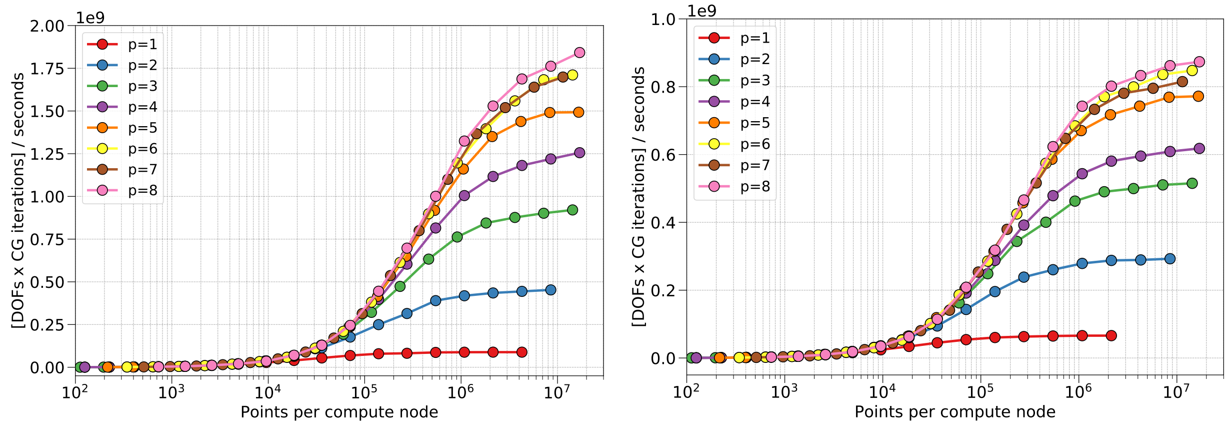


Figure 5: Left: MFEM benchmark performance of the experimental hipMAGMA backend for the mass problem (BP1) on a single AMD MI50 GPU with ROCm 3.7, for basis functions of order $p = 1$ to 8. Right: The diffusion benchmark (BP3).

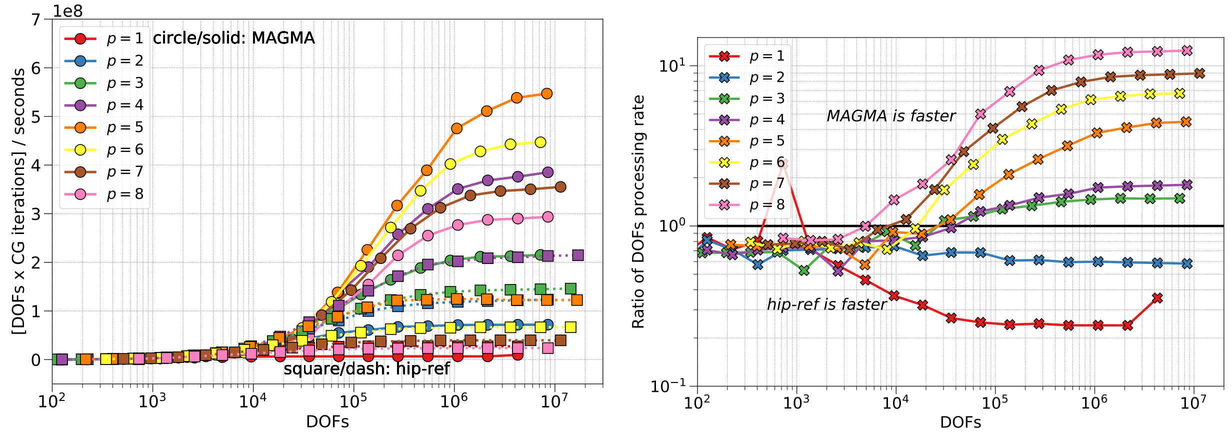
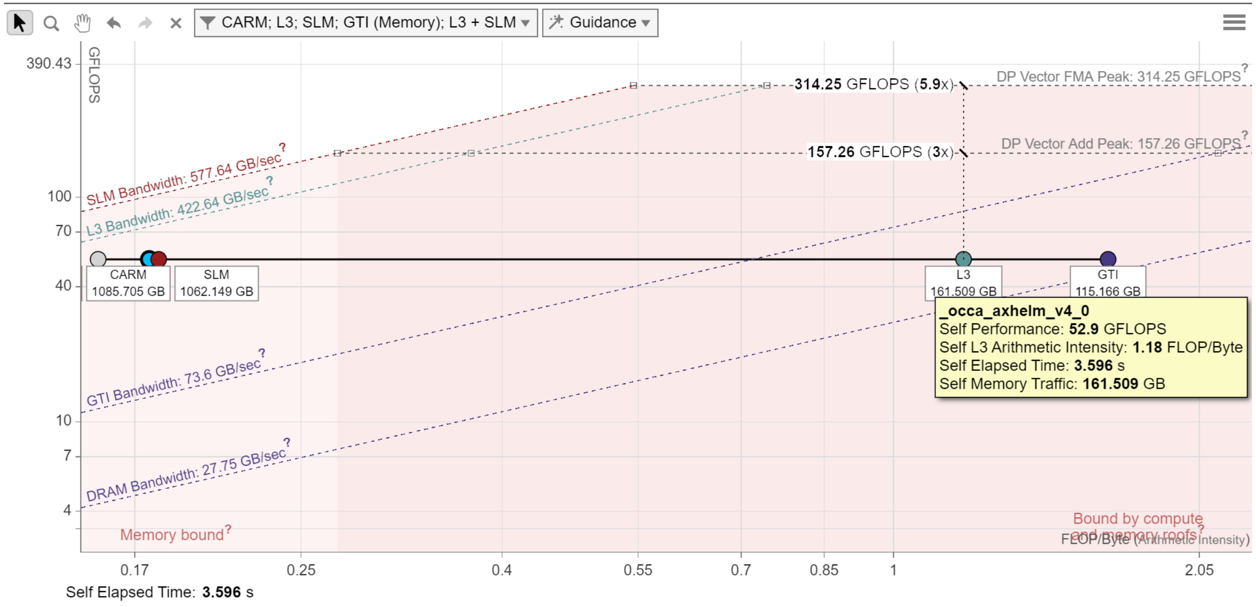


Figure 6: Left: MFEM benchmark performance of the experimental hipMAGMA backend (circle/solid) and the `hip-ref` backend (square/dash) for the diffusion problem (BP3) on a single AMD MI50 GPU with ROCm 3.7. Right: The ratio of DOFs processed for hipMAGMA versus `hip-ref`.

4.1 CEED benchmarks analysis with Intel Advisor

The CEED team has connections to Intel through our interactions with ALCF lead Scott Parker and CEED-dedicated staff member Kris Rowe. In addition, UIUC Ph.D. student and CEED team member Thilina Rathnayake is a part-time intern with Intel's Aurora group.

The CEED team is working to generate a battery of diagnostic tools and benchmarks relevant to high-order methods. Nekbench is a tool that directly reflects many of the performance-critical kernels in NekRS. It includes a variety of communication tests including elementary operations (e.g., ping-pong and vector reductions) and more sophisticated multi-node communication tests that reflect near-neighbor gather-scatter (halo) exchanges used in iterative solution of PDEs discretized by high-order methods. Nekbench also includes several high-performance implementations of key bake-off kernels coming from libParanumal, which is developed by Tim Warburton's group [3, 19]. For portability, libParanumal and NekRS are written in the *open concurrent compute abstraction* (OCCA), which is a pragma-augmented version of C and also coming from the Warburton group [15]. OCCA provides a lightweight wrapper around various heterogeneous



_occa_axhelm_v4_0
Self Performance: **52.9** GFLOPS
Self L3 Arithmetic Intensity: **1.18** FLOP/Byte
Self Elapsed Time: **3.596** s
Self Memory Traffic: **161.509** GB

Figure 7: Intel Advisor roofline analysis for NekBench kernel BK5, v4.

programming models or back-ends, including OpenMP, OpenCL, CUDA, HIP, and Metal.

For Aurora, we have worked with Kris Rowe to specify relevant benchmarks, particularly as they apply to NekRS and dependent ECP applications (e.g., ExaSMR). Kris has explored several variants of the BK5 kernel with the OpenCL backend to see if this approach will be viable on Aurora. (Currently OpenMP does not support device off-load.) BK5 executes k iterations of the 3D spectral element Poisson operator (cf. (4.4.7) in [5]) for E elements of order p. Per iteration, the leading-order costs for this kernel are $12E(p+1)^4+18E(p+1)^3$ operations and $7E(p+1)$ memory references. The fp64 flop-to-byte ratio is $\approx (.214p+.34)$. Trials were run on an Intel Iris node at Argonne with $k=1000$, $E=3200$, and $p=7$ (corresponding to a flop-to-byte ratio of ≈ 2). Presently, the best version of this kernel achieves 52.9 Gflops, whereas the roofline analysis provided by Intel Advisor (Fig. 7) indicates that 70-100 Gflops should be achievable with further tuning. As our target is the actual Aurora architecture and not Iris, we are not pursuing additional tuning until access to Aurora hardware is available.

Another development path for Aurora is a direct OpenMP port of BP5. This bake-off problem uses the BK5 kernel inside a Jacobi-preconditioned conjugate gradient (CG) iteration. As such, it tests the gather-scatter communication and global vector reductions as well as the additional streaming operations required for CG. As this work is part of Thilina Rathnayake's Intel internship, results are proprietary at this time but will be available in the near future.

5. LIBCEED-0.7 RELEASE

libCEED version 0.7 was released in September 2020. Notable features of this release are a new HIP backend, a rebuilt and improved OCCA backend to facilitate future performance enhancements, improved GPU backends, and improvements to the suite of PETSc bakeoff problems (BPs) to reduce noise due to multiple calls to `mpiexec`.

Several interface changes and additions are present in this release: Linear operators can be assembled as diagonal matrices or point-block diagonal matrices with `CeedOperatorLinearAssembleAddDiagonal` and `CeedOperatorLinearAssembleAddPointBlockDiagonal`, respectively, for improved integration with codes such as MFEM that compose the action of `CeedOperator`s external to libCEED; `CeedQFunctionContext` object was added to manage user QFunction context data and reduce copies between device and host memory; `CeedOperatorMultigridLevelCreate`, `CeedOperatorMultigridLevelCreateTensorH1`, and `CeedOperatorMultigridLevelCreateH1` to facilitate creation of multigrid prolongation, restriction, and coarse grid operators using a common quadrature space; the function `CeedVectorTakeAray` was added to sync and remove libCEED read/write access to an allocated array and pass ownership of the array to the caller (this function is recommended over `CeedVectorSyncArray` when the `CeedVector` has an array owned by the caller that was set by `CeedVectorSetArray`); and finally, the diagonal assemble interface changed to accept a `CeedVector` instead of a pointer to a `CeedVector` to reduce memory movement when interfacing with calling code. In this release, the `/gpu/cuda/reg` backend has been removed, with its core features moved into `/gpu/cuda/ref` and `/gpu/cuda/shared`.

Performance on CPUs and GPUs

The best performing CPU backends in libCEED use the LIBXSMM library. Recent performance enhancements in the suite of PETSc BPs in libCEED led to the throughput results shown in Figure 8 on AMD EPYC 7452. On CUDA GPUs, the best performing backend uses NVRTC, with performance shown in Figure 9. A HIP backend has also been developed, but is currently less optimized than this CUDA backend. We are also experimenting with Scalable Vector Extension (SVE) for ARM CPUs.

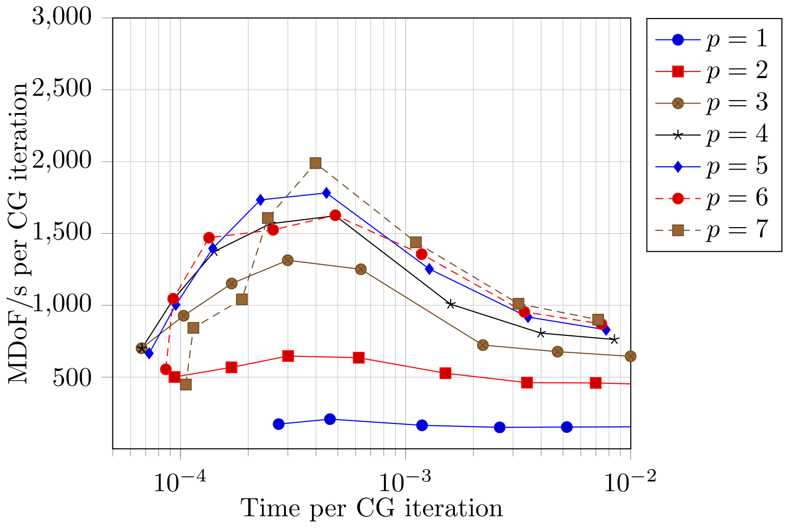


Figure 8: Throughput vs latency for the libCEED `/cpu/self/xsmm/blocked` backend solving BP3 on a 2-socket AMD EPYC 7452.

6. APPLICATION COLLABORATIONS

We perform full application simulation and analyze its performance and predict further speedup potential and extension to a wide range of applications. We demonstrate recent developments and results from collaboration with application teams including ExaSMR, MARBL, ExaWind and ExaAM.

6.1 NekRS performance on full machine of Summit for ExaSMR

Here we consider ExaSMR reactor problems. Figures 11– 14 show a variety of diagnostics from large-scale simulations on Summit for two ExaSMR test cases. The first case (CASE I) is the *full-core* reactor bundle, which comprises 37 arrays, each comprising a 17×17 array of fuel rods. The NekRS simulations model the flow around these rods, with the primary flow direction being parallel to the rods. The second case (CASE II) consists of a single (long) 17×17 fuel bundle. Both cases were run out to $E = 175$M elements—roughly twelve times larger than 15M-element "hero calculations" performed on Mira just a year ago. The tremendous size of these computations puts pressure on all aspects of the NekRS workflow, not just the PDE solver, so we are

case	node	rank	E	E/gpu	N	nstep	Δt	CFL	$t_{step}(s)$	avg $t_{step}(s)$
I	1810	10860	174233000	16044	7	100	3.0e-04	0.58	1.65e-01	2.17e-01
	2715	16290	174233000	10696	7	100	3.0e-04	0.58	1.43e-01	1.39e-01
	3620	21720	174233000	8021	7	100	3.0e-04	0.58	9.48e-02	1.18e-01
	4525	27150	174233000	6417	7	100	3.0e-04	0.58	9.34e-02	1.22e-01
II	2536	15216	175618000	11542	7	100	3.0e-04	0.56	1.19e-01	1.51e-01
	4608	27648	175618000	6351	7	100	3.0e-04	0.56	8.05e-02	1.03e-01

Table 1: NekRS strong-scaling, corresponding to the blue line in Figure 11, top-left, for a full core reactor geometry and the red line in Figure 11, top-left, from ExaSMR, using 6 GPUs per node on Summit. Timings are in seconds for the walltime per step, t_{step}, at 100 and the averaged-walltime per step, avg t_{step}, using 101-200 steps. We used dealiasing with 9th-order, projection in time, CHEBYSHEV+ASM, PCG+FLEXIBLE, pressure tol= 1.e-04, velocity tol= 1.e-06, $Re = 5000$, and BDF3+EXT3. **CASE I** represents the full-core reactor mesh ($E = 174,233,000$) consisting of 37 arrays of 17×17 rod bundles using 1700 layers of two-dimensional full core mesh (E=$1,024,900$). **CASE II** represents 17×17 rod-bundle mesh ($E = 175,618,000$) using 6340 layers extruded from a two-dimensional 17×17 rod bundle (E=$27,700$).

case	node	rank	E	E/gpu	N	nstep	Δt	CFL	$t_{step}(s)$	avg $t_{step}(s)$
	271	1626	10249000	6303	7	100	3.0e-04	0.58	6.16e-02	6.58e-02
	813	4878	30747000	6303	7	100	3.0e-04	0.58	7.96e-02	9.68e-02
I	1626	9756	61494000	6303	7	100	3.0e-04	0.58	8.64e-02	1.05e-01
	3253	19518	122988000	6301	7	100	3.0e-04	0.58	9.59e-02	1.18e-01
	4608	27648	174233000	6301	7	100	3.0e-04	0.58	9.01e-02	1.21e-01
	87	522	3324000	6367	7	100	3.0e-04	0.56	7.53e-02	8.57e-02
	320	1920	12188000	6347	7	100	3.0e-04	0.56	8.08e-02	8.67e-02
II	800	4800	30470000	6347	7	100	3.0e-04	0.56	1.00e-01	9.11e-02
	1600	9600	60940000	6347	7	100	3.0e-04	0.56	9.75e-02	9.33e-02
	3200	19200	121880000	6347	7	100	3.0e-04	0.56	8.02e-02	9.71e-02
	4608	27648	175618000	6351	7	100	3.0e-04	0.56	8.05e-02	1.03e-01

Table 2: NekRS weak-scaling for a full core reactor geometry from ExaSMR, using 6 GPUs per node on Summit. Timings are in seconds for the walltime per step, t_{step}, at 100 and the averaged-walltime per step, avg t_{step}, using 101-200 steps. We used dealiasing with 9th-order, projection in time, CHEBYSHEV+ASM, PCG+FLEXIBLE, pressure tol= 1.e-04, velocity tol= 1.e-06, $Re = 5000$, and BDF3+EXT3. **CASE I** represents full core meshes increased in size by 100, 300, 600, 1200, and 1700 layers of two-dimensional full core mesh ($E = 1,024,900$), corresponding to Figure 12, top left. **CASE II** represents 17×17 rod-bundle meshes using 120, 440, 1100, 2200, 4400, and 6340 layers of two-dimensional mesh (E=$27,700$), corresponding to Figure 14, top left.

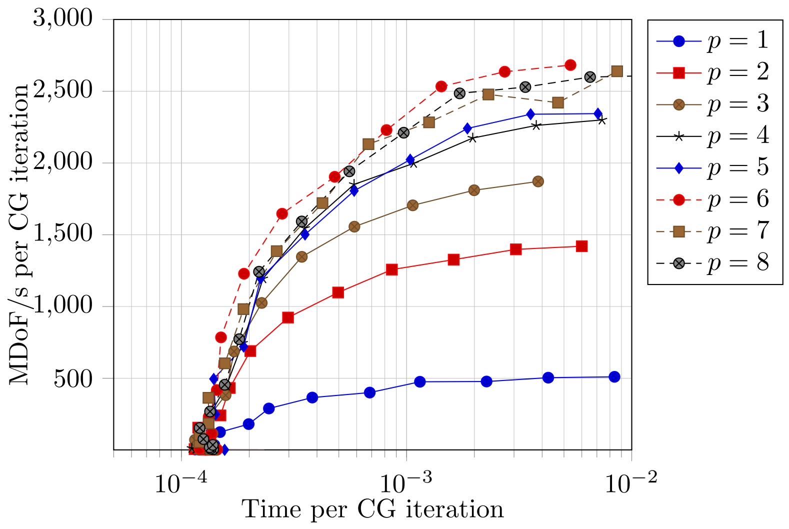


Figure 9: Throughput vs latency for the libCEED `/gpu/cuda/gen` backend solving BP3 on a NVidia V100.

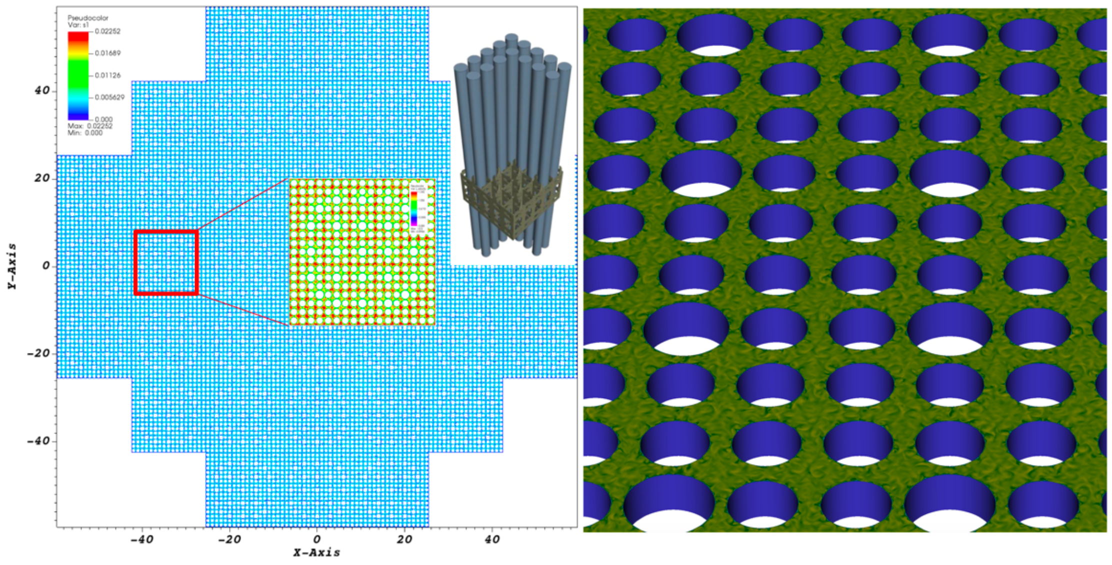


Figure 10: NekRS simulation for turbulent flows past a full-core mesh consisting of 37 arrays of 17 × 17 rod bundles.

currently tracking multiple metrics to understand the overall scalability of the targeted GPU-based exascale architectures.

Figure 11, top-left, is a log-log plot of averaged-walltime per Navier-Stokes timestep versus number of nodes (6 V100 GPUs per node) for the full-core and 17×17 rod cases. The number of elements is (approximately) E=175M for each case and the polynomial order is $N = 7$, for a total number $n = EN^3$ =60B grid points. We see that the minimum solution time for the full-core (in blue) is realized at about 2.7M points per GPU, whereas the 17×17 case continues to scale well out to all of Summit, that is n/P=2.1M, where P is the number of V100s. The scalability is more evident in Figure 1, top-right, which shows the parallel efficiency. We see that 80% efficiency is sustained for n/P=2.1M even at P=27648 GPUs. Figure 1, 2nd-left shows the walltime-per-step for the first 200 timesteps of the full-core simulation and Figure 1, 2nd-right gives a breakdown of times for the key NekRS kernels as a function of the number of nodes: *makef* corresponds to explicit evaluation of the nonlinear advection terms; *vsolv* corresponds to the 3-component velocity solves; *psolv* corresponds to the pressure solve times; and *crs* corresponds to the communication-intensive coarse-grid solve that is part of the multilevel pressure solve. Clearly, the pressure solve is the largest part of the solution cost, which is expected given the relatively long tails of the Poisson equation Green's functions. Figure 11, 3rd-left and right show the number of pressure (left) and velocity (right) iterations per step for each value of P. We see that the iteration counts are essentially independent of the number of processors, as would be expected for the algorithms in Nek5000/RS. From these plots, we conclude that the fall-off in efficiency of the full-core configuration is due to an increase in communication overhead as P is increased.

The communication increase is evident in Figure 11, 4th-right, which shows communication statistics for the internode gather-scatter operator. Two communication strategies are considered: the crystal-router [11], which is a generalized (and scalable) many-to-many, and the pairwise exchange. Both are implemented in the Nek5000 *gslib* library, which picks at runtime the fastest strategy. In this case, the pairwise strategy would be selected and the maximum time ("pwmax") is the relevant metric as it reflects the longest time to execute the pairwise exchange over all P MPI ranks. We see that there is an up-tick in pwmax as the number of ranks is increased from 3620 to 4520, which is consistent with the strong-scaling results discussed above.

Finally, we also track total problem setup time, shown as "init" in Figure 11, 4th-left. The principal constituents are the parallel recursive-spectral bisection (rsb, [18]), and the parallel read times (re2). There is also a parallel recursive coordinate bisection (RCB), which is used to presort the mesh elements prior to starting RSB. Using RCB leads to compact element sets on each rank and hence less communication when implementing the Lanczos algorithm to find the Fiedler vector required for RSB. We are currently developing a full-approximation multigrid scheme for RSB that should be significantly faster than straight Lanczos.

Figure 12 shows a similar sequence of metrics for the full-core case, but now under weak-scaling conditions where the number of elements per GPU (or node) is fixed. In this problem, one can add more layers of elements in the axial (z) direction, so there is some self-similarity that makes weak-scale studies possible. Figure 12 top-left shows the averaged-walltime per step as we weak-scale this problem from 271 to 4608 nodes (1626 to 27648 GPUs). The time increases by roughly a factor of two over this range, corresponding to a parallel efficiency of $\approx 50\%$ on all of Summit. As discussed above, this increased wall time is due to increased communication overhead, witnessed by the increase in times in Figure 12, 2nd-right and 4th-right, and by the constant iteration counts in Figure 12, 3rd-left and right.

As part of the analysis of the communication overhead, we considered a different data partition by replacing the RSB partition with one based on RCB. In Figure 13, top left, we see that this approach yields perfect weak scaling, but the results are uniformly inferior to those of Figure 12, top left, for the exact same problem. Similar conclusions hold for the other graphs in this Figure, which we include for completeness.

In contrast to the full-core case, the 17×17 case in Figure 14 shows perfect weak scaling out to all of Summit. We suspect that this scaling holds because of the geometry of this particular domain, which is very long in the axial direction. Each of the early cuts in RSB (or even RCB) will yield relatively small separators and communication overhead will be relatively low.

6.2 GPU port of MARBL's mesh optimization phase

During this milestone the CEED team developed a GPU port of MARBL's mesh optimization method. This method is based on the Target-Matrix Optimization Paradigm (TMOP), where the mesh optimization problem is posed as a variational minimization of a nonlinear functional. This variational form avoids low-level geometric operations, allowing the use meshes that are based on high-order finite elements (FE), independent of the element type, FE order, and space dimension. All mathematical details about TMOP with high-order FE can be found in [7, 8, 6] The variational formulation of the problem also allows the use of optimized partial assembly computations, which was the focus of this work. The development was originally performed in MFEM's mesh optimization miniapp, and then ported to MARBL, as both codes use the same core TMOP algorithms. Below we give a short overview of the main computational kernels that were rewritten through partial assembly and ported to GPU.

TMOP optimizes the mesh positions x, where x is a high-order FE function, by minimizing a global objective function $F(x)$ that depends on the local quality measure throughout the mesh. This minimization is performed by solving $\partial F/\partial \mathbf{x} = 0$ with Newton's method, where $\mathbf{x}$ is the vector of FE coefficients corresponding

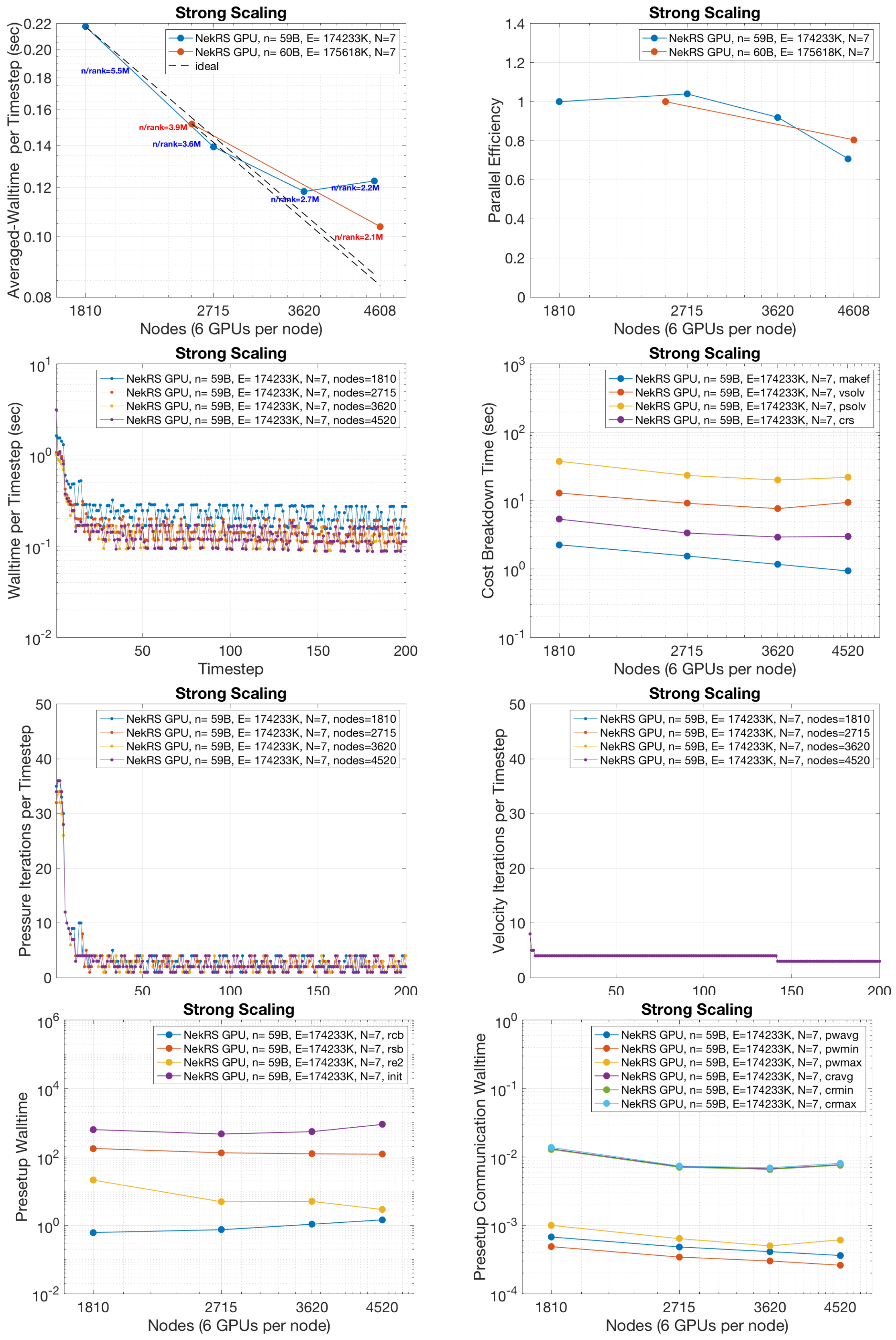


Figure 11: NekRS strong scaling for a full-core reactor mesh ($E = 174,233,000$). The blue lines in the 1st-left and 1st-right are for full-core case, compared to red lines for a (long length) 17×17 rod-bundle mesh in the 1st-left and 1st-right.

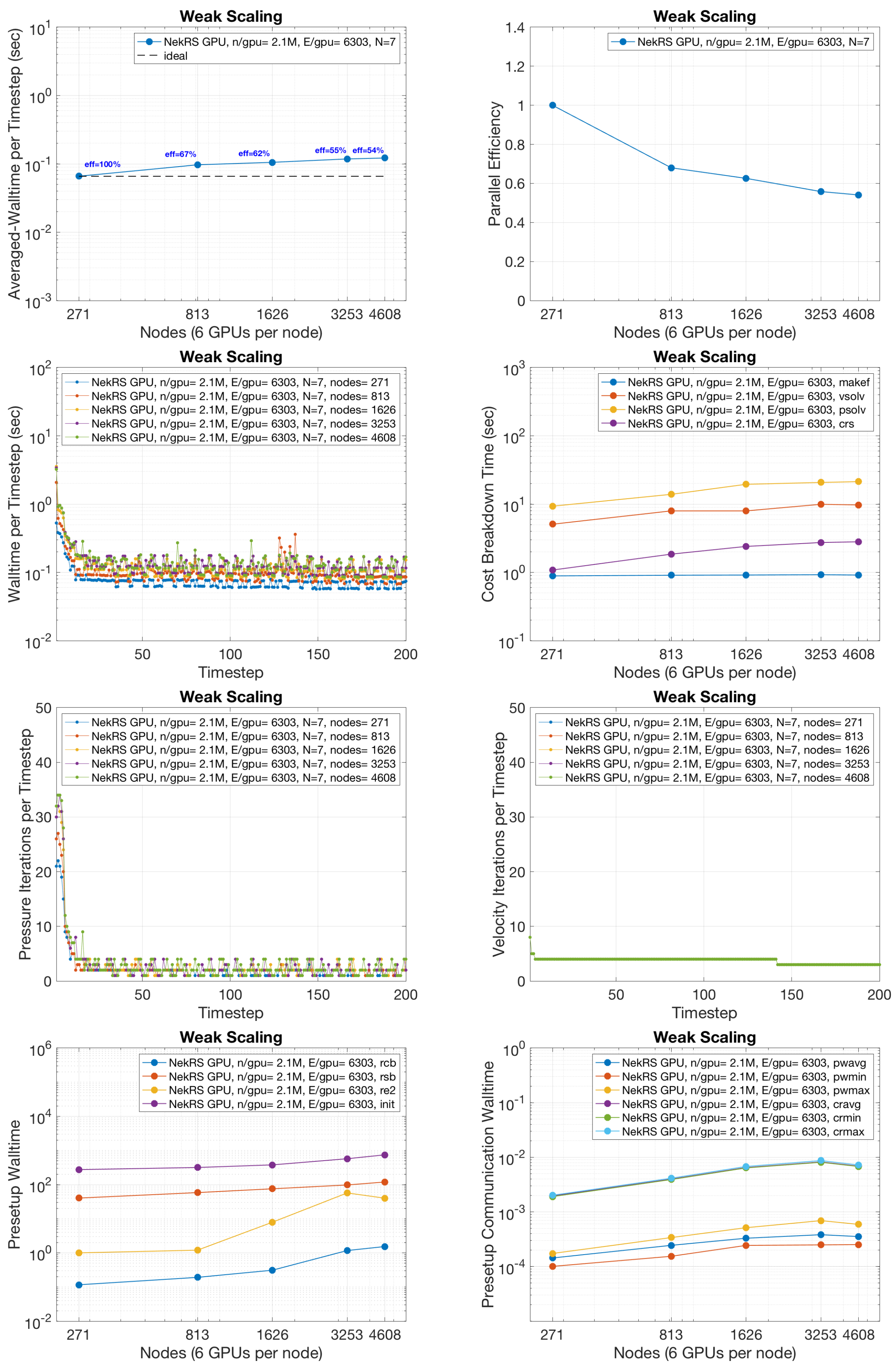


Figure 12: NekRS weak scaling for a full-core reactor mesh ($E = 174, 233, 000$).

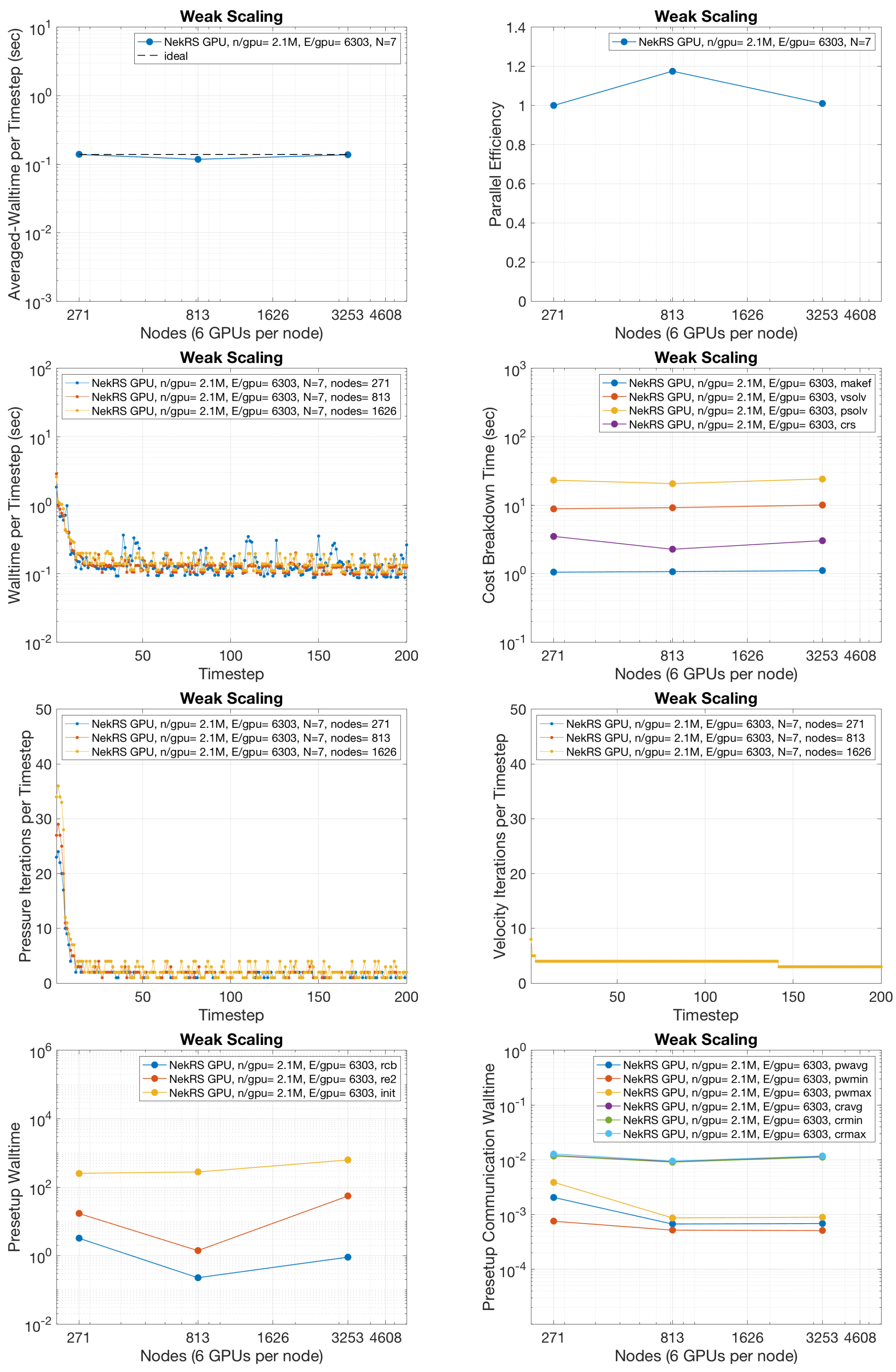


Figure 13: (RCB-only test) NekRS weak scaling for a full-core reactor mesh ($E = 174,233,000$).

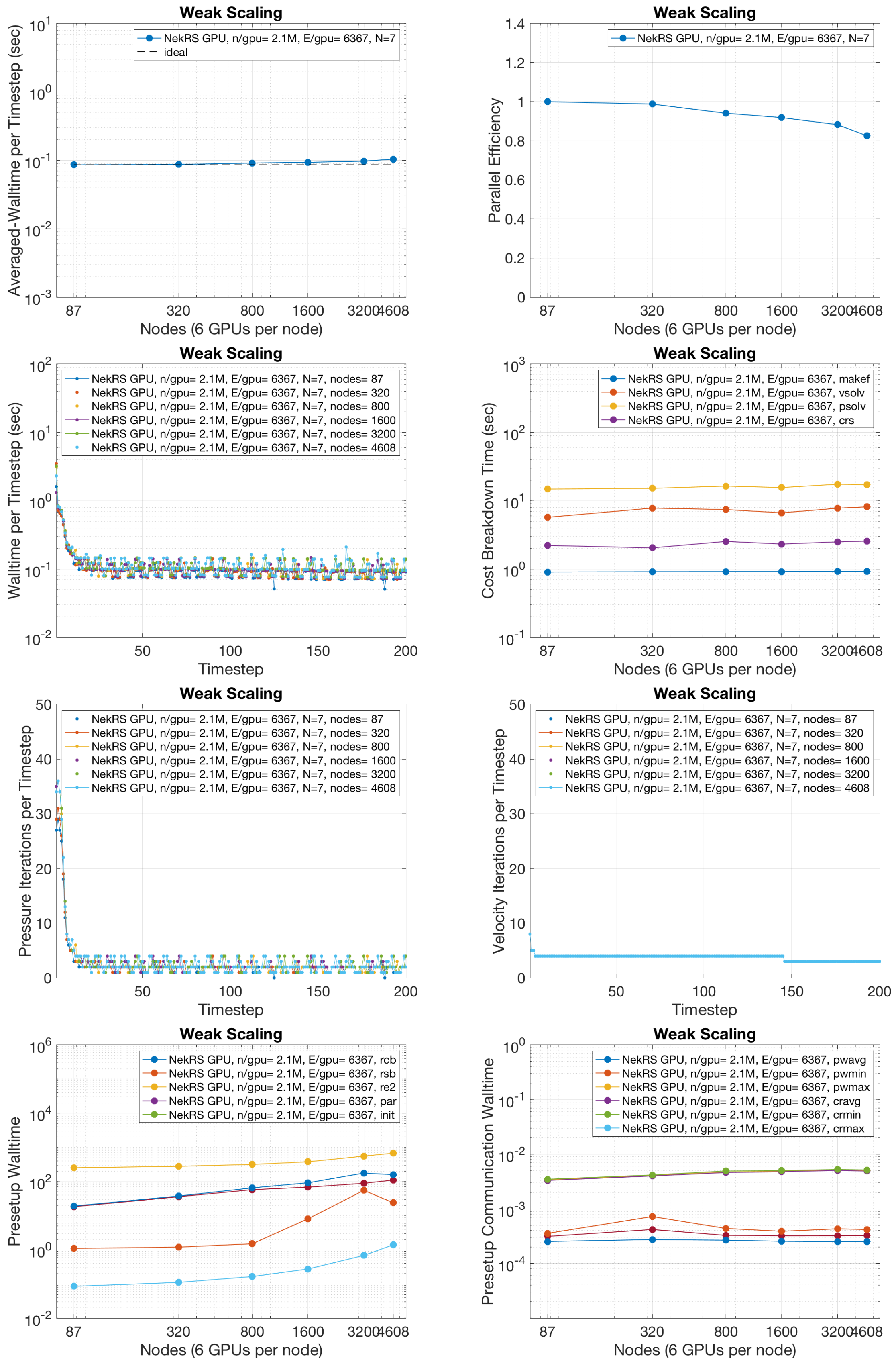


Figure 14: NekRS weak scaling for a long length of 17×17 rod-bundle mesh ($E = 175,610,000$).

to x. The objective function has the following general form:

$$F(x) = \frac{1}{n}\sum_{s=1}^{n} \frac{\sum_{E(x)} \int_{E_t} \omega_s(x)\mu_{i_s}(T(x))dx_t}{\sum_{E(x_0)} \int_{E_t} \omega_s(x_0)\mu_{i_s}(T_0(x_0))dx_t} + c\sum_{E}\int_{E_t} \xi(x - x_0, \delta(x_0))dx. \qquad (1)$$

The right-most term is used to limit the node displacements during optimization, see [6]. The other term represents a normalized explicit combination of n mesh quality metrics $\mu_{i_1}, \dots \mu_{i_n}$, where T is a Jacobian matrix that represents the transformation from the user-defined target positions to physical positions, and ω_s are user-defined weights of each metric. The metric integrals in (1) are computed as

$$\sum_{E\in\mathcal{M}} \int_{E_t} \omega(x_t)\mu(T(x_t))dx_t = \sum_{E\in\mathcal{M}} \sum_{x_q\in E_t} w_q \det(W(\bar{x}_q))\,\omega(x_q)\mu(T(x_q)), \qquad (2)$$

where $\mathcal{M}$ is the current mesh, E_t is the target element corresponding to the physical element E, w_q are the quadrature weights, and the point x_q is the image of the reference quadrature point $\bar{x}_q$ in the target element. Performing optimization through Newton's method requires computation of $\partial F/\partial \mathbf{x}$ and $\partial^2 F/\partial \mathbf{x}^2$, followed by inversion of the $\partial^2 F/\partial \mathbf{x}^2$ operator, which is performed by a preconditioned MINRES solver. When the mesh optimization is *adaptive*, the T matrices depend on discrete finite element fields that are defined on the initial mesh. This case involves solving a linear advection equation, after each Newton step, which is used to transfer discrete fields from one mesh to another, see Section 4.2 of [8].

Partial assembly kernels and new GPU implementations were developed to compute the following:

- The integrals involved in $F(x)$, resulting in a scalar value for the global mesh.
- The local action of the $\partial F/\partial \mathbf{x}$ nonlinear form, resulting in a vector of the same size as $\mathbf{x}$.
- The local action of the Hessian operator $\partial^2 F/\partial \mathbf{x}^2$, which is then used in a matrix-free manner during the MINRES iterative solve.
- Matrix-free assembly of the diagonal of the Hessian operator $\partial^2 F/\partial \mathbf{x}^2$, which is then used for preconditioning purposes during the MINRES iterative solve.
- Data at every quadrature point needed for the computations related to $F(x), \partial F/\partial \mathbf{x}$, and $\partial^2 F/\partial \mathbf{x}^2$. This data consists of the target matrices $T(x)$ and the related geometric data and coefficients; the quality metric $\mu(x)$; the first derivatives $\partial\mu/\partial x$, which are $d\times d$ matrices where d is the space dimension; the second derivatives $\partial^2\mu/\partial x^2$, which are 4-tensors with dimension $d\times d\times d\times d$.
- In the case of adaptivity, the TMOP GPU procedures utilize the existing MFEM kernels for the action of convection mass operators. These are used to solve the advection equation for field transfer between meshes.

All kernels were implemented through the MFEM's device abstraction, so that they can be executed on all supported devices.

The newly developed capabilities allow MARBL to perform Arbitrary Lagrangian-Eulerian simulations that have all their phases ported on the GPU. The current state of MARBL's GPU capability provides around 15× speedup on the main benchmark problem, which is a multi-material ALE simulation on a 3D unstructured mesh, see Figure 15. This comparison uses 4 CPU nodes (144 cores) of LLNL's *rzgenie* machine versus 4 GPU nodes (16 GPUs) of LLNL's *rzansel* machine. The GPU version of the mesh optimization phase is currently around 6 times faster than the CPU one. The CEED and MARBL teams will continue to collaborate on achieving further speedups, e.g., by deriving more efficient partial assembly preconditioners and optimizing of the core TMOP kernels.

6.3 ExaConstit advancements for ExaAM

As described in prior CEED reports, the ExaAM project in collaboration with the CEED team is creating the ExaConstit application for finite element modeling of crystal-mechanics-based constitutive models. The

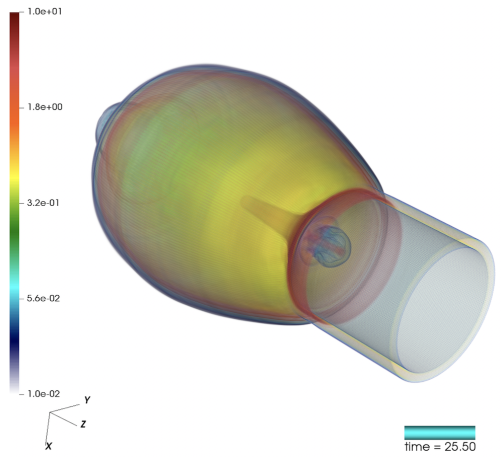


Figure 15: 3D multi-material ALE simulation that is used as a GPU performance benchmark in MARBL.

ExaConstit first release is focused on the ExaAM local property analysis, but as required, additional physics, inline results processing, and other features will be added to meet wider ExaAM needs.

Building off of the GPU element assembly work provided in MFEM, a new element assembly method was added to ExaConstit based on an efficient formulation provided in [13]. This new formulation provided about 14.5× speed-up per node on Summit over the equivalent CPU full assembly implementation. In comparison to the earlier partial assembly GPU implementation, the new GPU assembly method provided an about 8× per node speed-up on Summit. Furthermore, the new element assembly formulation provided a solid basis to build a new additional incompressible material finite element formulation based on [14] within ExaConstit. This new integrator is particularly important when using linear hexahedron elements where the incompressibility of the plastic flow of the material results in an artificially stiffer response in the material when plastic flow occurs. The performance gains from these new methods are vital in being able to run the hundreds of ExaConstit simulations needed to obtain these local properties in a reasonable time on Frontier.

As part of the ExaAM challenge problem calculation, a simulation was performed on a part of the Truchas-PBF and ExaCA generated microstructure which represents 20 layers of the whole small leg from the challenge problem. The portion taken from these 20 layers (425 microns) represented a roughly $500 \times 500 \times 425$ micron cube with 6.7 million linear hexahedron elements from the in-plane middle of the leg as seen in Figure 16. Results from this simulation were in line with on-going studies investigating the representative volume elements (RVEs) of the microstructure for the additively manufactured NIST AMB2018-01 challenge problem part that were all taken out to 5% strain. Future steps will be to automate this process of running different RVEs across the part for a wide range of loading directions and temperatures for the challenge problem in order to build up the necessary local properties needed in the process simulations.

6.4 Atmospheric boundary layer flows for ExaWind

Efficient simulation of atmospheric boundary layer flows (ABL) is important for the study of wind farms, urban canyons, and basic weather modeling. In collaboration with the ExaWind team, we identified an atmospheric boundary layer benchmark problem to serve as a point of comparison for code and modeling strategies [4]. We have addressed cross-verification and validation of our LES results and corresponding wall models. We demonstrated the suitability of high-order methods for a well-documented stably stratified atmospheric boundary layer benchmark problem, the Global Energy and Water Cycle Experiment (GEWEX) Atmospheric Boundary Layer Study (GABLS) as shown in Figure 17. This collaboration will be extended to perform scaling studies to compare the performance of several ABL codes on CPU and GPU platforms.

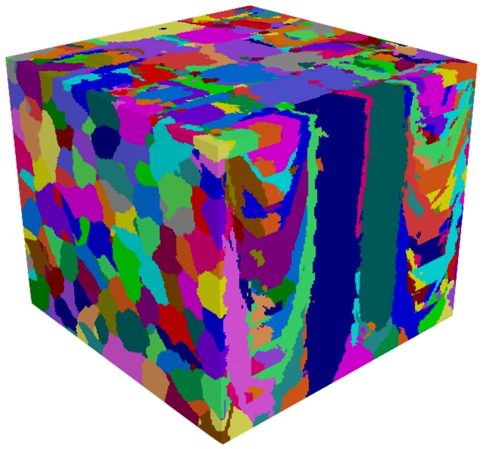


Figure 16: Middle portion of the Truchas-PBF and ExaCA generated microstructure of 20 layers of the whole small leg from the NIST AMB2018-01 challenge problem.

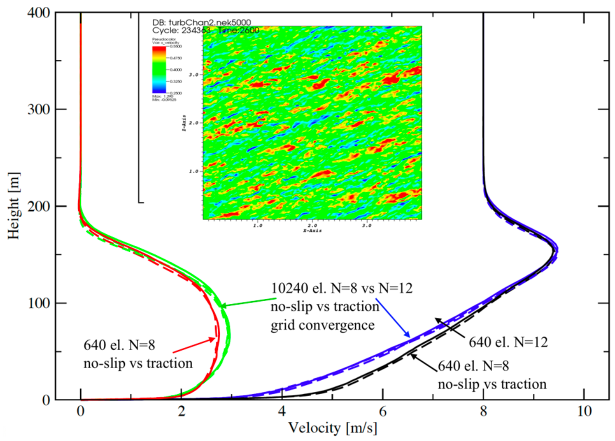


Figure 17: Nek5000 LES simulation for GABLS benchmark studies for wind velocities with no-slip and traction boundary conditions for $E = 640$ and $E = 10240$ for varying $N = 8, 12$.

6.5 NEAMS, VTO, and COVID-19 Applications

Pebble-Bed Reactors. Working the DOE NEAMS project, the CEED team has developed novel scalable meshing strategies for generating high-quality hexahedral element meshes that ensure accurate representation of densely packed spheres for complex pebble-bed reactor geometries. Our target is to capture highly turbulent flow structures in the solution at minimal cost by using relatively few elements ($\sim$300 per sphere) of high order ($N = 7$). Algorithmic strategies includes efficient edge collapse, tessellation, smoothing, and projection along with quality measurements, flow simulations, validation, and performance results for pebble bed geometries ranging from hundreds to thousands of pebbles as shown in Figure 18(a) for a case of 3344 pebbles in an annular domain using 1.1M spectral elements.

Internal Combustion. Turbulence in IC engines presents a challenge for computational fluid dynamics due primarily to the broad range of length and time scales that need to be resolved. Specifically, simulations need to predict the evolution of a variety of flow structures in the vicinity of complex domains that are moving. Executing these simulations accurately and in a reasonable amount of time can ultimately lead to engine design concepts with improved efficiency.

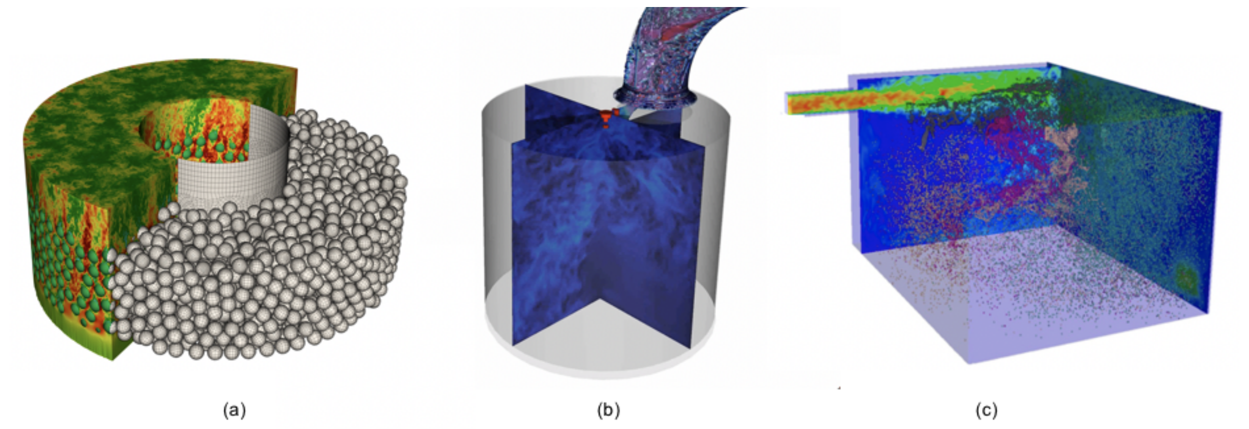


Figure 18: Other applications: (a) DOE NEAMS: Turbulent flows around 3344 pebbles with an all-hex mesh. (b) DOE VTO: Exhaust stroke TCC engine modeling. (c) COVID19: LES Lagrangian particle tracking simulation for 500,000 aerosols.

The CEED team has been working with researchers at ETH Zurich [12] and ANL's Energy Systems Division (under support from DOE's Vehicle Technologies Office) on detailed studies of turbulence in the IC engine cycle. We developed a characteristic-based spectral element method for moving-domain problems [17], and demonstrated it for the TCC III engine model illustrated in Figure18(b).We also added a significantly enhanced capability for handling complex moving geometries by adding scalable support for overset grids, referred to as NekNek, based on generalized Schwarz overlapping methods [16]. The NekNek multimesh coupling is based entirely on the kernels in Nek's gslib communication library, which has scaled to millions of MPI ranks. A newly developed preconditioner based on the SEM/FEM spectral equivalence was shown to be effective for solving the pressure-Poisson systems in these configurations [2].

Aerosol Transport Modeling. Related to the current COVID-19 pandemic, the Nek5000 team, in collaboration with NVIDIA and Utah State is researching aerosol transport analysis. High-resolution LES coupled with Lagrangian particle tracking is used for predicting the dynamics of virus-laden aerosols in indoor classroom environments [9]. Figure 18(c) demonstrates a recent simulation, using 70 million grid points and 500,000 five micron aerosols with a future target with 1 billion polydisperse aerosols in a full classroom size. We will have access to Theta-GPU (A100s) and perform several cases of study including the dynamics of the droplets depending on ambient temperature and back-ground turbulence, and also considering complex geometries when the indoor settings having items such as furniture.

7. OTHER PROJECT ACTIVITIES

7.1 Fourth CEED annual meeting

The fourth annual meeting of the CEED co-design center took place virtually Aug 11-12, 2020 with 99 researchers from 36 organizations registered (9 national labs + 14 universities + 13 industry). Participants reported on the progress in the center, deepened existing and established new connections with ECP hardware vendors, ECP software technologies projects and other collaborators, planned project activities and brainstormed / worked as a group to make technical progress.

7.2 MFEM an R&D100 finalist

The MFEM finite element library was selected as a finalist for this year's R&D100 awards is a finalist in the Software/Services category for its development of advanced discretization algorithms for HPC applications.

7.3 Outreach

CEED researchers were involved in a number of outreach activities, including presentations at the 2020 Argonne Training Program on Extreme-Scale Computing (ATPESC20); submission of two minisymposium proposals for the SIAM CSE21 conference (16 talks total): ”Exascale software and algorithms for high-order PDE solvers” and ”Exascale applications for high-order PDE solvers”; 7 papers, including a submission to the ECP special issue on co-design in IJHPCA.

8. CONCLUSION

In this milestone we helped CEED-enabled ECP applications in their preparations for the Aurora and Frontier architectures. This work included collaboration with ECP vendors and porting and optimization of CEED's benchmarks and miniapps to early access hardware.

As part of this milestone, we also released the next version of libCEED, libCEED-0.7, including improved bake-off problems and bake-off kernel implementation from Nek, MFEM, libParanumal and external community. We also organized, in virtual form, the fourth CEED Annual meeting (CEED4AM), which included representatives from ECP applications, vendors and software technology projects.

ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations—the Office of Science and the National Nuclear Security Administration—responsible for the planning and preparation of a capable exascale ecosystem—including software, applications, hardware, advanced system engineering, and early testbed platforms—to support the nation's exascale computing imperative.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, LLNL-TR-815247.

REFERENCES

[1] A. Austin, N. Chalmers, and T. Warburton. Initial guesses for sequences of linear systems in a GPU-accelerated incompressible flow solver. *submitted*, 2020.

[2] P. Bello-Maldonado and P.F. Fischer. Scalable low-order finite element preconditioners for high-order spectral element Poisson solvers. *SIAM J. Sci. Comput.*, 41:S2–S18, 2019.

[3] N. Chalmers, A. Karakus, A. P. Austin, K. Swirydowicz, and T. Warburton. libParanumal: a performance portable high-order finite element library, 2020. Release 0.3.2.

[4] M.J. Churchfield, Sang Lee, and P.J. Moriatry. Adding complex terrain and stable atmospheric condition capability to the OpenFOAM-based flow solver of the simulator for on/offshore wind farm application (SOWFA). Technical Report NREL/CP-5000-58539, NREL, 2000.

[5] M.O. Deville, P.F. Fischer, and E.H. Mund. *High-order methods for incompressible fluid flow.* Cambridge University Press, Cambridge, 2002 (500 pages).

[6] Veselin A. Dobrev, Patrick Knupp, Tzanio V. Kolev, Ketan Mittal, Robert N. Rieben, and Vladimir Z. Tomov. Simulation-driven optimization of high-order meshes in ALE hydrodynamics. *Comput. Fluids*, 2020.

[7] Veselin A. Dobrev, Patrick Knupp, Tzanio V. Kolev, Ketan Mittal, and Vladimir Z. Tomov. The Target-Matrix Optimization Paradigm for high-order meshes. *SIAM J. Sci. Comp.*, 41(1):B50–B68, 2019.

[8] Veselin A. Dobrev, Patrick Knupp, Tzanio V. Kolev, and Vladimir Z. Tomov. *Towards Simulation-Driven Optimization of High-Order Meshes by the Target-Matrix Optimization Paradigm*, pages 285–302. Springer International Publishing, 2019.

[9] Som Dutta, Paul Fischer, and et. al. On turbulence and particle transport in closed rooms. *American Physical Society, Division of Fluid Dynamics*, 2020. submitted.

[10] P.F. Fischer. Projection techniques for iterative solution of $A\underline{x} = \underline{b}$ with successive right-hand sides. *Comput. Methods Appl. Mech. Engrg.*, 163:193–204, 1998.

[11] G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker. *Solving Problems on Concurrent Processors.* Prentice-Hall, Englewood Cliffs, NJ, 1988.

[12] G.K. Giannakopoulos, C.E. Frouzakis, P.F. Fischer, A.G. Tomboulides, and K. Boulouchos. LES of the gas-exchange process inside an internal combustion engine using a high-order method. *Flow, Turbulence and Combustion*, 2019.

[13] Ajaya K. Gupta. Efficient numerical integration of element stiffness matrices. *International Journal for Numerical Methods in Engineering*, 19(9):1410–1413, 1983.

[14] Thomas J. R. Hughes. Generalization of selective integration procedures to anisotropic and nonlinear media. *International Journal for Numerical Methods in Engineering*, 15(9):1413–1418, 1980.

[15] David S. Medina, Amik St.-Cyr, and Timothy Warburton. OCCA: A unified approach to multi-threading languages. *CoRR*, abs/1403.0968, 2014.

[16] Ketan Mittal, Som Dutta, and Paul Fischer. Nonconforming Schwarz-spectral element methods for incompressible flow. *Computers and Fluids*, 191, 2019.

[17] S. Patel, P. Fischer, M. Min, and A. Tomboulides. A characteristic-based, spectral element method for moving-domain problems. *J. Sci. Comp.*, 79:564–592, 2019.

[18] A. Pothen, H.D. Simon, and K.P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11:430–452, 1990.

[19] K. Świrydowicz, N. Chalmers, A. Karakus, and T. Warburton. Acceleration of tensor-product operations for high-order finite element methods. *Int. J. of High Performance Comput. App.*, 33(4):735–757, 2019.