15th World Congress on Computational Mechanics (WCCM-XV)
8th Asian Pacific Congress on Computational Mechanics (APCOM-VIII)
Virtual Congress: 31 July – 5 August 2022
S. Koshizuka (Ed.)

PREPARING THE PATH FOR THE EFFICIENT SIMULATION OF TURBULENT COMPRESSIBLE INDUSTRIAL FLOWS WITH ROBUST COLLOCATED RK-DG SOLVERS

Rasha Al Jahdali[1]*, Lisandro Dalcin[1], and Matteo Parsani[1]

[1] King Abdullah University of Science and Technology (KAUST), Extreme Computing Research Center (ECRC), Computer, Electrical and Mathematical Sciences & Engineering (CEMSE)
Thuwal, 23955-6900, Kingdom of Saudi Arabia
rasha.aljahdali@kaust.edu.sa, dalcinl@gmail.com, matteo.parsani@kaust.edu.sa.

Key words: Compressible flows, Entropy stable schemes, Robust Runge–Kutta schemes, Cloud Computing

Abstract. *We present an analysis of the performance of some standard and optimized explicitly Runge–Kutta schemes that are equipped with CFL-based and error-based time step adaptivity when they are coupled with the relaxation procedure to achieve fully-discrete entropy stability for complex compressible flow simulations. We investigate the performance of the temporal integration algorithms by simulating the flow past the NASA juncture flow model using the in-house KAUST SSDC hp-adaptive collocated entropy stable discontinuous Galerkin solver. In addition, we present a preliminary analysis of the performance of the SSDC framework on the Amazon web service cloud computing. The results indicate that SSDC scales well on the most recent and exotic computing architectures available on the Amazon cloud platform. Our findings might help select a more robust and efficient temporal integration algorithm and guide the choice of the EC2 AWS instances that give the best price and wall-clock-time performance to simulate industrially relevant turbulent flow problems.*

1 Introduction

During the last two decades, a lot of effort has been devoted to developing robust and efficient adaptive high-order accurate solvers for compressible computational fluid dynamics (CFD). Discontinuous Galerkin, spectral difference, and flux reconstruction approaches are some of the potential possibilities among current, unstructured high-order methods. Although these approaches are ideally suited for smooth solutions, they are often designed in the context of linear stability. Hence, they often suffer from numerical instabilities if the flow, for instance, has unresolved physical features (e.g., under-resolved turbulent structures).

For nonlinear systems of conservation laws, such as the compressible Navier–Stokes equations, the entropy stability framework [1] allows mimicking some of the continuous system's key features closely. This has resulted in the development of entropy-stable high-order approaches that are provably nonlinearly stable (entropy stable) so long as the positivity of the thermodynamic variables is maintained [2, 3]. High-order collocated entropy stable discontinuous Galerkin methods are used in this work. Specifically, we use the high-performance entropy stable solver SSDC presented in [4].

Recently, the relaxation procedure has been developed to ensure the conservation, dissipation, or other

*Corresponding author.

solution qualities with respect to any convex functional by multiplying each Runge–Kutta update by a relaxation parameter [5, 6]. Over the last year, the relaxation method has been adopted and utilized effectively in numerous fluid dynamics problems. Herein, we use some standard and optimized explicitly Runge–Kutta (ERK) schemes that are equipped with time step adaptivity and the relaxation approach to attain the fully-discrete entropy stability. Fully-entropy stability is a feature that has been highlighted in the NASA CFD vision 2030 as an essential element of future CFD frameworks [7] because it provides provably robustenss for industrial complex problems in the aeronautics and aerospace fields.

Very often, industrially-relevant simulations require vast computations that can only be done on supercomputers available to governments or the most well-heeled corporations. Therefore, grids and on-demand cloud resources should be seriously considered as potential alternatives for traditional local computing clusters in many contexts and realities. A significant benefit of cloud computing is the ability to acquire hardware without the need for upfront capital investment or ongoing IT maintenance costs, and software can be obtained on a pay-per-use basis rather than an annual license. Cloud computing allows for large-scale high-performance computing resources to be rented on a cost-effective basis. Therefore, because the industry is extremely interested in using this on-demand technology for performing engineering simulations, testing and assessing the performance of a unique prototype for next generation compressible CFD solvers, such as SSDC, is relevant.

In this work, we perform the simulation of the NASA juncture flow experiment, which is a new effort specifically designed to collect validation data in the juncture region of a wing-body configuration [8]. Current turbulence models routinely employed by Reynolds-averaged Navier–Stokes CFD solvers are inconsistent in their prediction of corner flow separation in aircraft juncture regions, so experimental data in the near-wall region of such a configuration are useful both for assessment as well as for turbulence model improvement [8]. The purpose of this study is to provide some results on the performance of the SSDC solver on the Amazon ParallelCluster, and to measure the capabilities of different instance types of Amazon cloud. An EC2 instance has a default number of CPU cores, which varies according to instance type. In this study, we restricted the instance to just utilize the available physical CPU cores, meaning that we disabled hyperthreading. In particular, we consider the some recent Intel, AMD, and Arm EC2 instances and up to 8 computing nodes.

2 Numerical discretization of the compressible Navier–Stokes equations

2.1 Spatial discretization

The compressible Navier–Stokes equations in Cartesian coordinates read

$$\begin{cases} \frac{\partial q}{\partial t} + \sum\limits_{m=1}^{3} \frac{\partial f^I_{x_m}}{\partial x_m} = \sum\limits_{m=1}^{3} \frac{\partial f^V_{x_m}}{\partial x_m}, & \forall (x_1,x_2,x_3) \in \Omega, \quad t \geq 0, \\ q(x_1,x_2,x_3,t) = g^{(B)}(x_1,x_2,x_3,t), & \forall (x_1,x_2,x_3) \in \Gamma, \quad t \geq 0, \\ q(x_1,x_2,x_3,0) = g^{(0)}(x_1,x_2,x_3), & \forall (x_1,x_2,x_3) \in \Omega, \end{cases} \tag{1}$$

where the vectors q, $f^I_{x_m}$, and $f^V_{x_m}$ are denoted as the conserved variables, the inviscid fluxes, and the viscous fluxes, respectively. The boundary data, $g^{(B)}$, and the initial condition, $g^{(0)}$, are assumed to be in $L^2(\Omega)$, with the further assumption that $g^{(B)}$ coincides with linear, well–posed boundary conditions, prescribed in such a way that either entropy conservation or entropy stability is achieved.

The vector of conserved variables is given by $q = [\rho, \rho\mathcal{U}_1, \rho\mathcal{U}_2, \rho\mathcal{U}_3, \rho\mathcal{E}]^{\mathrm{T}}$, where ρ denotes the density, $\mathcal{U} = [\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3]^{\mathrm{T}}$ is the velocity vector, and $\mathcal{E}$ is the specific total energy.

The compressible Navier–Stokes equations given in (1) have a convex extension (a redundant sixth equation constructed from a nonlinear combination of the mass, momentum, and energy equations) that, when integrated over the physical domain, only depends on the boundary data and on negative semi-definite dissipation terms. Such convex extension provides a mechanism for proving stability in the L^2 norm; it depends on an entropy function, $\mathfrak{s}$, that is constructed from the thermodynamic entropy as $\mathfrak{s} = -\rho s$, where s is the thermodynamic entropy. The approximation of the compressible Navier–Stokes equations proceeds by partitioning the domain Ω into K non-overlapping elements. On the κ-th element, the generic entropy stable discretization of (1) reads

$$\frac{\mathrm{d}\boldsymbol{q}_\kappa}{\mathrm{d}t} + \sum_{m=1}^{3} 2\mathsf{D}_{x_m}^{I,\kappa} \circ \mathsf{F}_{x_m}\left(\boldsymbol{q}_\kappa, \boldsymbol{q}_\kappa\right) \mathbf{1}_\kappa = \sum_{m,j=1}^{3} \mathsf{D}_{x_m}^{V_1,\kappa} \left[\mathsf{C}_{m,j}\right] \boldsymbol{\theta}_j + \mathrm{SAT}^I + \mathrm{SAT}^V + \mathrm{diss}^I + \mathrm{diss}^{V_1}, \tag{2}$$

where the vector $\boldsymbol{q}_\kappa$ is the discrete solution at the mesh nodes, and the vectors diss^I and diss^V are interface dissipation terms for the inviscid and viscous portions of the equations, respectively. To mimic the continuous entropy stability analysis, the derivatives of the inviscid fluxes are approximated in a special way [9], i.e.,

$$\frac{\partial f_{x_m}^I}{\partial x_m} \approx 2\mathsf{D}_{x_m}^{I,\kappa} \circ \mathsf{F}_{x_m}\left(\boldsymbol{q}_\kappa, \boldsymbol{q}_\kappa\right) \mathbf{1}_\kappa, \tag{3}$$

where $\mathsf{D}_{x_m}^{I,\kappa}$ is a differentiation matrix for the x_m direction, $\circ$ is the Hadamard product (entry-wise multiplication), $\mathsf{F}_{x_m}\left(\boldsymbol{q}_\kappa, \boldsymbol{q}_\kappa\right)$ is a two-point flux function matrix, and $\mathbf{1}_\kappa$ is a vector of ones. Numerically, we solve PDEs in computational coordinates, $(\xi_1, \xi_2, \xi_3) \subset \mathbb{R}^3$, where each cell is locally transformed to the reference element using a pull-back curvilinear coordinate transformation. The matrix $\mathsf{D}_{x_m}^{I,\kappa}$ is an SBP operator constructed by discretizing the skew-symmetric split of the Cartesian derivative [9].

The differentiation matrix $\mathsf{D}_{x_m}^{I,\kappa}$ is constructed as

$$\mathsf{D}_{x_m}^{I,\kappa} \equiv \frac{1}{2}\mathsf{J}_\kappa^{-1} \sum_{l=1}^{3} \left(\mathsf{D}_{\xi_l} \left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right]_\kappa + \left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right]_\kappa \mathsf{D}_{\xi_l} \right), \tag{4}$$

where J_κ denotes the determinant of the discrete Jacobian, D_{ξ_l} is an SBP operator approximating $\frac{\partial}{\partial \xi}$, while $\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right]_\kappa$ denotes the discrete metric terms that need to satisfy a discrete version of the geometric conservation law (GCL) conditions [9]. On the other hand, the differentiation matrix, $\mathsf{D}_{x_m}^{V_1,\kappa}$, is constructed as

$$\mathsf{D}_{x_m}^{V_1,\kappa} \equiv \mathsf{J}_\kappa^{-1} \sum_{l=1}^{3} \mathsf{D}_{\xi_l} \left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right]_\kappa. \tag{5}$$

The metric terms $\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right]_\kappa$ and $\left[\mathcal{J} \frac{\partial \xi_a}{\partial x_j} \right]_\kappa$ have been color-coded to highlight that these terms can be computed in different ways.

The terms SAT are the Simultaneous Approximation Terms that weakly couple neighboring elements (see, for instance, [10, 9]) or impose boundary conditions (see, for instance, [11, 12]). Here, they are composed of an inviscid and a viscous contribution, *i.e.*, $\mathrm{SAT} = \mathrm{SAT}^I + \mathrm{SAT}^V$, and are generically constructed from the difference of the flux on the element boundary Γ_κ and a numerical flux, in order to properly approximate

$$\int_{\Gamma_\kappa} \mathfrak{v} \left(f_{x_m} - f_{x_m}^* \right) n_{x_m} \mathrm{d}\Gamma,$$

where n_{x_m} is the m-th component of the outward facing unit normal, and v is a smooth test function. Moreover, $f^*_{x_m}$ is the unique (for conservation) numerical flux function. The specific form of this flux function will depend on what type of SAT is being enforced, i.e., interface or boundary condition. In addition, SAT terms are designed to extend the mimetic properties of the scheme and its stability from the element level to the full discretization. Stability proofs also hold in the presence of p- and hp-nonconforming interfaces between elements by using SBP-preserving interpolation operators and non-standard approximations of the metric terms; see [13] for p-nonconforming interfaces, and [9] for hp-nonconforming interfaces.

At the beginning of the simulation, the metric terms are computed in order to satisfy the discrete GCL conditions. Explicit Runge–Kutta schemes are then used to integrate the system of nonlinear ordinary differential equations (ODEs) arising from the spatial discretization of the compressible Navier–Stokes equations.

2.2 Temporal discretization

A general (explicit or implicit) Runge–Kutta method with s stages can be represented by its Butcher tableau which is composed by a square matrix A of dimensions $s \times s$, and two column vectors b and c of length s. All these coefficients are real numbers. In the context of the relaxation procedure [5], the basic idea to enforce conservation, dissipation, or other solution properties with respect to a convex functional is to scale the weights b_i of a given Runge–Kutta method by a real-value parameter $\tilde{\gamma}$. Thus, using an s-stage Runge–Kutta scheme, a time step from $u^n \approx u(t_n)$ is given by

$$y_i = u^n + \Delta t \sum_{j=1}^{s} a_{ij} f(t_n + c_j \Delta t, y_j), \qquad u^{n+1}_{\tilde{\gamma}} = u^n + \tilde{\gamma}_n \Delta t \sum_{i=1}^{s} b_i f(t_n + c_i \Delta t, y_i), \tag{6}$$

where y_i are the stage values of the Runge–Kutta method. For the global relaxation procedure, $\tilde{\gamma}_n$ is a root of a global nonlinear algebraic equation for η [5]. On the contrary, for the local relaxation approach, $\tilde{\gamma}_n = min_{\kappa} \tilde{\gamma}_{n,\kappa}$, *i.e.*, the minimum among the roots of a local equation for η_κ for each cell, κ, of the discretized spatial domain [6]. If $\tilde{\gamma}_n = 1$, we recover the standard Runge–Kutta method.

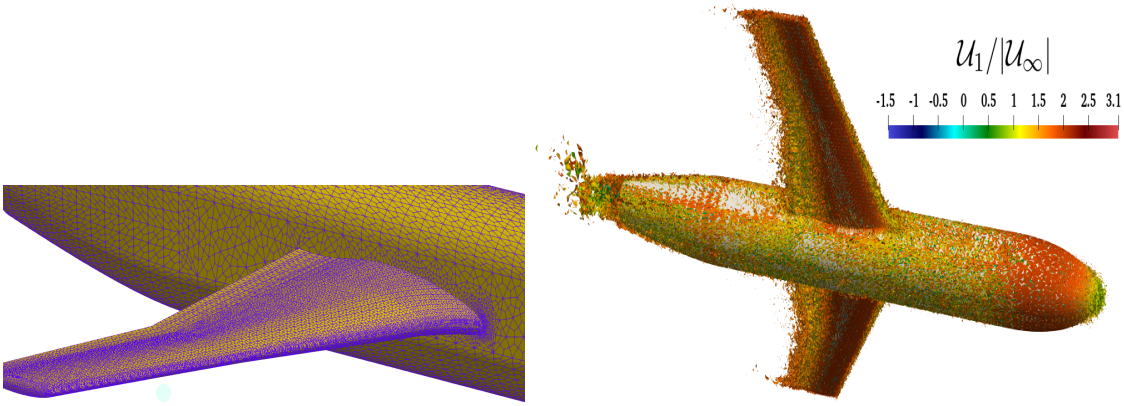
As required by the relaxation theory [5], we select ERK schemes with positive weights, b_i. In particular, we use two second-order schemes with CFL-based timestep controller, *i.e.*, ERK(4,2) and ERK(8,2) [14], one third-order scheme with CFL-based timestep controller, *i.e.*, ERK(5,3) [14], three third-order schemes with error-based timestep controller, *i.e.*, BSRK(4,3), SSP3(2)4[3S*+] and RK3(2)5F[3S*+] [15], the classical four-stage fourth-order scheme with CFL-based timestep controller, *i.e.*, ERK(4,4), one fifth-order scheme with error-based timestep controller, *i.e.*, BSRK(8,5), and one sixth-order scheme with error-based timestep controller, *i.e.*, VRK(9,6). All these schemes have been throughly tested and verified when used in combination with an adaptive timestep strategy [14]. In addition, some of these ERK schemes (e.g., ERK(4,4), BSRK(4,3), and BSRK(8,5)) are the work-horses of commercial and national laboratories compressible computational fluid dynamics solvers.

3 Numerical results

This section presents an analysis of the performance of certain standard and optimized explicitly Runge–Kutta schemes equipped with CFL-based and error-based time step adaptivity coupled with the local and global relaxation techniques. In addition to that, we analyze the performance of the SSDC solver on the Amazon EC2 cluster.

3.1 NASA juncture flow configuration

We use the NASA juncture configuration with a wing based on the DLR-F6 geometry and a leading edge horn to mitigate the effect of the horseshoe vortex over the wing-fuselage juncture [8]. The Reynolds number is $\mathrm{Re} = 2.4 \times 10^6$ and the freestream Mach number is $\mathrm{Ma} = 0.189$. The angle of attack is $\mathrm{AoA} = -2.5^\circ$. We perform simulations in free air conditions, ignoring both the sting and the mast. An illustration of the grid structure is shown in Fig. 1 (a), whereas the vortical features visualized using isocontours of the Q-criterion colored by normalized instantaneous velocity $\mathcal{U}_1/|\mathcal{U}_\infty|$ are shown in Fig. 1 (b).



(a) (b)

Figure 1: NASA juncture flow experiment.

3.2 Performance of relaxation and adaptive schemes

To assess the performance of the selected adaptive explicit Runge–Kutta schemes coupled with the global or local relaxation procedure, we perform 600 time steps starting from a fully developed turbulent flow. The simulations are carried out using 512 CPU cores of the supercomputer Shaheen XC40 hosted at KAUST. The number of function evaluations and the arithmetic average of the wall-clock time for the three configurations without, with the global, and with local relaxation procedures are reported in Table 1. The CFL-based and error-based time step controllers are highlighted in yellow and gray, respectively. As we can see in Table 1, more function evaluations are needed when the relaxation technique is active. Moreover, approximately the same number of function evaluations are required for the global and local relaxation procedures. This leads to the fact that the relaxation approaches require more wall-clock time than the same scheme without relaxation. When the number of function evaluations with and without relaxation is close to each other, we point out that the difference in wall-clock time is practically due to time spent by nonlinear algebraic solvers on computing the relaxation parameter, $\tilde{\gamma}_n$. Remarkably, the error-based controllers are significantly more efficient than CFL-based controllers for third-order accurate schemes, regardless of whether the relaxation method is enabled. Furthermore, the fifth- and sixth-order adaptive schemes with error-based time step adaptivity are faster than the second and third-order accurate methods whose time step is controlled by the CFL number. As demonstrated in Table 1,

the relaxation Runge–Kutta schemes with CFL-based timestep controller require a number of function evaluations very close to the schemes without the relaxation approach. According to this, it appears that with the existing CFL-based controller, the time step approaches the value required for entropy stability, *i.e.*, $\tilde{\gamma}_n \approx 1$. We highlight that the latter behavior is not observed for the computations performed with the error-based controller. Finally, we would like to point out that all simulations are stable, even without using a relaxation method.

ERK scheme	Approach	# Function evaluations	Wall-clock time [s]
ERK(4,2)	Without relaxation	1,004	1.4062e+02
	Global relaxation	1,008	1.4108e+02
	Local relaxation	1,008	1.4081e+02
ERK(8,2)	Without relaxation	1,000	1.3995e+02
	Global relaxation	1,024	1.4340e+02
	Local relaxation	1,008	1.4125e+02
BSRK(4,3)	Without relaxation	817	1.1783e+02
	Global relaxation	1,084	1.5248e+02
	Local relaxation	1,088	1.5327e+02
SSP3(2)4[3S*+]	Without relaxation	1,356	1.9057e+02
	Global relaxation	1,352	1.9029e+02
	Local relaxation	1,356	1.9117e+02
RK3(2)5F[3S*+]	Without relaxation	817	1.1780e+02
	Global relaxation	1,084	1.5228e+02
	Local relaxation	1,088	1.5299e+02
ERK(5,3)	Without relaxation	1,080	1.5131e+02
	Global relaxation	1,080	1.5140e+02
	Local relaxation	1,085	1.5187e+02
ERK(4,4)	Without relaxation	1,492	2.0858e+02
	Global relaxation	1,504	2.1078e+02
	Local relaxation	1,496	2.0941e+02
BSRK(8,5)	Without relaxation	904	1.2841e+02
	Global relaxation	952	1.3411e+02
	Local relaxation	952	1.3404e+02
VRK(9,6)	Without relaxation	873	1.2462e+02
	Global relaxation	999	1.4170e+02
	Local relaxation	999	1.4168e+02

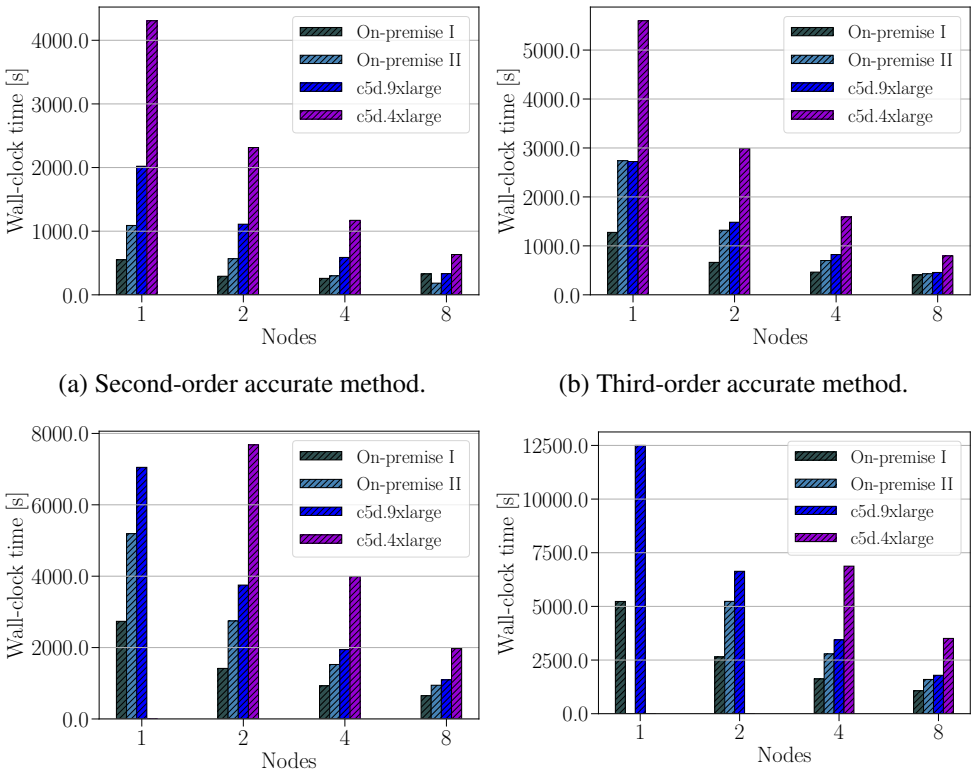
Table 1: Performance of the schemes used for the simulation of the flow past the NASA juncture flow model.

3.3 Performance evaluation of on-premises cluster and AWS cloud cluster

In this section, we conduct a detailed, quantitative analysis to evaluate the performance of SSDC on the on-premises cluster Ibex at KAUST and the Amazon EC2 when used to run complex flow past the NASA juncture flow experiment. The goal of this study is to determine the suitability of Amazon's cloud-based high-performance computing HPC offering for resolving complicated industrial flow problems and define a range of architectures that give the minimum time-to-solution at the minimum cost. The wall-clock time in seconds for on-premises cluster and EC2 instances for different architectures of CPUs are reported in Figs. 2 and 4. For the simulations on the on-premises cluster, we run this test case using 40

(dark green) and 20 (light blue) physical cores, that is the maximum and half of the maximum physical cores per nodes. In both cases, each core is assigned a single MPI thread. For the on-premises cluster with the Intel Cascade Lake architecture, the wall-clock time decrease as the number of nodes increases, as shown in Fig. 2. Clearly, for the same node counts, the on-premises cluster with 40 physical cores provides a speed-up of approximately 50% more than that with 20 physical cores. However, this speed-up percentage generally decreases as the number of nodes increases. We observe this behavior for 4 and 8 nodes in both setups, *i.e.,* when using 40 or 20 physical cores. Together with the problem size, the main reasons responsible for the degradation in performance on-premise cluster are i) the nodes are non-exclusive with a daily occupancy of about 75% of medium-to-large scales jobs, and therefore, the more computing nodes, the higher the chances to share the computing resources with other users ii) the more computing nodes, the more partitioned the workload on each node and hence, increasing communication messages which are also sent and received using the shared computing resources. As a result, if the problem size is not "sufficiently large", the network connectivity is not "sufficiently fast" and shared with other users, the performance degrades to the point that a larger number of computing nodes is detrimental.

For the EC2 c5d instances and all orders of accuracy, the simulations are speed-up as the number of CPUs increases. In particular, going from one to eight computing nodes yields a speed-up factor of approximately 6.5 for both instances and the second- and third-order accurate methods. Among the EC2 instances, the smaller time-to-solution is given by the c5d.9xlarge instance for all the orders of accuracy. It is also important to note that, the on-premises cluster runs with 40 and 20 MPI threads are still faster than the EC2 instances. For four computing nodes, the runs on the on-premises cluster with 20 MPI threads are slightly faster or on par with the runs performed with the c5d.4xlarge instance. Thus, it appears that using half of the physically available cores on a node has positive effects on the wall-clock time because shared intra-node network is used by only half of the physically available cores, and importantly, the workload per core allows to hide better communications behind computations. We further highlight that, for higher-order accurate methods, the number of degrees of freedom increases, and therefore a large number of computing nodes is needed. Consequently, some cases cannot be run on the smaller node count due to the limitations of the computing resources. This occurs when we use 1 and 2 nodes for the fourth- and fifth-order accurate methods, as shown in Figs. 2c and 2d.



(a) Second-order accurate method.

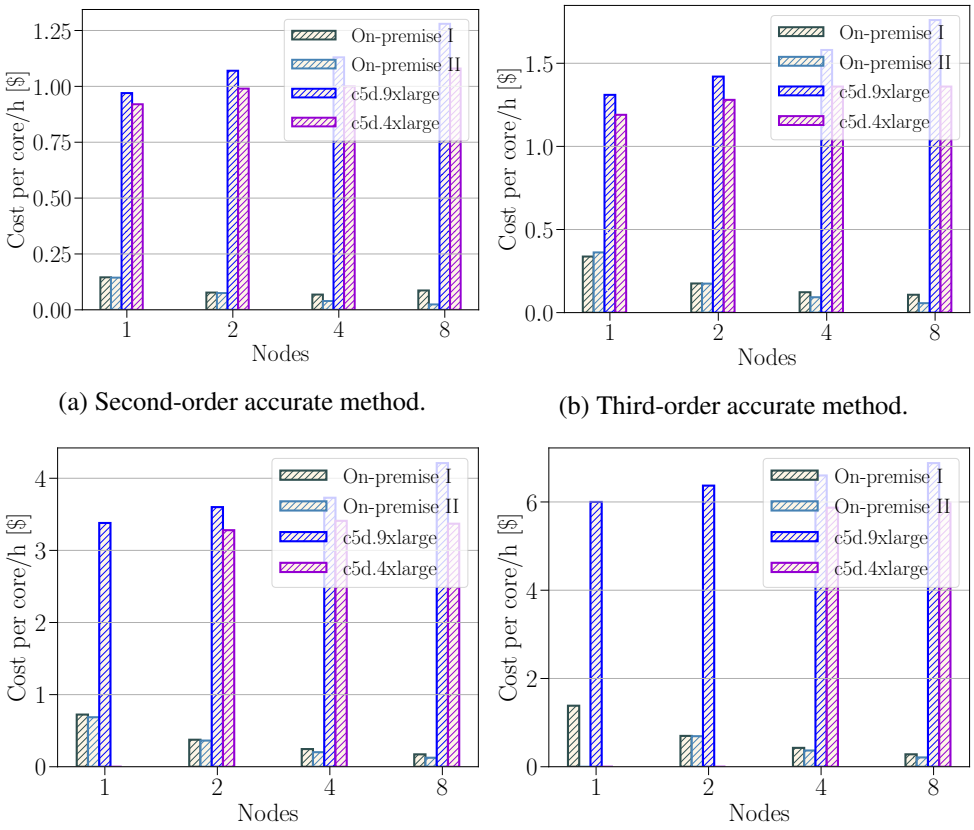
(b) Third-order accurate method.

(c) Fourth-order accurate method.

(d) Fifth-order accurate method.

Figure 2: Wall-clock time vs. number of Intel nodes for the on-premises cluster, and c5d instances.

In terms of wall-clock time, the results are in favor of the simulations ran on 8 nodes on the on-premises cluster. In particular, the on-premises cluster with 40 physical cores delivers the results in the least amount of time. On the other hand, in terms of cost, the plots shown in Fig. 3, lead to different conclusions: For any order of accuracy of the solver and number of nodes tested, it is more convenient to run on the on-premises cluster. In particular, the simulations performed with 20 and 40 physical cores cost at least a factor of two less than the simulations performed on the EC2 instances. Running on the on-premises cluster is always convenient also when using the maximum number of physical cores available (*i.e.*, 40 cores).



(a) Second-order accurate method.

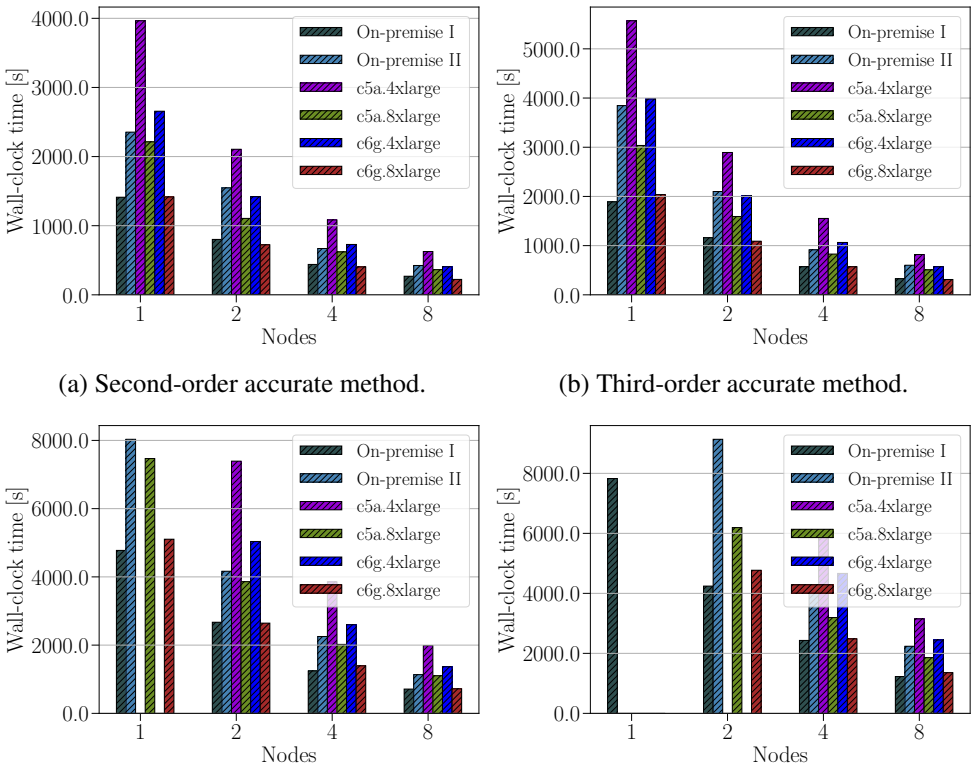
(b) Third-order accurate method.

(c) Fourth-order accurate method.

(d) Fifth-order accurate method.

Figure 3: Price performance of Intel nodes for on-premises cluster "Ibex", and c5d instances.

For the on-premises cluster with the AMD Rome architecture (see Fig. 4), we noticed that by increasing the number of nodes, the wall-clock time decreases substantially for all the order of accuracy. The time-to-solution delivered by the EC2 AMD EPYC (*i.e.*, c5a) also decreases by increasing the number of nodes. In addition, we observe that by going from the c5a.4xlarge instances to the c5a.8xlarge, the wall-clock time decreases by about 35%-to-40%. The instances c6g corresponds to the Arm-based AWS Graviton2 processors and deliver the best performance, *i.e.*, the least wall-clock time. In addition, we observe that on the Arm-based architecture, doubling the number of nodes leads to approximately halving the time-to-solution. This corresponds to almost a perfect scaling. We also observe that the time-to-solution delivered by four Arm nodes is very close to the time-to-solution required by eight AMD EPYC nodes. Remarkably, the wall clock time on the on-premises cluster for eight nodes with 40 MPI processes per node is practically the same as the wall-clock time required by the Arm-based architecture c6g.8xlarge. Finally, we notice that the test case does not fit in memory for the fourth- and fifth-order accurate schemes and some of the on-premises and AWS instances.



(a) Second-order accurate method.

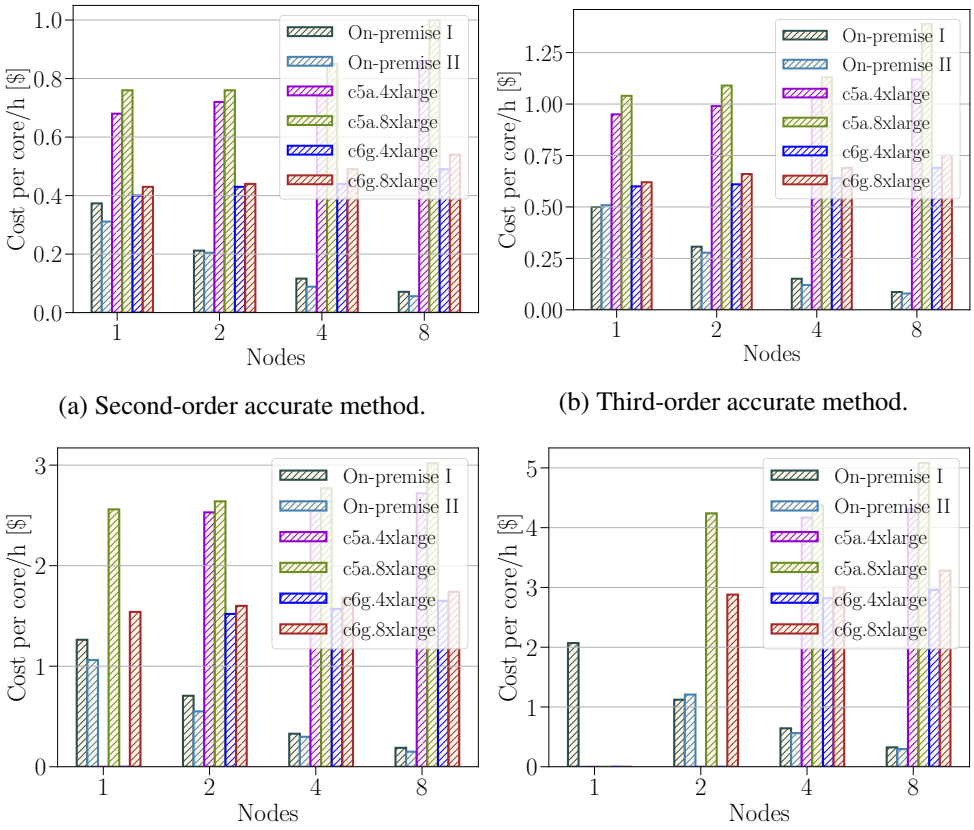
(b) Third-order accurate method.

(c) Fourth-order accurate method.

(d) Fifth-order accurate method.

Figure 4: Wall-clock time vs. the number of AMD nodes for on-premises cluster "Ibex", and c5a and c6g instances.

To conclude this section, we analyze the cost for simulating the test case with AMD and Arm architectures. As shown in Figures 5, the computations on the on-premises cluster are substantially cheaper than those performed on the EC2 parallel cluster. Among the EC2 architectures, Arm is the cheapest one for both c6g.4xlarge and c6g.8xlarge.



(a) Second-order accurate method.

(b) Third-order accurate method.

(c) Fourth-order accurate method.

(d) Fifth-order accurate method.

Figure 5: Price performance of AMD nodes for on-premises cluster "Ibex", and c5a and c6g instances.

4 Conclusions

In this work, we reported the performance of the adaptive RK-DG entropy stable solver with and without the relaxation procedure in time implemented in the SSDC framework. We tested the solver using the on-premises Ibex cluster and AWS ParallelCluster. In general, the adaptive ERK schemes coupled with the relaxation procedures are more expensive than the adaptive schemes without relaxation. In addition, we found that the schemes equipped with the error-based controller are substantially more efficient than those using the CFL-based controller, whether or not the relaxation procedure is activated. By performing a preliminary study on the AWS ParallelCluster, we have observed that the instances that allow performing the simulation in the least amount of time and cost are the Arm-based processors. However, the numerical results show that the on-premises cluster Ibex performs reasonably well in terms of price and wall-clock-time.

REFERENCES

[1] E. Tadmor, "Entropy stability theory for difference approximations of nonlinear conservation laws and related time-dependent problems," Acta Numerica, vol. 12, no. 1, pp. 451–512, 2003.

[2] M. H. Carpenter, M. Parsani, E. J. Nielsen, and T. C. Fisher, "Towards an entropy stable spectral element framework for computational fluid dynamics," in 54th AIAA Aerospace Sciences Meeting, p. 1058, 2016.

[3] T. Chen and C. W. Shu, "Entropy stable high order discontinuous Galerkin methods with suitable quadrature rules for hyperbolic conservation laws," Journal of Computational Physics, vol. 345, pp. 427 – 461, 2017.

[4] M. Parsani, R. Boukharfane, I. R. Nolasco, D. C. Del Rey Fernández, S. Zampini, B. Hadri, and L. Dalcin, "High-order accurate entropy-stable discontinuous collocated Galerkin methods with the summation-by-parts property for compressible CFD frameworks: Scalable SSDC algorithms and flow solver," Journal of Computational Physics, vol. 424, p. 109844, 2021.

[5] H. Ranocha, M. Sayyari, L. Dalcin, M. Parsani, and D. I. Ketcheson, "Relaxation Runge-Kutta methods: Fully-discrete explicit entropy-stable schemes for the compressible Euler and Navier–Stokes equations," SIAM Journal on Scientific Computing, vol. 42, pp. A612–A638, 03 2020.

[6] H. Ranocha, L. Dalcin, and M. Parsani, "Fully discrete explicit locally entropy-stable schemes for the compressible Euler and Navier–Stokes equations," Computers & Mathematics with Applications, vol. 80, no. 5, pp. 1343–1359, 2020.

[7] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis, "CFD vision 2030 study: A path to revolutionary computational aerosciences," NASA-CR-2014-218178, 2014.

[8] C. L. Rumsey and J. H. Morrison, "Goals and status of the NASA juncture flow experiment," NATO, STO-MP-AVT-246, 2016.

[9] D. C. Del Rey Fernández, M. H. Carpenter, L. Dalcin, S. Zampini, and M. Parsani, "Entropy stable h/p-nonconforming discretization with the summation-by-parts property for the compressible Euler and Navier–Stokes equations," SN Partial Differential Equations and Applications, vol. 1, no. 2, pp. 1–54, 2020.

[10] M. Parsani, M. H. Carpenter, and E. J. Nielsen, "Entropy stable discontinuous interfaces coupling for the three-dimensional compressible Navier–Stokes equations," Journal of Computational Physics, vol. 290, pp. 132–138, 2015.

[11] L. Dalcin, D. Rojas, S. Zampini, D. C. Del Rey Fernández, M. H. Carpenter, and M. Parsani, "Conservative and entropy stable solid wall boundary conditions for the compressible Navier–Stokes equations: Adiabatic wall and heat entropy transfer," Journal of Computational Physics, vol. 397, p. 108775, 2019.

[12] M. Parsani, M. H. Carpenter, and E. J. Nielsen, "Entropy stable wall boundary conditions for the three-dimensional compressible Navier–Stokes equations," Journal of Computational Physics, vol. 292, pp. 88–113, 2015.

[13] D. C. D. R. Fernández, M. H. Carpenter, L. Dalcin, L. Fredrich, A. R. Winters, G. J. Gassner, and M. Parsani, "Entropy-stable p-nonconforming discretizations with the summation-by-parts property for the compressible navier–stokes equations," Computers & Fluids, vol. 210, p. 104631, 2020.

[14] R. Al Jahdali, L. Dalcin, I. R. Nolasco, E. D. Keyes, and M. Parsani, "Optimized explicit Runge–Kutta schemes for collocated discontinuous Galerkin methods with the summation-by-parts property for compressible fluid dynamics," Computers & Mathematics with Applications, 2022.

[15] M. Parsani, D. I. Ketcheson, and W. Deconinck, "Optimized explicit Runge-Kutta schemes for the spectral difference method applied to wave propagation problems," SIAM Journal on Scientific Computing, vol. 35, no. 2, pp. A957–A986, 2013.