Hindawi
Scientific Programming
Volume 2020, Article ID 9083620, 11 pages
https://doi.org/10.1155/2020/9083620

Research Article

Scalability of OpenFOAM Density-Based Solver with Runge–Kutta Temporal Discretization Scheme

Sibo Li,[1] **Roberto Paoli**,[1,2] **and Michael D'Mello**[3]

[1] *Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL, USA*
[2] *Computational Science Division and Leadership Computing Facility, Argonne National Laboratory, Lemont, IL, USA*
[3] *Intel Corporation, Schaumburg, IL, USA*

Correspondence should be addressed to Roberto Paoli; robpaoli@uic.edu

Received 5 December 2019; Revised 2 February 2020; Accepted 11 February 2020; Published 11 March 2020

Academic Editor: Manuel E. Acacio Sanchez

Compressible density-based solvers are widely used in OpenFOAM, and the parallel scalability of these solvers is crucial for large-scale simulations. In this paper, we report our experiences with the scalability of OpenFOAM's native rhoCentralFoam solver, and by making a small number of modifications to it, we show the degree to which the scalability of the solver can be improved. The main modification made is to replace the first-order accurate Euler scheme in rhoCentralFoam with a third-order accurate, four-stage Runge-Kutta or RK4 scheme for the time integration. The scaling test we used is the transonic flow over the ONERA M6 wing. This is a common validation test for compressible flows solvers in aerospace and other engineering applications. Numerical experiments show that our modified solver, referred to as rhoCentralRK4Foam, for the same spatial discretization, achieves as much as a 123.2% improvement in scalability over the rhoCentralFoam solver. As expected, the better time resolution of the Runge–Kutta scheme makes it more suitable for unsteady problems such as the Taylor–Green vortex decay where the new solver showed a 50% decrease in the overall time-to-solution compared to rhoCentralFoam to get to the final solution with the same numerical accuracy. Finally, the improved scalability can be traced to the improvement of the computation to communication ratio obtained by substituting the RK4 scheme in place of the Euler scheme. All numerical tests were conducted on a Cray XC40 parallel system, Theta, at Argonne National Laboratory.

1. Introduction

With the continuous development of hardware and software supporting high-performance computing, eventually leading to exascale computing capabilities in a near future, high-fidelity simulations of complex flows that were out of reach until a decade ago are now becoming feasible on supercomputer infrastructures. Direct numerical simulations (DNS) and large-eddy simulations (LES) are natural candidates for high-fidelity simulations as they can capture all relevant spatial and temporal characteristic scales of the flow. Indeed, there is consensus in the computational fluid dynamics (CFD) community that DNS/LES codes incorporating high-order time integration and spatial discretization methods are preferable for ensuring minimal influence of numerical diffusion and dispersion on the flow physics. While these numerical constraints have been traditionally integrated in the simulation of academic flows on simple geometries, they are also being considered for industrial and more complex applications where accurate prediction of local or instantaneous flow properties are required (e.g., in combustion, multiphase and reacting flows).

In this context, the OpenFOAM package [1] represents a popular open-source software originally designed for use in CFD. Its operation is composed of solvers, dictionaries, and domain manipulation tools. It provides several different kinds of solvers for different partial differential equations (PDEs) and framework to implement third-party solvers for customized PDEs. The solvers integrated in the standard distributions of OpenFOAM are robust, but they generally lack precision with 2nd-order accuracy at most in both space and time. Therefore, there is a growing interest in the CFD community to develop and implement higher-order methods in OpenFOAM for transient flow calculations. These include, for example, numerical algorithms for DNS/LES of incompressible flows [2], compressible flows [3–5], and reacting flows [6].

In addition to high-order numerical schemes, parallel efficiency and scalability are of course crucial for high-performance computing of complex flows. Parallelism in OpenFOAM is implemented by means of MPI (Message Passing Interface) although previous scalability analysis of incompressible OpenFOAM solvers showed limited speedup [7, 8]. The improvement of scaling performance in OpenFOAM has been a concern in many recent studies. In order to optimize the performance, it is crucial to first conduct performance analysis to find the bottleneck of the simulation process. Culpo [7] found that the communication is the bottleneck in scalability of solvers in OpenFOAM for large-scale simulations. Duran et al. [9] studied the speedup of icoFoam solver for different problem sizes and showed that there is a large room for scalability improvement. Lin et al. [10] proposed a communication optimization method for multiphase flow solver in OpenFOAM. Ojha et al. [11] applied optimizations to the geometric algebraic multigrid linear solver and showed an improved scaling performance.

In this work, we are primarily interested in compressible flow solvers for aeronautical applications. In the standard distribution of OpenFOAM, rhoCentralFoam is the only density-based solver for transient, compressible flows [12]. It is built upon a second-order nonstaggered central scheme based on KT (Kurganov and Tadmor [13]) and KNP (Kurganov, Noelle, and Petrova [14]) methods. There are a few studies that tried to improve the numerical algorithm of density-based solver. Heyns et al. [15] extended the rhoCentralFoam solver through the implementation of alternative discretization schemes to improve its stability and efficiency. More recently, Modesti and Pirozzoli [4] developed a solver named rhoEnergyFoam relying on the AUSM scheme by Liou and Steffen Jr. [16]. By advancing using a low-storage third-order four-stage Runge–Kutta algorithm for time advancement, their solver showed reduced numerical diffusion and better conservation properties compared to rhoCentralFoam in both steady and unsteady turbulent flows.

In this work, we present a new OpenFOAM solver, named rhoCentralRK4Foam, which is derived from rhoCentralFoam by replacing its first-order time advancement scheme with a third-order, four-stage Runge–Kutta scheme. The aim of developing this solver is twofold: (i) to improve the scaling performance of rhoCentralFoam, especially in large-scale simulations; (ii) to improve time accuracy and overall time-to-solution using high-order Runge–Kutta scheme [17]. Instead of trying to optimize the parallelism embedded in OpenFoam, which has been shown to be very hard to achieve (see, e.g., Culpo [7]), our proposed approach is to select a different numerical integration scheme that shows improved CPU and scalability performances with minimal modification of the standard distribution. The cases are configured to solve the PDEs through the rhoCentralFoam and rhoCentralRK4Foam solvers. We investigate the parallel performance of rhoCentralFoam and rhoCentralRK4Foam on Cray XC40 system Theta [18]. These two solvers are benchmarked on two cases, an inviscid transonic flow over the ONERA M6 wing, and a supersonic flow over the Forward-facing step to validate the new solver's shock capturing capability. The TAU (Tuning and Analysis Utilities) Performance System analyzer [19] is used to collect the hotspot profiles of the two solvers. The strong and weak scaling tests of the benchmark problems are conducted on Theta up to 4,096 cores. For the time-to-solution analysis, we considered the Taylor–Green vortex problem and determined the time-to-solution needed by the two solvers to get a solution with the same accuracy (same numerical error) with respect to the analytical solution. The present study is also aimed at providing users a handle on parameter choices for performance and scalability.

The rest of the paper is organized as follows: in Section 2, a description of numerical methods for the two solvers as well as the hardware and profiling tools are presented. The benchmark test cases and the results for the scalability analysis are presented in Section 3. In Section 4, we present the results for the time-to-solution analysis. Section 5 shows the conclusion.

2. Methods

2.1. Governing Equations and Spatial Integration. The solver rhoCentralFoam is the most widely used compressible flow solver in the baseline distribution of OpenFOAM. It relies on the full discretization of the convective fluxes through the central TVD scheme (total variation diminishing) of KT and KNP schemes. rhoCentralFoam solves the governing fluid equations in an Eulerian frame of reference for three conservative variables, specifically density (ρ), momentum density ($\rho\mathbf{u}$), and total energy density (ρE):

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0, \tag{1}$$

$$\frac{\partial\rho\mathbf{u}}{\partial t} + \nabla\cdot(\rho\mathbf{u}\mathbf{u}) = -\nabla p + \nabla\cdot\boldsymbol{\tau}, \tag{2}$$

$$\frac{\partial\rho E}{\partial t} + \nabla\cdot(\rho\mathbf{u}E) + \nabla\cdot(p\mathbf{u}) = \nabla\cdot(T\cdot\mathbf{u}) + \nabla\cdot\mathbf{J}. \tag{3}$$

In the above equations, $E \equiv e(T) + 1/2\mathbf{u}\cdot\mathbf{u}$, where $e(T)$ is the internal energy and T is the temperature; p is the thermodynamic pressure, related to the temperature and density through the equation of state for ideal gases: $p = \rho RT$, where R is the gas constant; and $\tau = 2\mu[(1/2)(\nabla\mathbf{u} + \nabla^T\mathbf{u}) - 1/3(\nabla\cdot\mathbf{u})\mathbf{I}]$ is the viscous stress tensor under the assumption of Newtonian fluid with dynamic viscosity μ while $\mathbf{I}$ is the unit tensor. Finally, $\mathbf{J} = -\lambda\nabla T$ is the heat flux vector where λ is the thermal conductivity.

The governing equations are discretized using a finite volume method where equations (1)–(3) are integrated over a control volume represented by a grid cell of volume V. Using Gauss theorem, all fluxes can be transformed into surface integrals across the cell boundaries, which are estimated by summing up the flux contributions from all faces f of the cell. The volume average of the state vector of conserved variables $\mathbf{\Psi} \equiv [\rho; \rho\mathbf{u}; \rho E]$ is given by

$$\overline{\Psi} = \frac{1}{V}\int_V \mathbf{\Psi} \mathrm{d}V. \tag{4}$$

So the integral form of equations (1)–(3) can be expressed as

$$\frac{\mathrm{d}\overline{\Psi}}{\mathrm{d}t} = \mathscr{L}(\overline{\Psi}), \tag{5}$$

where $\mathscr{L}(\overline{\Psi})$ denotes the operator returning the right-hand side of equations (1)–(3) containing all inviscid and viscous fluxes. These fluxes have to be estimated numerically using the volume-averaged state variables $\overline{\Psi}$ of adjacent cells. In particular, the convective fluxes are obtained as

$$\int_V \nabla \cdot (u\mathbf{\Psi})\mathrm{d}V = \int_S \mathbf{u\Psi} \cdot \mathrm{d}\mathbf{S} \approx \sum_{f\in S}\left(\mathbf{u}_f \cdot \mathbf{S}_f \mathbf{\Psi}_f\right) = \sum_{f\in S} \phi_f \mathbf{\Psi}_f, \tag{6}$$

where subscript f identifies the faces of the cell volume, whereas the terms $\mathbf{\Psi}_f, \mathbf{u}_f, \mathbf{S}_f$, and $\phi_f \equiv \mathbf{u}_f \cdot \mathbf{S}_f$ represent the state vector, velocity, surface area, and volumetric flux at the interface between two adjacent cells, respectively. The product $\phi_f \rho_f$ is obtained by applying the KT scheme:

$$\phi_f \mathbf{\Psi}_f = \frac{1}{2}\left(\phi_f^+ \mathbf{\Psi}^+ + \phi_f^- \mathbf{\Psi}_f^-\right) - \frac{1}{2}\alpha_{\max}\left(\mathbf{\Psi}_f^- - \mathbf{\Psi}_f^+\right), \tag{7}$$

where + and − superscripts denote surface values interpolated from the owner cell and neighbor cell, respectively, while $\alpha_{\max}$ is an artificial diffusive factor (see [12] for details). In the implementation of rhoCentralRK4Foam, the Eulerian fluxes are firstly calculated explicitly as shown in Algorithm 1 where "phi" represents $\phi_f \rho_f$, "phiUp" represents $\phi_f \rho_f \mathbf{U}_f + \mathbf{S}_f p_f$ with $p_f = 1/2\,(p_f^+ + p_f^-)$, and "phiEp" represents $\phi_f \rho_f E_f + \phi_f p_f$.

2.2. Time Integration. After calculating fluxes and gradients according to equations (6) and (7), equation (5) can be numerically integrated between time t_n and $t_{n+1} = t_n + \Delta t$ using multistage Runge–Kutta scheme. Denoting by N_s the number of stages, this yields

$$\overline{\Psi}_n^l = \overline{\Psi}_n^0 + \alpha_l \Delta t \mathscr{L}\left(\overline{\Psi}_n^{l-1}\right), \quad l = 1, \ldots, N_s, \tag{8}$$

where $\overline{\Psi}_n^0 \equiv \overline{\Psi}(t_n)$ and $\overline{\Psi}_n^{N_s} \equiv \overline{\Psi}(t_{n+1})$. In the proposed solver rhoCentralRK4Foam, we use a four-stage Runge–Kutta scheme, which is obtained by setting $N_s = 4$ and $\alpha_1 = 1/4, \alpha_2 = 1/3, \alpha_3 = 1/2$, and $\alpha_4 = 1$. The implementation in OpenFOAM is reported in Algorithm 2, and the C++ code is publicly available at https://github.com/SiboLi666/rhoCentralRK4Foam. Note that the original Euler scheme implemented in rhoCentralFoam can be simply obtained from equation (8) by setting $N_s = 1$ and $\alpha_1 = 1$.

2.3. Hardware and Profiling Tools. The simulations were run on supercomputer platform "Theta" at Argonne National Laboratory. Theta is a Cray XC40 system with a second-generation Intel Xeon Phi (Knights Landing) processor and the Cray Aries proprietary interconnect. The system comprises 4,392 compute nodes. Each compute node is a single Xeon Phi chip with 64 cores, 16 GB Multi-Channel DRAM (MCDRAM), and 192 GB DDR4 memory. To avoid any possible memory issues with OpenFoam, the simulations were run using 32 cores of each computing node (half of its maximum capacity).

We assess the scalability and parallel efficiency of the two solvers, rhoCentralFoam and rhoCentralRK4Foam, by analyzing their speedup on the same grid and by monitoring the portions of CPU time spent in communication and in computation, respectively. These measurements are performed with performance tools adapted to the machine. In the strong scaling tests, the tools are set to start counting after the initial setup stage is completed, in our case, after 20 time steps, lasting for an extra 75 time steps. The strong scaling tests are performed on Theta. In order to measure the time spent in communication, we rely on the TAU Performance system [19]. The TAU performance tool suite is an open-source software developed at the University of Oregon and offers a diverse range of performance. On Theta, the OpenFOAM makefile is modified to utilize the TAU wrapper functions for compilation. TAU can automatically parse the source code and insert instrumentation calls to collect profile and/or trace data, which allow us to measure the total time spent in communication during the targeted 75 time steps and identify the hotspots for the two solvers.

3. Results for the Scalability Analysis

3.1. Test Cases Description. For the scalability analysis, we tested the new solver rhoCentralRK4Foam in two different benchmark cases: (i) the three-dimensional ONERA M6 transonic wing; (ii) the two-dimensional supersonic forward facing step. The two cases are steady flows; they are first used for validating the new solver's shock-capturing capability and then used for detailed parallel performance analysis of the solver in the next section. For the two cases, we used the decomposePar tool in OpenFOAM to decompose the generated mesh. The scotch decomposition method [20] is used to partition the mesh into subdomains. A brief description of the two cases is presented next.

3.1.1. Transonic M6 Wing. In the ONERA M6 wing case, the mean aerodynamic chord is $c = 0.53$ m, the semispan is $b = 1$ m, and the computational domain extends to $20c$. The angle of attack is 3.06°, and the free stream mach number is Ma = 0.8395. The Reynolds number in the original experiment was $\mathrm{Re} = 11.72 \times 10^6$, and so the flow is certainly turbulent; however, in order to capture the pressure distribution and the shock location along the wing, inviscid calculations can be safely employed and are indeed customary (see, for example, Modesti and Pirozzoli, [4]). The geometry is meshed using hexahedral elements, and three grids with different sizes are generated: Grid1 has 1 million cells (shown in Figure 1(a)), Grid2 has 5 million cells, and

```
Result: construct Eulerian fluxes explicitly
while runTime.loop() do
  Saving quantities at preavious time steps if rk4 then
    /* The following firstly constructs the Eulerian fluxes by applying the Kurganov and Tadmor (KT) scheme. */
    surfaceScalarField phi
    ("phi,"
    $(aphiv\_pos * rho\_pos + aphiv\_neg * rho\_neg)$);
    surfaceVectorField phiUp
    ("phiUp,"
    $(aphiv\_pos * rhoU\_pos + xaphiv\_neg * rhoU\_neg)$
    $+\ (a\_pos * p\_pos + a\_neg * p\_neg * mesh.Sf)$);
    surfaceScalarField phiEp
    ("phiEp,"
    $(aphiv\_pos * (rho\_pos * (e\_pos + 0.5 * magSqr(U\_pos)) + p\_pos)$
    $+ aphiv\_neg * (rho\_neg * (e\_neg + 0.5 * magSqr(U\_neg)) + p\_neg)$
    $+ aSf * p\_pos - aSf * p\_neg)$);
    /* Then, the divergence theorem (Gauss's law) is applied to relate the spatial integrals to surface integrals.
    */
    volScalarField phiSum ("phiSum," $fvc::div(phi)$);
    volVectorField phiUpSum2 ("phiUpSum2," $fvc::div(tauMC)$);
    volVectorField phiUpSum3 ("phiUpSum3," $fvc::laplacian(muEff, U)$);
  else
  end
end
```

Algorithm 1: The Eulerian fluxes are calculated explicitly so that the ordinary differential equations system can be constructed and further integrated in time by using explicit Runge–Kutta algorithm.

```
Result: Calculate the three conservative variables
for ($int\ cycle = 0;\ cycle < rkloop.size;\ cycle++$){
  /* The following calculates the three conservative variables, specifically density ($\rho$), momentum density ($\rho\mathbf{U}$) and total energy density ($\rho E$) */
  $rho = rhoOld + rk[cycle] * runTime.deltaT * (-phiSum)$;
  $rhoU = rhoUOld + rk[cycle] * runTime.deltaT * (-phiUpSum + phiUpSum2 + phiUpSum3)$;
  $rhoE = rhoEOld + rk[cycle] * runTime.deltaT * (-enFl + enFl2)$;
}
```

Algorithm 2: The time integration of the resulting ordinary differential equations system is carried out by a third-order, four-stage Runge–Kutta algorithm.

Grid3 has 43 million cells. The flow-filed analysis was conducted on Grid1 (which is sufficient for grid convergence), while Grid2 and Grid 3 were used for scaling analysis. Figure 1(b) shows the pressure distribution on the wing surface computed by using rhoCentralFoam and rhoCentralRK4Foam. In Figure 1(c), the pressure coefficient at the inner wing section (20% span) computed by the two solvers is compared with the experimental data (blue circle symbols [21]). We can observe that the newly developed solver captures the main flow features. It generates a pressure distribution on the wing surface similar to that obtained with rhoCentralFoam and captures the shock location precisely.

3.1.2. Supersonic Forward-Facing Step. Computations of inviscid flow over a forward-facing step are used to further verify the shock-capturing capability of rhoCentralRK4Foam solver. The flow configuration used by Woodward and Colella [22] is considered in this work. The supersonic flow mach number is $\mathrm{Ma}_\infty = 3$. The grid has $N_x \times N_y = 240 \times 80$ cells in the coordinate directions. The shock pattern represented by density distribution shown in Figure 2 confirms that the rhoCentralRK4Foam is capable of capturing strong shocks for supersonic flows.

3.2. Strong Scaling Tests. In the strong scaling tests, we tested the rhoCentralFoam and rhoCentralRK4Foam solvers on three different mesh sizes of ONERA M6 wing. Here, we present the scalability results by showing the speedup as well as the speedup increment percentage for rhoCentralRK4Foam for Grid1 up to 1024 ranks in Table 1, Grid2 up to 2048 ranks in Table 2, and Grid3 up to 4096 ranks in Table 3. Note that the scaling is presented as CPU time normalized by the CPU time obtained with 128 ranks, which

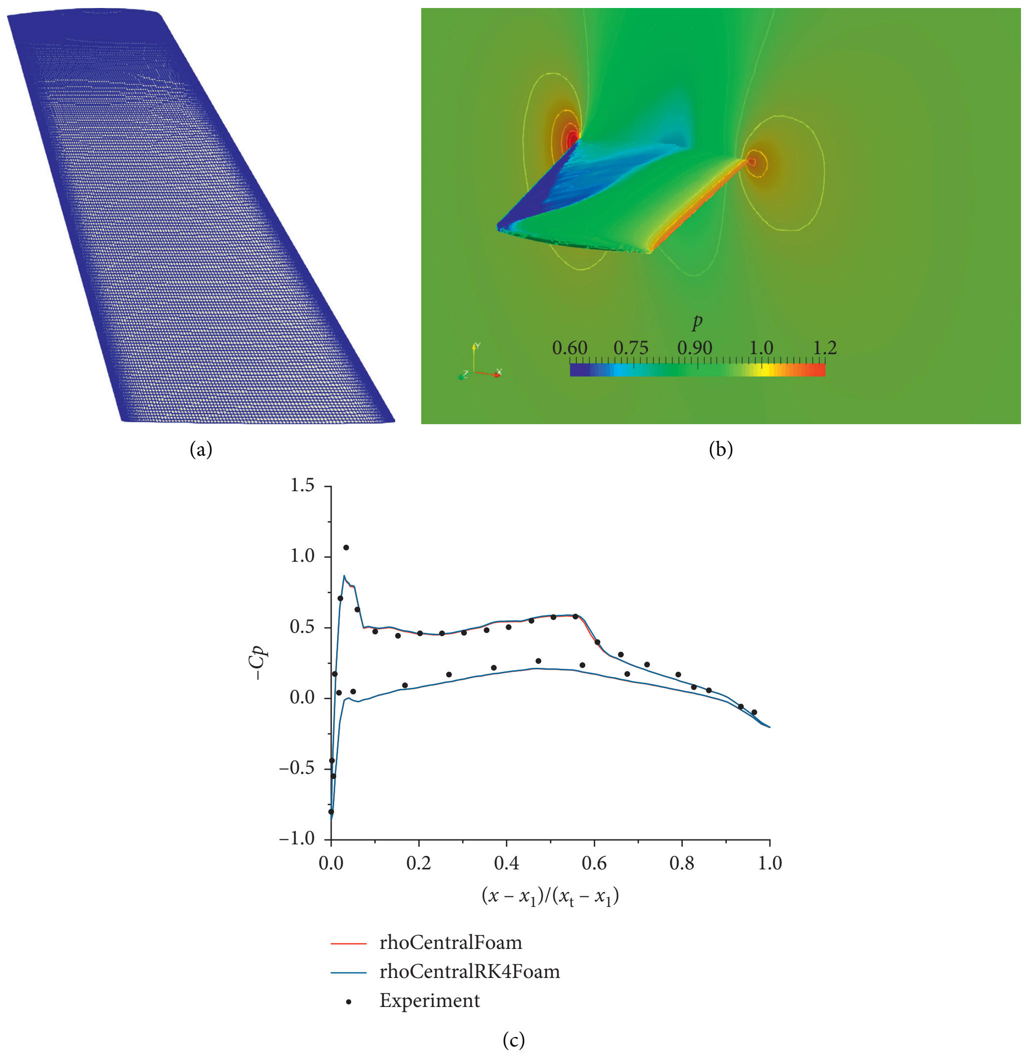


Figure 1: ONERA M6 transonic wing results: (a) computational mesh (Grid 1); (b) pressure distribution computed with the proposed solver rhoCentralRK4Foam; (c) comparison of pressure coefficient at section located (20% span) computed with rhoCentralRK4Foam and the original rhoCentralFoam.

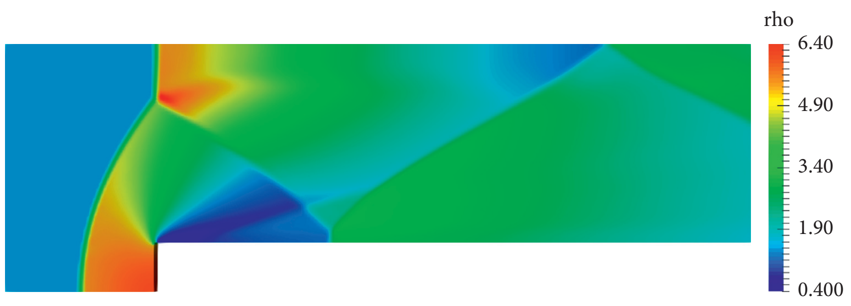


Figure 2: Density distribution in supersonic forward facing step simulations obtained using rhoCentralRK4Foam.

is the minimum number of ranks that fit the memory requirements for the largest grid. For example, for 4096 ranks, the ideal speedup would then be $4096/128 = 32$. It can be observed that rhoCentralRK4Foam outperforms rhoCentralFoam in speedup in each case. For the same grid size, the speedup increment percentage increases as the number of ranks increases. To better illustrate and analyze the scaling performance improvement, the results reported in the three tables are summarized in Figure 3. First, we can observe that for the Grid3, there is a dramatic increase in speedup when the number of ranks rises from 1024 to 2048 and from 2048 to 4096. The speedup of rhoCentralRK4Foam

Table 1: Normalized speedup for Grid1 (ONERA M6 wing) up to 1024 ranks.

#Ranks	rhoCentralFoam	rhoCentralRK4Foam	% increment
128	1	1	0
256	1.341	1.367	1.939
512	1.594	1.814	13.802
1024	2.364	2.747	16.201

Table 2: Normalized speedup for Grid2 (ONERA M6 wing) up to 2048 ranks.

#Ranks	rhoCentralFoam	rhoCentralRK4Foam	% increment
128	1	1	0
256	1.574	1.601	1.715
512	2.308	2.423	4.983
1024	2.916	3.575	22.599
2048	4.541	6.005	32.240

Table 3: Normalized speedup for Grid3 (ONERA M6 wing) up to 4096 ranks.

#Ranks	rhoCentralFoam	rhoCentralRK4Foam	% increment
128	1	1	0
256	1.871	1.896	1.336
512	3.275	3.527	7.695
1024	5.098	5.809	13.947
2048	6.341	9.115	43.747
4096	6.895	15.39	123.205

has a 1.6 times increase from 1024 ranks to 2048 ranks and a 1.7 times increase from 2048 ranks to 4096 ranks for Grid3. The reason is that rhoCentralFoam scales very slowly after 1024 ranks, eventually reaching a plateau, which is an indication the solver attained its maximum theoretical speedup. It is indeed instructive to estimate the serial portion f of CPU time from the speedup of the two solvers using Amdahl's law [23]:

$$S(N_r) = \frac{N_r}{1 + (N_r - 1)(1 - f)}, \quad \text{with } f = \frac{t_p}{t_p + t_s}, \tag{9}$$

where t_p and t_s are the CPU time spent in parallel and serial (i.e., not parallelizable) sections of the solver, respectively. From the previous equations, we have

$$f = \frac{1 - (1/S(N_r))}{1 - (1/N_r)} \longrightarrow f = 1 - \frac{1}{S_\infty}, \tag{10}$$

where $S_\infty \equiv S(N_r \longrightarrow \infty)$ is the maximum theoretical speedup obtained for $N_r \longrightarrow \infty$. By measuring S_∞, we can directly evaluate f from equation (10) for the cases featuring asymptotic speedup: for rhoCentralFoam (Grid3), this yields $f \simeq 99.88\%$. We can also estimate f but best fitting $S(N_c)$ to Amdahl's formula in equation (9) for rhoCentralRK4Foam, which yields $f \simeq 99.98\%$. Of course, these results can be affected by other important factors such as latency and bandwidth of the machine that are not taken into account in Amdahl's model. We also observe that the scalability becomes better as the problem size increases as the workload of

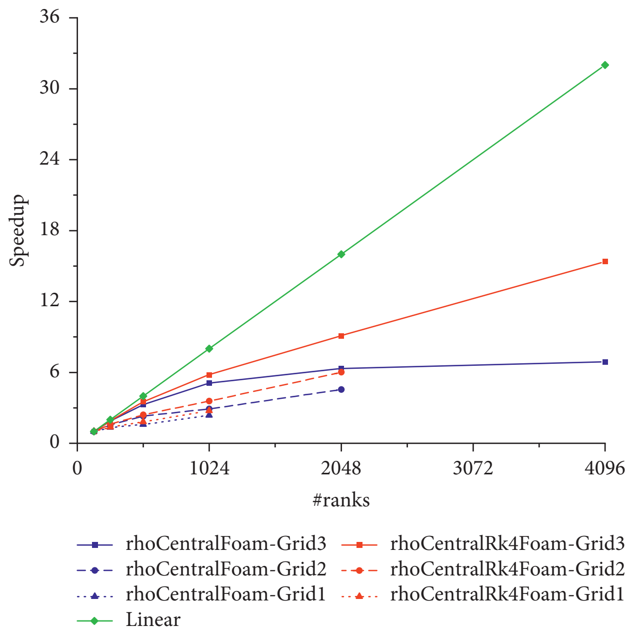
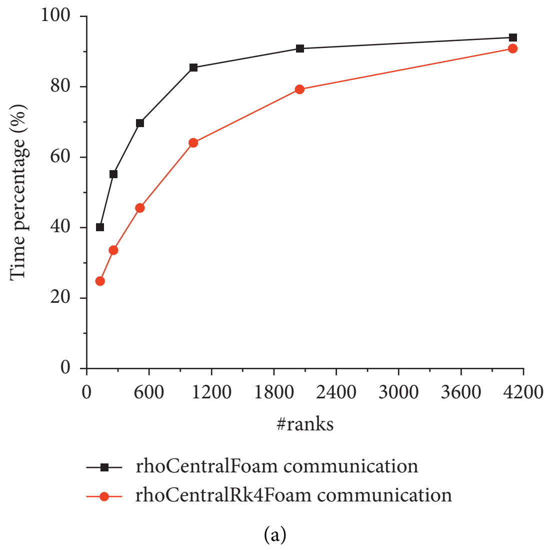


Figure 3: Strong scaling normalized speedups of rhoCentralFoam and rhoCentralRK4Foam for three grids of the M6 wing test case.

communication over computation decreases. For example, with 2048 ranks, the speedup for rhoCentralRK4Foam is 6.005 for Grid2 (5 million cells), while it becomes 9.115 for Grid3 (43 million cells). Moreover, we can also see that as the grid size increases, the maximum speedup increment percentage also increases, which corresponding to 16%, 32%, and 123% for Grid1, Grid2, and Grid3, respectively. As a final exercise, the TAU performance tool was applied to profile the code for Grid3 with 2048 ranks. Figure 4 also shows the communication and computation time percentage for rhoCentralFoam and rhoCentralRK4Foam on Grid3 from 128 to 4,096 ranks. We can observe that the rhoCentralRK4Foam solver takes less time to communicate, which lead to the improved parallel performance found in previous tests. In addition, among the MPI subroutines that are relevant for communication (i.e., called every time step in the simulation), MPI_waitall() was the one that employed most of the time (see Figure 5), which is in line with previous profiling (see, for example, Axtmann and Rist [24]).

3.3. Weak Scaling Tests. In order to further analyze the parallel scalability of rhoCentralRK4Foam solver, we conducted a weak scaling analysis based on the ONERA M6 wing case and the forward-facing step case. The grid size ranges from 700,000 to 44,800,000 cells in ONERA M6 wing case and from 1,032,000 to 66,060,000 cells in the forward-facing step case. The configurations of the weak scaling test cases are the same as the one in the previous test cases. The number of ranks ranges from 16 to 1024 and 64 to 4096, respectively. The number of grid points per rank stays constant at around 43×10^3 and 16×10^3 for two cases, which ensures that the computation workload is not impacted much by communication. Denoting by τ the CPU time for a given run, the relative CPU time is obtained by



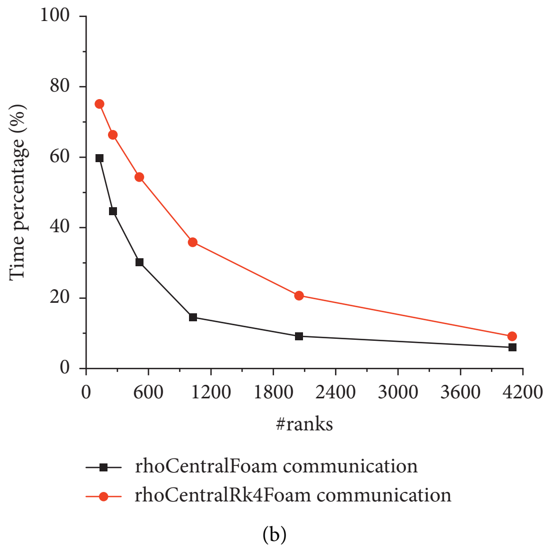


FIGURE 4: Communication (a) and computation (b) time percentage for the strong scaling analysis (M6 test case, Grid 3).

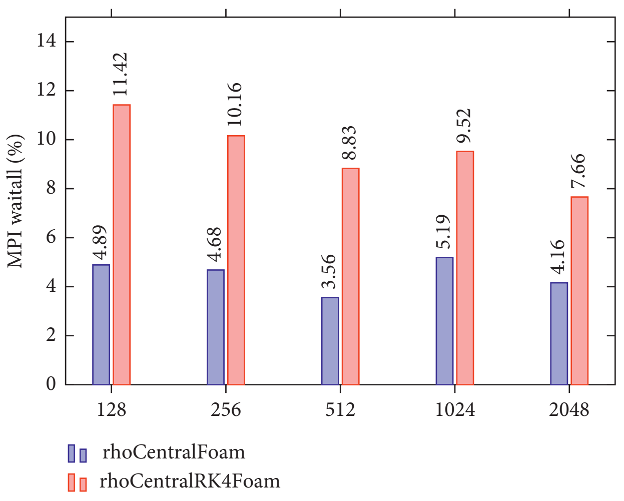


FIGURE 5: MPI waitall() time percentage for rhoCentralFoam and rhoCentralRK4Foam for M6 test case (Grid3) up to 2,048 ranks.

normalizing with respect to the values obtained for 16 and 64 ranks: $\tau/\tau_{\#\text{Ranks}=16}$, for 16 ranks and $\tau/\tau_{\#\text{Ranks}=64}$, for 64 ranks. The relative CPU time are recorded in Tables 4 and 5. The comparison of the two solvers is also plotted in Figures 6 and 7. The weak scaling tests give us indication about how well does the two solvers scale when the problem size is increased proportional to process count. From Tables 4 and 5 and Figures 6 and 7, it can be observed that for lower MPI tasks (16 to 64), both of the two solvers scale reasonably well. However, for higher MPI tasks (128 to 1024), the rhoCentralRK4Foam solver scales better. Remarkably, we can observe a distinguishable relative time difference between the two solvers as the number of ranks increases from 512 to 1024. The profiling results from TAU show that in the test case with 1024 ranks, rhoCentralRK4Foam spends around

TABLE 4: Relative time results $\tau/\tau_{\#\text{Ranks}=16}$ of weak scaling test (ONERA M6 wing).

#Ranks	rhoCentralFoam	rhoCentralRK4Foam
16	1	1
32	1.085	1.062
64	1.106	1.105
128	1.148	1.057
256	1.191	1.100
512	1.213	1.158
1024	1.255	1.148

TABLE 5: Relative time results $\tau/\tau_{\#\text{Ranks}=64}$ of the weak scaling test (forward-facing step).

#Ranks	rhoCentralFoam	rhoCentralRK4Foam
64	1	1
128	1.034	1.002
256	1.085	1.018
512	1.153	1.031
1024	1.186	1.041
2048	1.212	1.067
4096	1.203	1.063

40% time on computation while rhoCentralFoam only has around 20% time spent on computation. As for the forward-facing step case, we were able to conduct the tests at larger cores count (up to 4096 cores) and it can be confirmed that rhoCentralRK4Foam solver has better scaling performance than rhoCentralFoam solver. Indeed, it scales better with larger grid size and the relative time of rhoCentralRK4Foam solver maintains at 1.063 with 4096 cores (the relative time is 1.148 with 1024 cores in ONERA M6 wing case). Generally, the rhoCentralRK4Foam solver outperforms the rhoCentralFoam solver at large-scale ranks due to communication hiding.

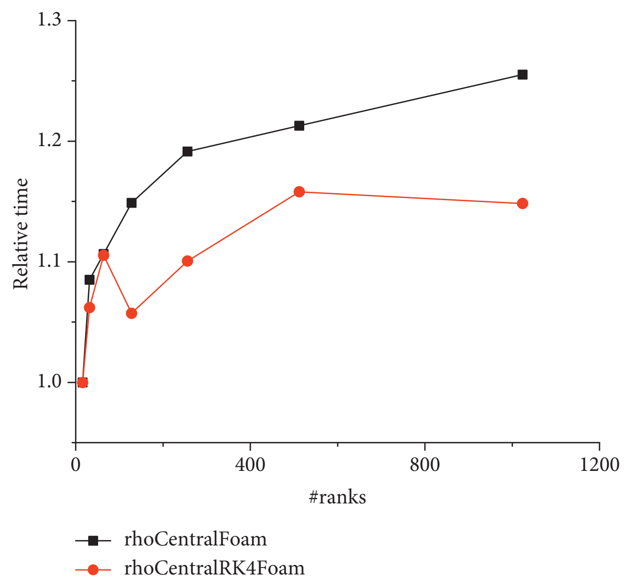


FIGURE 6: Relative time $\tau/\tau_{\#\text{Ranks}=16}$ of weak scaling tests for ONERA M6 wing case.

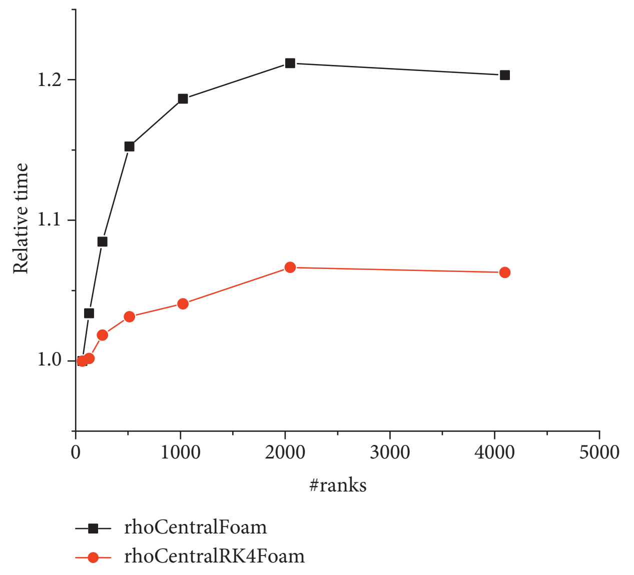


FIGURE 7: Relative time $\tau/\tau_{\#\text{Ranks}=64}$ of weak scaling tests for forward-facing step.

4. Results for the Time-to-Solution Analysis

The scaling analysis is an important exercise to evaluate the parallel performance of the code on multiple cores; however, it does not provide insights into the time accuracy of the numerical algorithm nor into the time-to-solution needed to evolve the discretized system of equations (equations (1)–(3)) up to a final state within an acceptable numerical error. For example, in the scaling study conducted on the M6 transonic wing, the number of iterations was fixed for both solvers rhoCentralFoam and rhoCentralRK4Foam and calculations were performed in the inviscid limit so that the implicit solver used in rhoCentralFoam to integrate the viscous fluxes was not active. In such circumstances, comparing the two solvers essentially amounts at comparing first-order (Euler) and fourth-order forward in time Runge–Kutta algorithms. Hence, it is not surprising that the CPU time for rhoCentralRK4Foam is about four times larger than rhoCentralFoam (for the same number of iterations) since it requires four more evaluations of the right-hand-side of equations (5) per time step. For a sound evaluation of the time-to-solution however, we need to compare the two solvers over the same physical time at which the errors with respect to an exact solution are comparable. Because the time accuracy of the solvers is different, time step is also different and so are the number of iterations required to achieve the final state.

4.1. Test Case Description.

For the time-to-solution analysis, we consider the numerical simulation of Taylor–Green (TG) vortex, a benchmark problem in CFD [25] for validating unsteady flow solvers [26, 27] requiring time accurate integration scheme as the Runge–Kutta scheme implemented in rhoCentralRK4Foam. There is no turbulence model applied in the current simulation. The TG vortex admits analytical time-dependent solutions, which allows for precise definition of the numerical error of the algorithm with respect to a given quantity of interest. The flow is initialized in a square domain $0 \le x, y \le L$ as follows:

$$\rho(x, y, t) = \rho_0, \tag{11}$$

$$u(x, y, 0) = -U_0 \cos kx \sin ky, \tag{12}$$

$$v(x, y, 0) = U_0 \sin kx \cos ky, \tag{13}$$

$$p(x, y, 0) = p_0 - \frac{\rho_0 U_0^2}{4} (\cos 2kx + \cos 2ky), \tag{14}$$

where u and v are the velocity components along x and y directions, $k \equiv 2\pi/L$ is the wave number, and U_0, ρ_0, and p_0 are the arbitrary constant velocity, density, and pressure, respectively. The analytical solution for the initial-value problem defined by equations (11)–(14) is

$$\rho(x, y, t) = \rho_0, \tag{15}$$

$$u(x, y, t) = -U_0 \cos kx \sin ky e^{-2\nu t k^2}, \tag{16}$$

$$v(x, y, t) = U_0 \sin kx \cos ky e^{-2\nu t k^2}, \tag{17}$$

$$p(x, y, t) = p_0 - \frac{\rho_0 U_0^2}{4} (\cos 2kx + \cos 2ky) e^{-4\nu t k^2}, \tag{18}$$

which shows the velocity decay due to viscous dissipation. As a quantity of interest, we chose to monitor both the velocity profiles u at selected location and the overall kinetic energy in the computational box, which can be readily obtained from equations (16)–(18) as

$$\varepsilon = \frac{1}{L^2} \int_0^L \int_0^L \frac{u^2 + v^2}{2} \mathrm{d}x\mathrm{d}y = \frac{U_0^2}{4} e^{-4\nu t k^2}. \tag{19}$$

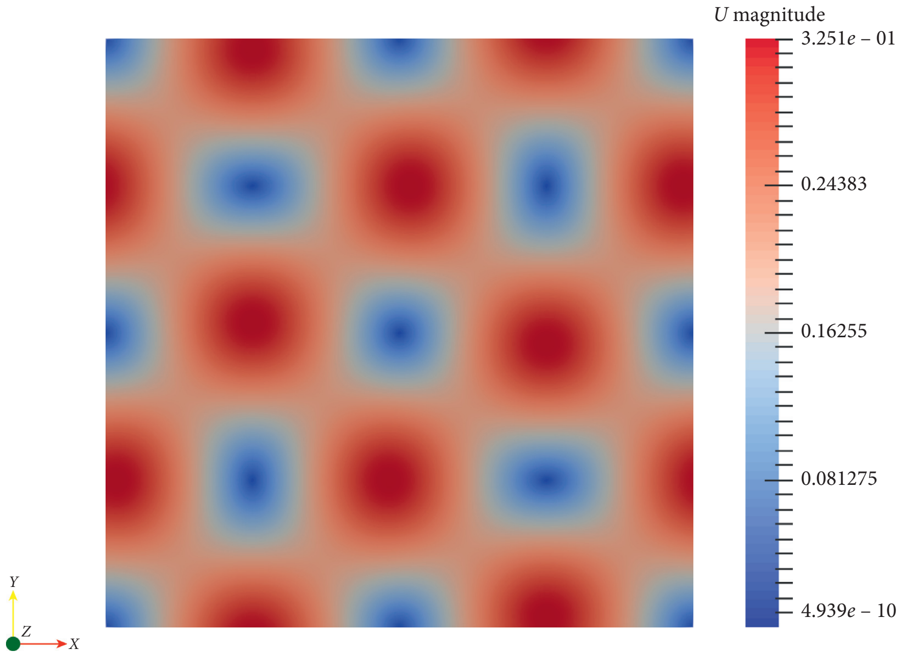


Figure 8: Contour of velocity magnitude $U \equiv (u^2 + v^2)^{1/2}$ for the Taylor–Green vortex at $t/t_{\text{ref}} = 0.0147$ for the case Re = 6.28.

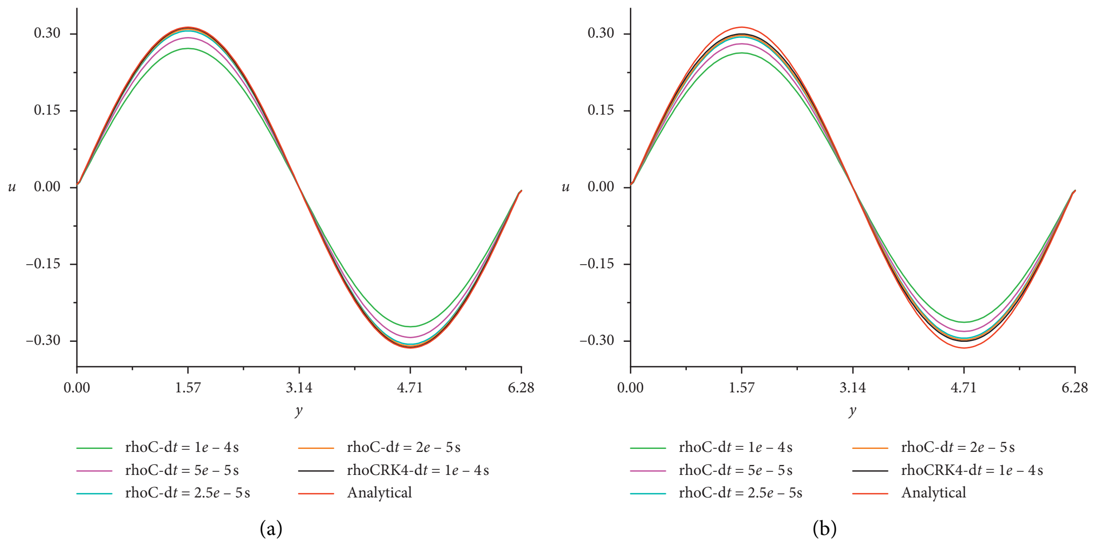


Figure 9: Velocity distribution in the Taylor–Green vortex problem at $t/t_{\text{ref}} = 0.0147$. (a)Re = 6.28; (b)Re = 125.6.

4.2. Comparison of Time-to-Solution. For the analysis of rhoCentralFoam and rhoCentralRK4Foam solvers we consider fixed box size $L = 2\pi$ m and $U_0 = 1$ m/s and two values of Reynolds number, $\text{Re} = LU_0/\nu$, to test the effect of viscosity. The computation is done on a 160×160 points uniform grid. The simulations run up to $t = 0.58$ s for Re = 6.28 and $t = 11.6$ s for Re = 125.6, which corresponds to the same nondimensional time $t/t_{\text{ref}} = 0.0147$ with $t_{\text{ref}} \equiv L^2/\nu$. At this time, the kinetic energy ε decreased to 10% from the initial value, as an illustrative example; Figure 8 shows the velocity magnitude contour for the Re = 6.28. Figure 9 presents the computed velocity profile $u(y)$ in the middle of the box, $x = L/2$. rhoC stands for rhoCentralFoam solver and rhoCRK4 stands for rhoCentralRK4Foam solver. It can be observed that rhoCentralRK4Foam shows an excellent agreement with the analytical profile for $\Delta t = 10^{-4}$ s (reducing the time step further does provide any further convergence for the grid used here). In order to achieve the same level of accuracy with rhoCentralFoam, time step has to be reduced by a factor of 5 to $\Delta t = 2 \times 10^{-5}$ s compared to rhoCentralRK4Foam. The same behavior is observed for the evolution of ε as shown in Figure 10.

In order to calculate the time-to-solution τ, we first evaluated the CPU time τ^0 per time step per grid point per

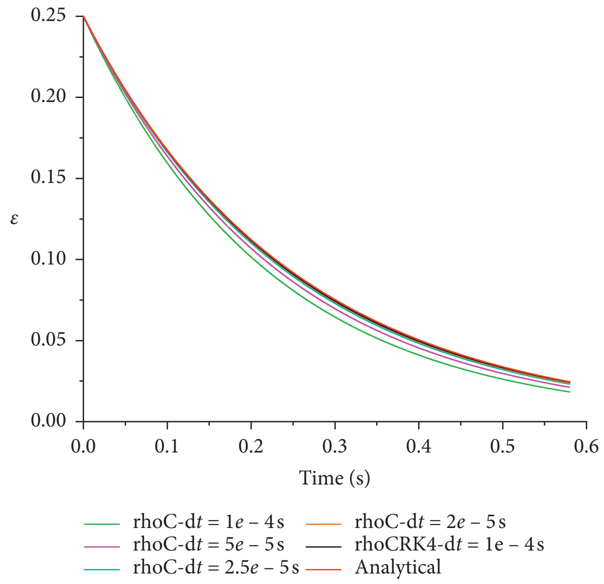


FIGURE 10: Evolution of total kinetic energy.

TABLE 6: CPU time per time step per grid point per core τ^0.

	$\tau^0_{\text{rhoCentral}}$	$\tau^0_{\text{rhoCentralRK4}}$	$\tau^0_{\text{rhoCentralRK4}}/\tau^0_{\text{rhoCentral}}$
Re = 6.28	$1.230e-06$ s	$3.590e-06$ s	2.919
Re = 125.6	$1.120e-06$ s	$3.630e-06$ s	3.241
Re = 1260	$1.189e-06$ s	$3.626e-06$ s	3.046
Re = 12600	$1.187e-06$ s	$3.603e-06$ s	3.035
Re = 126000	$1.191e-06$ s	$3.622e-06$ s	3.044

core for both solvers. We used an Intel Xeon E5-2695v4 processor installed on Theta and Bebop platforms at Argonne National Laboratory and reported the results of the tests in Table 6. Even though τ^0 depends on the specific processor used but the ratio $\tau^0_{\text{rhoCentralRK4}}/\tau^0_{\text{rhoCentral}}$ does not and so is a good indication of the relative performance of the two solvers. In the present case, this ratio is consistently found to be 3 for all Reynolds number; we made additional tests with much higher Reynolds up to Re $\sim 10^5$ to confirm this (see Table 6). Denoting by t_f the physical final time of the simulation, by N_{iter} the number of iterations and by N_{points} the number of grid points, the time-to-solution for the two solvers can be obtained as

$$\tau = \tau^0 \times N_{\text{iter}} \times N_{\text{points}}, \quad \text{with } N_{\text{iter}} = \frac{t_f}{\Delta t}, \tag{20}$$

and is reported in Table 7. It is concluded that to achieve the same level of accuracy, $\tau_{\text{rhoCentralRK4}}$ is about 1.5 to 1.6 times smaller than $\tau_{\text{rhoCentral}}$. Additionally, as we discussed in the scaling analysis, rhoCentralRK4Foam can achieve up to 123% improvement in scalability over rhoCentralFoam when using 4096 ranks. Thus, the reduction in time-to-solution for rhoCentralRK4Foam is expected to be even larger for large-scale, time-accurate simulations utilizing thousands of parallel cores.

TABLE 7: Time-to-solution τ for rhoCentralFoam and rhoCentral-RK4Foam.

	$\tau_{\text{rhoCentral}}$ $\Delta t = 2\times10^{-5}$ s	$\tau_{\text{rhoCentralRK4}}$ $\Delta t = 10^{-4}$ s	$\tau_{\text{rhoCentral}}/\tau_{\text{rhoCentralRK4}}$
Re = 6.28	848 s	532 s	1.59
Re = 125.6	826 s	539 s	1.53

5. Conclusion

In this study, we presented a new solver called rhoCentralRK4Foam for the integration of Navier–Stokes equations in the CFD package OpenFOAM. The novelty consists in the replacement of first-order time integration scheme of the native solver rhoCentralFoam with a third-order Runge–Kutta scheme which is more suitable for the simulation of time-dependent flows. We first analyzed the scalability of the two solvers for a benchmark case featuring transonic flow over a wing on a structured mesh and showed that rhoCentralRK4Foam can achieve a substantial improvement (up to 120%) in strong scaling compared to rhoCentralFoam. We also observed that the scalability becomes better as the problem size grows. Hence, even though OpenFOAM scalability is generally poor or decent at best, the new solver can at least alleviate this deficiency. We then analyzed the performance of the two solvers in the case of Taylor–Green vortex decay and compared the time-to-solution, which gives an indication of the workload needed to achieve the same level of accuracy of the solution (i.e., the same numerical error). For the problem considered here, we obtained a reduction of time-to-solution by a factor of about 1.5 when using rhoCentralRK4Foam compared to rhoCentralFoam due to the use of larger time steps to get to the final solution with the same numerical accuracy. This factor will be eventually even larger for larger-scale simulations employing thousands or more cores, thanks to the better speedup of rhoCentralRK4Foam. The proposed solver is potentially a useful alternative for the simulation of compressible flows requiring time-accurate integration like in direct or large-eddy simulations. Furthermore, the implementation in OpenFOAM can be easily generalized to different schemes of Runge–Kutta family with minimal code modification.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Argonne National Laboratory through grant #ANL 4J-30361-0030A titled “Multiscale modeling of complex flows." Numerical simulations were performed using the Director Discretionary allocation "OF_SCALING" in the Leadership Computing Facility at Argonne National Laboratory.

References

[1] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, "A tensorial approach to computational continuum mechanics using object-oriented techniques," *Computers in Physics*, vol. 12, no. 6, pp. 620–631, 1998.

[2] V. Vuorinen, J.-P. Keskinen, C. Duwig, and B. J. Boersma, "On the implementation of low-dissipative Runge-Kutta projection methods for time dependent flows using OpenFOAM," *Computers & Fluids*, vol. 93, pp. 153–163, 2014.

[3] C. Shen, F. Sun, and X. Xia, "Implementation of density-based solver for all speeds in the framework of openFOAM," *Computer Physics Communications*, vol. 185, no. 10, pp. 2730–2741, 2014.

[4] D. Modesti and S. Pirozzoli, "A low-dissipative solver for turbulent compressible flows on unstructured meshes, with OpenFOAM implementation," *Computers & Fluids*, vol. 152, pp. 14–23, 2017.

[5] S. Li and R. Paoli, "Modeling of ice accretion over aircraft wings using a compressible OpenFOAM solver," *International Journal of Aerospace Engineering*, vol. 2019, Article ID 4864927, 11 pages, 2019.

[6] Q. Yang, P. Zhao, and H. Ge, "ReactingFoam-SCI: an open source CFD platform for reacting flow simulation," *Computers & Fluids*, vol. 190, pp. 114–127, 2019.

[7] M. Culpo, *Current Bottlenecks in the Scalability of OpenFOAM on Massively Parallel Clusters*, Partnership for Advanced Computing in Europe, Brussels, Belgium, 2012.

[8] O. Rivera and K. Furlinger, "Parallel aspects of OpenFOAM with large eddy simulations," in *Proceedings of the 2011 IEEE International Conference on High Performance Computing and Communications*, pp. 389–396, Banff, Canada, September 2011.

[9] A. Duran, M. S. Celebi, S. Piskin, and M. Tuncel, "Scalability of OpenFOAM for bio-medical flow simulations," *The Journal of Supercomputing*, vol. 71, no. 3, pp. 938–951, 2015.

[10] Z. Lin, W. Yang, H. Zhou et al., "Communication optimization for multiphase flow solver in the library of OpenFOAM," *Water*, vol. 10, no. 10, pp. 1461–1529, 2018.

[11] R. Ojha, P. Pawar, S. Gupta, M. Klemm, and M. Nambiar, "Performance optimization of OpenFOAM on clusters of Intel Xeon Phi™ processors," in *Proceedings of the 2017 IEEE 24th International Conference on High Performance Computing Workshops (HiPCW)*, Jaipur, India, December 2017.

[12] C. J. Greenshields, H. G. Weller, L. Gasparini, and J. M. Reese, "Implementation of semi-discrete, non-staggered central schemes in a colocated, polyhedral, finite volume framework, for high-speed viscous flows," *International Journal for Numerical Methods in Fluids*, vol. 63, pp. 1–21, 2010.

[13] A. Kurganov and E. Tadmor, "New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations," *Journal of Computational Physics*, vol. 160, no. 1, pp. 241–282, 2000.

[14] A. Kurganov, S. Noelle, and G. Petrova, "Semidiscrete central-upwind schemes for hyperbolic conservation laws and Hamilton–Jacobi equations," *SIAM Journal on Scientific Computing*, vol. 23, no. 3, pp. 707–740, 2006.

[15] J. A. Heyns, O. F. Oxtoby, and A. Steenkamp, "Modelling high-speed viscous flow in OpenFOAM," in *Proceedings of the 9th OpenFOAM Workshop*, Zagreb, Croatia, June 2014.

[16] M. S. Liou and C. J. Steffen Jr., "A new flux splitting scheme," *Journal of Computational Physics*, vol. 107, no. 23, p. 39, 1993.

[17] D. Drikakis, M. Hahn, A. Mosedale, and B. Thornber, "Large eddy simulation using high-resolution and high-order methods," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1899, pp. 2985–2997, 2019.

[18] K. Harms, T. Leggett, B. Allen et al., "Theta: rapid installation and acceptance of an XC40 KNL system," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 1, p. 11, 2019.

[19] S. S. Shende and A. D. Malony, "The TAU parallel performance system," *The International Journal of High Performance Computing Applications*, vol. 20, no. 2, pp. 287–311, 2006.

[20] C. Chevalier and F. Pellegrini, "PT-Scotch: a tool for efficient parallel graph ordering," *Parallel Computing*, vol. 34, no. 6–8, pp. 318–331, 2008.

[21] V. Schmitt and F. Charpin, "Pressure distributions on the ONERA-M6-wing at transonic mach numbers," in *Experimental Data Base for Computer Program Assessment and Report of the Fluid Dynamics Panel Working Group 04*, AGARD, Neuilly sur Seine, France, 1979.

[22] P. Woodward and P. Colella, "The numerical simulation of two-dimensional fluid flow with strong shocks," *Journal of Computational Physics*, vol. 54, no. 1, pp. 115–173, 1984.

[23] G. M. Amdahl, "Validity of the single processor approach to achieving large-scale computing capabilities," in *Proceedings of the 1967, spring joint computer conference on—AFIPS'67 (Spring)*, vol. 30, pp. 483–485, Atlantic, NJ, USA, April 1967.

[24] G. Axtmann and U. Rist, "Scalability of OpenFOAM with large-leddy simulations and DNS on high-performance sytems," in *High-Performance Computing in Science and Engineering*, W. E. Nagel, Ed., Springer International Publishing, Berlin, Germany, pp. 413–425, 2016.

[25] A. Shah, L. Yuan, and S. Islam, "Numerical solution of unsteady Navier-Stokes equations on curvilinear meshes," *Computers & Mathematics with Applications*, vol. 63, no. 11, pp. 1548–1556, 2012.

[26] J. Kim and P. Moin, "Application of a fractional-step method to incompressible Navier-Stokes equations," *Journal of Computational Physics*, vol. 59, no. 2, pp. 308–323, 1985.

[27] A. Quarteroni, F. Saleri, and A. Veneziani, "Factorization methods for the numerical approximation of Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 188, no. 1–3, pp. 505–526, 2000.